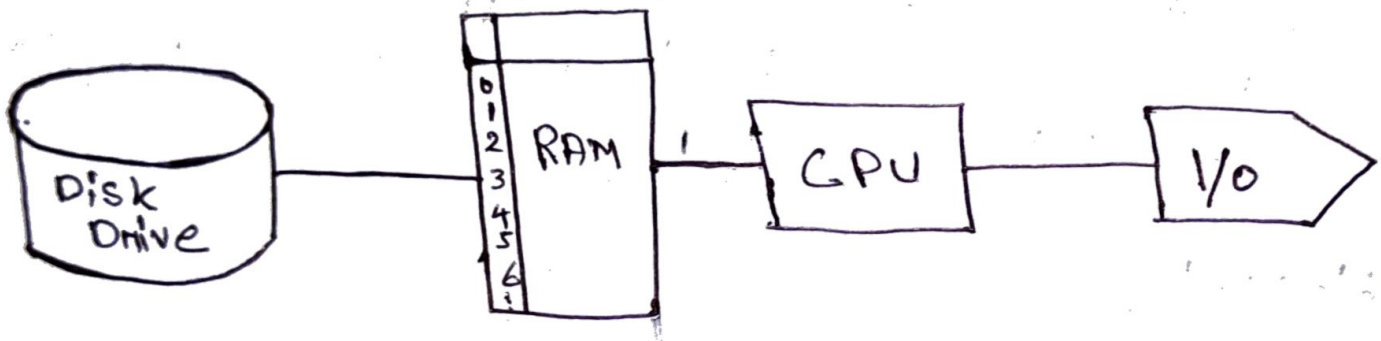
OS Concepts,

A computer is made up of a CPU, random access memory (RAM), input/output device (I/O) devices and long term storages (HDD, SSD),



An operating system is the interface between the users of a computer and the hardware of the Computer,

In particular, an operating system manages the ways applications access the resources in a Computer, including its disk drives, CPU, main memory, input devices, output devices and network interfaces,

It is the "glue" that allows users and applications to interact with the hardware of a Computer,

Operating systems allow users and applications to write programs without having to handle low-level details such as how to deal with every possible hardware device.

Example: like the hundreds of different kinds of printers that a user could possibly connect to his or her computer.

Thus, operating systems allow application programs to be run by users in a relatively simple and consistent way.

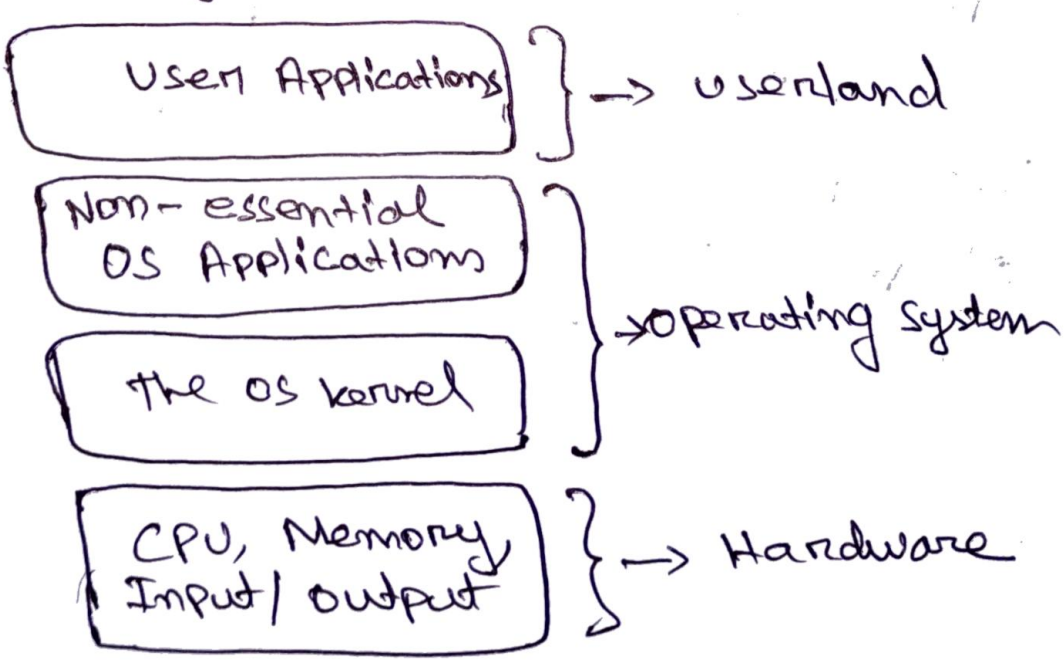
Multitasking,
Give each running program a "slice" of the CPU's time
The CPU is running so fast that to any user it appears that the computer is running all of the programs simultaneously.

Kernel

Kernel is the core component of the operating system.

It handles the management of low-level hardware resources, including memory, processors, and input/output, (I/O) devices, such as a keyboard, mouse, or video display.

Most operating systems define the tasks associated with the kernel in terms of a layer metaphor, with hardware components, such as the CPU, memory, and input/output devices being on the bottom, and users and applications being on the top.



I/O devices,

The input/output devices of a computer include many things,
keyboard, mouse, video display, and network card as well as other more optionale devices like a scanner, wifi interface, video camera, USB ports, and other input/output ports.

Each such device is presented in an operating system using a device driver, which encapsulates the details of how interaction with that device should be done.

The application programmer interface (API), which the device drivers present to application programs, allows those programs to interact with those devices at a fairly high level,

The operating system does the "heavy lifting" of performing the low-level interactions that make such devices actually work.

System Calls,

User applications don't communicate directly with low-level hardware components, and instead delegate such tasks to the kernel, via System call, or syscall for short.

System calls are contained in a collection of programs, that is, a library such as the C library (libc)

for example,

> Performing file I/O (open, close, read, write) and running application Programs (exec)

Process, of a Program

A process is an instance that is currently executing,

The actual contents of all programs are initially stored in persistent storage, such as a hard drive, but in order to actually be executed, the program must be loaded into random-access memory (RAM) and uniquely identified as a process.

In this way, multiple copies of the same program can can be run by having multiple processes initialised with the same program code.

For example, we can be running four different instances of a word processing program at the same time, each in a different window.

When a user creates a new process by making a request to run some program, the kernel sees this as an existing process (such as a shell program or graphical user interface program) asking to create a new process.

Thus Processes are Created by a mechanism Called forking, where a new process is Created (that is forked) by an existing Process.

The existing Process in this action is known as the parent process and the one that is being forked is known as the child process.

On most systems, the new child process inherits the permissions of its parent

Unless the parent deliberately forks a new child process and choose to give the child process fewer permissions if needed.

Due to the forking mechanism for process Creation, Processes have parent-child relationships and are organised in a rooted tree.

known as the process tree.

In Linux, the root of this process is called init process. Init Process starts executing during the boot Process right after the kernel is loaded.

Process IDs,

Each process running on a given computer is ~~identifi~~ identified by a unique ~~so~~ non-negative integer, called the process ID (PID).

In Linux, the root of the process tree is init, with PID 0.

Given the PID of a process we can then associate its CPU time, memory usage, user id (UID), Program name and so on.

Process Privilege,

To grant appropriate privileges to processes, an operating system associates information about the user on whose behalf the process is being executed with each process.

for Example, Unix-based systems have an ID system where each process has a user ID(uid) which identifies the user associated with this process.

Also, there is notion of group ID(gid) identifies a group of users for this process.

The uid is a number between 0 and 32,767 (hexadecimal notation) that uniquely identifies each user.

Typically, uid 0 is reserved for the root (administrator) account.

The gid is a number within the same range that identifies a group the user belongs to.

Each group has a unique identifier, and an administrator can add users to groups to give them varying levels of access.

Process Security

To protect a computer while it is running, it is essential to monitor and protect the processes that are running on that computer.

The trust that we place on the processes running on a computer is an inductive belief based on the integrity of the processes that are loaded when the computer is turned on.

This chain of trust starts from the boot process.

Boot Sequence,

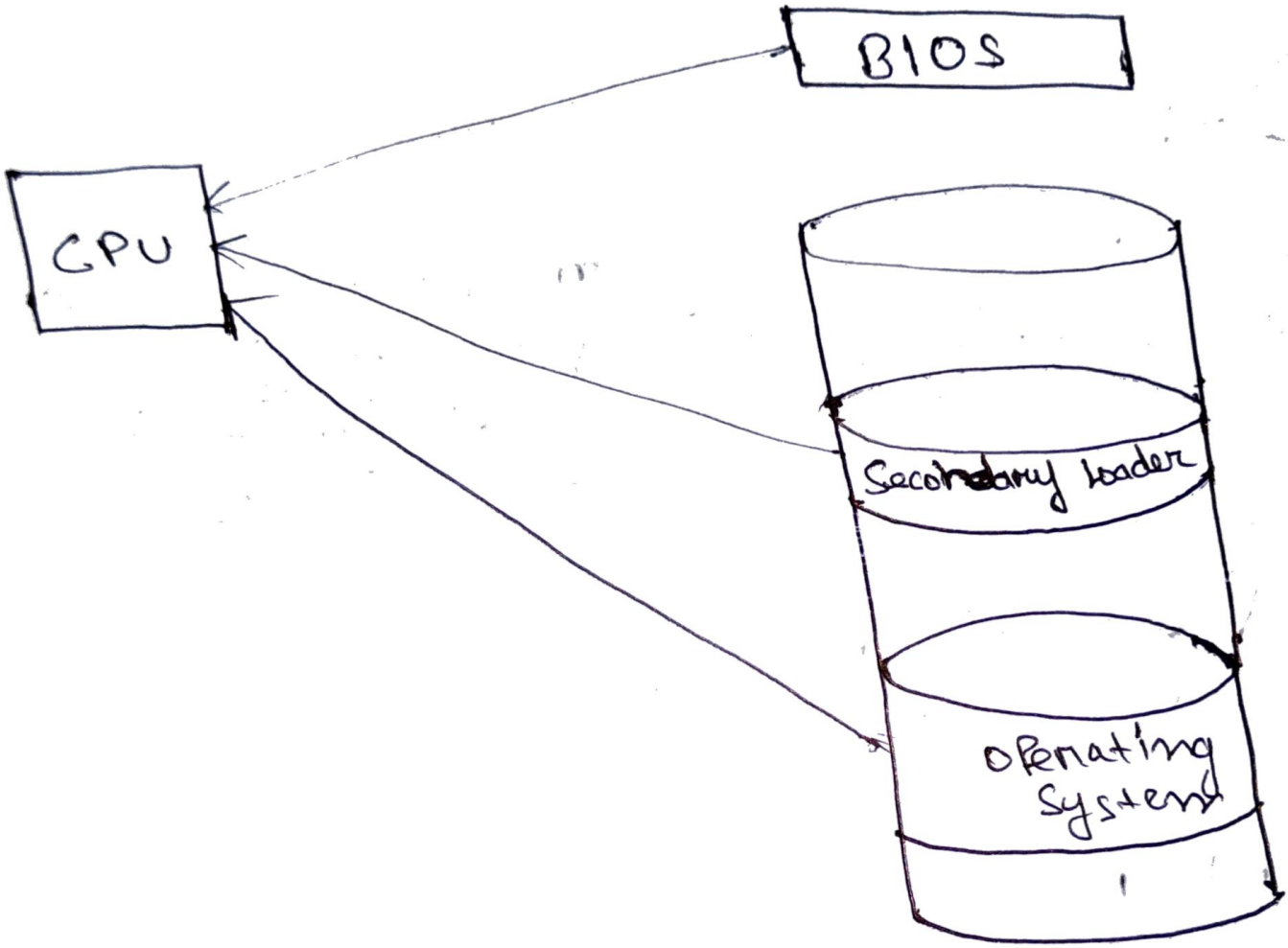
The action of loading an operating system into memory from a powered-off state is known as booting, originally bootstrapping.

All of the operating system's code is stored in hard drive.

In order for operating System to execute, it must be loaded into main memory RAM.

When a computer is turned ON, it first executes code stored in a firmware Component known as the BIOS ~~(basic)~~ (basic input / output system)

On modern systems, the BIOS loads into memory the second-stage boot loader, which handles loading the rest of the operating system into memory and then passes control of execution to the operating system.


BIOS
CPU
Secondary loader
Operating System

A malicious user could potentially seize the execution of a computer at several points in the boot process.

The attacker could even alter the boot sequence.

To prevent an attacker from initiating/altering the first stages of booting, many computer feature a BIOS password that does not allow a second-stage boot loader to be executed without proper authentication.

Monitoring, Management, and Logging

- One of the most important aspects of operating systems security is something military people call "situational awareness", keeping track of:
  - what processes are running
  - what other machines have interacted with the system via the internet.

- If the operating system has experienced any unexpected or suspicious behavior can often leave important clues:
  - For troubleshooting ordinary problems, helps in finding and fixing regular issues with the system.
  - For determining the cause of a security breach, Assists in figuring out what caused a security problem, such as a hack or malware.

Process Monitoring

- There are several scenarios where we would like to find out exactly which processes are currently running on our computer.
  - for example, sluggish performance: If your computer is running slowly, checking the processes can help identify an application that's using too much CPU or memory.
- Or we may suspect that our computer has been compromised by a virus and we want to check for suspecious processes.
  - Task manager application in windows,
  - PS, top, pstree, and kill commands in Linux.