Operating Systems – Learning Phase I

Shriram K Vasudevan B.E., M.Tech., M.B.A., Ph.D.,

- **Author of 22 Books**
- **Author of 86 Journal papers.**
- **Awarded by Infosys, Wipro, IBC (Cambridge), SOS Ventures (USA), Intellectual Ventures, IBM, ICTACT, ASDF Thailand, Amrita, VIT, SASTRA and few more.**

- Simple, keep calm and listen.
- If you have queries, well, you are free to raise.
- If you disturb me, you will be out of the class.
- There will be 10 mins of questioning in every session, if you fail to answer, you are out of the class.
- If you want to sleep, better stay back in room or home.
- Welcome aboard!

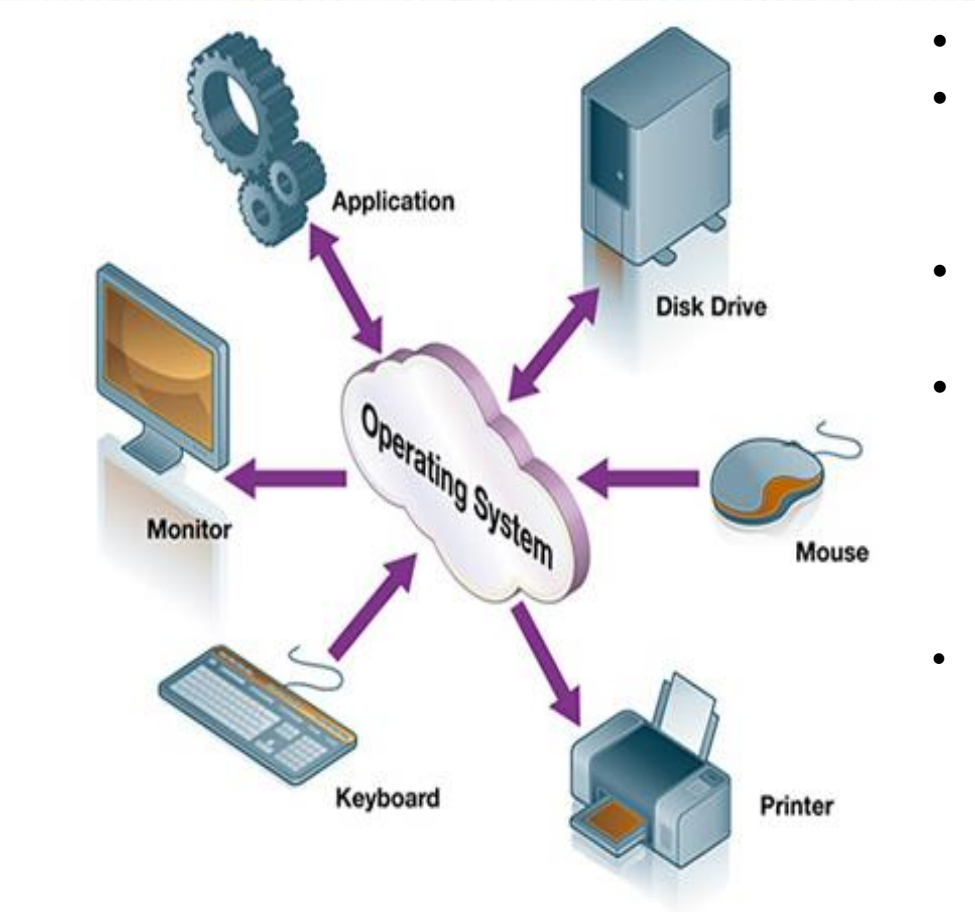
What do I Expect?

Unit 1

Introduction to operating systems: overview - types of systems - computer system operations - hardware protection - operating systems services - system calls - system structure - virtual machines. Process management: process concepts - process scheduling - operations on process - cooperating process - interprocess communication - multi threading models - threading issues - thread types - CPU scheduling – scheduling algorithms.

AGENDA

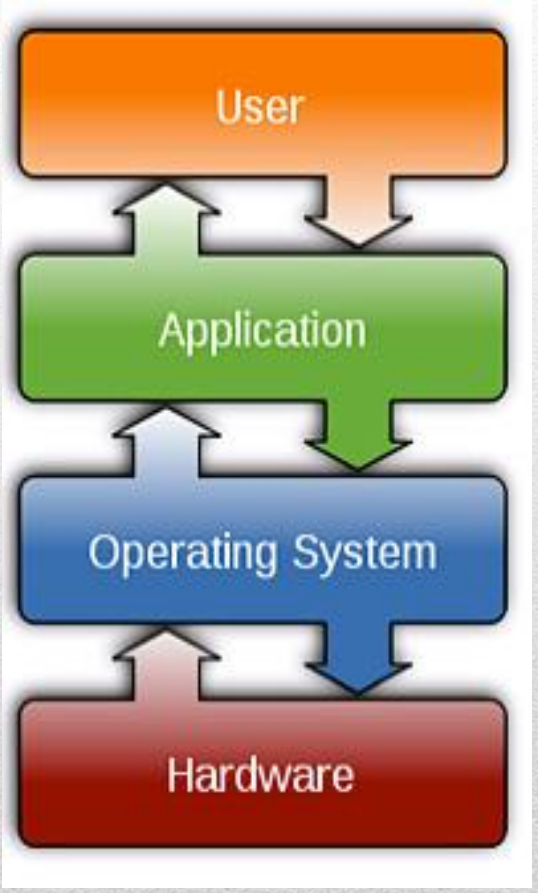
How many of you have brain? Hope everyone has!!



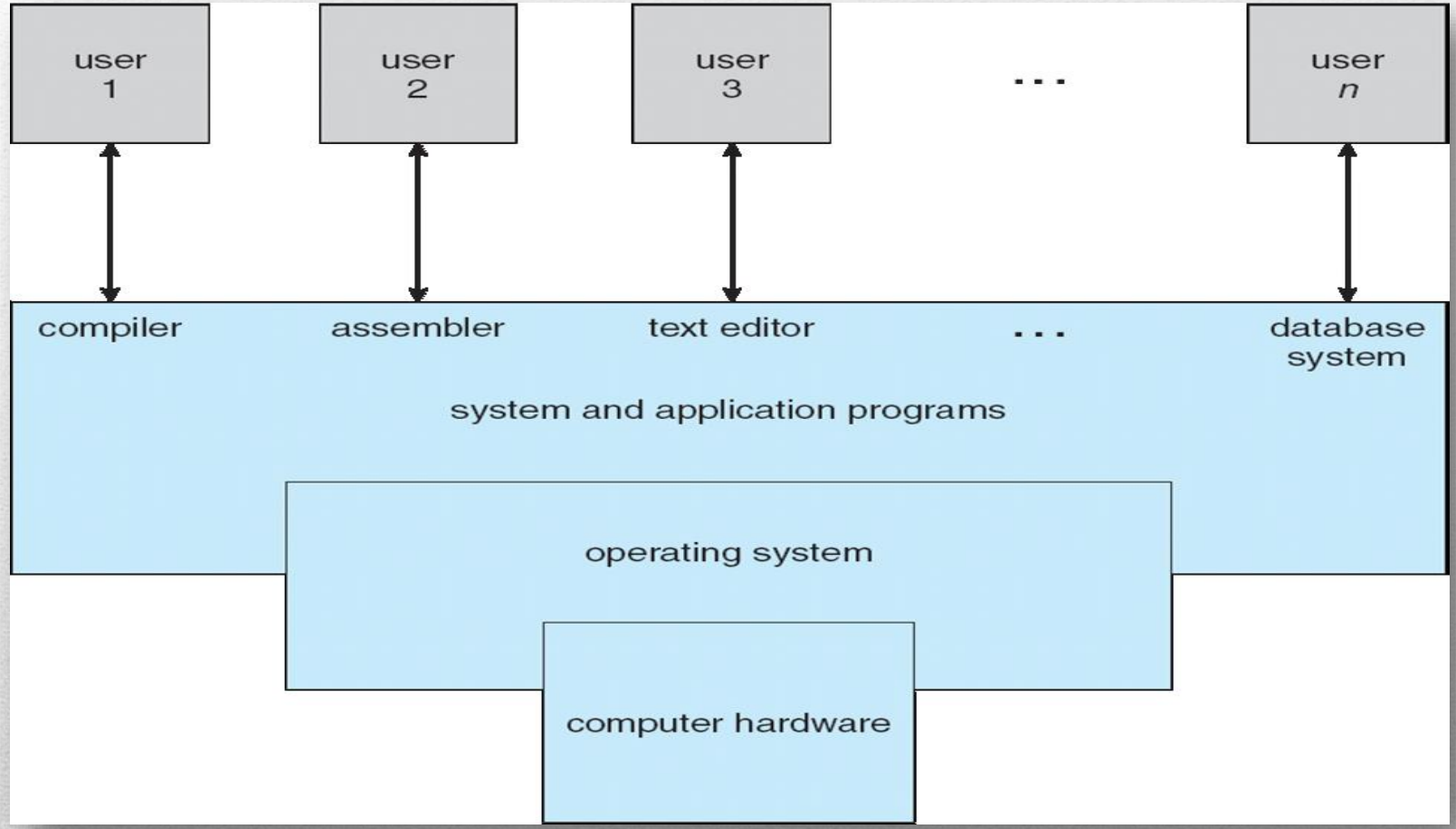
- Operating System is a resource Manager.
- It manages all the resources of a machine, just like a brain does for you to be alive and active!
- It also acts as an interface between the hardware and the user.
- general-purpose computer or Real time computing machine must have an operating system to run other programs.
  - **What is general purpose system?**
  - **What is real time system?**
- perform basic tasks,
  - **Recognizing input from the keyboard**
  - **Sending output to the display screen**
  - **Keeping track of files and directories**
  - **Managing communication**
  - **Manages peripherals (What are they?)**

What is OS?

- It also provides a platform for other software packages to run, these can be also know as “application programs”.
- The operating system not only acts as an interface for hardware and applications but it also offers a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls.
- **Users may also interact with the operating system with a software user interface (UI) by typing commands in to the command line interface (CLI) or using a graphical user interface (GUI).**



Contd.,



Four Basic Components . . A Simple Schematic

- Without an OS, A computer would be just collection of components, really not adding any value to the user.

- Windows or Linux - for personal computers
- MacOS, iOS- for Macs and iPhone and iPad
- Unix - for mainframes
- Symbian, Android - for mobile phones

- Providing a user interface
- Managing the computer's memory
- Managing the hardware

Contd.,

Types of Operating Systems

- **Single User Operating System**
  - Single user, single application
    - **Has to deal with only one person at a time and one application being launched at a time.**
    - **IPod is an example. It will have one user at a time and it will have only one application, max!!! ☺**
    - **Designed with keeping one user in mind, but capable of handling multiple tasks. Example, your laptop, mobile phone etc.,**

- **Multi-user operating systems**
  - **Multi – user and multi – tasking**
- Personal computers can multi-task very well, especially for the type of things that most of us want to do, for example, reading emails, writing letters, working on spread sheets, listening to music, surfing the web and watching videos.
- However, there comes a time when only a *really* powerful computer will do the job in hand.
- Some instances like:

Contd.,

- You are an engineer or scientist and want to run a very complicated simulation
- You are a weather scientist and want run a forecast
- You are a financial person and want to work on thousands of stock market share movements
- You work in a bank and want to handle customer accounts.
- You are an architect and want to see your full design
- You work at a University as an academic along with hundreds of other academics
- You are a film animator and want to work in 3D

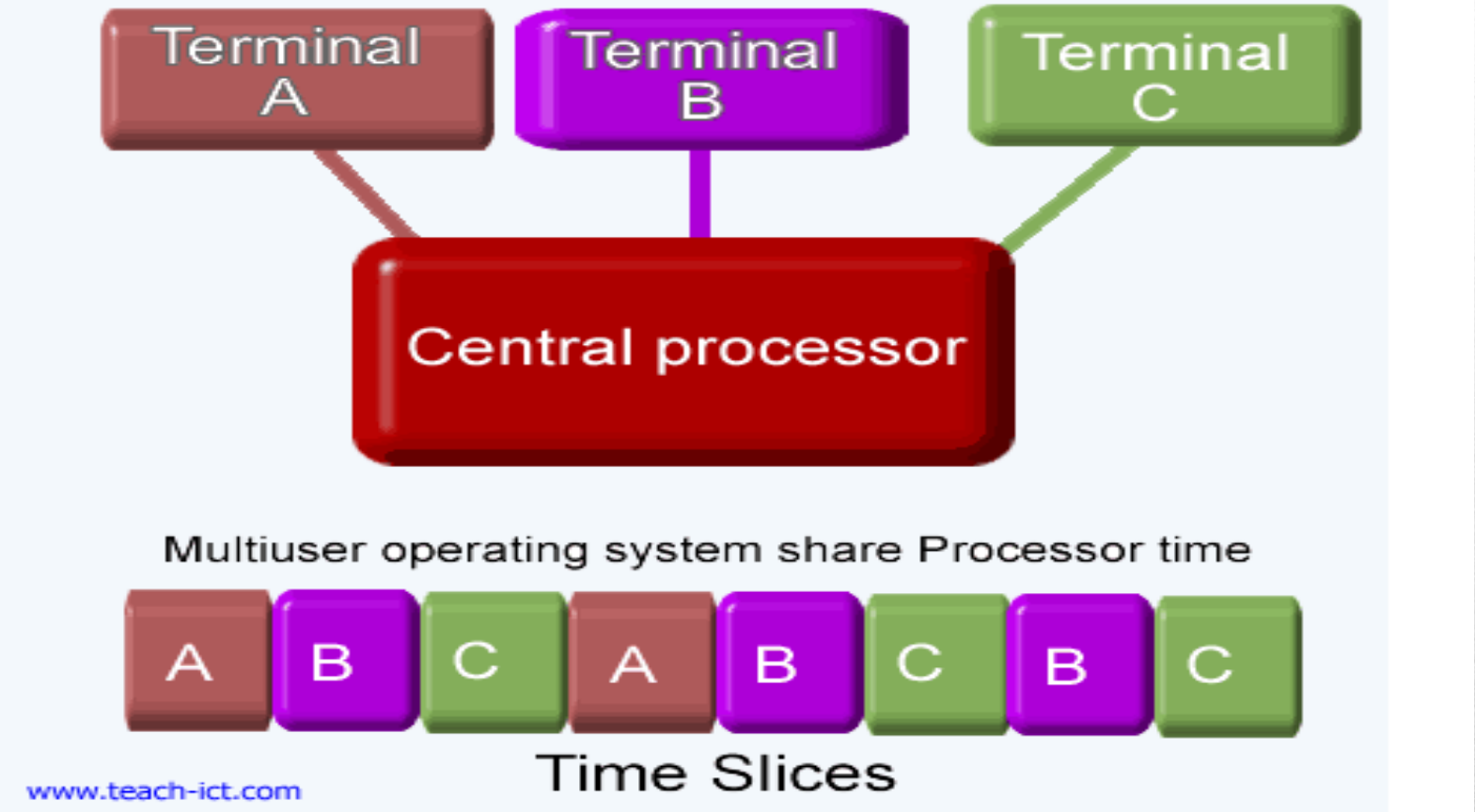
- Most personal computers can't handle these kind of tasks. Instead, a mainframe or supercomputer is required for this kind of work.

Contd.,

- **Supercomputer or mainframe costs millions to buy and maintain. There is no way that such an expensive machine could be used by just a single person.**
- To make it economic, this computer has to be shared. This means it needs a multi-user operating system. This allows more than one user to log on and can use the computer and its resources at the same time.
- Furthermore, each person needs to be able to run more than one application at a time, so it needs to be multi-tasking as well.

- So a powerful computer needs a **multi-user, multi-tasking** operating system to make maximum use of the machine. Each person can draw on the vast power of the computer in a shared way.
- Now the operating system has to manage
  - **Each user logged on to the system, their workspace and so on.**
  - **Allocate resources to the jobs they want to run.**
  - **Keep logs of how much processing time and resources they use**
  - **Work out the most efficient use of computer processing cycles**
  - **Maintain security**

Contd.,


Terminal A
Terminal B
Terminal C
Central processor
Multiuser operating system share Processor time
A
B
C
A
B
C
B
C
Time Slices
www.teach-ict.com

Contd.,

- **Networked system**
- Another common example of a multi-user environment is a standard personal computer running a single user, multi-tasking operating system connected to a network.
- On the network itself is present a server loaded

Contd.,

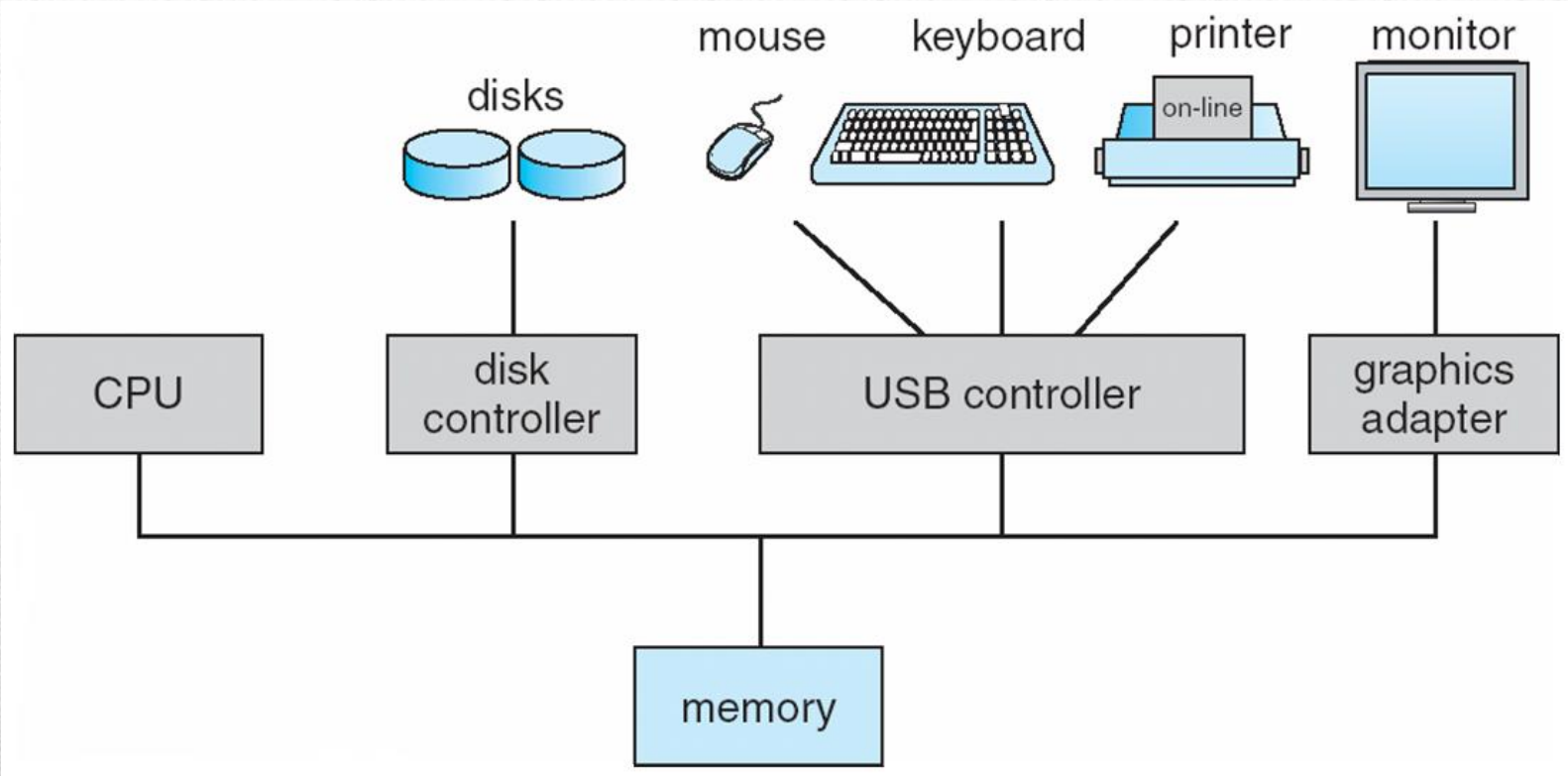
- **What is Real Time?**
- **How is Real Time OS made?**
- **Is RTOS related to Embedded Systems?**
- **What is an embedded system?**

- Let’s answer one by one!!! ☺

Real Time OS

Computer System Operation

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory
  - What is a bus?
  - How does it work?
- Concurrent execution of CPUs and devices competing for memory cycles.
- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, and video displays).
- The CPU and the device controllers can execute concurrently, competing for memory cycles.
- To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory.


mouse
keyboard
printer
monitor
disks
on-line
CPU
disk controller
USB controller
graphics adapter
memory

Contd.,

- When a system is powered up or rebooted-it needs to have an initial program to run. **(What do you call this??)**
- **It is referred to be as Boot-Strap program and it is the first one to wake up in the machine!!**
- Where do we store the Boot Strap code?
  - Simple. ROM!! (Read Only Memory)
  - What is the address in which it will be stored in ROM?
  - **Answer why?**
  - **This is referred also to be as firmware!!**

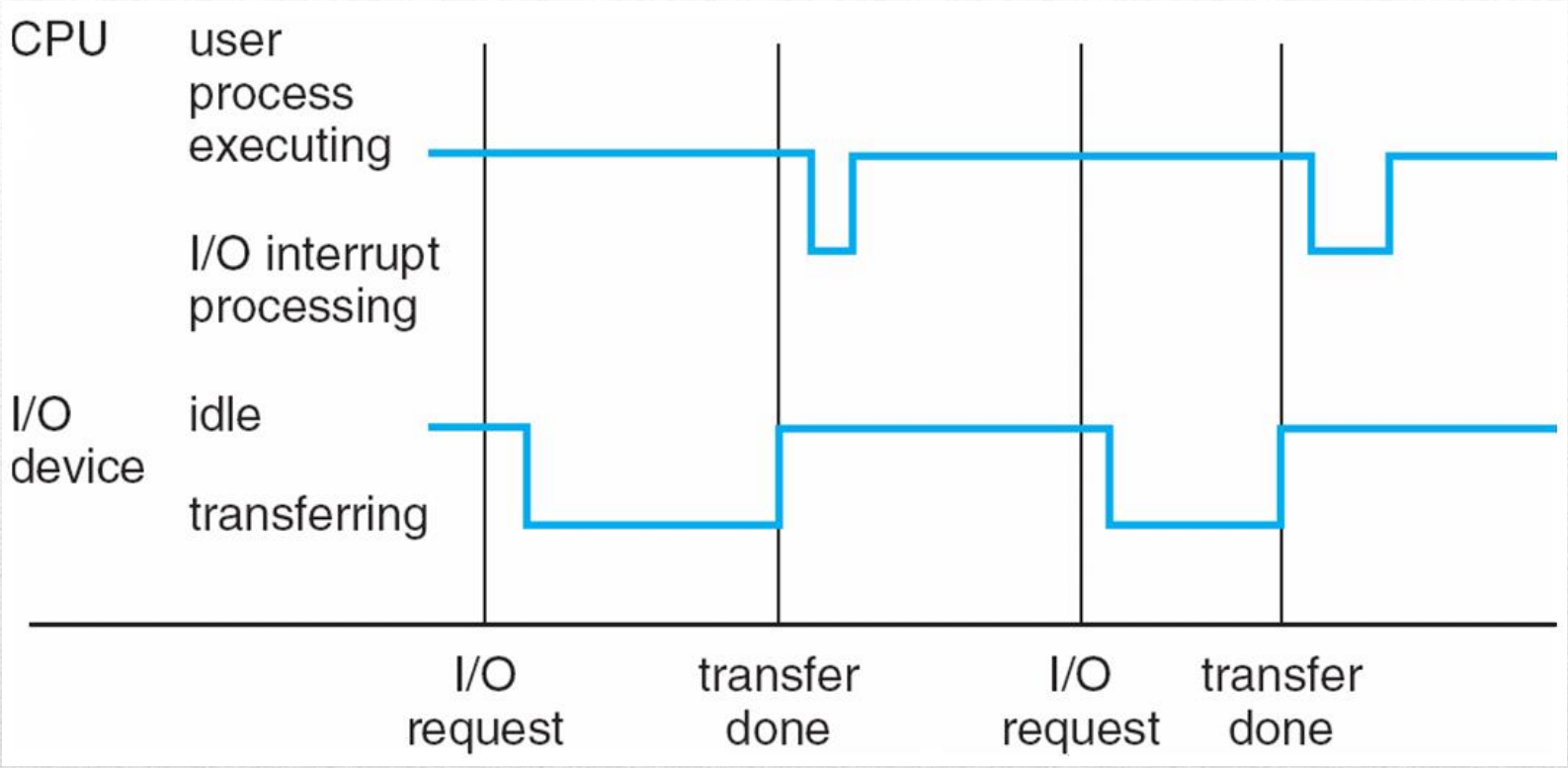
Contd.,

- It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program must know how to load the operating system and how to start executing that system.
- To accomplish this goal, the bootstrap program must locate and load into memory the operating system kernel.
- The operating system then starts executing the first process, such as "init," and waits for some event to occur.
- **Let’s see where is init in my machine!!!**

Contd.,

- What is an interrupt?
  - **How is it different from Polling?**
  - **An example please??**
- The occurrence of an event is usually signalled by an from either the hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus. Software may trigger an interrupt executing a **System call.**
  - **What is a system call?**
  - **We will learn it through demos!! Hold on till then!**

Interrupt..



Interrupt Cycle example

- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located.
- The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.
  - ***One example here, I ask you questions right??***
- Events are almost always signalled by the occurrence of an interrupt or a t r a p . (division by zero or invalid memory access)

Contd.,

- Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism, but several functions are common.
- **The interrupt must transfer control to the appropriate interrupt service routine.**
- **The straightforward method for handling this transfer would be to invoke a generic routine to examine the interrupt information; the routine, in turn, would call the interrupt-specific handler.**
- **However, interrupts must be handled quickly. Since only a predefined number of interrupts is possible, a table of pointers to interrupt routines can be used instead to provide the necessary speed.**
- **The interrupt routine is called indirectly through the table, with no intermediate routine needed. Generally, the table of pointers is stored in low memory (the first hundred or so locations).**
- ***An Example : Car braking system, Flight Landing Gear Setup!***

Contd.,

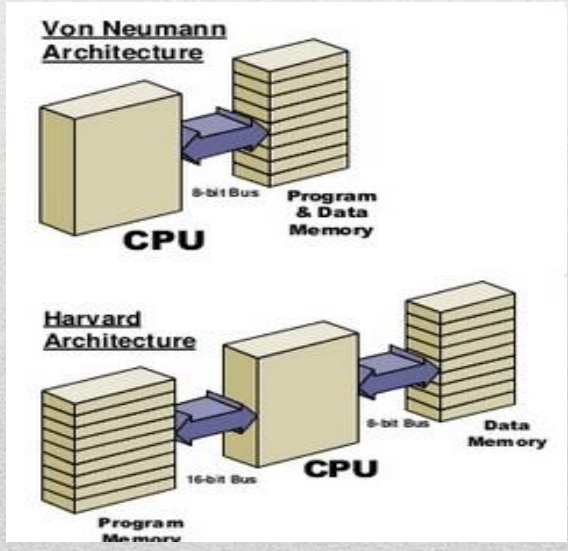
- These locations hold the addresses of the interrupt service routines for the various devices.
- This array is called ***Interrupt Vector!!***
- ***How will you come back to the original place, after servicing the interrupt request?***
  - ***Store the return address on to the stack!!***
  - ***Context Switching!!!***

Contd.,

- The CPU can ***load instructions only from memory***, so any programs to run must be stored there. Now you understand, ***why do we need memory?***
- General-purpose computers run most of their programs from rewriteable memory, called main memory (also called or ***RAM***).
- Because the ***read-only memory (ROM)*** can't be changed, only static programs are stored there. The immutability of ROM is of use in game cartridges.
- ***EEPROM*** can't be changed frequently and so contains mostly static programs. For example, smartphones have EEPROM to store their factory installed programs

Storage Structure

- A typical instruction-execution cycle, as executed on a system with a Von-neumann architecture, first fetches an instruction from memory and stores that instruction in the instruction register. (**What is the other architecture available?)**
- The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register.
  - ***What is a register?***
  - ***What is it made of?***



Contd.,

- After the instruction on the operands has been executed, the result may be stored back in memory. **I don't care about how the result has arrived!! I care on the result and not on how the result!**
- Mostly, we want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following two reasons:
- **Main memory is usually too small to store all needed programs and data permanently.**
- **Main memory is a *volatile* storage device that loses its contents when power is turned off or otherwise lost.**
- **So what is the solution?**
  - Keep extension memory.
  - May be in format of Hard Drive or Similar Structure.

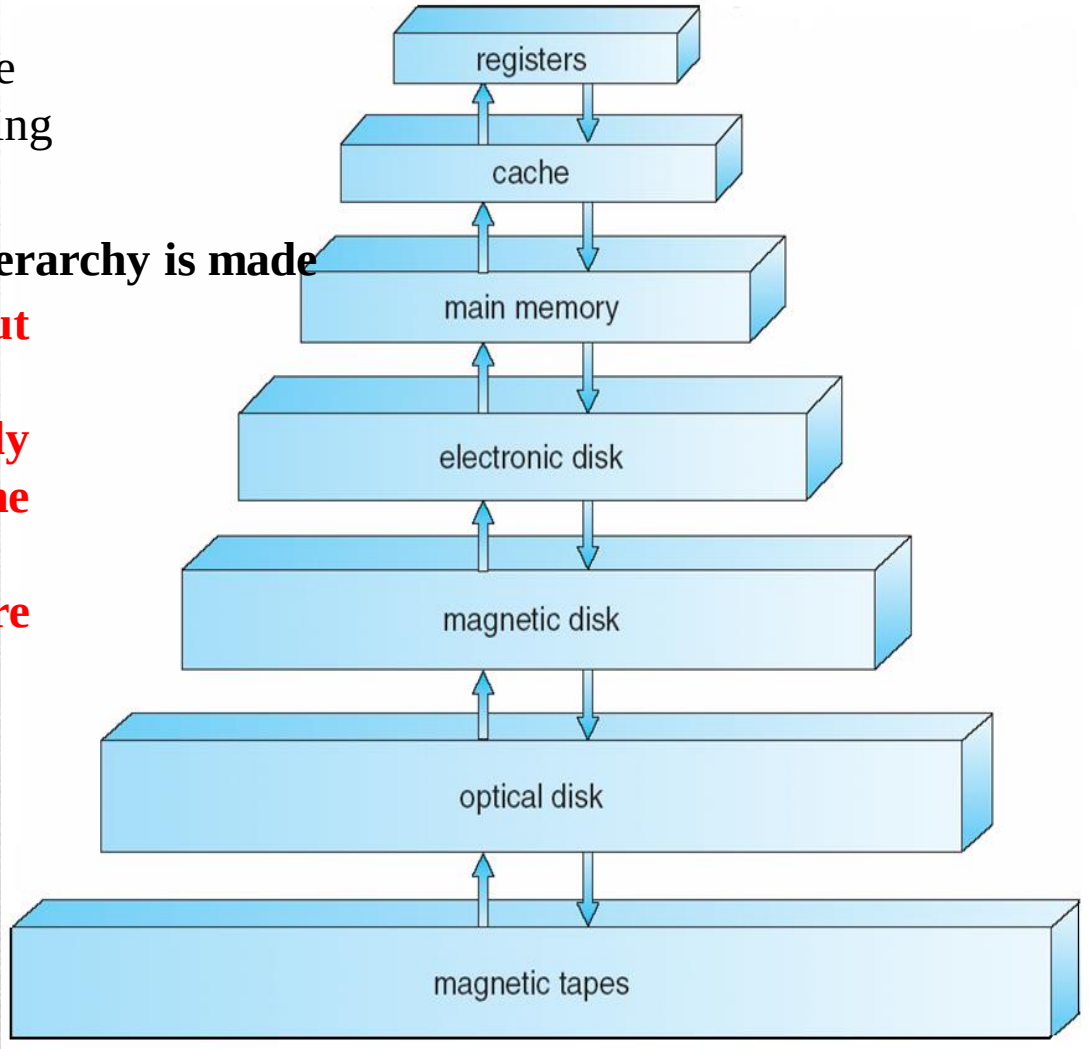
Contd.,

- The main requirement for secondary storage is that it be able to **hold large quantities of data permanently**.
- The most common secondary-storage device is a **Magnetic Disk** which provides storage for both programs and data.
- **Most programs (system and application) are stored on a disk until they are loaded into memory.**
- **Many programs then use the disk as both the source and the destination of their processing.**
- **So handling it is a challenge… We will see it shortly!!**

Contd.,

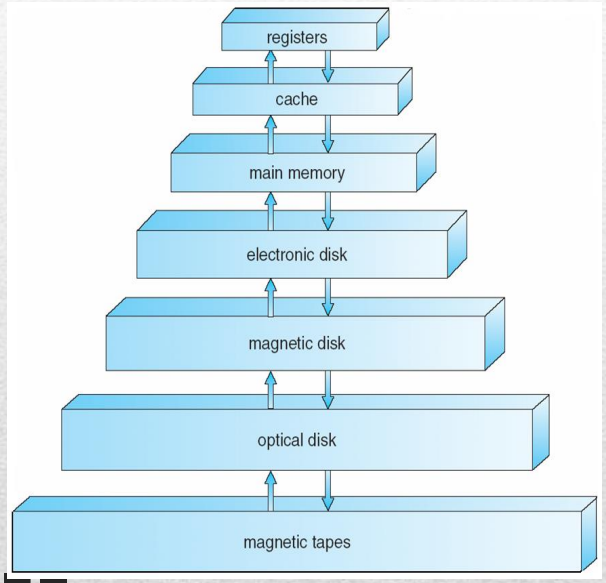
Hierarchy

- **Speed, cost, size, and volatility** are the parameters hold a lot of value in deciding right medium for storage.
- **According to speed and cost, The hierarchy is made**
  - **The higher levels are expensive, but they are fast.**
  - **The cost per bit generally decreases, whereas the access time generally increases.**
  - **The various storage systems are either volatile or non-volatile**
  - **What is volatility?**
    - **Loses its contents when the power to the device is removed.**



- In the absence of **expensive battery and generator backup** systems, data must be written to for safekeeping.

VOLATILE



NON - VOLATILE

An electronic disk can be designed to be either volatile or non-volatile.

Where Non-Volatile Systems are needed?

- Anything and everything that happens from the computer system is through the I/O.
- Storage even is supported through I/O alone.
- So, it is inevitable to know the I/O structure!! ☺
- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
  - ***What is a bus?? How does it work?***

I/O Structure

- **Each device controller is in charge of a specific type of device**. Depending on the controller, there may be more than one attached device.
- For instance, **seven or more devices** can be attached to the **small computer-systems interface (SCSI)** controller.
- A device controller maintains some **local buffer storage** and a set of **special-purpose registers.**
  - **What is a special purpose register?**
  - **What is it made of?**
- The device controller **is responsible for moving the data between the peripheral devices** that it controls and its local buffer storage.

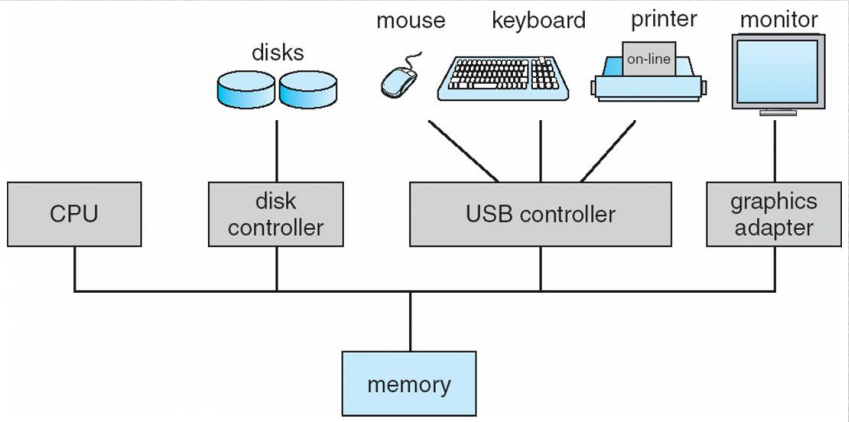
Contd.,

- **Device driver is there for every device which is being attached to the system.**
- Device driver **understands the device controller** and presents a uniform interface to the device to the rest of the operating system. **(Example??? When you mount your USB to your PC!)**
- **How does it work????**

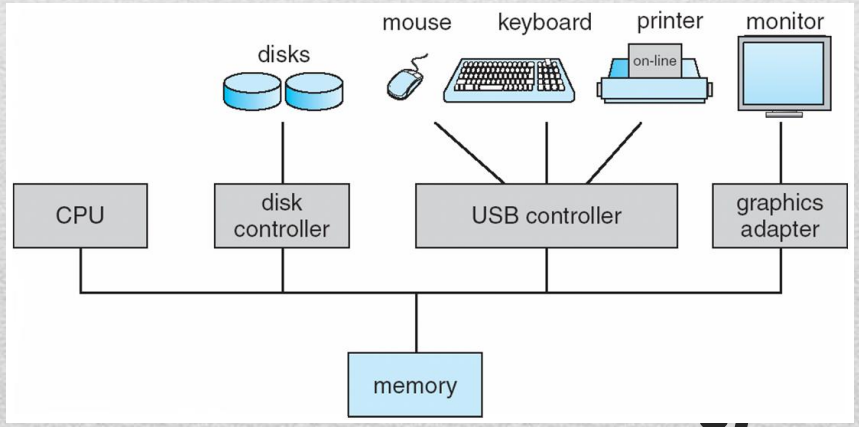
Contd.,

- **To start an I/O operation ➔**
- The device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as "read a character from the keyboard")
- The controller starts the transfer of data from the device to its local buffer.

Contd.,



- Once the transfer of data is complete, **the device controller informs the device driver via an interrupt** that it has finished its operation.
- The device driver then returns control to the operating system, **possibly returning the data or a pointer to the data** if the operation was a read.
- For other operations, the device driver returns status information.
- This is **interrupt driven IO**



Contd.,

- Interrupt driven IO is suitable for smaller chunk of data. How about big volumes of data??
- Answer is **DMA (Direct Memory Access)**
- **Lets look into it!!**

Contd.,

DMA

AN EXPLORATION

HOW DMA WORKS?? DIRECT MEMORY ACCESS!!!

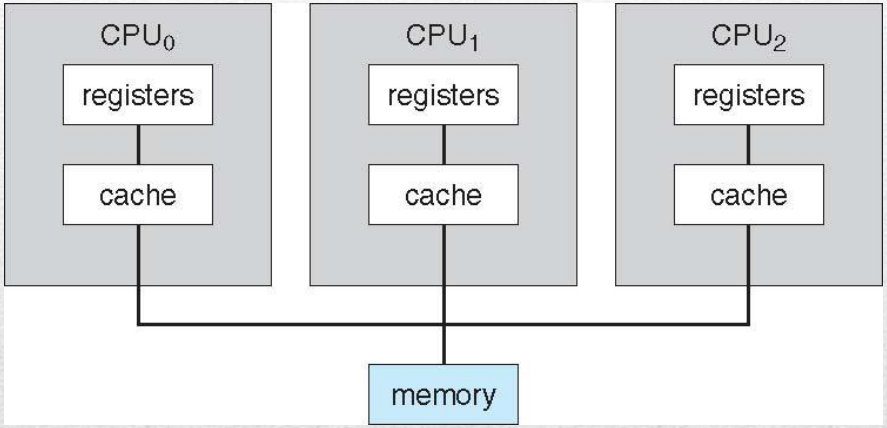
Computer-System Architecture

- **Single-Processor Systems**
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability – graceful degradation** or **fault tolerance**

- The multiple-processor systems in use today are of two types.
- Some systems use **asymmetric multiprocessing,** in which each processor is assigned a specific task.
- A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks.
- This scheme defines a master-slave relationship. The master processor schedules and allocates work to the slave processors.

Contd.,

- The most common systems use **symmetric multiprocessing (SMP),** in which each processor performs all tasks within the operating system.
- SMP means that all processors are peers; no master-slave relationship exists between processors.



- ***Assignment – Clustered Systems***

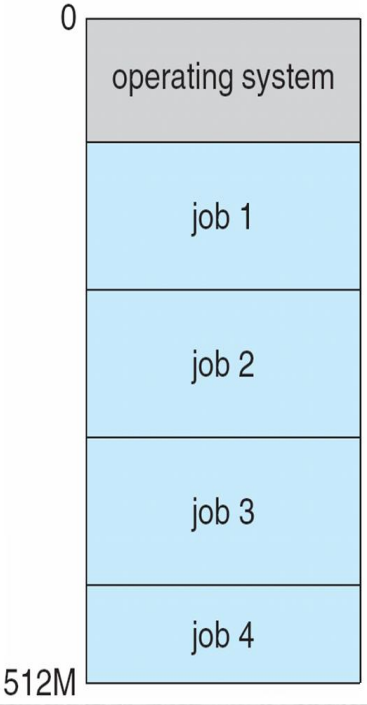
Contd.,

Operating System Structure

- ***One of the most important aspects of operating systems is the ability to multiprogram. - This is what makes you browse while you listen to songs!!***
- **Multiprogramming** increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute. ***We call it scheduling!***

Operating-System Structure

- ***The operating system keeps several jobs in memory simultaneously. Refer the figure in parallel!***
- ***The operating system picks and begins to execute one of the jobs in memory.***
- Eventually, the job may have to wait for some task, such as an I/O operation, to complete.



Contd.,

- In a multiprogrammed system, the operating system simply switches to, and executes, another job.
- When *that* job needs to wait, the CPU is switched to *another* job, and so on. **(can we have an example here??)**
- Eventually, ***the first job finishes waiting and gets the CPU back.***
- As long as at least one job needs to execute, the CPU is never idle.

Contd.,

- One more example would be awesome!! ☺
- A ***lawyer does not work for only one client at a time***, for example.
- While one case is waiting to go to trial or have papers typed, the lawyer can work on another case. If he has enough clients, the lawyer will never be idle for lack of work.

- **Multiprogrammed systems** provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively.
- **Time sharing** (or **multitasking)** is a logical extension of multiprogramming.
- In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

Contd.,

- Time sharing requires an **interactive** (or **hands-on) computer system,** which provides direct communication between the user and the system.
- The user gives instructions to the operating system or to a program directly, using a input device such as a keyboard or a mouse, and waits for immediate results on an output device.
- Accordingly, the **response time** should be short typically less than one second. **(If more what will happen??)**

Contd.,

- **A time-shared operating system allows many users to share the computer simultaneously.**
- Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to his use, even though it is being shared among many users. (**Virtual!!!!)**

Contd.,

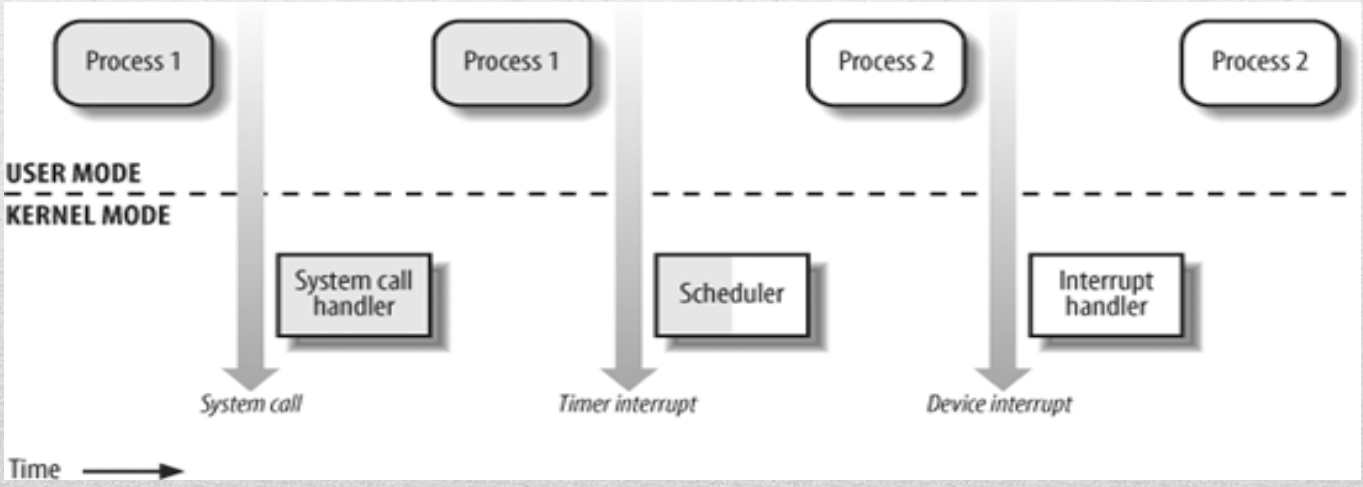
- A program in execution! What is it?? **Let's see it in Linux!!! It gets better here buddy.**
  - **How do you track a process??**
  - **What will be a process id?**
  - **Can you kill a process??**
- **Process Management**
  - **Creating and deleting both user and system processes**
  - **Suspending and resuming processes**
  - **Providing mechanisms for process synchronization**
  - **Providing mechanisms for process communication**
  - **Providing mechanisms for deadlock handling**

Process!!! What is it?

- Time-sharing and multiprogramming require several jobs to be kept simultaneously in memory.
- Since in general main memory is too small to **accommodate all jobs**, the jobs are kept initially on the disk in the **job pool.**
- If several jobs are ready to be brought into memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is **job scheduling.**
- When the operating system selects a job from the job pool, it loads that job into memory for execution. Having several programs in memory at the same time requires some form of **memory management**.
- In addition, if several jobs are ready to run at the same time, the system must choose among them. Making this decision is **CPU scheduling**
- **Virtual Memory** is as well an important component of the system.

Contd., What will constitute OS!

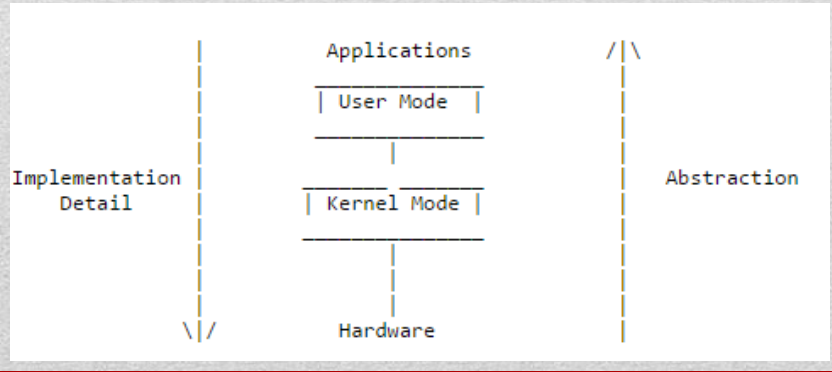
- **Kernel Mode** - Fully privileged mode and has complete control and access to the hardware**.**
- **User Mode -** When the computer system is executing on behalf of a user application, the system is in user mode. when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfil the request.

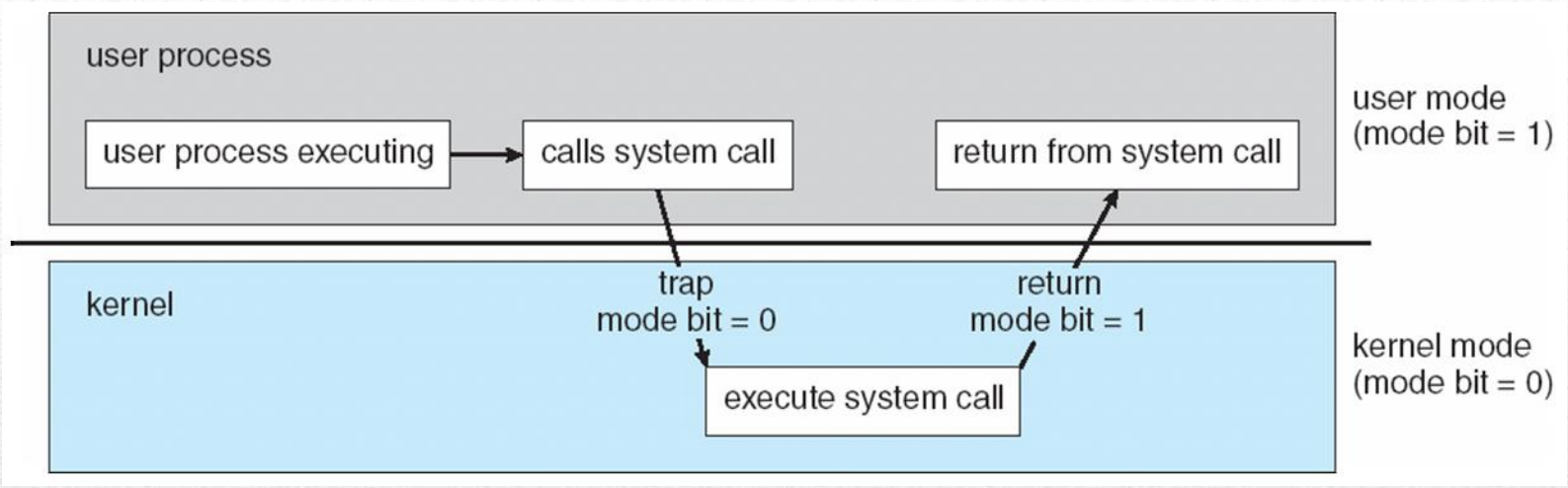


Two Modes of Operations

- At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user mode.
- Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0). Thus, whenever the operating system gains control of the computer, it is in kernel mode.
- The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program

Contd.,




user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

Contd.,

What Next??

Operating-System Operations

Let’s see through this!!!