Unit 1 . CSC-4 Operating System

Operating system is an Interface between a computer user and computer hardware.

It is a software which performs all basic tasks like file management, memory management, Process management, handling Input/output and controlling controlling peripherals like disk drives & printers.

Some popular operating system includes DOS, Linux, Unix, windows operating system etc.

Goals (1) Primary goal → convinience. (2) Secondary goal → Efficiency

Characteristics and Functions of operating system —

(a) Memory Management → It refers to the management of the primary memory or Main memory.

Main memory provides a fast storage that can be accessed directly by CPU. For a program to be executed, it must be in main memory.

OS does following activities for memory management

→ keeps tracks of primary memory ie what part of it are in use by whom, what part are not in use.

→ In multiprogramming OS decides which process will get the memory, when & how much.

→ Allocates the memory when a process requests it to do so

→ Deallocates the memory when process no longer needs it or —

e Process has been terminated

PROCESSOR MANAGEMENT

In multiprogramming environment, OS decides which process gets the processor when and for how much time. This function is called Process SCHEDULING.

OS keeps track of following activities for Processor Management

→ Keeps track of Processor & State of the Process
→ Allocates CPU to a Process
→ Deallocates CPU from a Process
→ Providing mechanism for Process communication and Process synchronization.

3) Device Management

An OS manages device management via their respective drivers. It does the following activities for device management.

→ Keeps tracks of all devices.
→ Decides which process gets the device when and for how much time.
→ Allocates the device in most efficient way.
→ Deallocate devices.

(4) File Management

→ Mapping files to Secondary Storage
→ Backup of files
→ Keeps track of information, location, status, uses.
→ Creating & Deleting files
→ Decides who gets the resources.
→ Allocates the resources. (H/w – Memory, Devices
S/w – files)
→ De-Allocates the resources.
→ Creating & Deleting Directories to organize files

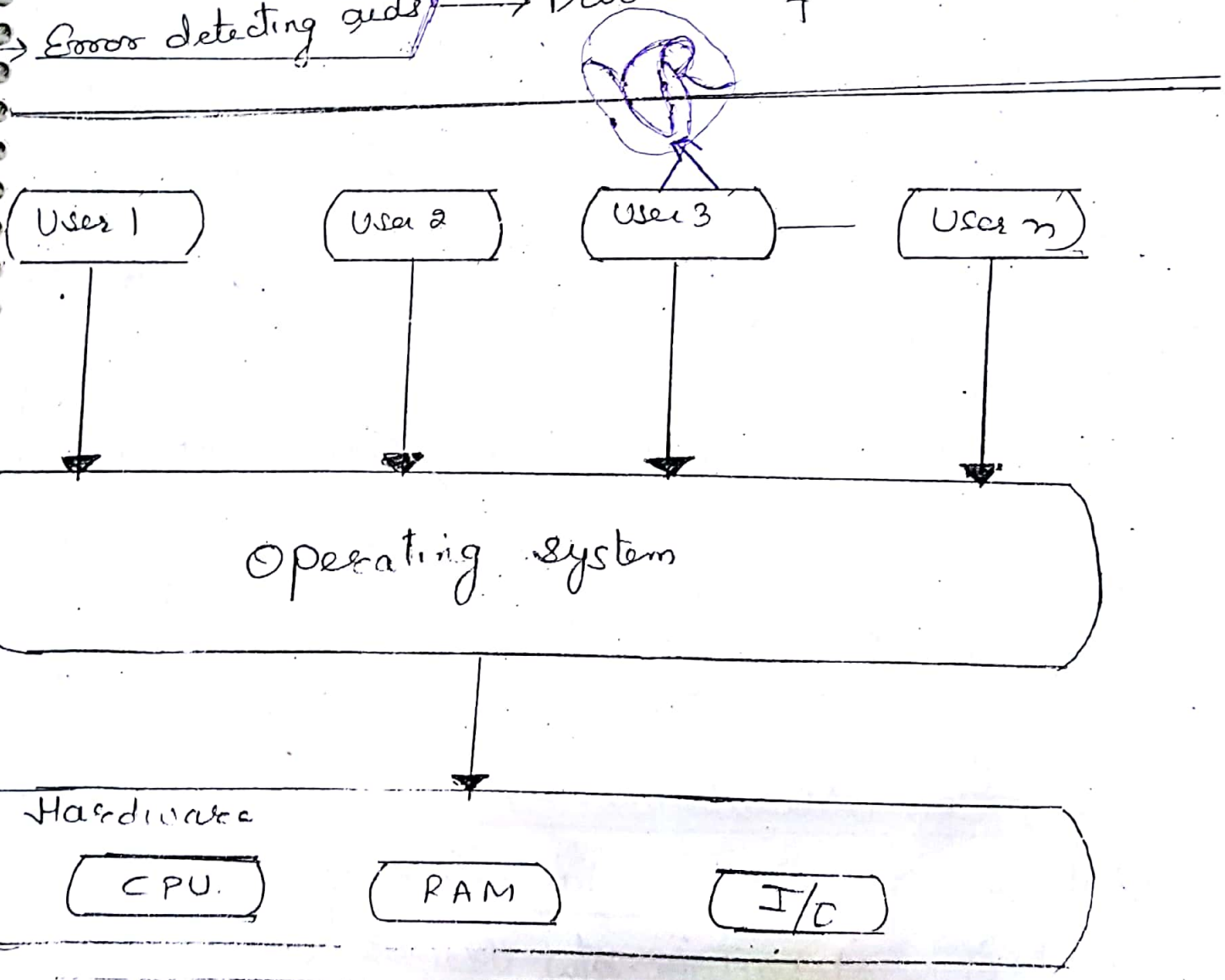
Here are some activities that a OS performs: -

→ Security → It provides security by means of password. It prevents unauthorized access to Programs & Data

→ Control over system Performance · Record Delay B/w request service & response time

→ Job accounting → keep track of time & resources used for various jobs & users

→ Error detecting aids → Production of error messages



Historic evolution of O/S | Different type of OS (2)

1) Serial Processing → From 1940's to mid 50's the user interacted directly with computer hardware. The m/c were run with a console consisting of display lights, some for of I/P device. Program in m/c code are loaded with I/P device like card reader. Programmers examined the registers & main memory to determine errors. If Program is success, O/P will appear on printer.

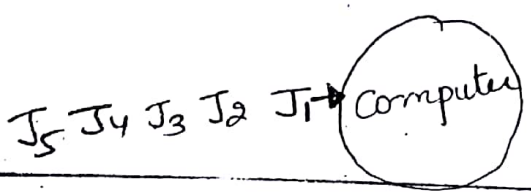
Limitation → Time consuming.

(2) Batch System → In a Batch OS, there is only one single computer called mainframe, Every user will give Program to mainframe and the Operator will place them in a queue and the jobs will be processed one by one

Limitation →

(1) If Job J_1 takes more time ie 10 days other jobs has to wait

(2) If a job J_1 has to go for Input-output



J_1, J_2, J_3, J_4, J_5 are jobs Processed in sequential manner

then CPU will be idle.

∴ CPU utilization is Low

(3) It can result in Starvation of Processes.

(4) Lack of Interaction b/w User & Job.

1) Multiprogrammed system → Then came Multiprogram system.

A single program user can't keep CPU or I/O-O/p devices busy all the time.

OS keeps multiple programs in memory simultaneously. This is called multiprogrammed system.

→ In multiprogrammed system multiple processes are residing in the main memory where only one process is running at a particular instant of time. The running process keeps executing till blocked for IO & next program in line takes turn for execution.

The goal is to optimize CPU utilization by reducing CPU idle time.

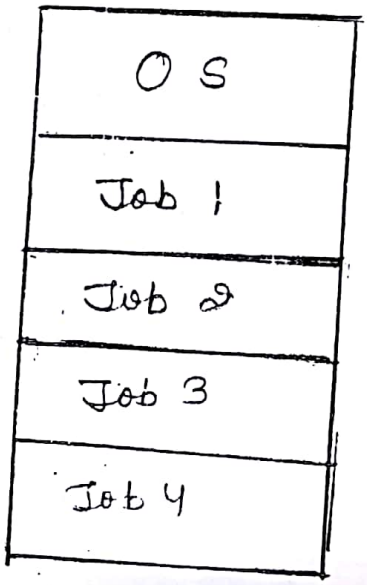


FIG.: Multiprogramming

1)

Time sharing systems → Timesharing or multitasking

multitasking is a logical extension of the multiprogramming. That is Processor time is shared among multiple users, & simultaneously is called time-sharing.

Multiple jobs are executed by the CPU by switching b/w them but switches occur so frequently. This the user can receive an immediate response. Eg If n users are present, each user will get the time quanta. The time quanta is of order of milliseconds

Advantage → (1) Less Response time.

eg IBM's OS/360.

The main diff b/w multiprogrammed and Time Sharing is that multiprogrammed main objective is to increase CPU utilization whereas Time-Sharing main objective is to minimize response time

Limitation of Timesharing →

1. Issue of Reliability
2. Problem of Data communication.

Real Time operating system

Real time system is defined as data processing system in which time interval required to process and respond to inputs is so small that it controls the environment

Real time systems have very Less response time.

Real time systems are used when there are requirements of time requirments.

Real time system must have well defined, fixed time constraints, Otherwise system will fail eg Scientific experiments, medical imaging system, robots, air traffic control system

Hard real time systems →

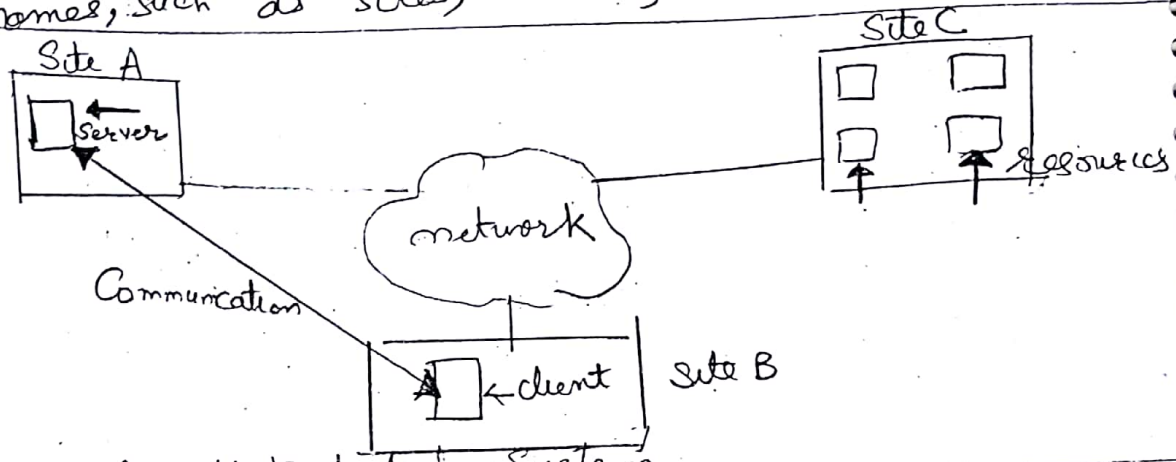
Hard real time systems guarantees that critical task complete on time. In Hard real time systems Secandary storage is Limited or missing. Virtual memory is almost never found eg components of Pace makers, aircraft control missiles system Systems.

Soft real time systems →

Soft real time systems are Less restrictive. Critical real time task gets Peiority over other tasks and retains the Priority until it completes. eg multimedia, Advance Scientific projects like undersea exploration etc.

5) Distributed systems → It is a collection of loosely coupled processors interconnected by a communication n/w.

They may include small microprocessors, workstation minicomputers and Large general-purpose computer systems. The processors are referred to by a no of names, such as sites, nodes, m/c and hosts.



A distributed System

4 advantages of Distributed System

1. Resource sharing → If no of different sites are connected to another, then user at one site may be able to use the resources available to another eg User at site A may be using printer at site B.

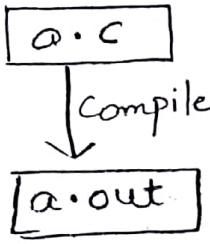
2. Computational speedup → If a particular computation can be partitioned into subcomputations that can run concurrently then a distributed system allows us to distribute these subcomputations among various sites. Subcomputations can be run concurrently - Provide speed up.

3. Reliability → If one site fails then the remaining sites can continue operating, giving system better reliability.

4. Communication :- When several sites are connected to each other via communication N/w, users at various sites can exchange information. People at geographically distant sites can collaborate on a project.

Process in Memory

Suppose there is a Program a.c in High Level Language and compiler converts it into a.out in binary form. Both a.c and a.out are Programs that reside in secondary memory. Operating system converts it into Process and stores them in main memory



Both a.c and a.out are Programs

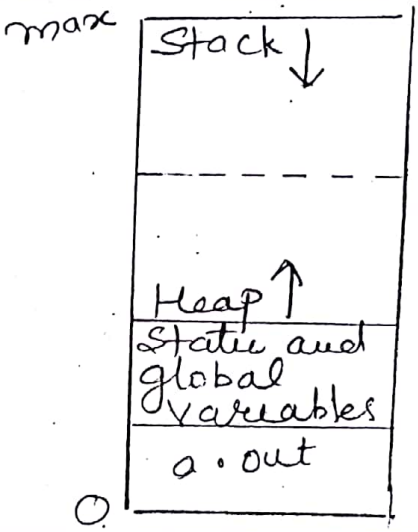


FIG SHOWING PROCESS IN MAIN MEMORY

In main memory a Process contains

1) a.out (Binary form of the Program)
2) Static and global variables that reside in memory throughout the lifetime of the Program
3) Heap Storage - It is dynamic storage that grows from Bottom to top at run time.
4) Stack Storage - It grows from top to bottom. It is used for recursive fn calls

Note → Heap & Stack grow in opposite directions because it is not known that how much space will be required for Heap and Stack Storage.

Process control Block |=▷

Each Process is represented in O.S called PCB (Process Control Block). It contains many information related to a Process.

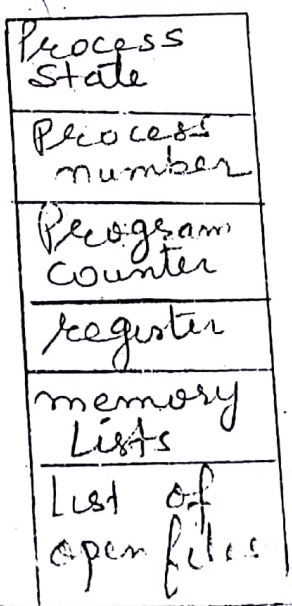
1) Process state → Process can be in any of the states

2) Program counter → It contains address of next I/S to be executed

3) CPU registers → They vary in no & size. They include accumulator, index registers

4) Memory Management → It contains value of base & Limit registers

5) List of open files → It contain list of open files which are opened by the Processes and list of Process using a Particular file.



PCB (Process Control Block)

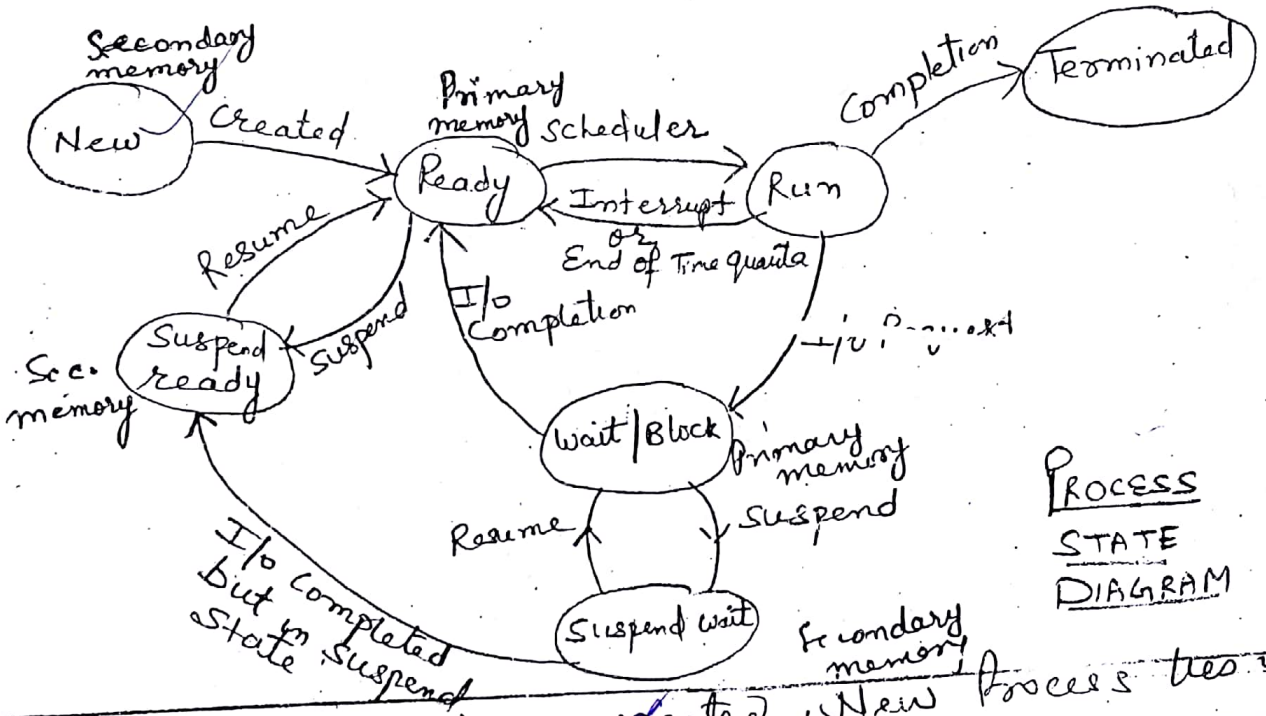
Process Management

(6)

A process can be thought as program in execution. A process will need certain resources such as CPU time, memory, files & I/O devices.

A process can be in many states

PROCESS STATES AND TYPES of SCHEDULERS



PROCESS STATE DIAGRAM

1) New → Process is being created, New process lies in Secondary memory.

2) Ready → Out of all the processes being created in secondary memory, it is the responsibility of Long Term Scheduler to bring process in main memory is in Ready state

③ Run → Out of the Processes in ready State it is the responsibility of Short term Scheduler to select the Process to be allocated CPU or in running State and then dispatcher the Process to CPU.

④ Terminated → If a Process has finished execution it goes to terminated State.

⑤ wait | Block → If a Process is running & requires I/O, then it goes to waiting State and after I/O completion it again goes to ready State

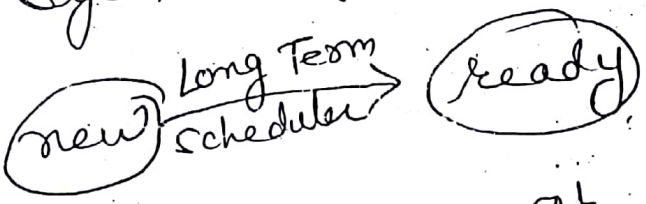
⑥ Suspend wait → If in waiting State I/O resources are not available then Process is moved from primary memory to secondary memory i.e from waiting State to Suspend wait State and after resources become available then again Process can be resumed & brought to main memory.

⑦ Suspend Ready → If ready State is already full & there are some Processes which are important & need to be moved to ready State, then some

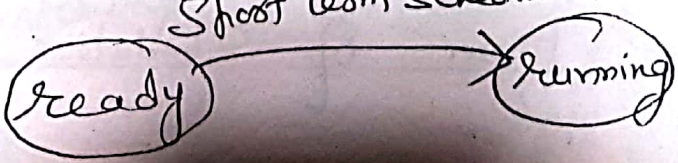
Processes from ready state will go to suspend ready state in secondary memory & can be later brought back.

Types of Scheduler

1) Long Term scheduler → It selects the Processes & brings them to main memory. It controls the degree of multiprogramming i.e no of Processes to be brought to main memory i.e ready state
Long term scheduler should select the Processes that are a mixture of CPU Bound & I/O Bound Instructions to keep the throughput of the system high.



2) Short term scheduler → It selects the Process from ready state and allocates CPU.
Short term scheduler should be such that Context switching time (time to transfer control of CPU from one Process to another) to be minimum

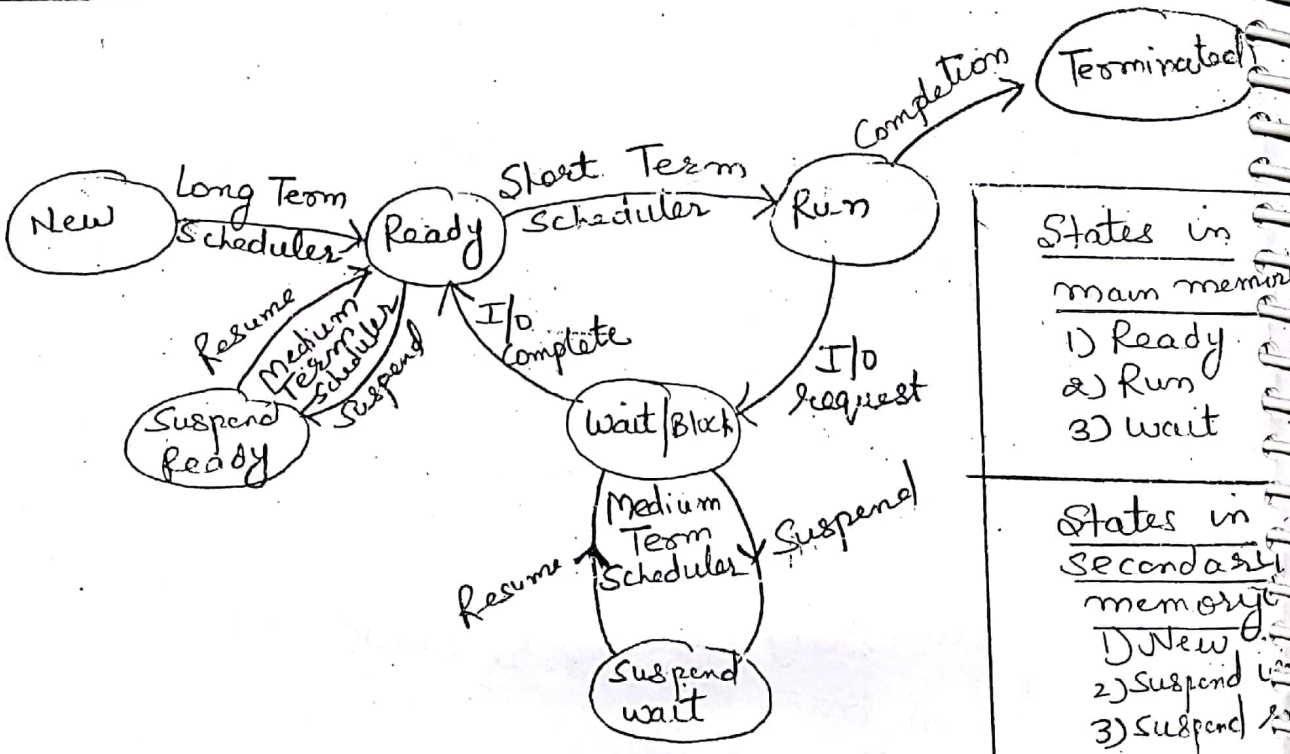


3 Medium Term Scheduler

It is responsible for suspending and resuming processes. It is also called Swapper. It swaps the processes b/w primary and secondary memory

Primary memory ⟷ (Medium Term Scheduler) Secondary memory

Medium term scheduler should be such so that there is a minimum swapping as swapping consumes a lot of time and decreases system throughput.



Different types of Schedulers in Process Sch

CPU Scheduling

Every process needs CPU to perform arithmetic operations. When CPU becomes idle it is the responsibility of CPU scheduler to select another process from ready queue to run next.

Preemptive Scheduling :- CPU scheduling decision takes place under one of four condition

1. When a process switches from running to waiting state for I/O.
2. When a process switches from running to ready state eg in response to an interrupt.
3. When a process switches from waiting state to the ready state ie at completion of I/O.
4. When a process terminates

The system is said to be non preemptive if a process starts running it keeps running until it either voluntarily blocks or until it finishes. Otherwise system is preemptive. ((1) & (4) condition)

Dispatcher

The fn of dispatcher is that it gives control of the CPU to the process selected by scheduler. The function involves

1. context switching

Scheduling Criteria

There are several criteria on which to select the scheduling statergies

1 CPU Utilization ---- → 40% to 90%. -- It should be b/w (Less busy) (more busy)

2 Throughput = No of operations / time.

3 Turnaround time → Time from submission to completion of a Process. Turnaround Time = Burst time + waiting time

4 waiting time → Time spend by Process in ready queue waiting their turn to get CPU.

5 Responsetime → Time from submission to first response by CPU. It should be Low

6 Burst time or Execution time → Execution time of a Program is called its burst time.

7 CPU Utilization → No of times CPU busy / Total time * 100

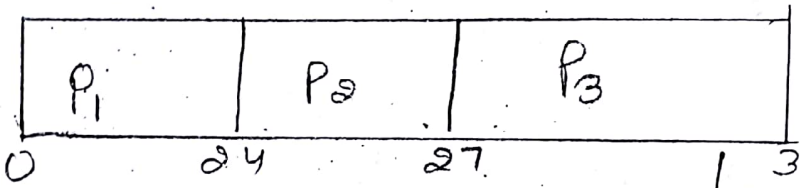
= 8/10 x 100 = 80%.

Scheduling Algorithms ⟶

(1) FCFS (First come First Serve)

It is very simple like customers waiting in a line at bank. It is a non preemptive scheduling technique.

Process	Burst Time
P_1	24
P_2	3
P_3	3



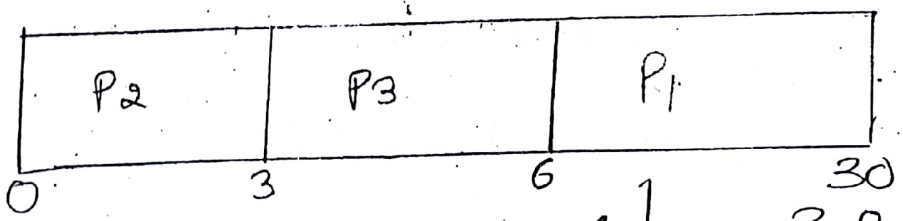
Gantt chart for FCFS

Avg waiting time = $(0 + 24 + 27)/3$
(P1) (P2) (P3)

$= 51/3 = 17$ ms

Avg Turnaround time = $24 + 27 + 30/3 = 81/3 = 27$ ms

Limitation → FCFS can yield very large waiting times



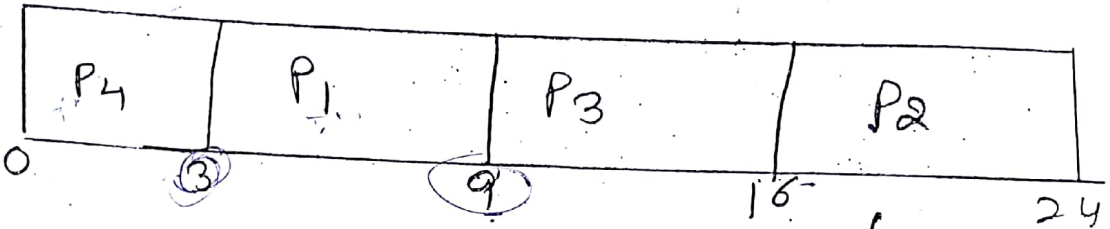
Avg waiting time = $(0 + 3 + 6)/3 = 3.0$ ms
(P2) (P3) (P1)

Avg Turnaround time = $3.0 + 3 + 6/3 = 13$ ms
P1 P2 P3

2) Shortest job first (SJF) →

The idea behind SJF is to pick the quickest little job that needs to be done.

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



Avg waiting time = $0 + 3 + 9 + 16 / 4$
(P_4) (P_1) (P_3) (P_2)

$= 28/4 = 7.0$ ms (as opposed to 10.25 ms for FCFS)

Avg Turnaround time = $19 + 24 + 16 + 3 / 4 = 52/4 = 13$ ms
(P_1 P_2 P_3 P_4)

SJF can be Primitive or Non Primitive

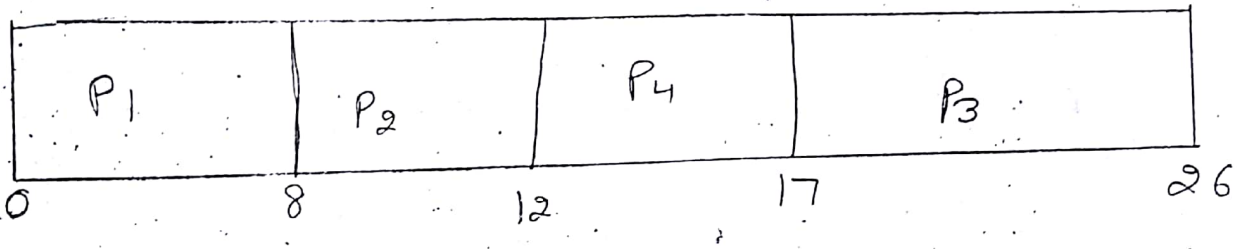
Preemption occurs when a new process (has 5) arrives in ready queue that has burst time shorter than the time remaining in the process whose burst is currently on CPU

Process	Arrival Time (ms)	Burst Time (ms)
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

(10)

1) Calculate Avg waiting time for Preemptive & Non Preemptive SJF.

1) Non Preemptive SJF ⟶



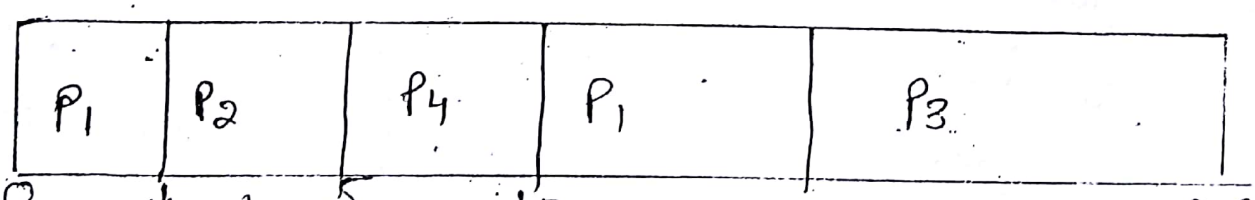
Gantt chart

Avg waiting time = $0 + 7 + 9 + 15 \,/\, 4$
(P_1) (P_2) (P_4) (P_3)

Avg waiting time = $31/4 = 7.75$ ms

Avg Turnaround time ⇒ $\underset{P_1}{8} + \underset{P_2}{11} + \underset{P_3}{24} + \underset{P_4}{14} \,/\, 4 = \frac{57}{4}$
⇒ 14.2 ms

2) Pre emptive SJF ⟶



Avg waiting time = $\dfrac{9}{(P_1)} + \dfrac{0}{(P_2)} + \dfrac{15}{(P_3)} + \dfrac{2}{(P_4)} \,/\, 4 = 26/4 = 6.5$ ms

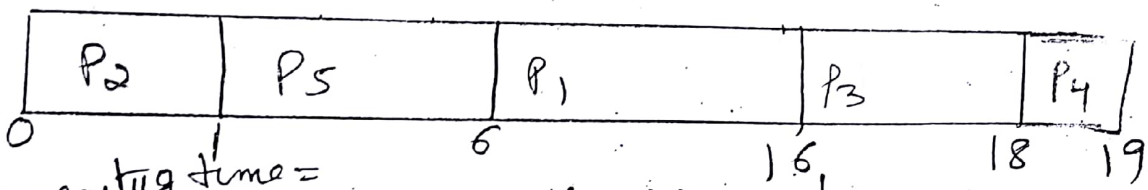
Priority scheduling →

Every job is assigned a priority and the job with highest priority gets scheduled first.

If 2 jobs have same priority then they are processed acc to FCFS.

Example

Process	BT	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



Avg waiting time = (P2) 0 + (P5) 1 + (P1) 6 + (P3) 16 + (P4) 18 / 5 = 41/5 = 8.2 ms

Avg Turnaround time = (P1) 16 + (P2) 1 + (P3) 18 + (P4) 19 + (P5) 6 / 5 = 60/5 = 12 ms

Limitation → Priority scheduling can suffer from indefinite blocking or starvation in which lower priority process has to wait forever by these are some jobs around that have higher priority.

This can be overcomed by aging, in which priority of jobs increase the longer they wait. Under this scheme a low priority job will eventually gets its priority raised high enough till its run.

(11)-a

Round Robin Scheduling ⟶

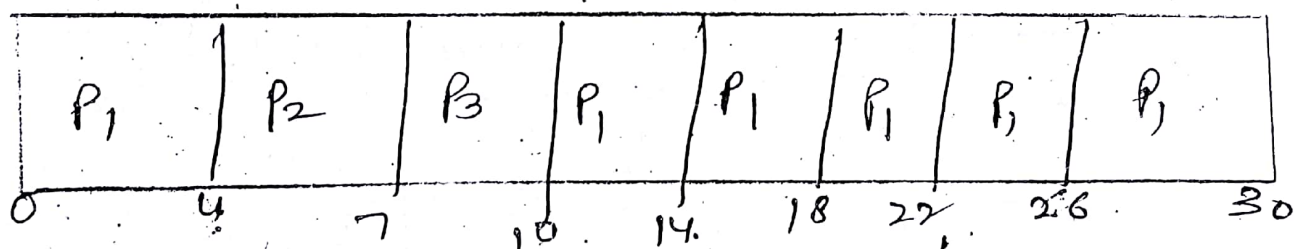
It is same as FCFS, except CPU bursts are assigned with time quanta.

Every process is given a time quanta after which next process will be executed in FCFS manner with same time quanta

Advantage ⟶ Fast response time

Process	Burst time
P1	24
P2	3
P3	3

Time quanta = 4ms



Avg waiting time = (6 + 4 + 7) / 3
(P1) (P2) (P3)

= 17/3 = 5.66ms

Avg Turnaround time = (30 + 7 + 10) / 3 = 47/3 = 15.66 ms
(P1) (P2) (P3)

* Context switch time is very less as compared to time quanta (10 to 100 ms)
(10 microsecond)

Multilevel Queue Scheduling →

When Processes can be readily categorized, then multiple separate queues can be established. Under this algo jobs are assigned queues Permanently. Each queue has its own scheduling algo.

Under this algo jobs can't switch from queue to queue, once they are assigned queue, that their queue until they finish. Each queue has priority over other. Lower queues will run Processes only after all higher queues have finished.

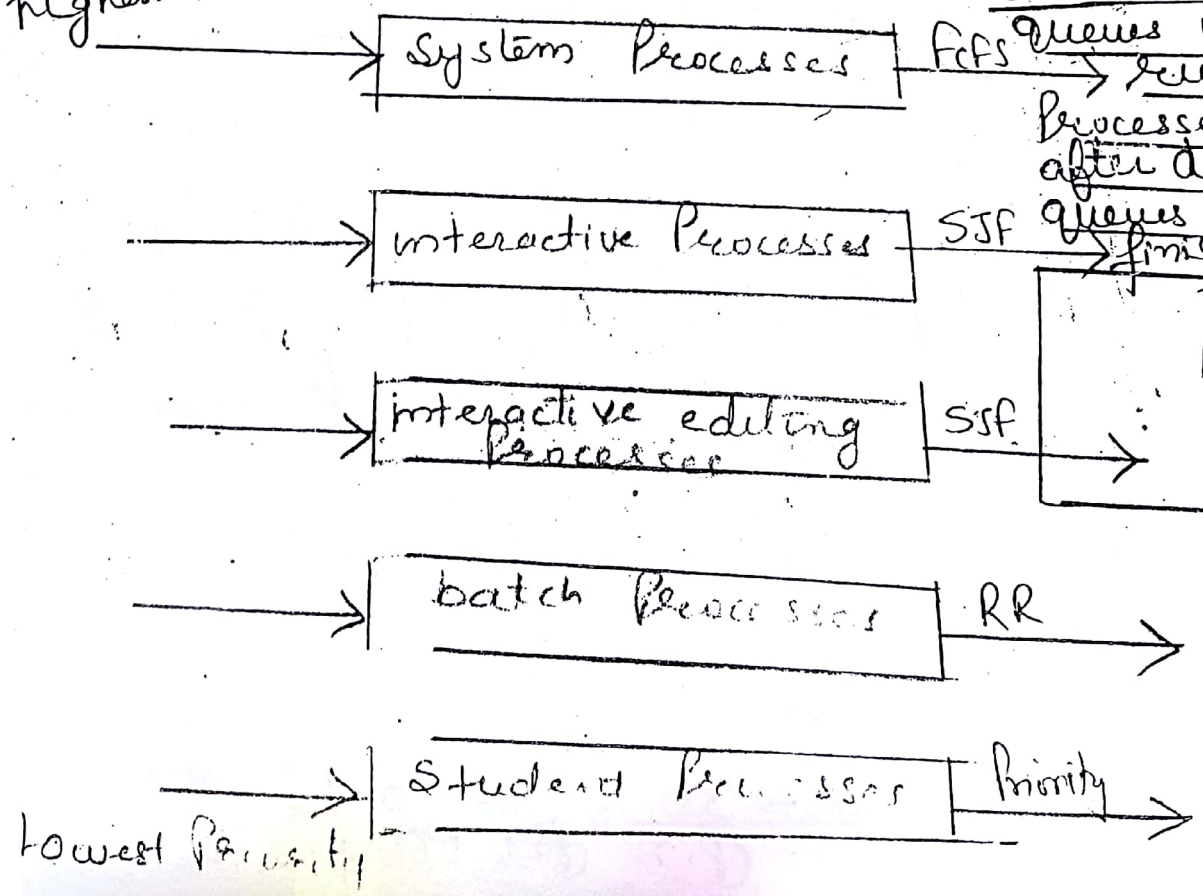


Fig Multilevel Queue scheduling

Multilevel Feedback Queue scheduling →

Multilevel feedback queue allows a process to move between queues. The idea is to separate processes according to characteristics of CPU bursts.
If a process uses too much of CPU burst time it will be moved to a lower priority queue.
In addition if a process waits too long in a lower priority queue, their process may be moved to a higher priority queue. This form of aging prevents starvation

eg Consider a multilevel feedback queue scheduler with 3 queues numbered from 0 to 2. The scheduler first executes process in queue 0. only when the queue 0 is empty, then it will execute process in queue 1, similarly when queue 1 is empty it will execute process in queue 2.

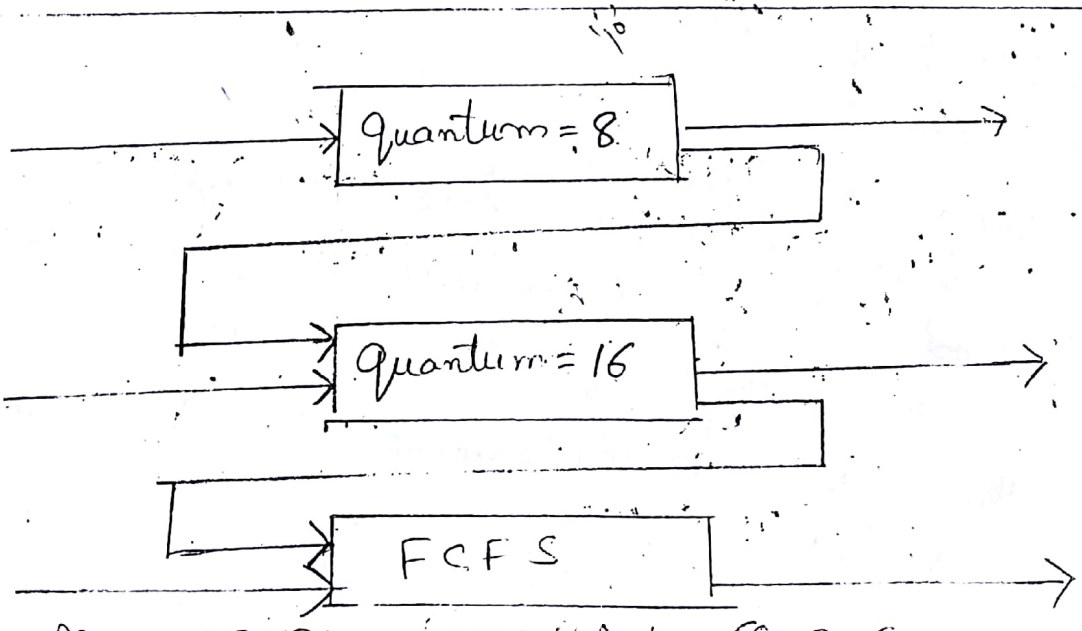


FIG MULTILEVEL FEEDBACK QUEUE

→ Process entering ready queue is put in queue 0. A Process in Queue 0 is given a time quanta of 8 ms. If it doesn't finish within that period it is moved to the tail of queue 1. If Queue 0 is empty than Process at head of Queue 1 is given time quanta of 16 ms. If it doesnot complete within 16ms then it is moved to queue 2. Processes in queue 2 are processed in FCFS basis but when queue 0 and queue 1 are empty.

This algorithm gives highest priority to a process whose burst time is 8 millisecond or less. Processes whose burst time is b/w 8 ms to 24 ms are also executed fast but long processes automatically sink to queue 2 and are processed in FCFS manner.

Multilevel feedback queue is defined by following parameters

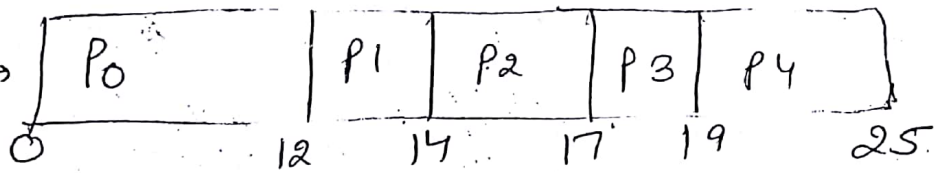
→ Number of queues.
→ Scheduling algorithm for each queue.
→ Method used to determine when to upgrade a process to higher priority queue.
→ Method used to determine when to demote a process to lower priority queue.

Comparitive Studies of CPU scheduling algos

Process ID	BT (ms)	Priority
P0	12	3
P1	2	1
P2	3	3
P3	2	4
P4	6	2

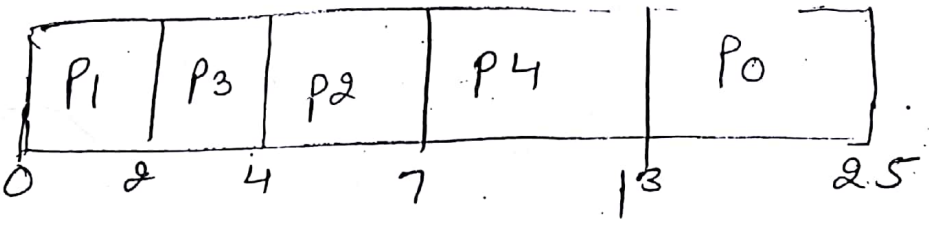
a) FCFS

Gantt Chart →



b) SJF

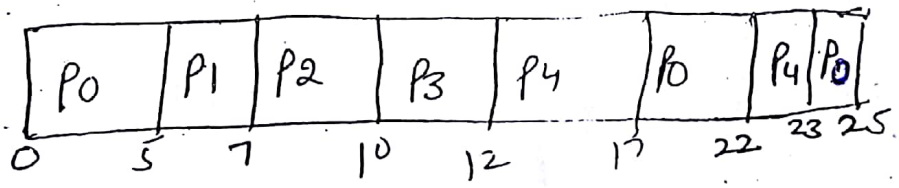
Gantt chart →



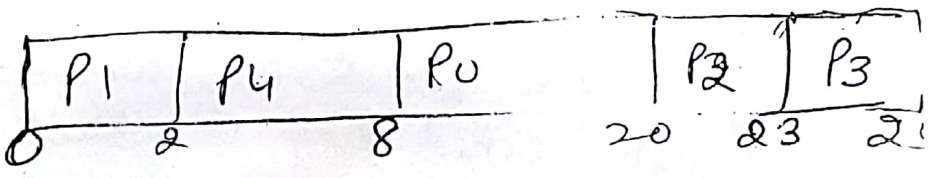
c) Round Robin →

Gantt chart

Time Quanta → 5ms



d) Priority Scheduling Gantt chart



Process ID	Waiting Time (ms) FCFS	SJF	Round Robin	Priority Scheduling
P0	0	13	13	8
P1	12	0	5	0
P2	14	4	7	20
P3	17	2	10	23
P4	19	7	17	2
Avg waiting time	12.4	5.2	10.4	10.6

Process ID	Turnaround time (ms) FCFS	SJF	Round Robin	Priority
P0	12	25	25	20
P1	14	2	7	2
P2	17	7	10	23
P3	19	4	12	25
P4	25	13	23	8
Avg Turnaround time	17.4	10.2	15.4	15.6

From above discussion it is clear SJF & FCFS is best for batch OS whereas Round Robin & Priority scheduling is best for Timesharing system.

Ques:- Consider the following set of Process with CPU burst time in m.s (14)

Process.	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2.

Processes are assumed to arrive in the order.
P_1, P_2, P_3, P_4, P_5 all at a time

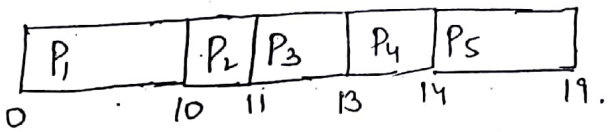
① Draw Ghantt chart that illustrate the execution of a process using following scheduling algorithm
→ FCFS, SJF, non-Preemptive, Priority, Round Robin (TQ=1 ms)

② What is turn around time for each process in part ①

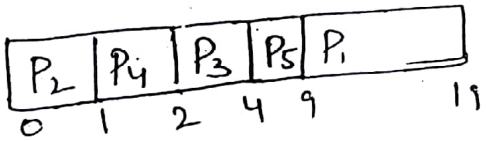
③ What is waiting time for each process in part ①

④ Which scheduling algorithm has the min. avg. waiting time

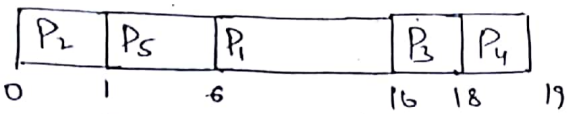
Soln :- ① FCFS



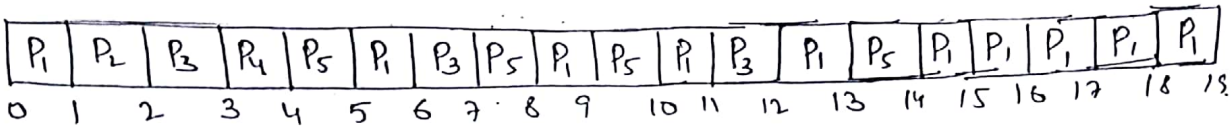
② SJF



③ Non- Preemptive



④ Round Robin



Average Waiting Time

	FCFS	SJF	Priority	Round Robin
P_1	0	9	6	9
P_2	10	0	0	1
P_3	11	2	16	5
P_4	13	1	18	3
P_5	14	4	1	9
	$\frac{48}{5} = 9.6$	$\frac{16}{5} = 3.2$	$\frac{41}{5} = 8.2$	$\frac{22}{5} = 5.4$
	$= 9.6$ ms	$= 3.2$ ms	$= 8.2$ ms	$= 5.4$ ms.

Turn around Time

	FCFS	SJF	Priority	Round Robin
P_1	10	19	16	19
P_2	11	1	1	2
P_3	13	4	18	7
P_4	14	2	19	4
P_5	19	9	6	14
	$= \frac{67}{5} = 13.4$	$\frac{35}{5} = 7$	$= 60/5 = 12$	$= \frac{46}{5} = 9.2$
	$= 13.4$ ms	$= 7$ ms	$= 12$ ms	$= 9.2$ ms

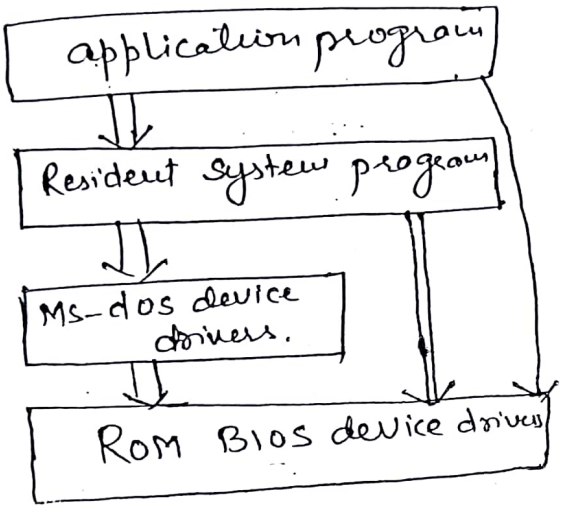
Operating System Structure :- (15)

→ The structure of operating system can be classified into 3 catagories:

1) Simple Structure or Monolithic structure
2) layered Approach
3) Virtual machine.

1) Simple Structure :- (Monotithic)

→ Some operating system do not have well-defined structure.
→ They are small, simple and limited system.
→ MS-Dos is an example of such system.
→ It was written to provide the most functionality in the least space, so it was not divided into modules carefully.
→ In MS-Dos, the interface and level of functionality are not well seprated.
→ o/o application program are able to access the basic I/O routines to write directly to the display and disk drives.



→ Such freedom leaves Ms-Dos vulnerable to malicious program, causing entire system crashes. when user program fail.

→ Another Example of limited structuring is the original UNIX operating system.

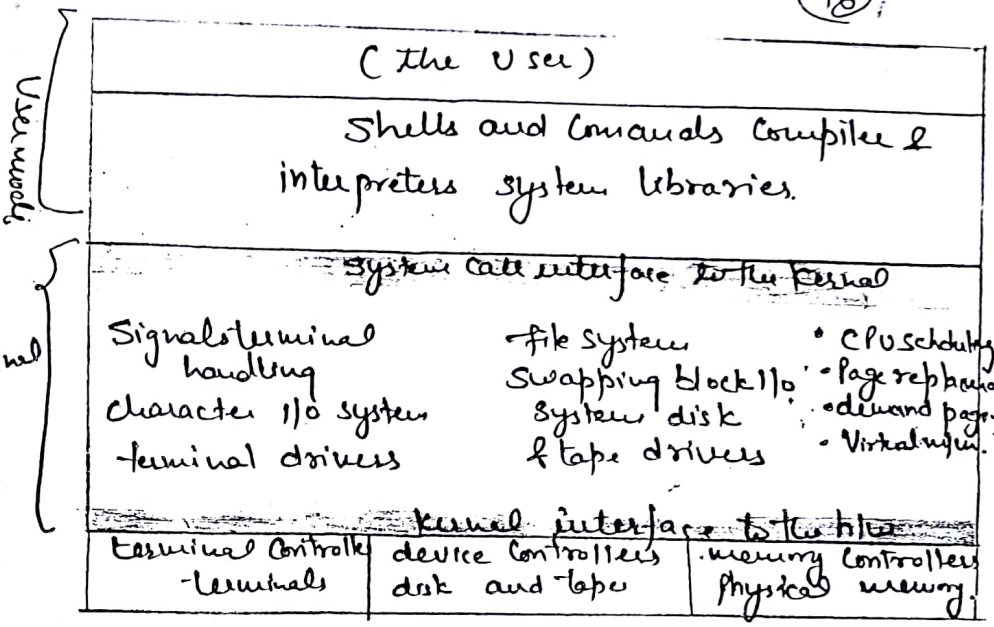
→ UNIX system initally was limited by hardware functionality.

→ It consist of two parts :

① kernal
② System program.

→ The kernal is further seperated into a series of interface & device drivers.

→ The view of UNIX operating system is shown as:



(UNIX system Structure)

→ The kernel provide the file system, CPU scheduling memory management and other operating system function through system calls.

2. Layered Approach :-

→ In this the Operating system can be broken into pieces that are smaller and more efficient than MS-DOS or UNIX system.

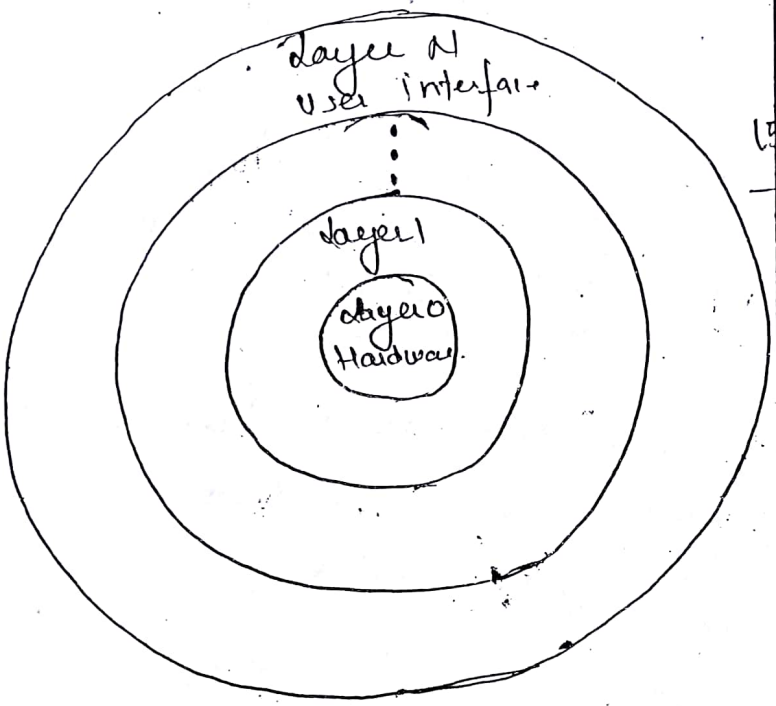
→ The operating system then retain much greater control over the computer and over the application that make use of that computer.

→ In layered approach, the operating system broken

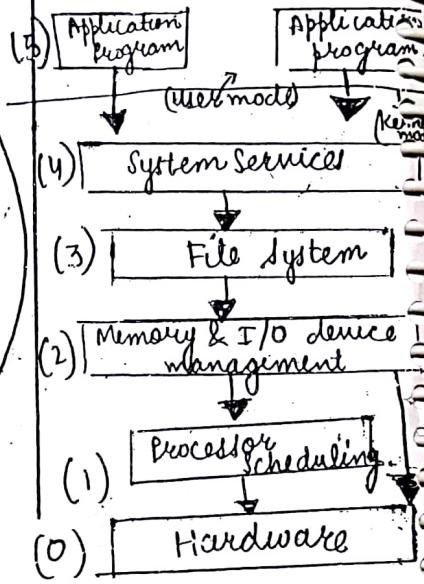
upto number of layers or levels.

→ The bottom layer is the hardware & the highest layer (layer N) is the user interface.

→ The layer structure is shown as:



Layered Architecture



→ In layered O.S, certain layers consist of data structure and a set of routines that can be invoked by higher-level layers.

→ The higher-level layer invoke operations on lower-level layer.

→ The main advantage of layered approach is simplicity of construction and debugging.

→ If an error is found during debugging of a particular layer, then the error must be on that layer because the layer below it are already debugged.

→ ∴ the design and implementation of the system is simplified.

→ Each layer is implemented with only those operations provided by lower level layer.

→ The major difficulty with the layered approach involves appropriatly defining the various layers. Because the layer can use only lower level layers.

→ The final problem with layered approach is that they tend to be less efficient than other types.

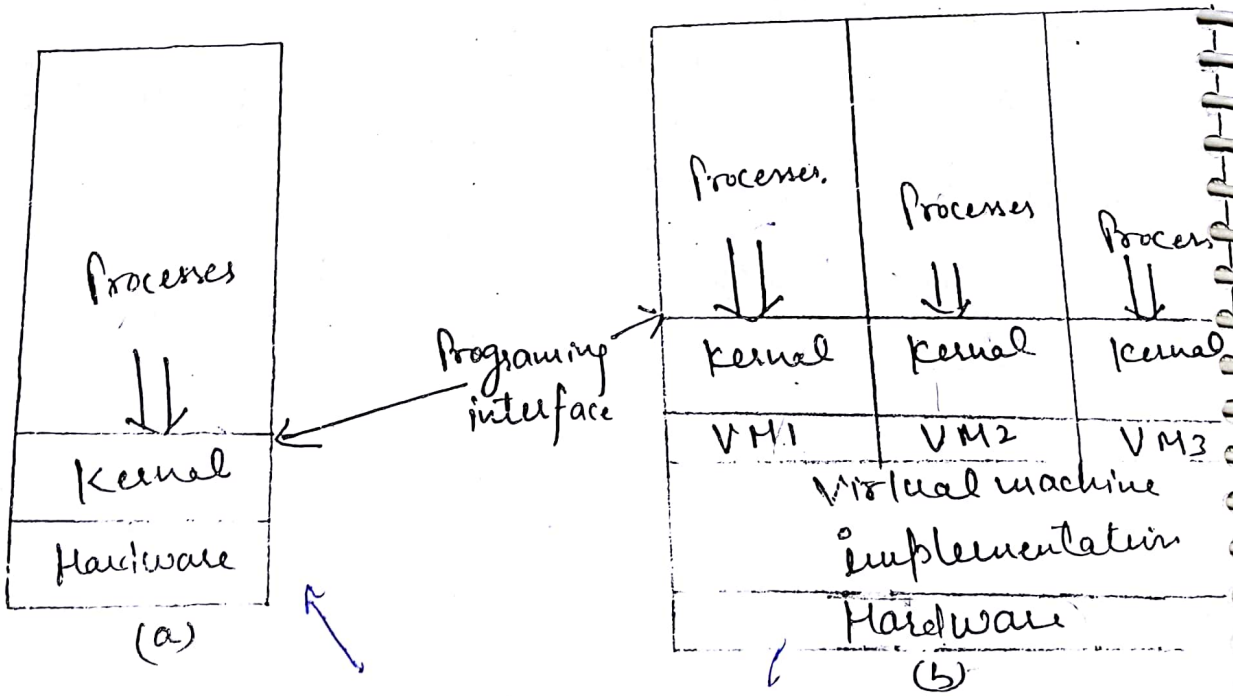
→ Each layers adds overhead to the system calls.

3 Virtual Machines :-

→ Virtual m/c abstract the hardware of a single computer into several different execution environment.

→ It creates the illusion that each separate execution environment is running its own private computer.

→ By using CPU scheduling and virtual m/m technique, an operating system can create the illusion that a process has its own processor with its own virtual memory.



a) Non-Virtual M/C b) Virtual m/c.

(18)

→ The main reason for creating a Virtual m/c is to share the same hardware by running several different execution environment concurrently.

→ The major difficulty with the Virtual-m/c approach involves disk system.

→ Suppose the physical m/c has three disk drivers but want to support seven virtual m/c. Clearly it cannot allocate a disk drive to each Virtual m/c, because the Virtual m/c Software itself needs disk space to provide Virtual memory and spooling.

→ The solution is to provide Virtual disk or minidisk

→ This disk is identical in all respects except Size.

→ The sum of size of all minidisk must be smaller than the size of the physical disk space available.

→ The virtual m/c Software can run in kernal mode since it is an operating system.

→ The virtual m/c itself can execute in only user mode.

→ As like physical m/c has two modes as user mode & kernal mode, similarly the Virtual m/c, also has two modes as Virtual user mode & Virtual kernal mode both of which run in a physical user mode.

Advantages :-

1) Complete protection of the various system resources
2) Each VM is completly isolated from all other Virtual machine, so there are no security problems
3) It is possible to define a n/w of VM's, each of which can send information over the Virtual Communication N/w.
4) Used in research and Development

Disadvantages :-

1) This architecture is difficult to implement as it require lot of effort to provide an exact duplicate of same m/c.

Example of Virtual m/c :-

1) VMware 2) JVM (Java Virtual m/c).

Diff b/w Program and Process

Program	Process
A sequence of instructions executed is called Program.	A Program in execution is called Process.
It is a passive entity	It is a active entity
A Program has only one instance.	A Process can have multiple instances eg we can open several notepad simultaneously
It is stored in secondary memory.	It is stored in primary memory/main mem
It depends on the logic of the problem.	It depends on the Program.
It doesnot use CPU registers.	It uses CPU registers. Every Process has a ID, State, list of open files and other parameters in Process control Block.

Programs are read into primary main memory and are executed by kernel

Differentiate :-

Long Term Scheduler	Medium Term Scheduler	Short Term Scheduler
- It is a Job scheduler.	It is process swapping scheduler.	CPU scheduler
2 Speed is less than short term scheduler.	Speed is in b/w LTS & STS.	Speed is fastest.
3 It controls the degree of multiprogramming	It reduces the degree of multiprogramming.	It has less control over degree of multiprogramming
4 It is almost absent or minimal in Time sharing system.	It is part of Time sharing system	It is also minimal in time sharing system
5 It selects the Processes from pool and loads them in memory for execution	It can introduce Process into memory and execution can be continued.	It selects those Processes which are ready to execute.

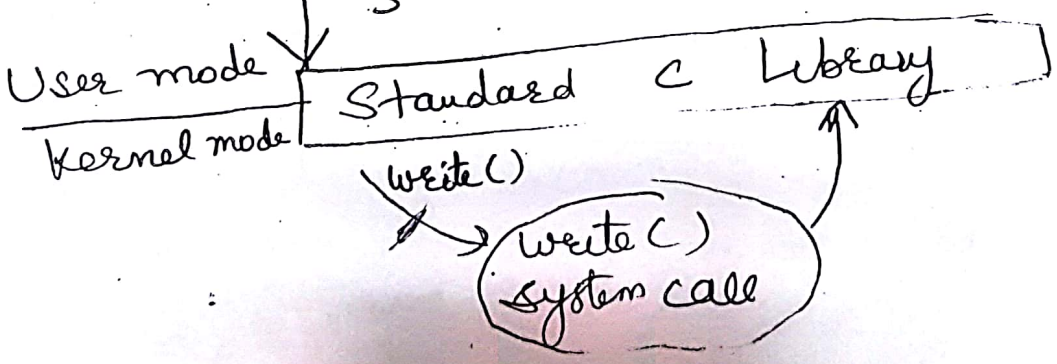
System Call

User Programs communicate with operating system and request services from it by making system calls. Each system call has library procedure that user program calls. The purpose of system calls is to request the operating system to perform some task. So system call provides interface b/w process and operating system. These calls are provided as routines in C/C++

Standard C Library Example of system call

C Program invoking printf() library call, which calls write() system call

```
#include <stdio.h>
int main()
{
    printf("Greetings");
    return 0;
}
```



Types of System Calls

	Windows	Unix
→ Process Control	CreateProcess() ExitProcess WaitForSingleObject	fork() exit() wait()
→ File manipulation	CreateFile() ✓ ReadFile() WriteFile() CloseHandle()	open() read() write() close()
→ Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe()	pipe()
Protection	SetFileSecurity() InitializeSecurityDescriptor()	chmod() umask()

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. CPU and device controllers can execute concurrently

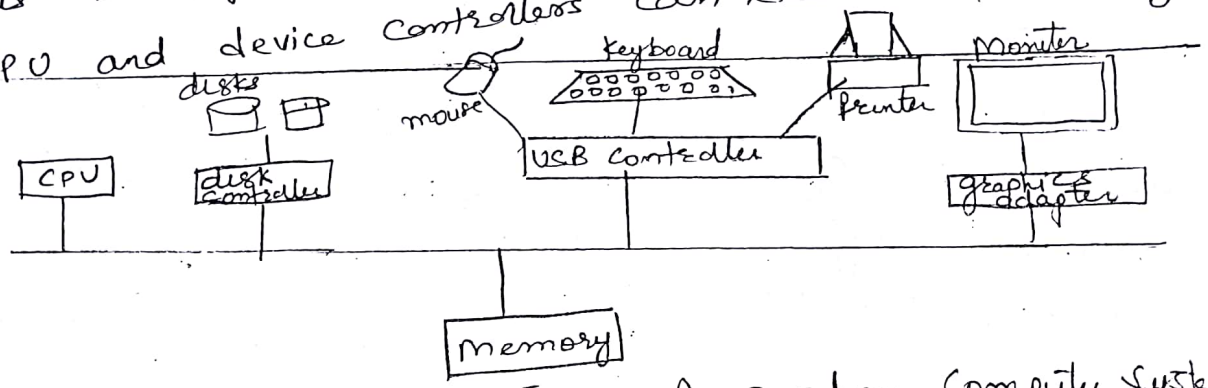


FIG:- A modern Computer System

For a system to start running - for instance, when it is powered up or rebooted - it needs to have the initial program or bootstrap program. Typically it is stored in read only memory (ROM). It initializes all aspects of system from CPU registers to device controllers to memory contents. The bootstrap (loader) program must know how to load the operating system and start executing the program. To accomplish this bootstrap must (know) 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.

Occurence of some event is usually triggered by an Interrupt either from Hardware or Software. Hardware may send an Interrupt by sending a signal to CPU by System bus, Software may trigger an interrupt by executing a special operation called System call.

When CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains starting address where service routine for interrupt is located. The Interrupt Service routine executes and on completing execution, CPU resumes the Interrupted process.

Interrupt routine is called indirectly through the table with no Intermediate routine needed. Generally table of pointers is stored in Low memory (first 100 or so locations). These locations

Contain the address of interrupt service routines
The interrupt architecture must also save the address of interrupted instruction. Now a days return address is stored in stack and after completion of Interrupt service routine the return address is reinstalled into the Program counter from stack

Storage structure and Storage Hierarchy

There are different types of memory hierarchies. The main need of the memory hierarchy is to achieve max. possible Speedup with min cost or to minimize the average access time.

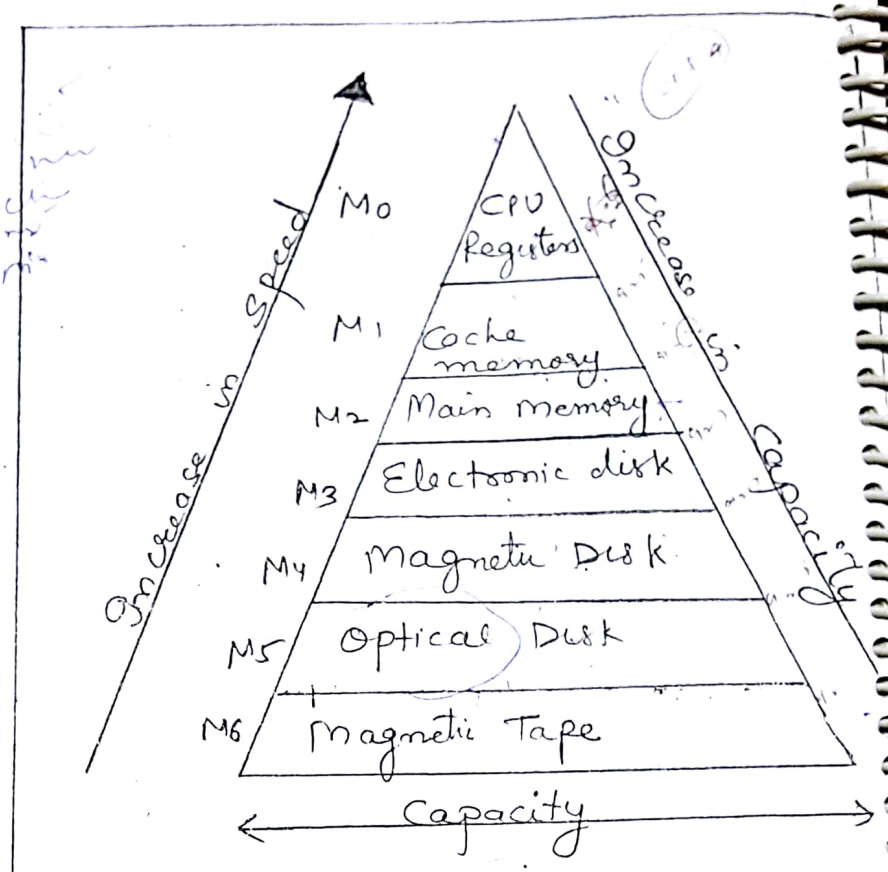


DIAGRAM: STORAGE HIERARCHY

At the Lowest level is Magnetic Tape which has the highest capacity. It can store data upto max. 185 TB of information. It is a Secondary Storage device. Its Speed is very Less. It is a magnetic device

After Magnetic Tape there is optical Disk which is based on Light. Examples of optical Disk include CD ROM, Blue ray disc. It can have a maximum Storage capacity of 1TB. In CDROM drive a small beam of Laser Light is responsible for reading the information.

After optical Disk, there is magnetic disk. Example of magnetic disk is Hard disk. It can store data upto 1TB. Data is stored in Hard disk in form of Tracks and Sector. Firstly the read/write head is positioned to a particular track. And after that to a particular sector. The time required to position R/W head to particular track is called Seek time & time required to position R/W head to particular Sector is rotational delay

After magnetic disk is Electronic Disk which include flash memory like memory card,

Pen drives it can store data upto 32 GB.
After that there is main memory which can have storage capacity upto 16 GB.
It's speed is better than Electronic disk.
After main memory there is cache memory
Size of cache memory is 1/10 of the main memory and has speed faster than main memory. Highest in the hierarchy is CPU registers, which are few & highest in speed.

① Principle of Inclusion

$M_0 \subseteq M_1 \subseteq M_2 \subseteq M_3 \subseteq M_4 \subseteq M_5 \subseteq M_6$

② Principle of coherence

Value of a variable must be same at all the memory levels