

Q.1. What is Operating System? Explain architecture of an OS.

Ans - An operating system is a software that manages the computer h/ws. The h/w must provide the appropriate mechanisms to ensure the correct operation of computer system and to prevent user from interfering with the proper operation of the system.

Architecture of OS:- The core software components of OS are collectively known as kernel. The kernel has unrestricted access to all of the resources on system.

Since there are many programs & resources are limited, the kernel also decides when and how long a program should run. This is called a scheduling.

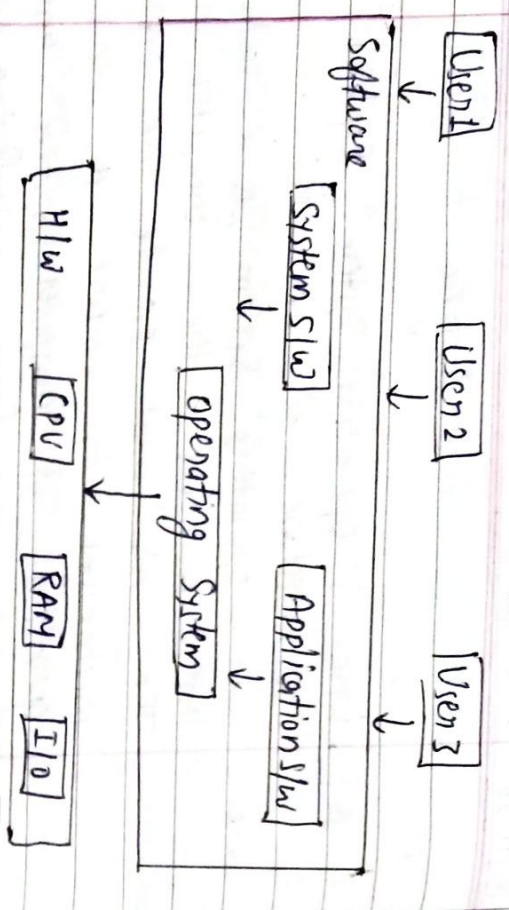


fig- Architecture of OS.

Q.2 explain the following:-

(i) Process & Program:-

Process contains a set of instructions designed to complete a specific task.

Program can be described as an instance of program running on a computer as entity that can be assigned to and executed on a processor.

(ii) Thread:- A thread is a path of execution within a process. Thread is a single sequence stream with

in a process. Threads have some properties as of process so they are called as light weighted process. They are executed one after another. but gives the illusion as if they are executing in parallel.

(iii) System Call:- A system call is a programmatic way in which computer program requests a service from kernel of OS it is executed on. It is way for program to interact with OS.

Q.3

1. Explain various services of an OS
OS provides services to both user & to the programs.
2. It provides programs an environment to execute.

2. It provide users the services to execute the programs in convenient manner.

Four common services provided by an OS

1. Program Execution

2. I/O operations

3. File System Manipulation

4. Communication
5. Error detection
6. Resource Allocation
7. Protection.

1. Program execution:- OS handles many kinds of activities from programs to system programs like printer etc.

2. I/O operation:- It means read or write operation with any file on any specific I/O device.

3. File System Manipulation:- Program needs to read a file or write a file OS gives the permission to program for operation on file.

4. Communication:- Multiple processes commⁿ with one another through common lines in network.

5. Error Handling:- Error may occur anywhere for possible error and takes an appropriate action to ensure correct

8. consistent computing.

6. Resource Management:- The OS manages all kinds of resources using schedulers. CPU scheduling also used for better utilization of CPU.

7. Protection:- OS ensures that all access to system resource is controlled. The OS ensures that external I/O devices are protected from invalid access attempts.

Q.4
List out various process states & briefly explain with suitable diagram.

Ans:- A process is program in execution. A program become a process when an executable file is loaded into memory.

State of process:-

(i) New:- Process is being created.

(ii) Ready:- The process is waiting to be assigned to a processor. Process that are ready to get executed by CPU are maintained in queue.

3. Running:- Instruction are being executed.

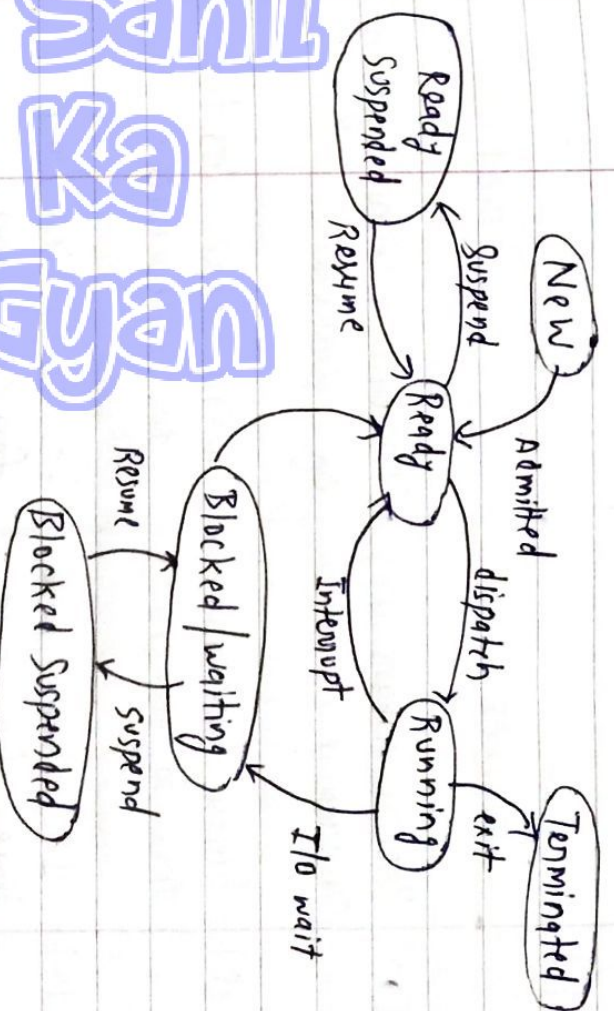
4. Waiting:- The process is waiting for some event to occur.

5. Terminated:- Process has finished execution & process control block is deleted.

6. Ready Suspended:- The process in ready suspended state is in secondary memory. They were initially in ready state in main memory but lack of memory forced them to be suspended and get placed in the secondary memory.

7. Blocked Suspended:- This is similar to the ready suspended they are initially in block state in main memory waiting for some event. A process may go from blocked suspended to ready suspended if its work is done.

Process state



Q.5
(i)

Differentiate b/w

User Thread

Kernel Thread

(a) User Thread are faster to create and manage.	(a) KLT are slower to create & manage.
(b) context switch time is less.	(b) context switch time is more.
(c) context switch requires no h/w support.	(c) H/w support is needed.
(d) OS doesn't recognize user level threads.	(d) KT are recognized by OS.
(e) If one ULT operation perform blocking operation then entire process will be blocked.	(e) If one KT perform blocking operation then another thread can continue execution.

Eg - Java thread, POSIX

Eg - Window solaris

(ii)

Processes

Thread

- | | |
|---|--|
| (a) Process means any program is in execution. | Thread means segment of process. |
| (b) Process takes more time to terminate. | Thread takes less time to terminate. |
| (c) It takes more time for creation. | It takes less time for creation. |
| (d) Process is less efficient in term of comm ⁿ process. | Thread is more efficient in term of comm ⁿ process. |
| (e) Consumes more resources. | Consumes less resources. |
| (f) Process is isolated. | Thread shares memory. |
| (g) Process is called heavy weight process. | Thread is called light weighted process. |

Q-1 What is critical section problem?
How semaphores are used for solving critical section problem?

Ans - The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a CS i.e. only one process can execute in its CS at a time. All other process have to wait to execute in their critical section.

A diagram that demonstrates critical section is as follows: -

```
do { Entry Section
    critical section
    Exit Section
    Remainder section
} while (TRUE);
```

In given diagram, the entry section handles the entry into critical section. It acquires the resources needed for execution by process.

The exit section handles exit from the critical section. It releases resource and

also informs the other process that critical section is free.

Solution to the Critical Section Problem:-

The CS problem needs a solution to synchronize the different processes. The solution to CS problem must satisfy the following conditions:-

(i) Mutual Exclusion:- Mutual Exclusion implies that only one process can be inside the CS at any time. If any other process requires CS, they must wait until it is free.

(ii) Progress:- Progress means that if a process is not using CS, then it should not stop any other process from accessing it. It means any process can enter in CS if it's free.

(iii) Bounded Waiting:- Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the CS.

Semaphores are integer variables that are used to solve the CS problem by using two atomic operation, wait & signal that are used for process synchronization. The definition of wait & signal are as follows:-

Wait:- The wait operation decrements the value of its argument if it's positive. If s is negative or zero, then no operation is performed.

```
wait(s)
{
    while (s <= 0);
    s--;
}
```

Signal:- The signal operation increments the value of its argument.

```
s.
signal(s)
{
    s++;
}
```

Q.2
What is scheduling? Difference b/w short term & long term scheduling
Process scheduling is the activity of process manager that handles the removal of running process from CPU.

and the selection of another process on the basis of particular strategy. Process Scheduling is an essential part of multiprogramming OS. Such OS allow more than one process to be loaded into executable memory at a time & loaded process shares CPU using time multiplexing.

CPU Scheduling is a process which allows one process to use CPU while execution of another process is on hold due to unavailability of any resource like I/O etc. Thereby making full use of CPU. The aim of CPU scheduling is to make system efficient, fast & fair.

Long term Scheduling	Short term
(i) It takes process from job pool.	It takes process from ready queue.
(ii) It is also known as Job scheduler.	It is also known as CPU scheduler.
(iii) The programs are setup in queue & as per the	No such job exists

requirement the best one job is selected.

- | | | |
|------|--|---|
| (iv) | It requires programs which are selected to system for processing | It ensures which program is suitable or important for processing. |
| (v) | Speed is less than S.T.S. | Speed is very fast as compared to L.T.S. |
| (vi) | It regulates more DOM (degree of multi programming.) | It regulates the less DOM. |

Q.3 Describe basic criteria to select a better CPU scheduling Algorithm.

Ans - Different CPU scheduling algorithms have different properties and the choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algo.

The criteria include the following:

1. CPU utilization: - The main objective of any CPU scheduling algo. is to keep the CPU as busy as possible. Theoretically, CPU utilisation can range from 0 to 100 but in a real time system, it varies from 40

to 90%. depending on load upon system.

2. Through put:-

A measure of work done by CPU is no. of processes being executed and completed per unit time.

This is called throughput.

The throughput may vary depending upon length or duration of process.

3. Turnaround Time:- An important

criterion is how long it takes to execute the process. The time elapsed from the time of submission of a process to time of completion is known as Turnaround time.

TT is sum of time spent waiting to get into memory, waiting in queue, executing in CPU & waiting for I/O.

4. Waiting Time:- A scheduling algo.

does not affect the time required to compute the process once it starts execution. It only

affects the waiting time of a process i.e. time spent by process waiting in ready queue.

5. Response Time: - In a interactive system, \downarrow TT is not best criteria.

A process may produce some output fairly early and continue computing new results are being output to the user. Thus another criteria is the time taken from submission of process of request until the first response is produced. This measure is called response time.

Basic of OS

POPU

Page No. :
Date : / /

Operating System is a ~~computer~~ system sw that manages all ~~computer~~ computer h/w, sw resources and provides a common service for communication. It hides the h/w complexity.

→ It runs in kernel mode.

OS as an Extended Machine \Rightarrow OS provide yet another layer of

abstraction for using disks. The abstraction is the key ~~to~~ ~~all~~ to managing all this complexity.

→ OS turn ugly h/w into beautify abstraction

→ OS's real customers are application programs.

OS as a Resource Manager \Rightarrow Bottom-up view holds that OS is

there to manage all pieces of complex system.

In this view, job of OS is to provide for an orderly & controlled allocation of processors, memories & I/O devices among the various programs wanting them.

RM includes multiplexing resources in time and in space.

History of OS :- Ist Generation - Vacuum Tubes

IInd Gen. - Transistors & Batch System

IIIrd Gen. - ICs & multiprogramming

IVth Gen. - Personal Computers

Vth Gen. - Mobile Computers

Types of OS :- (i) Mainframe (ii) Server OS (iii) Multiprocessor

(iv) Personal Computer OS (v) Handheld Computer OS

(vi) Embedded OS (vii) Sensor-node OS (viii) Real time OS

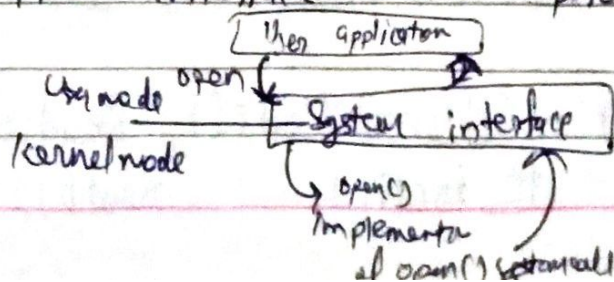
(ix) Smart card OS

Virtual Machine:- A virtual machine is an emulation of particular computer system. So, a VM is an OS on an application environment that is installed on emulated h/w instead of being physically installed on dedicated h/w. The end user has the same experience on a VM as they would have on dedicated h/w.

BIOS:- [Basic Input Output System] The BIOS is the built in s/w that determines what a computer can do without accessing programs from a disk. On personal computers the BIOS contains all code required to control the keyboard, display screen, disk drives and serial communications.

System Calls:- System calls are only entry points into kernel system.

A s/c how a program request a service from an OS kernel. This may include h/w related services, creating & executing new processes & commⁿ with integral kernel services. System calls provides an essential interface b/w process & an OS.



→ Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier.

→ Context Switching Steps :-

- (i) Save context of process that is currently running on CPU. Update the process control block & other important fields.
- (ii) Move PCB of above process into relevant queue such as ready queue, I/O queue etc.
- (iii) Select a new process for execution
- (iv) Update PCB of selected process. This includes updating the process state to running.
- (v) Update memory management data structure as required.
- (vi) Restore context of process that was previously running. This is done by loading previous value of process-PCB and registers.

Threads :- Thread is a single sequence stream within a process. It is a path of execution within a process. A process can contain multiple threads.

Type of Thread :-

- (i) User level Thread (ULT) :- It is not created using system calls. Thread switch doesn't need to call OS and to cause interrupt to kernel. Kernel doesn't know about the ULT and manages them as if they were single threaded process.
 - (ii) Kernel level Thread (KLT) :- Kernel knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table that keeps track of all threads in system. But it is slow.
- Each ULT has process table that keeps track of threads using thread table.
- Each KLT has Thread table (TCB) as well as process table (PCB).

Multitasking programming is of two types :-

(i) Process Based Multitasking :- 2 or more process and programs can be run concurrently. It has slower data state multi-tasking.

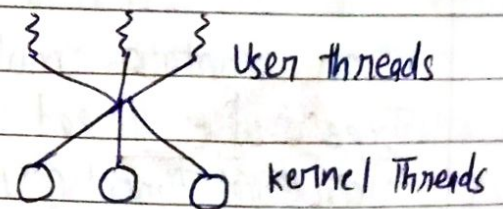
Eg- listen to music and browse internet at the same time.

(ii) Thread Based Multitasking :- 2 or more threads can be run concurrently. Threads share same address space.

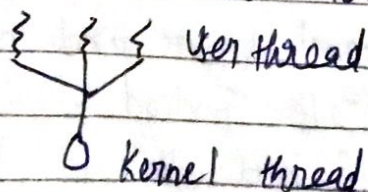
Eg- Using a browser we can navigate through webpage at same time download a file.

Multi Threading Models in Process Management \Rightarrow

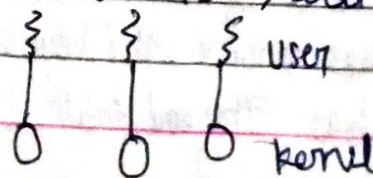
(i) Many to Many Model :- If a user thread is blocked we can schedule others user thread to other kernel thread.



(ii) Many to one model :- Only one user thread can access kernel at a time, so multiple threads aren't able access multiprocessor at the same time.



(iii) One to One model :- One to One relationship



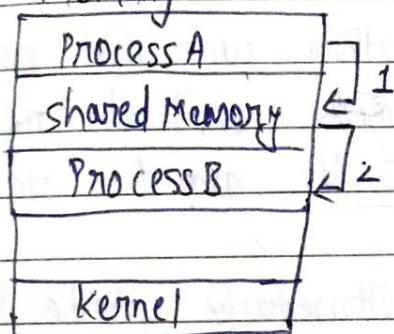
Benefit of Multithreading:- Responsiveness, Resource sharing, Economy, Scalability

InterProcess Communication (IPC):- It is a mechanism that involves commⁿ of one process with another process. This usually occurs only in one system.

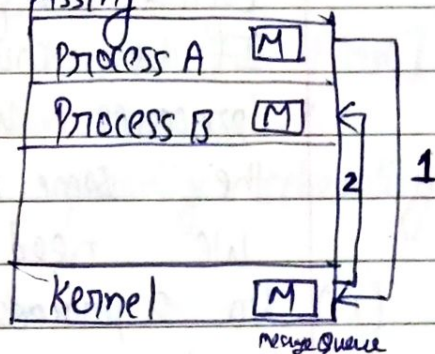
It is used for exchanging data b/w multiple threads in one or more processes or programs.

Process can commⁿ with each other through both:

(i) Shared Memory



(ii) Message Passing

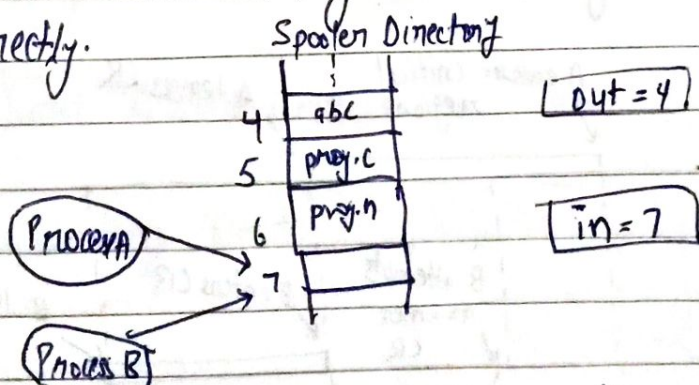


Process in OS are of 2 types:-

(i) Independent process

(ii) Co-operation process

Race Condition:- It occurs when a c/w program depends on the timing of one or more processes to function correctly.



Where two or more processes are reading or writing some shared data & final result depends on who runs precisely when, are called Race conditions.

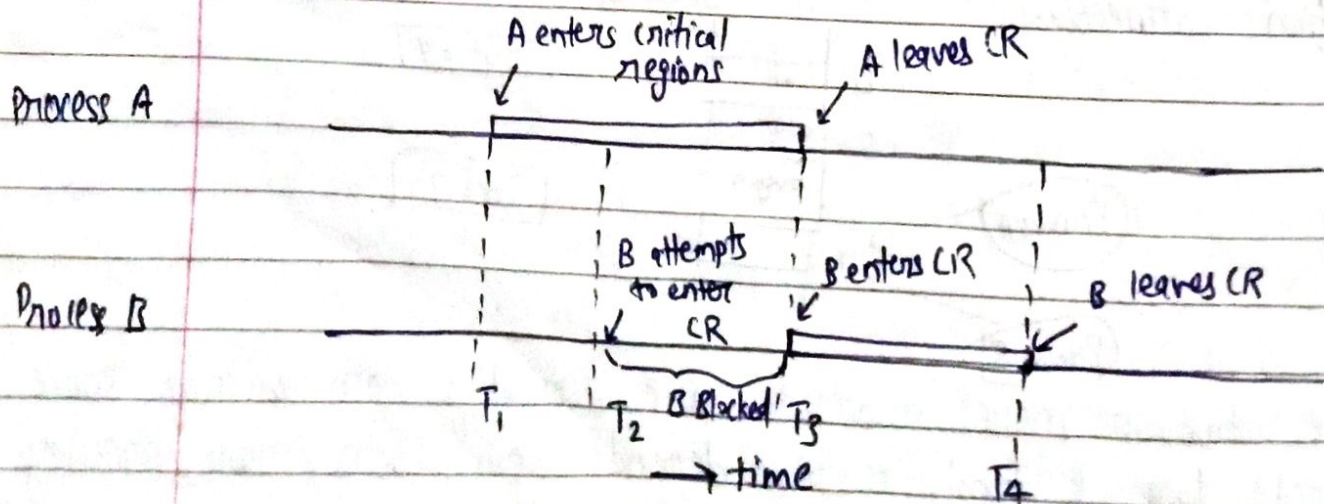
Critical Regions :- How do we avoid race conditions?

The key to preventing trouble here and in many other situations involving shared memory, shared files & shared everything else is to find some way to prohibit more than one process from reading & writing the shared data at the same time. In other words ^{we need} Mutual Exclusion. Sometimes a process has to access shared memory that can lead to races. That part of program where shared memory is accessed is called the critical region.

If we could arrange matters such that no two processes were ever in their critical regions at the same time, we could avoid races.

We need 4 conditions:-

- (i) No 2 processes may be simultaneously inside their critical regions.
- (ii) No assumptions may be made about speed of CPUs.
- (iii) No process running outside its critical region may block any process.
- (iv) No process should have to wait forever to enter its critical region.



Eg - Mutual Exclusion using critical regions

Process Scheduling Algorithms

1. Pre-emptive algorithms

2. Non preemptive algorithms

① SRTF

[Shortest Remaining Time first]

② Preemptive priority

③ Round Robin

① FCFS

[First Come First Serve]

② SJF

[Shortest Job first]

③ Non preemptive priority scheduling

"Gantt Chart" ^{representation} ~~representation~~ of jobs

AT → Arrival Time

BT → Burst Time

CT → Computation Time

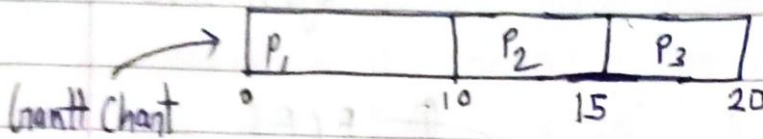
TAT → Turn Around Time

WT → Waiting Time

→ FCFS = First Come, First Served

non-preemptive

Processes	Burst Time
P ₁	10
P ₂	5
P ₃	5



Waiting Time of

P₁ = 0ms

P₂ = 10ms

P₃ = 15ms

Average waiting time = $\frac{0+10+15}{3} = \frac{25}{3} = 8.33\text{ms}$

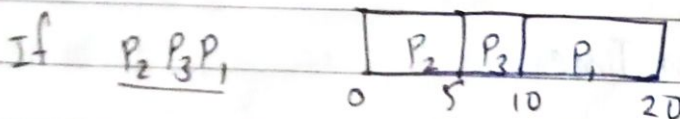
Turnaround Time = Waiting Time + Burst Time

TT of P₁ = 0 + 10 = 10

TT of P₂ = 10 + 5 = 15ms

TT of P₃ = 15 + 5 = 20ms

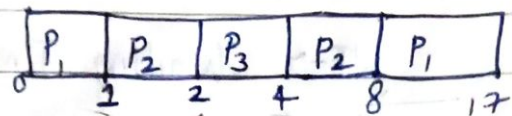
Average turnaround Time = $\frac{10+15+20}{3} = \frac{45}{3} = 15\text{ms}$



→ SJF [Shortest Job First] →

preemptive :-

Process	Burst	Arrival
P ₁	10	0
P ₂	5	1
P ₃	2	2



↓ A.T.

WT of P₁ = 0 + (8-1) - 0 = 7ms

WT of P₂ = 1 + (4-2) - 1 = 2ms

WT of P₃ = 2 - 2 = 0ms

WT = CT - AT - BT

= 11 - 0 - 10 = 1

WT = 8 - 1 - 5 = 2

WT = 4 - 2 - 2 = 0

$$TT \text{ of } P_1 = 7 + 10 = 17 \text{ ms}$$

$$TT \text{ of } P_2 = 2 + 5 = 7 \text{ ms}$$

$$TT \text{ of } P_3 = 0 + 2 = 2 \text{ ms}$$

$$\text{Average of } TT = \frac{17 + 7 + 2}{3} = \frac{26}{3} = 8.66 \text{ ms}$$

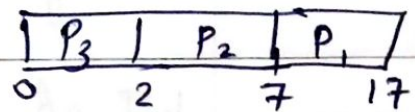
Non-preemptive type:-

Process | CPU-Burst Time

P_1 10

P_2 5

P_3 2



$$WT \text{ of } P_1 = 7 \text{ ms}$$

$$WT \text{ of } P_2 = 2 \text{ ms}$$

$$WT \text{ of } P_3 = 0 \text{ ms}$$

$$A.V. \text{ of } WT = \frac{7 + 2 + 0}{3}$$

$$= 3 \text{ ms}$$

$$TT \text{ of } P_1 = 7 + 10 = 17 \text{ ms}$$

$$TT \text{ of } P_2 = 2 + 5 = 7 \text{ ms}$$

$$TT \text{ of } P_3 = 0 + 2 = 2 \text{ ms}$$

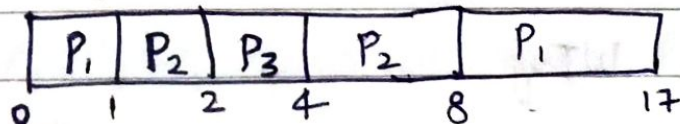
$$\text{Average of } TT = \frac{17 + 7 + 2}{3} = 26 \text{ ms}$$

Priority Scheduling :-

Process	BT	Priority	AT
P_1	10	3	0
P_2	5	2	1
P_3	2	1	2

Preemptive

Indefinite Blocking/
Starvation
↓
It is solved
by Aging



$$WT \text{ of } P_1 = 0 + (8 - 1) - 0 = 7$$

$$WT \text{ of } P_2 = 1 + (4 - 2) - 1 = 2$$

$$WT \text{ of } P_3 = 2 - 2 = 0$$

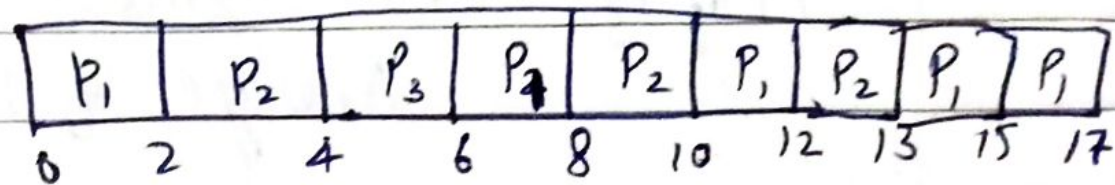
$$TT \text{ of } P_1 = 17$$

$$TT \text{ of } P_2 = 7$$

$$TT \text{ of } P_3 = 2$$

Round Robin \rightarrow Time Quantum = 2ms

Process	BT
P ₁	10
P ₂	5
P ₃	2



CT-BT-AT

17-10=7

$$\text{WT of } P_1 = 0 + (6-2) + (10-8) + (13-10) \\ 4 + 2 + 1 = 7$$

$$\text{WT of } P_2 = 2 + (8-4) + (12-10) = 8$$

$$\text{WT of } P_3 = 4$$

$$\text{TT of } P_1 = 7 + 10 = 17$$

$$\text{TT of } P_2 = 8 + 5 = 13$$

$$\text{TT of } P_3 = 2 + 4 = 6$$

$$\text{Av. TT} = \frac{17+13+6}{3}$$

$$= \frac{46}{3} = 15.33$$

Non-preemptive

① FCFS Algorithm
[First come First Serve]

Process	BT	CT	TAT = CT - AT		WT = TAT - BT	
P ₁	20	20	20	20	0	0
P ₂	4	24	24	24	20	20
P ₃	3	27	27	27	24	24
P ₄	2	29	29	29	27	27
	<u>29</u>		<u>100/4 = 25</u>		<u>71/4 = 17.75</u>	

P_1	P_2	P_3	P_4	
0	20 (+4)	24 (+3)	27 (+2)	29

② SJF algorithm [Shortest Job First] Non-preemptive

	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P ₀	0	8	8	8	0
P ₁	1	5	20	19	14
P ₂	2	2	11	9	7
P ₃	3	5	25	22	17
P ₄	4	4	15	11	7
P ₅	5	1	9	4	3
		<u>25</u>		<u>4</u>	<u>3</u>

$$\text{Avg} = 73/6 = 12.1$$

$$\text{Avg} = 48/6 = 8$$

P_0	P_5	P_2	P_4	P_1	P_3	
0	8 (+1)	9 (+2)	11 (+4)	15 (+5)	20 (+1)	25

③ Non preemptive priority Scheduling

	AT	BT	Priority	CT	TAT = CT - AT	WT = TAT - BT
P ₁	0	10	2	10	10	0
P ₂	2	5	1	35	33	28
P ₃	3	2	0	37	34	32
P ₄	5	20	3	30	25	5
		37				

$$\text{Avg} = \frac{102}{4} = 25.5$$

$$\text{Avg} = \frac{65}{4} = 16.25$$

Gantt chart: [P₁ | P₄ | P₂ | P₃]
 0 10 (+20) 30 (+5) 35 (+2) 37

Pre-emptive Algorithm
 ① SRTF [Shortest Remaining Time First]
 { Pre-emptive version of SJF }

	AT	BT	CT	TAT = CT - AT	WT = TAT - BT
P ₁	0	8-1=7	18	18	10
P ₂	1	4-1=3	6	5	1
P ₃	2	9-1=8	26	24	15
P ₄	3	5-1=4	11	8	3

Gantt chart: [P₁ | P₂ | P₃ | P₂ | P₄ | P₁ | P₃]
 0 (+1) 1 (+1) 2 (+1) 3 (+3) 6 (+5) 11 (+1) 18 (+8) 26

$$\text{Avg of TAT} = \frac{55}{4} = 13.75$$

$$\text{Avg of WT} = \frac{29}{4} = 7.25$$

② Pre-emptive priority algorithm

	AT	Priority	BT	CT	TAT	WT
P ₁	0	2	10-2=8	33	33	23
P ₂	2	1	5-1=4	37	35	30
P ₃	3	0	2-2=0	5	2	0
P ₄	5	3	20	25	20	0
			37			

$\text{Avg} = \frac{30}{4} = 7.5$ $\text{Avg} = \frac{53}{4} = 13.25$

0 (42) 2 (+1) 3 (+2) 5 (420) 25 (+8) 33 (+4) 37

ms

③ Round Robin Algorithm [Pre-emptive] TQ=3

Process	AT	BT	CT	TAT	WT = TAT - BT
P ₁	5	8	26	21	16
P ₂	4	6	24	20	14
P ₃	3	4	30	27	20
P ₄	1	8	29	28	19
P ₅	2	7	6	4	2

-	P ₄	P ₅	P ₃	P ₂	P ₁	P ₄	P ₃	P ₂	P ₁	P ₄	P ₃	
0	1 (+3)	4 (+2)	6 (+3)	9 (+3)	12 (+3)	15 (+3)	18 (+3)	21 (+3)	24 (+3)	26 (+3)	29 (+1)	30

$\text{Avg. of TAT} = \frac{21 + 20 + 27 + 28 + 4}{5} = \frac{100}{5} = 20 \text{ ms}$

$\text{Avg. of WT} = \frac{71}{5} = 14.2 \text{ ms}$

Q.1 Explain about deadlock. What are the necessary conditions for deadlock to occur?

Ans A deadlock is a state in which each member of a group is waiting for another member, including itself to take action such as sending a message or more commonly releasing a lock. Deadlock is a common problem in multiprocessing systems where s/w & h/w locks are used to arbitrate shared resources and implement process synchronization.

In OS, a deadlock occurs when a process thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another waiting process.

If a process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then system is said to be in deadlock.

The 4 necessary conditions for deadlock to occur are: -

1. Mutual Exclusion: - Resources involved must be unsharable

, otherwise process would not be prevented from using the resource when necessary.

2. Hold & wait :- It is for partial allocation. The process must hold the resources they have already been allocated while waiting for other resources.

3. No - preemption :- The process must not have resources taken away while that resource is being used.

4. Resource waiting or circular wait :-
A circular chain of processes with each process holding resources which are currently being requested by the next process in the chain, can't exist.

Q.2 Explain the fragmentation & difference b/w Internal & External fragmentation.

Ans -

Fragmentation :-

As processes are loaded & removed from memory. The free memory space is broken into little pieces. It happens after some times that

processes can't be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as fragmentation.

<u>Internal Fragmentation</u>	<u>External Fragmentation</u>
(i) In internal fragmentation, fixed size memory blocks square measure appointed to process.	Whereas, variable sized memory blocks square measure appointed to method
(ii) It happens when the method or process is larger than memory.	It happens when method or process is removed.
(iii) The solution of internal frag. is best fit block.	Solution of external frag. is compaction, paging etc.
(iv) Occurs when memory is divided into fixed sized partitions.	Occurs when memory is divided into variable sized partitions.

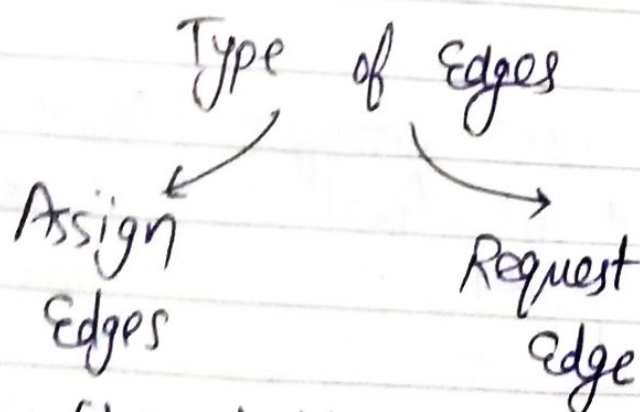
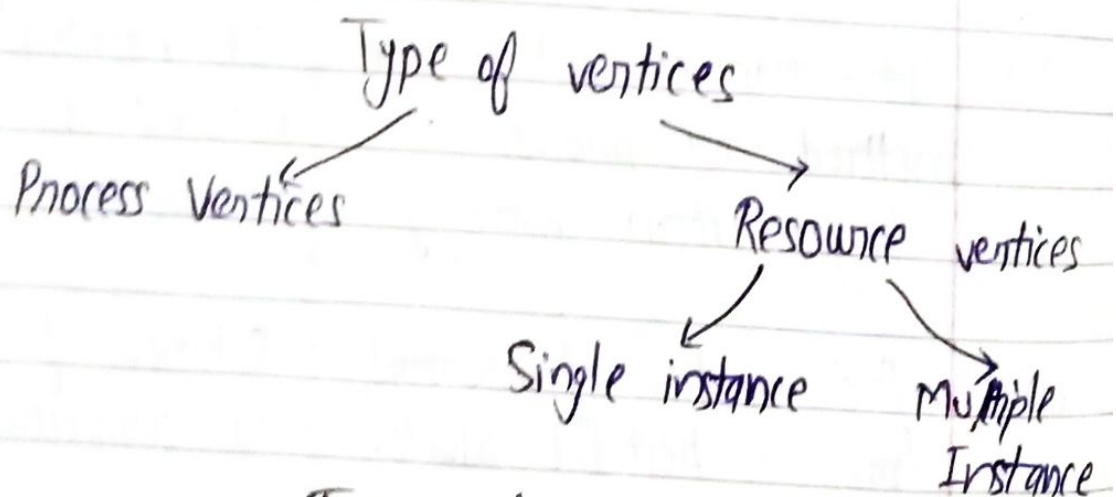
Q.3 Explain (i) Resource Allocation Graph

Ans - The RAG is pictorial representation of the state of system. As its name suggests the RAG is complete information about

all processes which are holding some resources or waiting for some resources. It also contains the information about all instances of all resources whether they are available or being used by the processes. In RA6, the process is represented by a circle while the resource is represented by a rectangle.

There are 2 components of RA6:-

- (i) Vertices
- (ii) Edges



(ii) Deadlock Characteristic:-

(a) Mutual Exclusion:- There should be a source that can only be held by one process at a time.

- (b) Hold & wait :- A process can hold multiple resources and still request more resources from other processes which are holding them.
- (c) No-preemption :- A resource can't be preempted from a process by force.
- (d) Circular Wait :- A process is waiting for resource held by second process, which is waiting for resource held by 3rd process & so on.

Q.4 What are memory management & explain Swapping.

Ans-

Memory Management :- It is the functionality of an OS which handles or manages primary memory and moves processes back and forth b/w main memory & disk during execution. Memory Management keeps track of each & every memory location regardless of whether it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at which time. It tracks

whenever some memory gets free or un-allocated and correspondingly it updates the states.

Swapping:- Swapping is a mechanism in which a process can be swapped temporarily out of main memory to secondary storage & make that memory available to other processes. At some later time, the system swaps back the process from secondary storage to main memory. Swapping is also known as technique for Memory Compaction.

Q.5 Explain

(i) Logical & physical Address Space

Ans- Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore it is also known as virtual Address. This address is used as a reference to access the physical memory location by CPU. The term logical address space is used for set of all logical address generated by program's perspective.

Physical Address identifies a physical location of ~~Locati~~ required data in memory. The user never directly deals with physical address but can access by its corresponding logical address. The term physical address space is used for all physical address corresponding to the logical address in a logical address space.

(ii) Relocation & Address transaction :—

Relocation is the process of assigning load addresses for position-dependent code and data of a program & adjusting the code and data to reflect the assigned addresses.

Relocation is typically done by the linker at link time, but it can also be done at load time by a relocating loader or at runtime by running program itself.

Address transaction concatenates the frame number with the offset part of a logical address to form a physical address. A page table base register, holds the base address

for the page table of the current process. It is a processor register that is managed by the Operating System.

Q.1

Chapter-4

Explain the following

(1)

Virtual Memory:- A computer can address more memory than amount physically installed on system. This extra memory is actually called virtual memory & it is a section of harddisk that's set up to emulate computer's RAM.

- Advantage of this is programs can be larger than phy. memory. It serves 2 purposes:
- First, it allows us to extend the use of physical memory by using disk.
- Second, it allows us to have memory protection bcz each virtual add. is translated to phy. address.
- Modern MCU intended for general purpose use, a MMU, is built into h/w. MMU's job is to translate virtual add. into phy. add. A basic example is given on next page.
- VM is commonly implemented by demand paging. It can also be implemented in segmentation system. Demand segmentation can also be used to provide

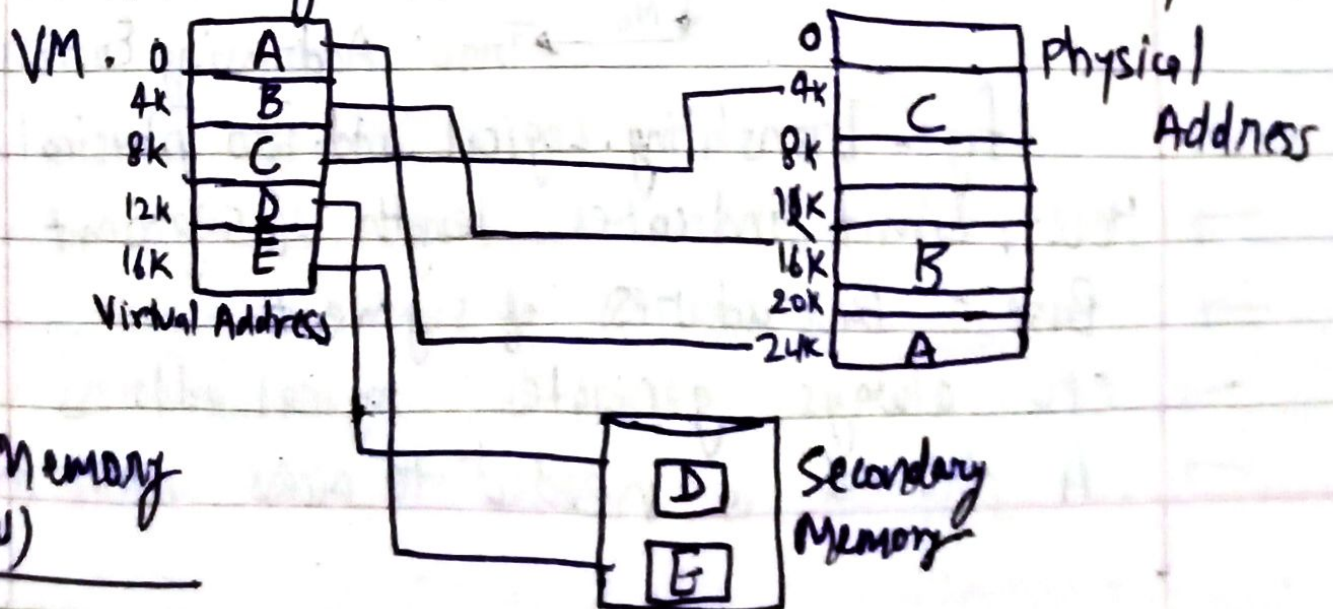


Fig: Ex. of
Virtual Memory
(Using MMU)

- (ii) Segmentation :- Like paging, segmentation is another non-contiguous memory allocation tech.
- Process is not divided blindly into fixed size pages. Rather, process is divided into modules for better utilization.
 - It is a variable size partitioning scheme. Here, Secondary m & main m are divided into partitions of unequal size & their size depends on length of modules.
 - The partitions of secondary memory are called segments.
 - Segment Tables: (i) It stores info. about each segment of process. It has 2 columns

- (ii) First column stores size or length. Second stores base add. or starting add. of segment in main memory
- (iii) Segment Table is stored as separate segment in MM
- (iv) It's base register stores the base address of segment Table

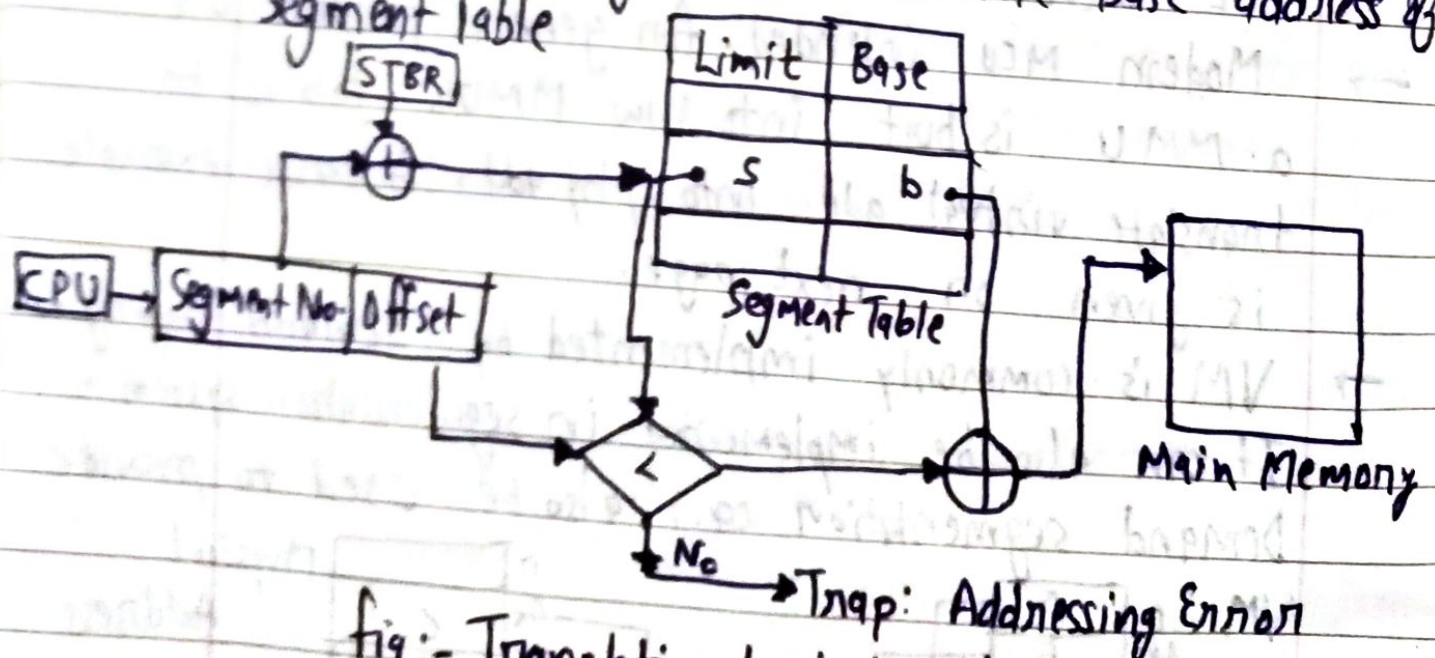


Fig:- Translating Logical add. into physical add.

- Here, Limit indicates length of segment
- Base = Base address of segment
- CPU always generates logical address
- A phy. add is needed to access main memory

Q.2 Explain various page replacement using example.

Ans → 0 Various page replacement Algorithm:—

(i) Optimal Page Replacement Algorithms ⇒ This algo. replaces page that will not be used for longest period of time.

Eg- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 no. of frames = 3

7	7	7	2	2	2	2	2	2	7								
0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	3	3	3	3	3	1	1	1	1	1	1	1	1	1

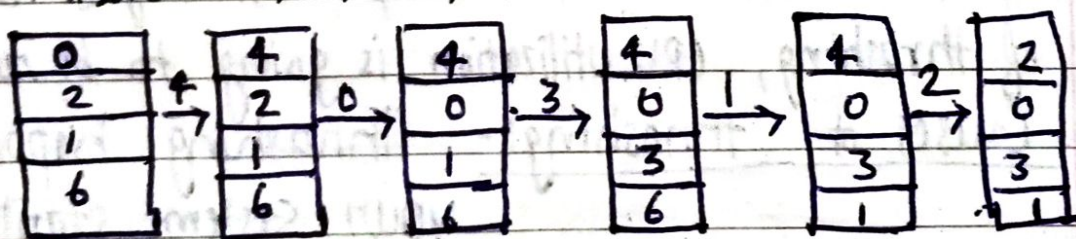
No. of page faults = 9

(ii) FIFO Page Replacement Algorithm ⇒ The oldest page, which has

spent the longest time in memory is chosen & replaced. A page is inserted at REAR end of queue & is replaced at FRONT of queue.

Eg- String:- 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses:- x x x x x x x x x x

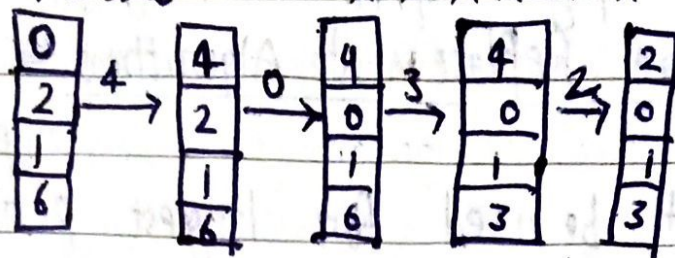


(iii) LRU (Least Recently Used) ⇒ This algo. replaces the pages that has not been

used for the longest period of time. It is based on observation that pages that have not been used for long time will probably remain unused for longest time and are to be replaced.

Eg - Reference String: - 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses: - X X X X X X X X

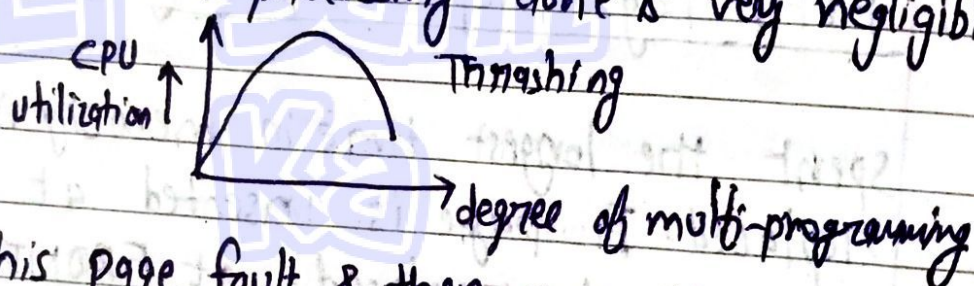


Q.3

A. - 0

Explain concept of Thrashing & TLB

Thrashing: - It is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible.

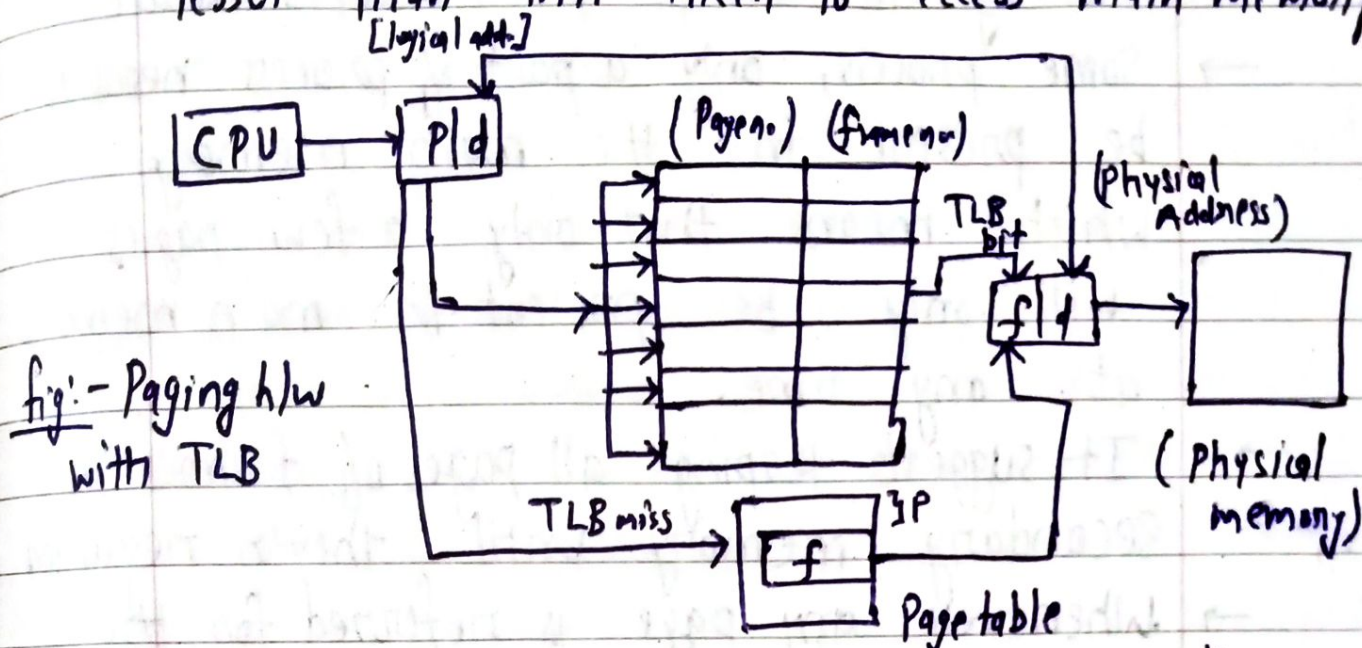


In this page fault & then swapping happening very frequently at higher rate, then OS has to spend more time to swap these pages. Because of thrashing, CPU utilization is going to be reduced.

Causes of thrashing: - Thrashing happens when your system starts spending more time in doing paging rather than performing computation. It results in severe performance problems.

• TLB [Translation Look-aside Buffer]: - TLB is an associative, high speed memory. It can be defined as memory which can be used

to reduce time taken to access the page table again & again. It is a memory cache which is closer to CPU & time taken by CPU to access TLB is lesser than that taken to access main memory.



There are taps & keys in TLB, with help of which, the mapping is done. TLB hit is condition where desired entry is found in translation look aside buffer. If this happens then the CPU simply access the actual location in main memory.

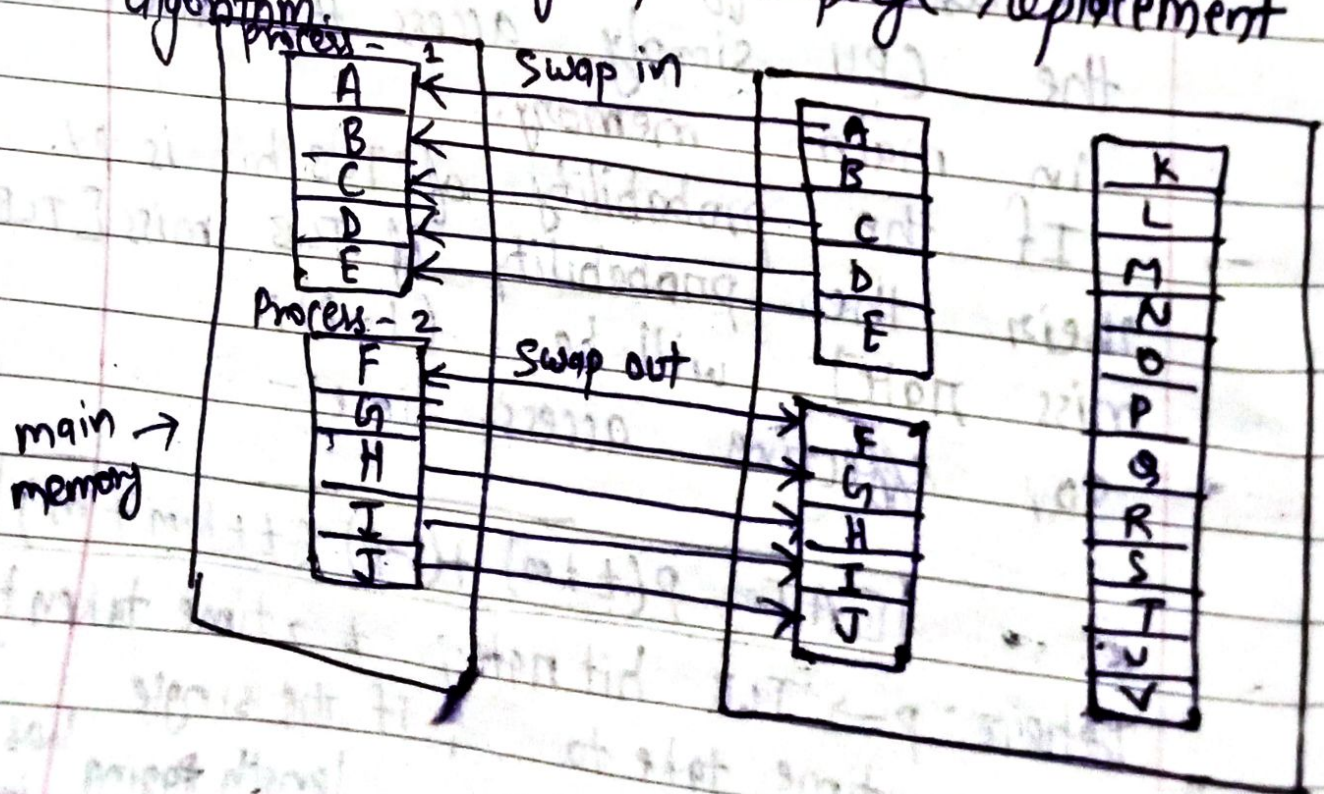
→ If the probability of TLB hit is $p\%$, then the probability of TLB miss [TLB miss rate] will be $(1-p)\%$.
So, effective access time:-

$$EAT = p(t+m) + (1-p)(t+k \cdot m + m)$$

Where $p \rightarrow$ TLB hit rate; $t \rightarrow$ time taken to access TLB
 $m \rightarrow$ time take to access main memory, if the single length paging has been implemented

Q.4
(i) Explain the following Demand Paging: — DP is a method of virtual memory management.

- According to concept, in order to execute
- Some process, only a part of process needs to be present in the main memory which means that only a few pages will only be present in main memory at any time.
 - It suggests keeping all pages of frames in secondary memory until they're required
 - Whenever any page is referred for the first time in MM then that page will be found in secondary memory.
 - After that it may or may not be present in MM depending upon page replacement algorithm.



(ii) Global versus local Allocation: — The no. of frames allocated to a process can dynamically change depending on whether global is used or local is used for replacing pages in case of page fault.

Global Replacement: — When process needs a page which is not in memory, it can bring in new page & allocate it a frame from ~~set of~~ all frames ~~from the~~ even if that frame is currently allocated to some other process;

Adv → It doesn't hinder the performance of process & hence results in greater system throughputs.

Disadv → The page fault ratio of process can't be solely controlled by process itself.

→ Local Replacement: — When a process needs a page which is not in memory it can bring in new page & allocate it a frame from its own set of allocated frames only.

Adv → The page in memory for particular process & page fault ratio is affected by paging behavior of only that process.

Disadv → A low priority process may hinder a high priority process by not making its frames available to high priority process.

Q.5 Consider least recent used algorithm using a matrix when pages are referred in order of 0, 1, 2, 3, 2, 1, 0, 3, 2, 3 and calculate page fault.

Ans - LRU :- This algorithm replaces the page that hasn't been used for the longest period of time. It is based on observation that pages that have not been used for long time will probably remain unused for longest time & are to be replaced.

Given order:- 0, 1, 2, 3, 2, 1, 0, 3, 2, 3

Let us consider 3 frames for storing process pages in main memory.

0	1	2	3	2	1	0	3	2	3
		2	2	2	2	2	3	3	3
	1	1	1	1	1	1	1	2	2
0	0	0	3	3	3	0	0	0	0

\checkmark \checkmark \checkmark \checkmark \times \times \checkmark \checkmark \checkmark \times
 Total no. of page faults occurred = 7

\Rightarrow Hit Ratio = $\frac{3}{10} \times 100\%$ or 0.3
 \Rightarrow Miss Ratio = $\frac{7}{10} \times 100\%$ or 0.7

BANKER'S Algorithm: — for multiple instances

n = no. of processes

m = no. of resource types

Available: Vector of length m . If $Available[j] = k$, there are k instances of resource type R_j available.

Max: $n \times m$ matrix. If $Max[i, j] = k$, then Process P_i may request

Allocation:-

Need: Max - Allocation

Safety Algorithm:- (i) $Work = Available$

$finish[i] = false$ for $i = 0, 1, \dots, n-1$

(ii) Find j and i such that both

(a) $finish[i] = false$

(b) $Need_i \leq Work$

If no such i exists, go to step ④

(iii) $Work = Work + Allocation$

$finish[i] = true$

goto step ②

(iv) If $finish[i] = true$ for all i , then system is in a safe state.

Resource Request Algorithm:-

$Request$ = request vector for P_i

(i)

$Request \leq Need$: go to step ②. Otherwise, raise error condition, process has exceeded its max. claim.

(ii)

$Request \leq Available$ go to step ③. Otherwise P_i must wait since resources are not available.

(iii) Pretend to allocated request resources to P_i by modifying state as

$$Avail = Avail - Request;$$

$$Alloca = Alloca + Request$$

$$Need = Need - Request$$

If safe \Rightarrow resources are allocated to P_i

If unsafe $\Rightarrow P_i$ must wait, & old resource allocation state is restored.

Q. 5 processes P_0 through P_4 : 3 resource types A(10 instance) B(5 instance) and C(7 instance)

	Allocation	Max	Available	Need	
	ABC	ABC	ABC		Work = available
P_0	0 1 0	7 5 3	3 3 2	7 4 3	= 332
P_1	2 0 0	3 2 2		1 2 2	= 532
P_2	3 0 2	9 0 2		6 0 0	= 743
P_3	2 1 1	2 2 2		0 1 1	= 745
P_4	0 0 2	4 3 3		4 3 1	Finish = 755
					= 1057

IF IF IF IF IF
0 1 2 3 4

Need = Max - Allocation Safety algorithm: -

ABC =

7 4 3

1 2 2

6 0 0

0 1 1

4 3 1

For process P_0

Finish[0] = false

Need \leq work

7 4 3 \leq 332

Work = 745 + 0 1 0

= 755

For process P_1

Finish[1] = false

Need \leq work

1 2 2 \leq 332

Work = 332 + 2 0 0

= 532

Process P_2

-

6 0 0 \leq 532

6 0 0 \leq 755

755

+ 3 0 2

= 1057

P_3

-

0 1 1 \leq 532

0 1 1 \leq 755

532 + 0 1 1

743

743

P_4

-

4 3 1 \leq 743

4 3 1 \leq 743

743 + 0 0 2

= 745

Safe sequence $\langle P_1, P_3, P_4, P_2, P_0 \rangle$

RRA: -

If P_1 requests (1,0,2), determine if it can be granted immediately.

$$P_1 \rightarrow R(102)$$

$$\text{Need}(P_1) = \text{Max} - \text{Allo.} = 322 - 200 = 122$$

① Request \leq need

$$102 \leq 122 \checkmark \text{ go to } ②$$

② Request \leq Available

$$102 \leq 332 \checkmark$$

$$\begin{aligned} ③ \quad \text{avail} &= \text{avail} - \text{request} \\ &= 332 - 102 = 230 \end{aligned}$$

$$\text{allo.} = \text{alloc} + \text{request} = 200 + 102 = 302$$

$$\text{need} = \text{need} - \text{request} = 122 - 102 = 020$$

New table

	Alloc.	Max.	Available	Need
P_0	010	753	(230)	743
P_1	(302)	322		(020)
P_2	302	902		600
P_3	211	222		011
P_4	002	433		431

Now apply Banker safety algo.

$$P_0: 743 \leq 230 \times$$

$$P_1: 020 \leq 230 \checkmark$$

$$P_2: 600 \leq 532 \times$$

$$P_3: 011 \leq 532 \checkmark$$

$$P_4: 431 \leq 743 \checkmark$$

$$P_0: 743 \leq 745 \checkmark$$

$$P_2: 600 \leq 755 \checkmark$$

$$W = 230 + 302 = 532$$

$$W = 532 + 211 = 743$$

$$W = 743 + 002 = 745$$

$$W = 745 + 002 = 755$$

$$W = 755 + 302 = 1057$$

C_1, P_3, P_0, P_2

Yes P_1 requests (102) can be granted immediately.

BANKER'S Algorithm:-

Q. 5 processes P_0 through P_4 : 3 resource types
 A (10 instances), B (5 instances) and C (7 instances)

	Allocation	Max	Available	Need
	ABC	ABC	A B C	
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2	5 3 2	1 2 2
P_2	3 0 2	9 0 2	7 4 3	6 0 0
P_3	2 1 1	2 2 2	7 4 5	0 1 1
P_4	0 0 2	4 3 3	7 5 5 10 5 7	4 3 1

(Need = Max - Allocation)

Safe sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$

Banker's Algorithm :-

Safety Algorithm

① $work = available = 332$

② $Need_i \leq work \Rightarrow work = work + allocation$

Finish

F	F	F	F	F
0	1	2	3	4

$P_0 : 743 \leq 332 \quad X$

$P_1 : 122 \leq 332 \quad \checkmark$

$W = W + alloc. = 332 + 200 = 532$

$P_2 : 600 \leq 532 \quad X$

$P_3 : 011 \leq 532 \quad \checkmark$

$W = W + alloc. = 532 + 211 = 743$

$P_4 : 431 \leq 743 \quad \checkmark$

$W = W + alloc. = 743 + 002 = 745$

$P_0 : 743 \leq 745 \quad \checkmark$

$W = W + alloc. = 745 + 010 = 755$

$P_2 : 600 \leq 755 \quad \checkmark$

$W = W + alloc. = 755 + 302 = 1057$

 $\overline{A} \quad \overline{B} \quad \overline{C}$

Finish

T	T	T	T	T
0	1	2	3	4

Safe Sequence = $\langle P_1 P_3 P_4 P_0 P_2 \rangle$

Resource Request Algorithm:-

If P_1 requests $(1, 0, 2)$ determine if it can be granted immediately

$$P_1 \rightarrow R(102)$$

$$\text{Need}(P_1) = \text{Max} - \text{alloc} = 322 - 200 = 122$$

① $\text{Request} \leq \text{need} \Rightarrow 102 \leq 122 \quad \checkmark \quad \text{goto } ②$

② $\text{Request} \leq \text{Available} \Rightarrow 102 \leq 332 \quad \checkmark \quad \text{goto } ③$

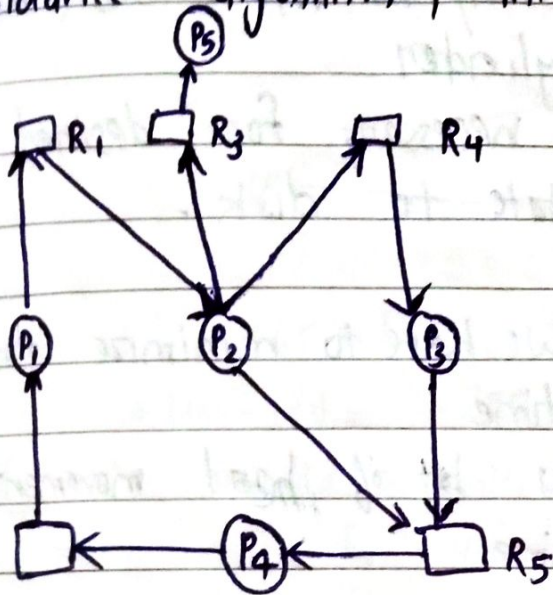
③ $\text{available} = \text{available} - \text{request}$

$$= 332 - 102 = 230$$

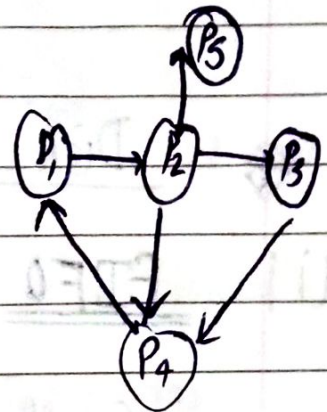
$$\text{Allocation} = \text{Allo.} + \text{req.} = 200 + 102 = 302$$

$$\text{Need} = \text{Need} - \text{Request} = 122 - 102 = 020$$

Deadlock Detection :- If system does not employ either a deadlock-prevention or a deadlock-avoidance algorithm, then a deadlock situation may occur.



(a) RAG



(b) wait for graph

How to Recover from deadlock :-

(i) Process Termination :-

- (a) Abort all deadlocked processes
- (b) Abort one process at a time until deadlock cycle is eliminated.

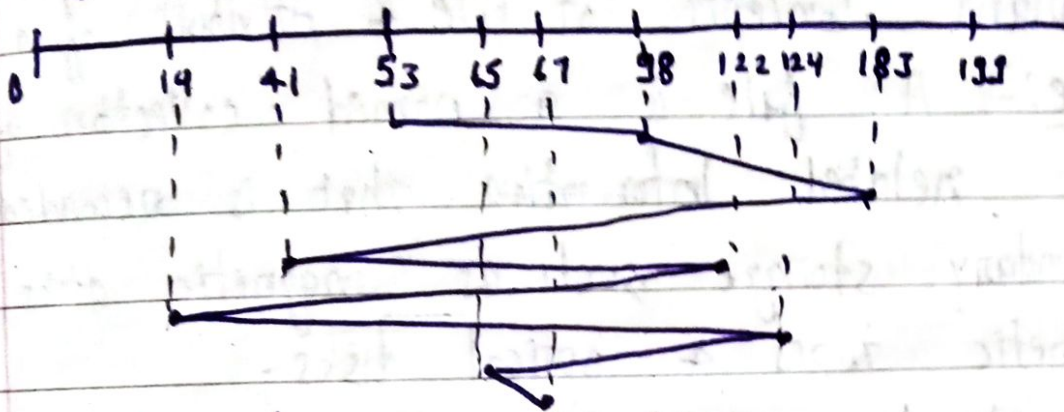
(ii) Resource Preemption :- [3 issues need to be addressed :]

- (a) Selecting a victim
- (b) Rollback
- (c) Starvation

Q.1 Explain various Disk Scheduling Algorithm in brief.

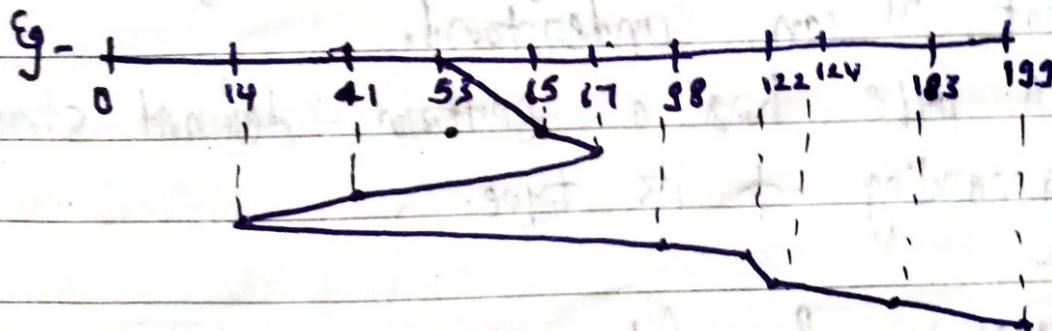
(i) FCFS:-

Eg - 98, 183, 41, 122, 14, 124, 65, 67, H = 53; 200 cylinders.



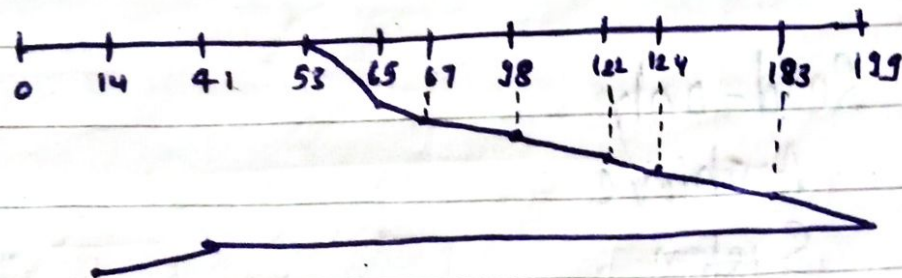
Total head movement = 632

(ii) SSTF:-



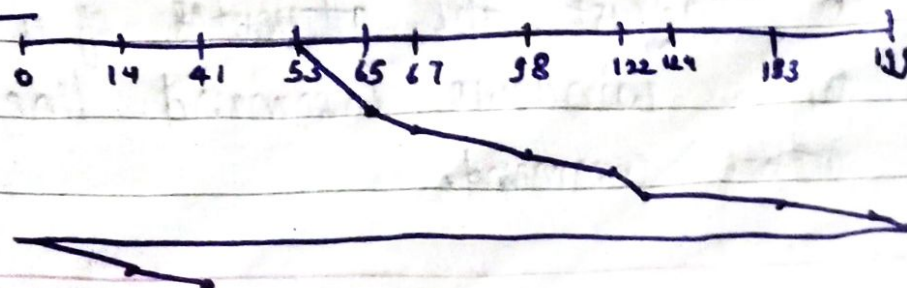
Total head = 236

(iii) SCAN:-



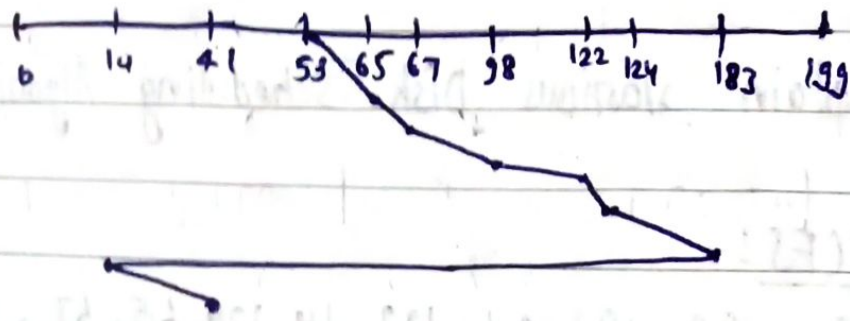
Total head = 332

(iv) CSAAN:-



Total head = 386

(v) Look : -



Q.2 Explain concepts of file & Attribute of a file.

Ans - File :- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes & optical disks.

File structure :- A file structure should be according to a required format that OS can understand.

→ A file has a certain defined structure according to its type.

Attributes of a file :-

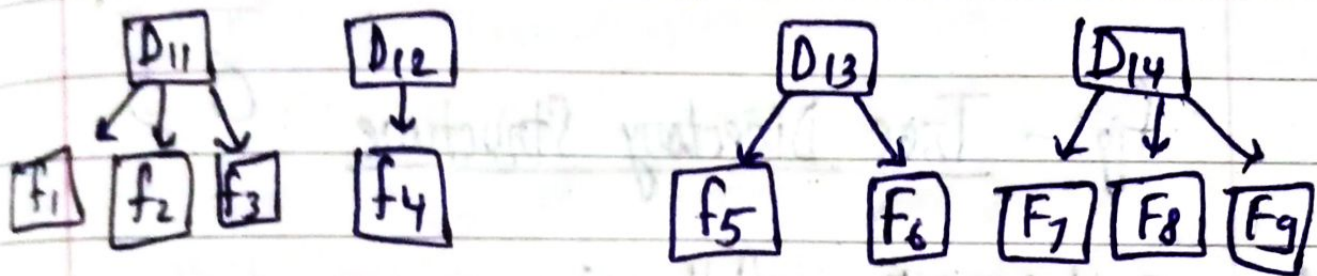
- (i) Read-only
- (ii) Archive
- (iii) System
- (iv) Hidden

To adjust the attributes of a file in MS-DOS or windows command line, use the attrib command.

Q.3 Explain directory structure & briefly explain about tree structured directory.

Ans - Directory:- A directory is a container that is used to contain folders & file. It organizes file & folders into a hierarchical manner.

D₁



There are several logical structure of a directory, there are given below:-

- (i) Single-level directory
- (ii) Two-level directory
- (iii) Tree-Structured directory

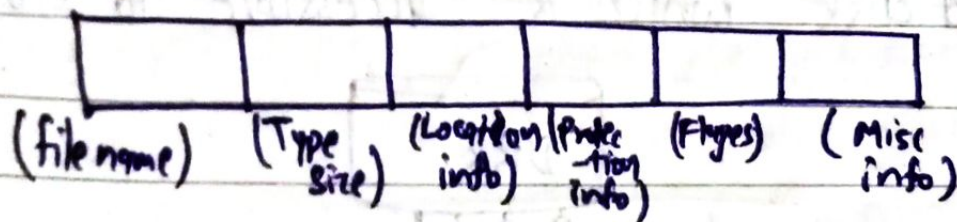


fig:- Directory structure.

Tree - Directory Structure:- In a TDS, except root directory, every directory or file has only one parent directory so there is total separation b/w users which provide complete naming freedom.

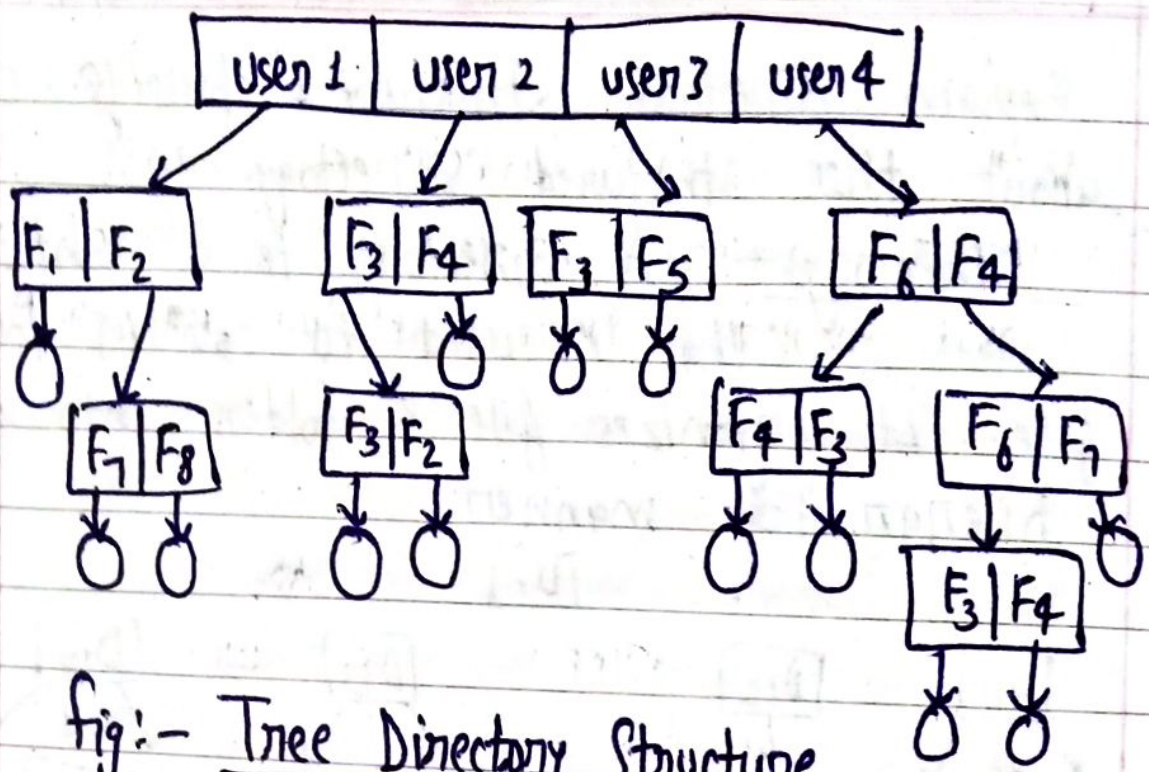


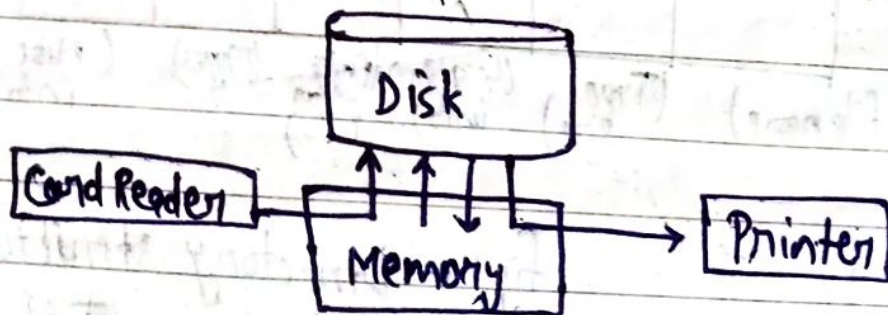
Fig:- Tree Directory Structure

Q.4

(1)

Explain the following:-

Spooling:- It is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.



- The spooling operation uses a disk as a very large buffer.
- Spooling is capable of overlapping I/O operation for one job with process or operations for another job.

(ii) File System Mounting:- Mounting refers to making a group of files in a FSM accessible to user or group of users. It is done by attaching a root directory from one file to that of another.

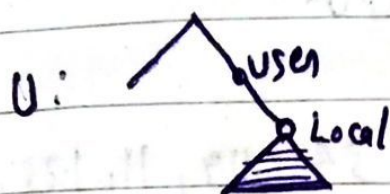
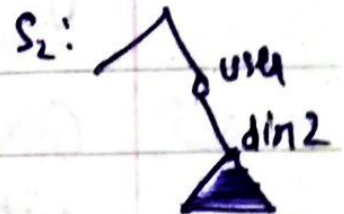
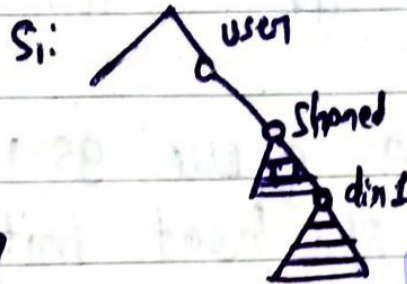


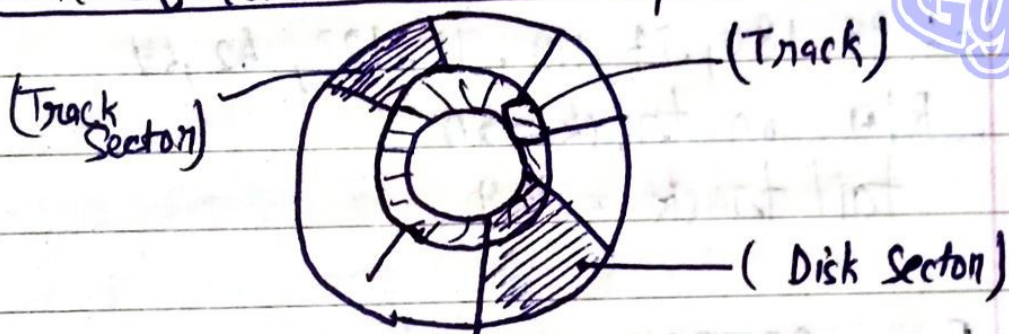
Fig:- Mounting of files



Er Sahil
Ka
Gyan

→ Mounting may be local or remote.

(iii) Disk Structure & Disk Operation:-



The disk is divided into tracks. Each track is further divided into sectors. Outer tracks are bigger in size than inner tracks but they contain same number of sectors & have equal storage capacity. This is because the storage density is high in sectors of inner tracks.

Disk Operation:- Disk platters are mounted on single spindle that spins at a

typical 10,000 rpm.

On EIDE & SCSI drives the disk controller is part of drive itself. It controls the drive's servomotors & translates the fluctuating voltages from head into digital data for CPU.

Q.5

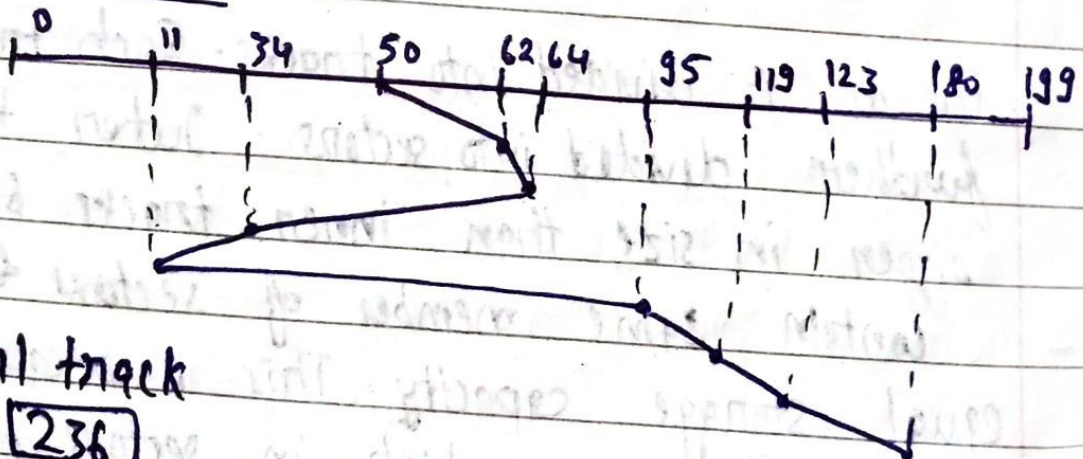
Given queue: 95, 180, 34, 119, 11, 123, 62, 64 with R/W head initially at track 50 & trail track being at 199 to calculate by SSTF & SCAN & look algorithm.

Ans -

-: 95, 180, 34, 119, 11, 123, 62, 64
R/W on track = 50
tail track = 199

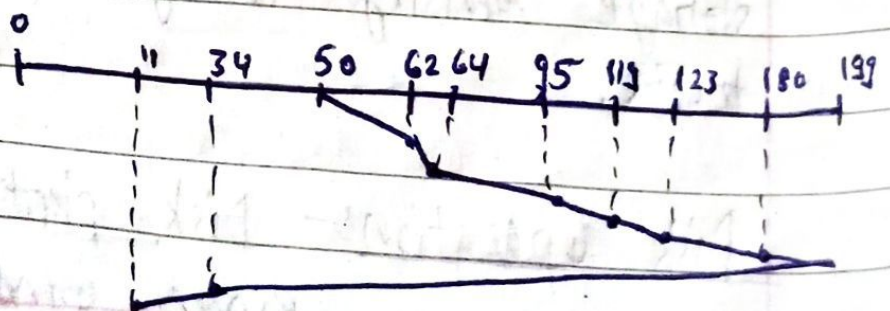
Er Sahil
Ka
Gyan

For SSTF:-



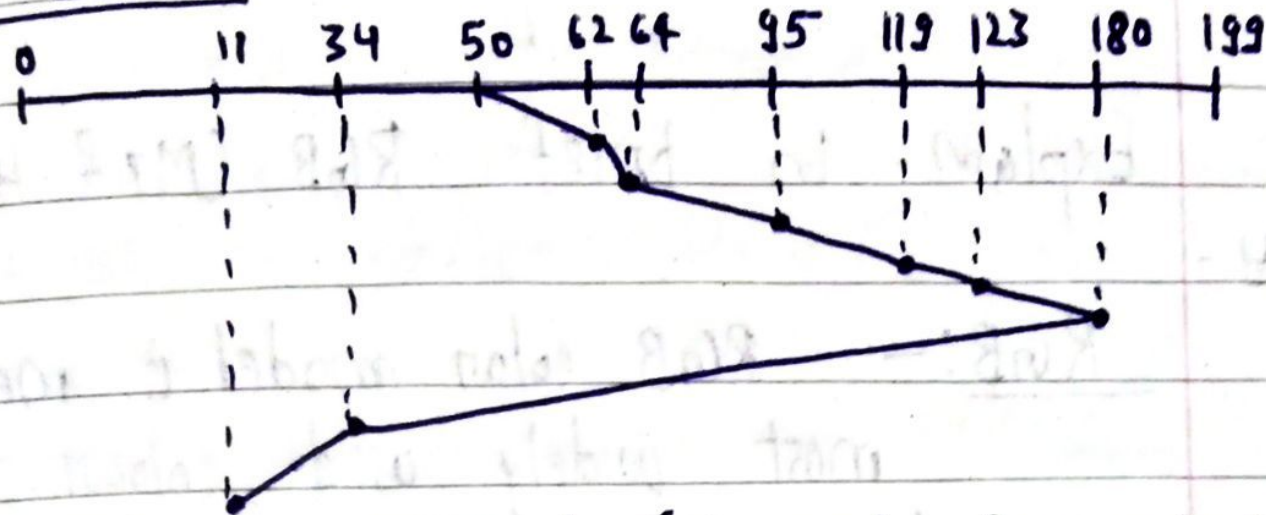
Total track
= 236

For SCAN:-



Total track
= 360

For Look:-



$$\text{Total head Movement} = \frac{1}{2} \{ (62-50) + (64-62) + (95-64) \\ + (119-95) + (123-119) + (180-123) \\ + (180-34) + (34-11) \}$$

$$= (12 + 2 + 31 + 24 + 4 + 57 + 146 + 23)$$

$$= \boxed{299}$$

Er Sahil
Ka
Gyan