Year 2022

① Define operating system with its function. Differentiate process and thread.

Operating system (OS) is a software program that manages the hardware and software resources of a computer. A program that acts as an intermediary between a user of the computer and the computer hardware. An operating system is like a government like it performs no useful function by itself. It simply provides an environment within which other program can do useful work. Hence an operating system is a program that controls the execution of application program and acts as an interface between user of a computer hardware some of the example of operating system are UNIX, Mach, MS-DOS, MS-Windows and so on.

The function of operating system are:-

(i) Implementing the user interface
(ii) Sharing hardware among users
(iii) Handling network communication
(iv) Facilitating parallel operations
(v) Allowing users to share data among themselves
(vi) Preventing users to share data among interferring with one another
(vii) Scheduling resource among users
(viii) Recovering from errors
(ix) Accounting for resource page
(x) Facilitating input/output

2) Differences between process and thread are:-

	Process	Thread
1)	Each process has their own address space.	Thread share the address space of process that created it.
2)	Each process has their own copy of data, code & file segment.	Threads share the same copy of data, code & file segment.
3)	Process must user inter process communication to communicate with sibling processes.	Thread can directly communicate with other thread of its processes.
4)	More resources are required.	Fewer resource are required.
5)	New processes required duplication of parent process.	New thread are easily created.
6)	Not suitable for parallelism.	suitable for parallelism.
7)	System call is involved in process.	There is no system called involved.
8)	context switching is slow.	context switching is faster.

3) Define Inter-Process communication and race condition. Consider set of process with their burst time and arrival time as shown below:

Process	Arrival Time	Burst time
P1	0	15
P2	1	7
P3	2	2
P4	3	8
P5	4	6
P6	6	3

Assume quantum time = 2sec. find better scheduling using shortest job first, round robin & first come first serve scheduling.

Interprocess communication is a mechanism that allow to communicate & synchronized their action without sharing address space. It is useful. In the distributed environment where communicating process may resides on different machine with a network.

IPC mechanism includes:

(I) shared Memory
(II) Message Passing

A race condition is a situation where multiple process concurrently read & write to a shared memory location & the result depends upon the order of execution. There are four condition for solution to eliminate race condition.

(I) Mutual exclusion
(II) Progress
(III) Bound waiting
(IV) Architectural Neutral

1) For SJF,

P1	P2	P3	P3	P2	P6	P2	P5	P4	P1

0 1 2 3 4 6 9 13 19 27 41

Process	AT	BT	CT	TAT = CT - AT	WT = (TAT - BT)
P1	0	15	41	41	26
P2	1	7	32	31	24
P3	2	2	6	4	2
P4	3	8	36	33	25
P5	4	6	31	27	21
P6	6	3	25	19	16

A.WT = (26+24+2+25+21+16)/6

= 19

ATAT = (41+31+4+33+27+19)/6

= 25.83 //

Using FCFS → Gantt chart

P1	P2	P3	P4	P5	P6

0 15 22 24 32 38 41

Process	AT	BT	CT	TAT(CT - AT)	WT(TAT - BT)
P1	0	15	15	15	0
P2	1	7	22	21	14
P3	2	2	24	22	20
P4	3	8	32	29	21
P5	4	6	38	34	28
P6	6	3	41	35	32

A·WT = (0+14+20+21+28+32)/6
= 19.17//

ATAT = (15+21+22+29+34+35)/6
= 26//

Round Robin (Quantum) = 2 sec

Ready Queue

P1	P2	P3	P1	P4	P5	P2	P6	P1	P4	P5	P2	P6	P1	P4	P5	P2	P1	P4	P1

Gantt chart

P1	P2	P3	P1	P4	P5	P2	P6	P1	P4	P5	P2	P6	P1	P4	P5

0 2 4 6 8 10 12 14 16 18 20 22 24 25 27 29 31

P2	P1	P4	P1

31 32 34 36 41

Process	AT	BT	CT	TAT(CT-AT)	WT(TAT-BT)
P1	0	15	41	41	26
P2	1	7	32	31	24
P3	2	2	6	4	2
P4	3	8	36	33	25
P5	4	6	31	27	21
P6	6	3	25	19	16

A·W·T = (26+24+2+25+21+16)/6 = 19
ATAT = (41+31+4+33+27+19)/6 = 25.83 //

Date
Page

4. Explain sleeping barber problem. What do you mean by Dining philosophers problem, explain the solution with example.

- The sleeping Barber problem is a classic illustration of synchronization challenges in concurrent systems, often used to demonstrate process synchronization in OS. Dijkstra introduced the sleeping barber problem in 1965. This problem is based on a hypothetical scenario where there is a barbershop with one barber. The barbershop is divided into two rooms, the waiting room & the workroom. The waiting room has n chairs for waiting customers, and the workroom only has a barber chair.

Now, if there is no customer, then the barber sleeps in his own chair. Whenever a customer arrives, he has to wake up the barber to get his haircut. If there are multiple customers & the barber is cutting a hair then remaining customers wait in waiting room with 'n' chairs or they leave. This problem may lead to race condition.

Dining Philosophers problem contain five philosopher sitting around a circular table. Dining table has 5 chopstick & bowl of rice in the middle. Philosopher can be in thinking, eating & hungry state. When philosopher want to eat, he uses two chopstick. When philosopher want to think, he put down both chopstick. Each philosopher can forever think & eat continuously alternatively. The challenge is to design a solution that avoids deadlock (when no philosopher can finish eating) & avoids contention for resources.

The example is:
As spaghetti is difficult to serve & eat with a single fork, it is assumed that a philosopher can only use the fork on his or her immediate left or right. It is further assumed that the philosophers are stubborn that a philosoph only put down the forks after eating.
code:

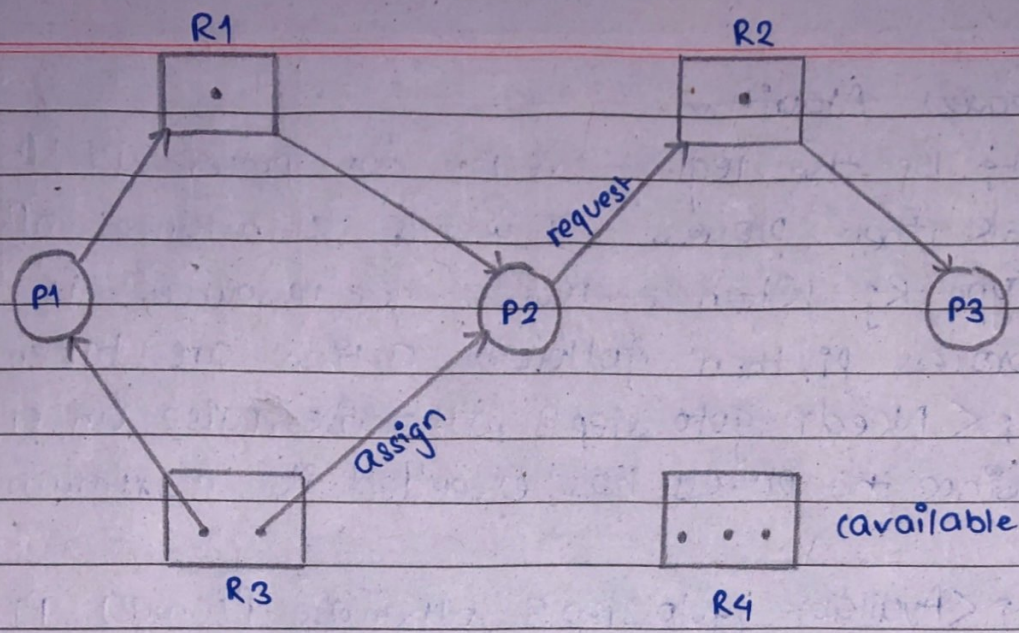
```
monitor dp
{ enum{ Thinking, Hungry, Eating};
  state[5];
  condition self[5];
  void pickup(int i)
  { state[i] = Hungry;
      test(i);
      if (state[i] != Eating)
        self[i].wait();
      }
    void putdown(int i)
    {
      state[i] = Thinking;
      test((i+4)%5);
      test((i+1)%5);
    }
    void test(int i)
  {
    if((state[(i+4)%5] != Eating) && (state[i] == Hungry)
    && (state[(i+1)%5] != Eating))
    { state[i] = Eating;
      self[i].signal();
```

```
    } }
initialization_code()
{
  for(int i = 0; i<5; i++)
    {
      state[i] = Thinking;
    }
  }
}
```

5. Explain about resource allocation graph with suitable example. Write an algorithm for Resource-request algorithm & safety algorithm.

- The resource allocation graph is the pictorial representation of a state of system. The RAG is the complex information about all the processes which are holding some resources or waiting for some resources. It also contain the information that all the instances of all the resources whether they are available or being used by the process. In RAG, process is represented by a circle & resource is represented by rectangle. A resource can have more than one instance. Each instance will be represented by dotted inside a rectangle.

Example:



In the diagram, the sets P, R, E

P = {P1, P2, P3}

R = {R1, R2, R3, R4}

E = {P1→R1, R3→P1, R3→P2, R1→P2, P2→R2, R2→P3}

Safety Algorithm

i) Lets work & finish be a vectors of length m & n respectively
Initialize work = Available & finish[i] = false, for i=0, 1, ... n-1

ii) Find an index i such that both
finish[i] == false
Need i ≤ 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 the system is in safe state.

Date
Page

Resource Request Algorithm

let Request$_i$ be the request vector for process P_i. If Request$_i$[j] = K, then process P_i wants K instance of resource type R_j. When a request for resources is made by process P_i, then following action are taken:-

i) If Request$_i$ < Need$_i$, goto step 2, otherwise, raise an error condition, since the process has exceeded its maximum claim.

ii) If Request$_i$ < Available, goto step 3, otherwise P_i must P_i must wait, since resource are not available.

iii) Have the system pretended to have allocated the request resources to process P_i by modifying the state as:

Available = Available - Request$_i$
Allocation$_i$ = Allocation$_i$ + Request$_i$
Need$_i$ = Need$_i$ - Request$_i$

6. Define first fit, next fit, quick fit & buddy system? consider the following page reference string.
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 8, 3, 2, 1, 2, 3
How many page fault occur in a) LRU
b) FIFO?

First fit - In this, the operating system allocates the first available memory block that is large enough to accomodate the process. It scans the memory from the beginning & allocates the first block it encounters that is large enough.

Next fit - Similar to first fit, next fit allocates the first available memory block that is large enough to accomodate the process. It starts the search for the next available block from the location where the last allocation occured.

Quick fit - Quick fit is an extension of the First fit that divides the memory into fixed-size partitions. Each partition holds memory blocks of a specific size, & the OS quickly allocates the first available block of the required size.

Buddy system - This system divides memory into fixed-size blocks & maintains a binary tree to represent the free blocks. When a process requests memory, the system finds the smallest available block that is large enough to accomodate the request. If the block is larger than needed, it is split into the two buddies, & the process is allocated one of them.

→ Soln:-

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 8, 3, 2, 1, 2, 3

Total no. of reference string = 19

number of page frame = 3

LRU,

1	2	3	4	2	1	5	6	2	1	2	3	7	8	3	2	1	2	3
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	8	8	8	1	1	1
*	*	*	*	H	*	*	*	*	*	H	*	*	*	H	*	*	H	H

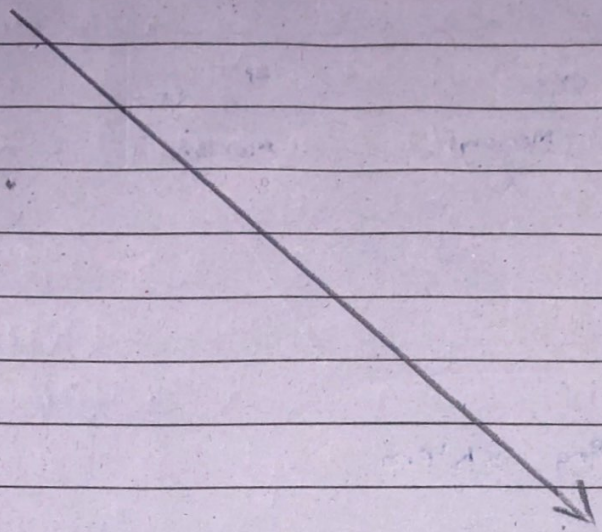
Total no. of page fault = 14 //

b) FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	8	3	2	1	2	3
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	8	8	8	8	8	3
*	*	*	*	H	*	*	*	*	*	H	*	*	*	H	*	*	H	*

Total no. of page fault = 15 //

Date
Page



7. Describe distributed OS with suitable diagram.

- Distributed OS is system software over a collection of independent software, networked, communicating & physically separate computational nodes. They handle jobs which are serviced by multiple CPUs. Each individual node holds a specific software subset of the global aggregate operating system. Each subset is a composite of two distinct service provisioners. The first is a ubiquitous minimal kernel, or microkernel that directly controls that nodes' hardware. Second level is a higher level collection of system management components that coordinate the node's individual & collaborative activities. These components abstract microkernel functions & support user applications. In this system, the OS on all the machines work together to manage the collective network resource. OSes on different computer running act like a single OS.

Next fit - Similar to first fit, this allocates the first Avail

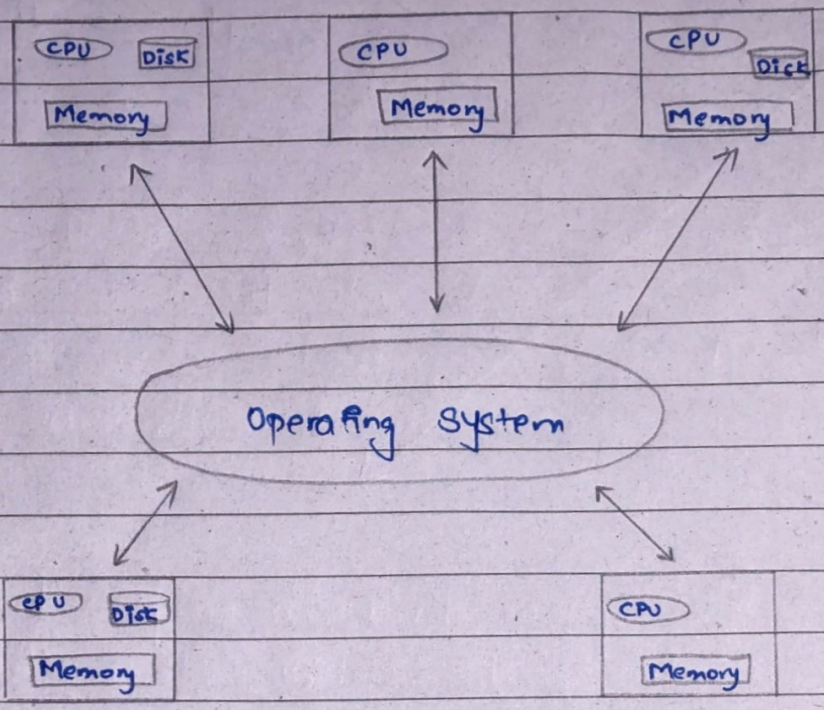


Fig: Distributed operating system

Advantage of Distributed OS

1) Load on host computer reduces
2) Delay in data processing reduces
3) Electronic mail increases the data exchange speed
4) Failure of one will not affect the other network communication.

Disadvantages of Distributed OS

1) Failure of main network will stop the entire communication
2) They are very expensive.
3) The language in distributed os is not properly defined yet.
4) Example of Distributed OS - LOTUS, etc.

Date
Page

8) compare Android OS and iOS with their characteristics features. Describe real time (RTOS) with its types.

	Android OS	IOS
1)	Offers a highly customizable user interface. Users can personalize their home screens, and twist various settings.	Follows a more uniform & controlled interface. customization options are limited compared to Android.
2)	Apps are available on the Google Playstore.	Apps are available on the Apple Appstore.
3)	Updates are rolled out by device manufactures & carriers, leading to a more fragmented distribution.	Updates are controlled by Apple & are generally available simultaneously for all supported devic
4)	Google Assistant is the default voice assistant.	Siri is in the built-in voice assistant.
5)	Allows the users to access the file system, providing more flexibility in managing files.	Has a more closed file system limiting direct access to the file structure.
6)	Has robc security measures, including app sandboxing & Google Play Protect.	Known for its strong-security features, including end-to-end encryption for iMessages, etc.
7)	supports more extensive multitasking, allowing floating apps, etc.	Has improved multitasking capabilitie with each update.

help from device drivers to handle all I/O devices. Device drivers encapsulate device-dependent code & implement a standard interface in such a way that code contains device-specific register reads/writes.

Device controller

Device controller works like an interface between a device & a device driver. I/O units (keyboard, mouse, printer, etc) typically consist of a mechanical component & an electronic component is called the device controller. There is always a device controller & a device driver for each device to communicate with the OS.

b) Message Passing

- Message Passing is a communication method used in OS & distributed systems, where the processes or components exchange information by sending & receiving messages. This communication allows different parts of a system to collaborate & share data. There are two main models of message passing:

1) Direct Message Passing

- Explicit communication - In direct message passing, processes explicitly send messages to each other.
- Synchronous or Asynchronous - It can be synchronous, where the sender is blocked until the receiver acknowledges the message, or asynchronous, where the sender continues execution without waiting for a response.

Date
Page

2) Indirect Message Passing

- Message Queues - In indirect message passing, processes communicate through a message queue or a mailbox.
- Asynchronous - Typically this method is asynchronous, where a process can deposit a message in a queue, & the other process can retrieve it later.

Advantage of Message Passing :

- Isolation - Message passing helps in achieving process isolation as processes communicate through well-defined interfaces.
- Modularity - components can be designed independently, & changes in one component do not necessarily affect others as long as the interface remains the same.

Message Passing in OS Ps Inter-Process communication (IPC)

- In OS, IPC mechanisms like pipes, sockets, & message queues are used for message passing between processes. Signals are a form of message passing where a process can send a signal to another process to notify it about a specific event to a request.

2019

1. What is an operating system? Discuss the evolution of operating system.

- An operating system (OS) is a software component that acts as an intermediary between computer hardware & the computer user. Its primary functions include managing hardware resources, providing services to applications, & facilitating communication between the hardware & software components of a computer system. The evolution of OS can be traced through several key stages:

1) No operating system (1940s)

- Early computers had no operating systems. Users interacted directly with the hardware, writing machine-level programs.
- Programs were loaded onto the computer using punched cards or other manual methods.

2) Batch Processing systems (1950s - 1960s)

- Batch processing system introduced the concept of running multiple jobs in sequence without user interaction.
- OS, like IBM's OS/360, managed the execution of batches of jobs, improving efficiency & resource utilization.

3) Time-sharing systems (1960s-1970s)

- Time sharing system allowed multiple users to interact with the computers simultaneously.
- OSs, like Multics & later Unix, provided each user with a virtual machine, giving the illusion of exclusive access to a system.

4) Personal computer (1980s)

- The advent of personal computers saw the rise of OS like

MS-DOS, Apple DOS, & early versions of Windows.

- Graphical User Interfaces (GUIs) became popular, making computers more accesible to non-technical users.

5) Networked Systems (1980s-1990s)

- The growth of networking led to the development of networked OS.
- Novell Netware & Microsoft Windows NT provided features for managing network resources.

6) Client-Server Architecture (1990s)

- Client-server computing became prevalent, with OS like Windows 95/98/ME, Windows NT, & Novell Netware.
- The internet became widely accesible, leading to the development of web-based applications.

7) Mobile operating system (2000s-Present)

- The rise of smartphones brought mobile OS like iOS, Android, and Windows Mobile.
- These OSs are optimized for touch interfaces & provide app-based ecosystems.

8) Modern operating systems (2000-Present)

- Contemporary operating systems include Microsoft Windows, macOS, Linux distributions, & Unix variants.
- Virtualization, & containerization technologies, such as VMware & Docker, have gained popularity.

9) Real-Time OS (RTOS)

- RTOS, like Free RTOS & VxWorks, are designed for applications with strict timing requirements, such as embedded systems & robotics.

10) Open source operating system

- Linux, an open-source Unix-like OS, gain widespread adopting in server environments & as the basis for Android.

The evolution in OS reflects advancements in hardware capabilities, changes in user needs, and innovations in software design.

2) What is critical section? Why mutual exclusion is required? How mutual exclusion can be achieved? Describe any one of them.

- Critical Section is the part of a program where shared variables and resources are accessed by multiple process. Mutual exclusion is required because

1) To control access to shared resources, maintaining program logic

2) To avoid deadlock by controlling access to resources

3) To ensures proper synchronization with condition variables

4) To prevent data inconsistency i.e. ensure shared data is accessed by only one process or thread at a time to avoid conflict.

5) To eliminate race condition by allowing only one thread to execute a critical section.

6) To enforces a consistent order of execution in a multi-threa environment.

Mutual Exclusion can be achieved by
1) Test & set lock
2) Strict Alternation method
3) Peterson's Method
4) Semaphore

Peterson's Method
- It is a software mechanism implemented at user mode. It is busy waiting solution & can be used for 2 process.

```
#define N2
#define True 1
#define false 0
int interest [N]=false;
int True ;
void entrysection (int process)
    { int other ;
      other = 1- process ;
      interest[process] = True;
      turn = process;
     while (interest (other ] == True && turn == process );
   }
void exit_section(int process)
    {interest [process] = false;
    }
```

3) Define process & its different states. What are the various operation on semaphore. How message passing is useful on IPC?

- A process is an instance of a program in execution. A program by itself is not a process, a program is a passive entity such as file containing a list of instruction stored on disks. A process is an active entity with a program counter specifying the next instruction to execute & a set of associated resources.

The different states are:-

i) New
ii) Running
iii) Waiting
iv) Ready
v) Terminated

The various operation on semaphore are:-

a) Initialization- A semaphore is created & initialized with an initial value. Initial value represents the no. of units available for allocation.

b) Wait (P/Down) operation

- Decrementing the value of semaphore by one. If the resulting value is non-negative the thread or process continues execution. If resulting value is negative, process is blocked. until next semaphore value become non-negative.

c) Signal (V/Up) operation

- Increment value of semaphore by 1. If there are any processes or threads waiting due to 'wait' operation, one of them is allowed to continue execution.

d) Blocking wait
- If a wait operation would result negative, semaphore value, process is blocked until value is non-negative.

e) Non-blocking wait (try wait)
Similar to wait but instead blocking, process continues execution even if semaphore value is negative.

f) Timed wait
- Similar to wait operation, but with additional time limit.

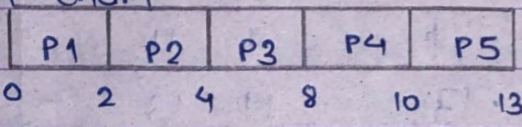
Message passing is useful for IPC because
a) Process can communicate through message without exposing internal details.
b) Enables processes to coordinate activities by exchanging messages.
c) Provides fault isolation, as the failure of one process doesn't necessarily impact others.

4. Compute average waiting time & average turn around times using FCFS, SJF, Priority (lowest no. represents highest priority) & Round-Robin (quantum=30s) scheduling algorithm for the following set of processes. Assume that all processes have arrived at time 0 in the order P1, P2, P3, P4 & P5.

Process	BT(ms)	Priority
P1	2	1
P2	2	1
P3	4	3
P4	2	4
P5	3	2

Arrival time = 0

1) FCFS Gantt chart



waiting time for P1 = 0

waiting time for P2 = 2

P3 = 4

P4 = 8

P5 = 10

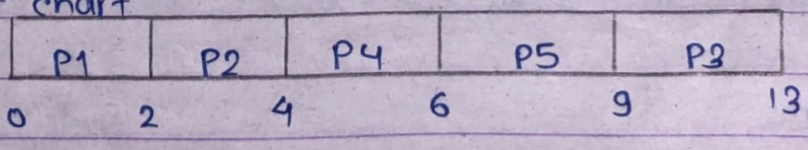
A.W.T = (0+2+4+8+10) /5

= 4.8

A.T.T = (2+4+8+10+13) /5

= 7.40 ms

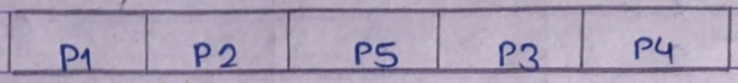
2) SJF, Gantt chart



$A.W.T = \frac{0+2+4+6+9}{5} = 4.2 ms$

$A.T.T = \frac{2+4+6+9+13}{5} = 6.8 ms$

3) Priority → Gantt chart



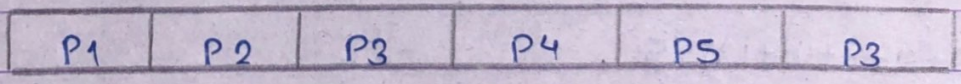
Process	BT	AT	C.T	TAT = CT - AT	WT = TAT - BT
P1	2	0	2	2	0
P2	2	0	4	4	2
P3	4	0	11	11	7
P4	2	0	13	13	11
P5	3	0	7	7	4

A.W.T = (0+2+7+11+4) / 5 = 4.80 ms

A.T.T = (2+4+11+13+7) / 5 = 7.40 ms

4) Round - Robin (Q.T) = 3ms

Gantt chart is



A.W.T = (0+2+(4-0)+(10-6)+6+8) / 5

= 4.8ms

A.T.T = (2+4+12+8+10) / 5

= 7.2 ms

5. Define the term Deadlock. What are the necessary conditions for a deadlock? Explain the deadlock avoidance algorithm.

→ A deadlock is a situation where a set of processes are blocked because each processes is holding some resources & waiting for another resources acquired by other processes.
The necessary conditions for a deadlock are:-

1) Mutual exclusion
- Two or more resources are non-shareable (only one process can use at a time)

2) Hold & wait
- A process must be holding at least one resource & waiting for at least one resource that is currently being held by some other processes.

3) No preemptive
- Resources cannot be preempted i.e. once a process is holding a resources then that resources cannot be taken from a process unless the process releases the resources.

4) Circular wait - A set $\{P_0, P_1, P_2, \ldots P_n\}$ of waiting processes must exist.

such that
→ P_0 is waiting for a resources held by P_1.
→ P_1 is waiting for a resources held by P_2.
→ P_2 is waiting for a resources held by P_3.
→ P_{n-1} is waiting for a resources held by P_n.
→ P_n is waiting for a resources held by P_0.

Deadlock avoidance

- Deadlock avoidance employs the most famous deadlock avoidance algorithm that is Banker's algorithm.

Banker's Algorithm is a resource allocation & deadlock avoidance algorithm develop by Dijkstra that test for the safety by simulating the allocation of predetermine maximum possible amount of resources. So, this algorithm can be used in banking system to ensure that the bank never allocates all its available cash such that it can no longer satisfy the need of entire customer. This algorithm is applicable to a system with multiple instance of resources of each type. When a new process enter into a system it must declare maximum no. of instances of each resource type that it may need. This algorithm prevents deadlock by denying the request if it determine that accepting a request put the system in unsafe state.

Deadlock avoidance algorithm (Banker's algorithm) contain safety algorithm and resource request allocation which removes the deadlock or avoid deadlock in the system.

6. Consider the following snapshot of a system.

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

Answer the following questions with Banker's algorithm:-

a) What is the content of the matrix need?
b) Is the system in a safe state?
c) If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

→ Soln:-

We know,

i) The contents of need matrix is

Need [i,j] = Max [i,j] - Allocation [i,j]

	Need matrix			
	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

ii) Applying the safety algorithm

For Pi,
if Need < Available
then Pi is in safe state.

For i=0, P0,
Need0 = (0,0,0,0)
Available = (1,5,2,0)

Need0 ≤ Available i.e. (0,0,0,0) ≤ (1,5,2,0) (True)

So, P0 will be in safe sequence.

Now, Available = Available + Allocation
= (1,5,2,0) + (0,0,1,2)
= (1,5,3,2)

For i=1, P1,
Need1 = (0,7,5,0)
Available = (1,5,3,2)

Need1 ≤ Available i.e. (0,7,5,0) ≤ (1,5,3,2) (false)

So, P1 must be waiting.

For i=2, P2,

Need 2 = (1,0,0,2)

Available = (1,5,3,2)

Need 2 ≤ Available i.e. (1,0,0,2) ≤ (1,5,3,2) (true)

So, P2 will be in safe sequence.

Now, Available = Available + Allocation

= (1,5,3,2) + (1,3,5,4)

= (2,8,8,6)

For i=3, P3,

Need 3 = (0,0,2,0)

Available = (2,8,8,6)

Need 3 ≤ Available i.e. (0,0,2,0) ≤ (2,8,8,6) (True)

So, P3 will be in safe sequence.

Now, Available = (2,8,8,6) + (0,6,3,2)

= (2,14,11,8)

For i=4, P4,

Need 4 = (0,6,4,2)

Available = (2,14,11,8)

Need 4 ≤ Available i.e. (0,6,4,2) ≤ (2,14,11,8) (True)

So, P4 will be in safe sequence.

Available = (2,14,11,8) + (0,0,14)
= (2,14,12,12)

For i = 1, P1
Need 1 = (0,7,5,0)
Available = (2,14,12,12)

Need 2 ≤ Available i.e. (0,7,5,0) ≤ (2,14,12,12) (True)

So, P2 is in safe sequence.

Available = (2,14,12,12) + (1,0,0,0)
= (3,14,14,12)

Hence, the safe sequence is

<P0,P2,P3,P4,P1> //

iii) Since the process P1 request some additional resource request (0,4,2,0).
To decides whether this request can granted or not.

we first check,
Requesti ≤ Available
i.e. (0,4,2,0) ≤ (1,5,2,0) (True)

So, the request of P1 may be granted.
To confirm that this request is granted we check the new state of system by applying safety algorithm that

the system is safe or not.

To define the state of new system because of arrival of P1, we follow the resource request algorithm as:

	Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	0	0				
P1	1	4	2	0	0	3	3	0	1	1	0	0
P2	1	3	5	4	1	0	0	2				
P3	0	6	3	2	0	0	2	0				
P4	0	0	1	4	0	6	4	2				

we must determine whether the state is safe or not.
Applying safety algorithm.

For Pi,
if Need $\leq$ Available
Pi is in safe state.

For i = 0, P0,
Need0 = (0,0,0,0)
Available = (1,1,0,0)
Need0 $\leq$ Available ie. (0,0,0,0) $\leq$ (1,1,0,0) (True)

So, P0 will be in safe state.

Available = (1,1,0,0) + (0,0,1,2)
= (1,1,1,2)

For i = 1, P1,

Need 1 = (0, 3, 3, 0)

Available = (1, 1, 1, 2)

Need1 ≤ Available i.e. (0, 3, 3, 0) ≤ (1, 1, 1, 2) (false)

So, P1 must be waiting.

For i = 2, P2,

Need 2 = (1, 0, 0, 2)

Available = (1, 1, 1, 2)

Need 2 ≤ Available i.e. (1, 0, 0, 2) ≤ (1, 1, 1, 2) (True)

So, P2 will be in safe sequence.

Available = (1, 1, 1, 2) + (1, 3, 5, 4)

= (2, 4, 6, 6)

For i = 3, P3,

Need 3 = (0, 0, 2, 0)

Available = (2, 4, 6, 6)

Need 3 ≤ Available i.e. (0, 0, 2, 0) ≤ (2, 4, 6, 6) (True)

So, P3 will be in safe sequence.

Available = (2, 4, 6, 6) + (0, 6, 3, 2)

= (2, 10, 9, 8)

For i=4, P4,

Need4 = (0,6,4,2)

Available = (2,10,9,8)

Need4 ≤ Available i.e. (0,6,4,2) ≤ (2,10,9,8)

So, P4 will be in safe sequence.

Available = (2,10,9,8) + (0,0,1,4)

= (2,10,10,12)

For i=1, P1

Need1 = (0,3,3,0)

Available = (2,10,10,12)

Need1 ≤ Available i.e. (0,3,3,0) ≤ (2,10,10,12) (True)

So, P1 will be in safe sequence.

The safe sequence is

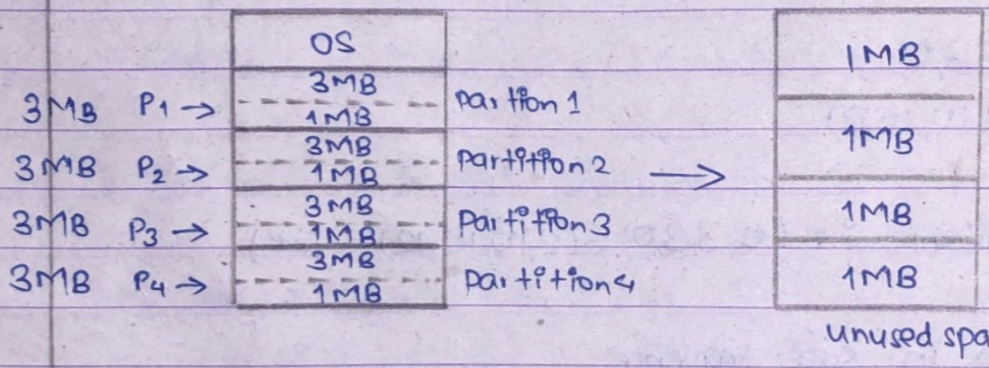
<P0, P2, P3, P4, P1>

Hence, we can immediately grant the grant request of P1.

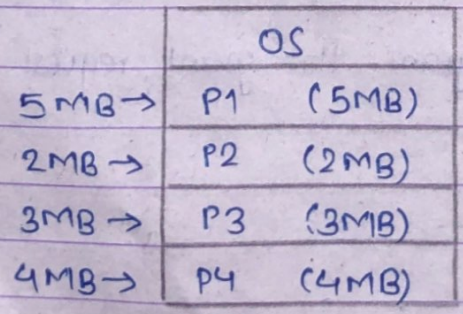
7. Explain the static & dynamic memory allocation techniques with examples. Use FIFO page replacement algorithm in the following reference string having 4 frames & calculate the no. of page faults.

0,1,2,3,0,1,2,3,0,1,2,3,4,5,6,7

- Static memory allocation has main memory divided into several fixed partition. They can be of same or different sizes. Each partition can hold single processes. In fixed position, degree of multiprogramming is fixed position & v. less due to fact that size cannot be changed.



Dynamic memory allocation tries to overcome problem of static partitioning. Hence, partition size is declared initially. Each process occupies only as much memory they require.



Dynamic Partitioning

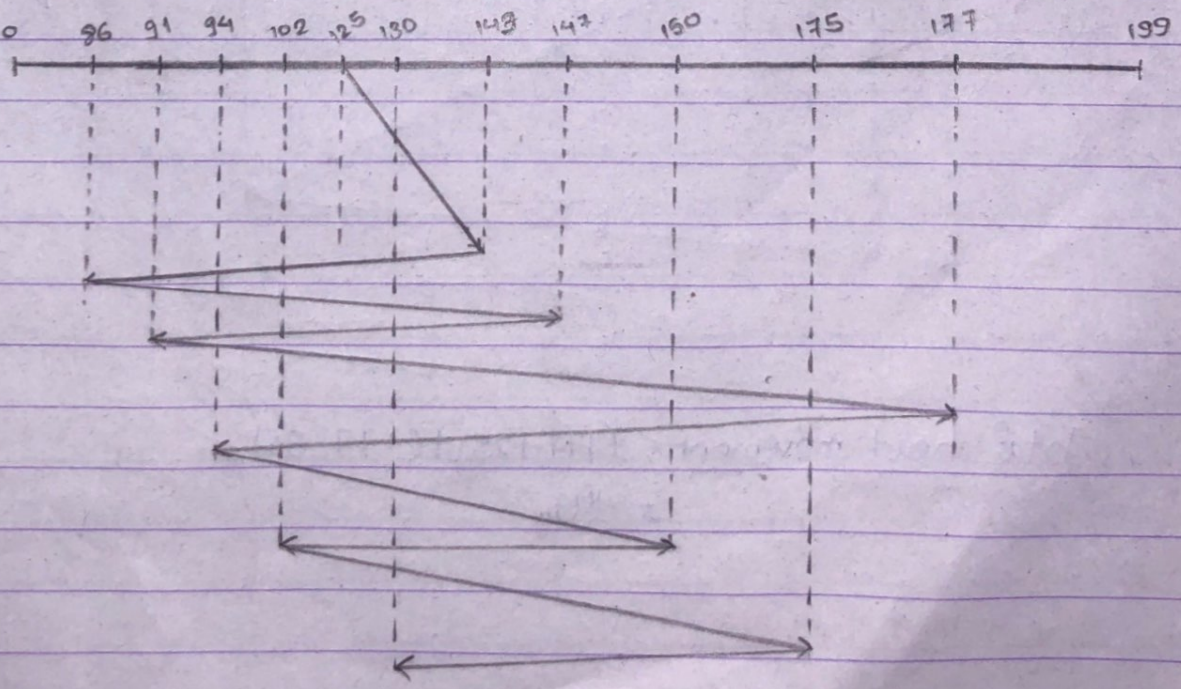
Using FIFO

0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	ø	4	4	4	4
	1	1	1	1	1	1	1	1	1	1	1	✗	5	5	5
		2	2	2	2	2	2	2	2	2	2	2	✗	6	6
			3	3	3	3	3	3	3	3	3	3	3	✗	7
*	*	*	*	H	H	H	H	H	H	H	H	*	*	*	*

Total no. of page faults = 8 //

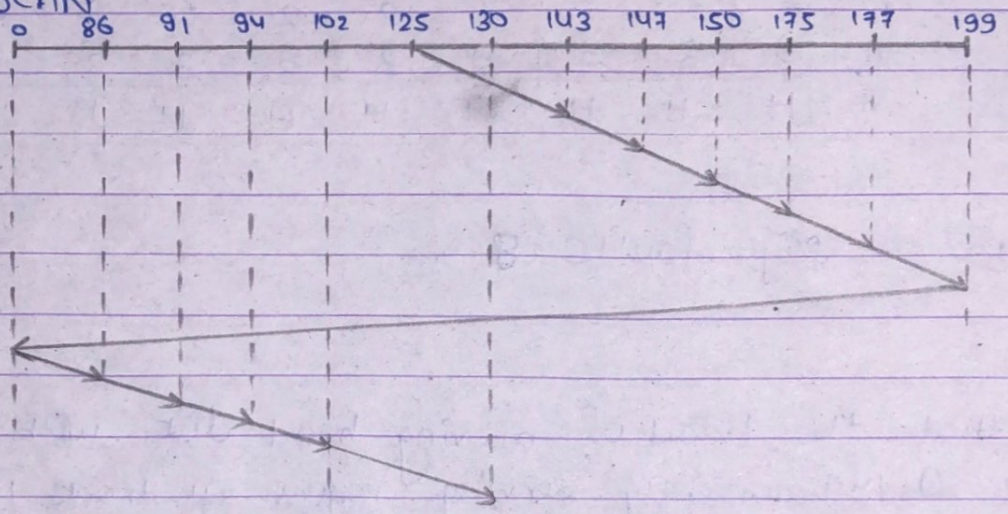
8. Suppose that the head of moving head disk with 200 tracks numbered 0-199, currently serving request at track 143, & just finished request at track 125. The queue of request is in FIFO order 86, 147, 91, 177, 94, 150 & 102, 175, 130. What is total head movement using FCFS, C-SCAN, SSTF & LOOK.

1) FCFS



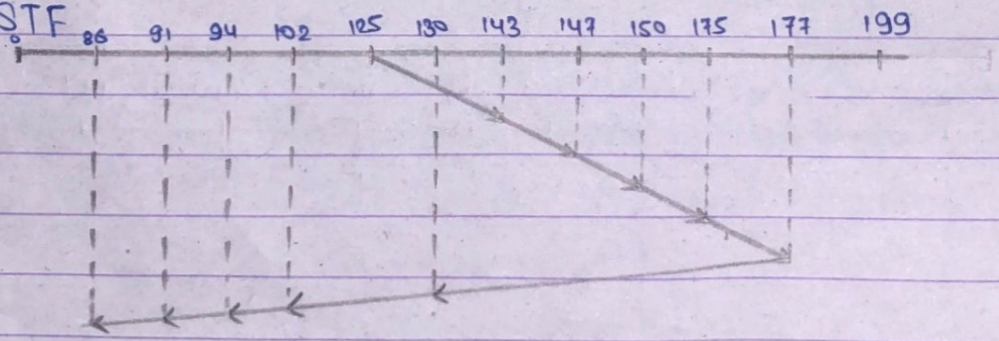
Total head movement = (143-125) + (143-86) + (147-86) + (147-91) +
(177-91) + (177-94) + (150-94) + (150-102) +
(175-102) + (175-130)
= 427 ||

2) C-SCAN



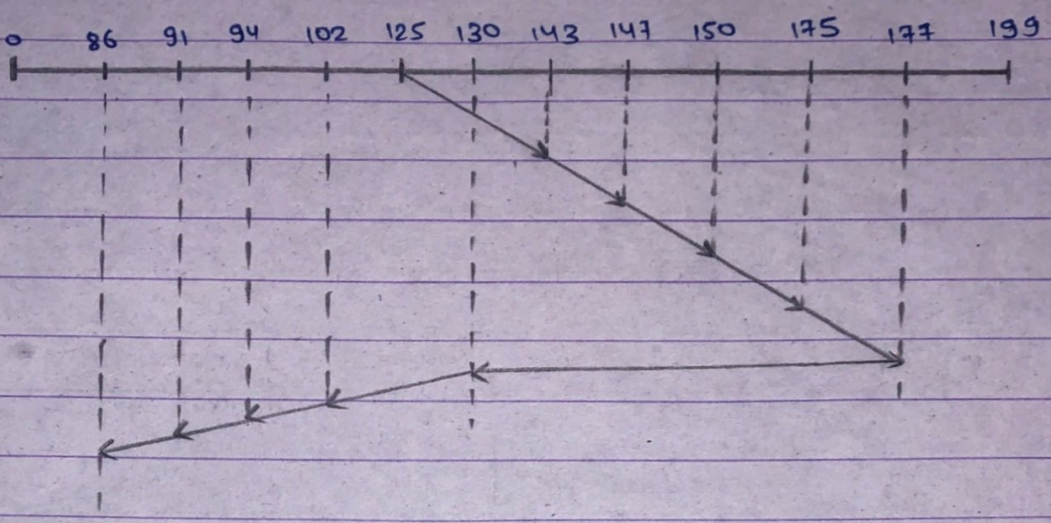
Total head movement = (199-125) + (199-0) + (130-0)
= 403 ||

3) SSTF



Total head movement = (177-125) + (177-86)
= 149 ||

4) LOOK



Total head movement = (177-125) + (177-86)

= 149