PAGE No.
DATE / /

Q) What is operating system?

Ans:

- An operating system acts as an interface between the user and hardware of the computer and also control the execution of application programs

- it is also called as resource manager

eg: windows, linux, macOS, Android, iOS

QB

PAGE No.
DATE / /

Q. Diff b/w monolithic & microkernel

Ans:

No.	Microkernel	Monolithic kernel.
(1)	in microkernel, user services and kernel services are kept in separate address space	in monolithic kernel, both user services and kernel services are kept in the same address space
(2)	OS is complex to design	OS is easy to design and implement.
(3)	microkernel are smaller in size	Monolithic kernel is larger than microkernel.
(4)	easier to add new functionalities	difficult to add new functionalities
(5)	to design microkernel, more code is required	less code when compared to microkernel.
(6)	execution speed is low	execution speed is high
(7)	it is easy to extend microkernel	it is not easy to extend monolithic kernel.
(8)	debugging is simple	debugging is difficult
(9)	It is simple to maintain	extra time and resources are needed for maintances
(10)	eg: mac os.	eg: Microsoft window 95

PAGE No.
DATE / /

Ans: Diff b/w preemitive & non-preemitive scheduling Algo

No.	Preemitive	Non-Preemitive.
(1)	The CPU is allocated to the processes for a certain amount of time	The CPU is allocated to the process till it ends its the fewer execution or switches to waiting state.
(2)	The executing process here is interrupted in the middle of execution	The executing process here is not interrupted in middle of execution
(3)	preemitive scheduling is more complex	~~It does not~~ Simple, but very inefficient
(4)	It also affects the design of the operating system kernel	it does not affect design of the OS kernel.
(5)	This is cost associative	This is not cost associative
(6)	preemitive scheduling incurs a cost associated with associative shared data	Non-preemitive scheduling does not increases the cost
(7)	eg: Round Robin Method	eg: First come first serve method.

PAGE No.
DATE / /

Q) objective of operating system

Ans:

Three objectives of O.S

- Convenience
- efficiency
- ability to evolve

(1) Convenience:

Computer system can be conveniently used due to operating system

(2) efficiency:

Computer system comprises of many resources. All these resources are utilized by user's application in efficient manner due to operating system

(3) Ability to evolve:

The design of the operating system permits the efficient developments, testing, it is also support for flexibility by allowing for addition of new system functions without interfering with ~~service~~. service.

Function of OS

Ans:

An operating system performs various functions to manage computer hardware and software resources efficiently. The function includes

Function of Operating System

- Process management
- Memory
- File
- Device
- Protection and security
- User interface or command interpreter (CLI)
- Booting the computer
- performance basic computer task.

Process Managements

- Creates, schedules and terminates processes
- Handles multitasking and process synchronization
- Use cpu scheduling algo like Round Robin, FCFS etc

Memory Managements

- Allocate and deallocates memory to processes
- implementing paging, segmentation and virtual memory for efficient use
- Prevents memory leaks and manage cache memory

PAGE No.
DATE / /

(3) File system management

- organize, stores, retrieves, and protects files
- implements file directories, permissions and access control.
- manage file operation likes read, write, delete, rname.

(4) Device management:

- devices drivers opened, closed and written by OS
- keeps eye on device driver. communicate, control and monitor the devices driver.
- Handles interrupts and error detection in devices

(5) protection & security

- provides authentication, encryption and access control.
- prevents unauthorized access and protects against malware
- implements fire walls

(6) User Interface (UI) managements.

- provides Graphical User interface (GUI) and command line interface (CLI)
- Allows User to interact with system resources

PAGE No.
DATE / /

(7) Booting the computer.

- The process of starting or restarting the computer is known as booting

- if computer is switched off completely and if turned on then is called cold booting
- A warm booting is the process of using the operating system to restart the computer

(8) Performs basic computer tasks

- The management of various peripheral devices such as the mouse, keyboard, and printers is carried out by operating system

- Today most of the operating system are plug and play

- file management

- I/O operation

PAGE No.
DATE / /

Q. What is system call & it's type

Ans:

- The interface b/w OS and user programs is defined by the set of system calls that the operating system offers.

- System calls is the call for the operating system to perform some task on behalf of the user's program

- Therefore system calls make up the interface b/w processes and the operating system.

- The system calls, ~~the~~ are function used in the kernel itself, from programmers POV, the system call is a normal function call.

- Types of system calls

  - → Process control
  - → device manipulation
  - → communications
  - → file manipulation
  - → information maintanence.

(1) Process control: These systems call manage processes by allowing a program to control its own execution and interaction with other processes.

fork(), exit(), wait(), exec(), kill()

(2) Device manipulation: These system calls interacts with hardware device such as printer, disk, drives etc.

read(), write(), open() / close()

(3) Communications: These system calls facillitate communication b/w processes, such as sending and receiving message.

pipe(), shmget(), mknod(), transfer status info, send, recieve

(4) File manipulation: These system calls handle file operations such as creating, writing and deleting files

open(), read(), write(), close(), delete(), rename()

PAGE No.
DATE / /

(5) information maintenance : These system calls deal with system information and the management of environment variables.

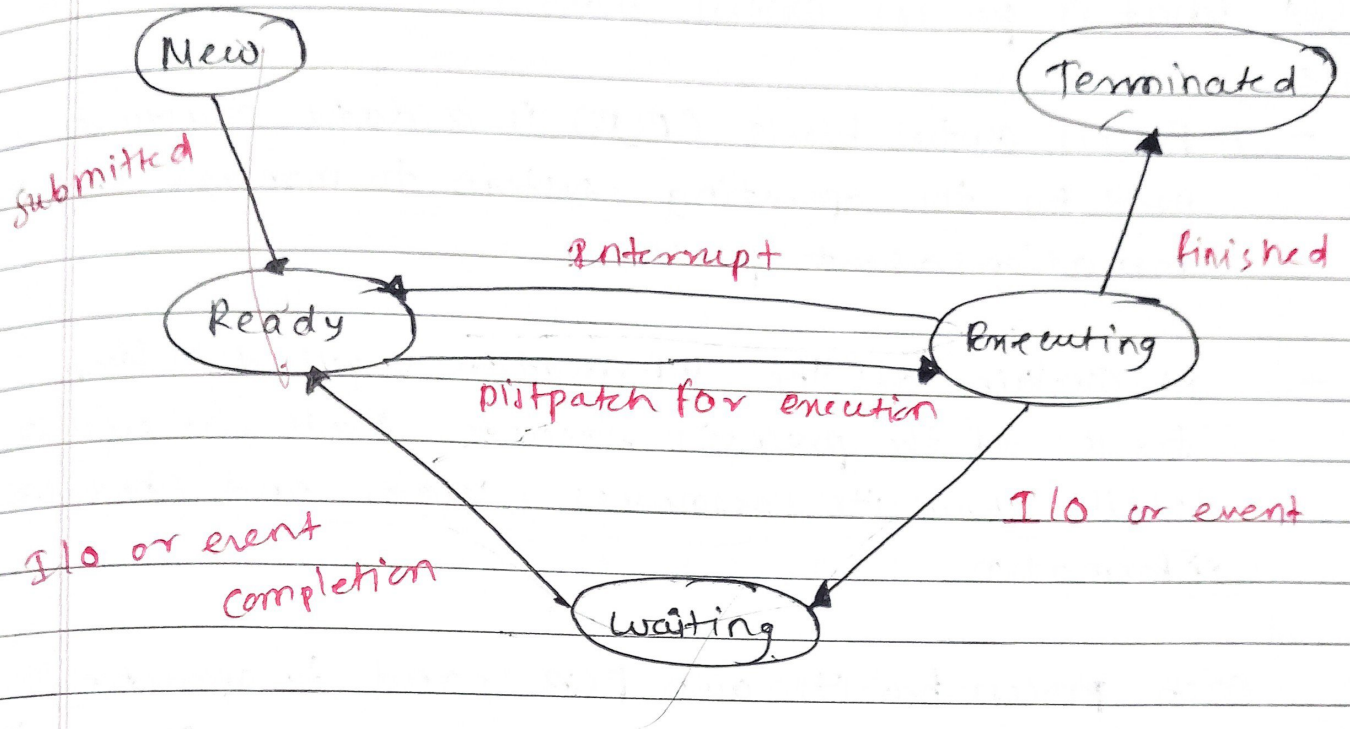
get time or date, set time or date, get process, set process file,

Q. What is process and describe it's states

Ans:

- A program or application under execution is called process
- A process includes the execution context, to complete execution, program needs many resources and complete for it. Now it become process
- During execution process changes its state
- process state shows the current activity of the process
- each process remains in one of these ~~five~~ states

5 States



(1) New state: The new process being created

(2) Ready state: A process is ready to run but it is waiting for CPU being assigned to it

(3) executing state: ~~A process is said to be run but it is waiting for CPU being~~ o

A process is said to be in running state if currently CPU is allocated to it and it is executing

(4) Waiting (blocked) state: A process can't continue the execution becoz it is waiting for event to happen such as I/O completion. Process is able to run when some external event happens.

(5) Terminated state: The process has completed execution

or explain ~~PCB~~ Role of PCB

PAGE No.
DATE / /

What is process control block

Ans:

- A Process control block (PCB) is a data structure used by the operating system to manage and store information about process

- It contains all the information required to keep track of the process's execute state, as well as details about its resources, usage and scheduling information.

each process has its own PCB, and the operating system uses it to manage the process across its life cycle.

Pointer	current State
Process ~~ID~~ ID Priority Program counter Registers Accounting memory allocation event information list of open files	
⋮	

PAGE No.
DATE / /

* pointer: This fields points to other's process's PCB
the scheduling list is maintained by pointer

* current state: currently processes can be in any of the state from new, ready, executing, waiting etc as described above.

* process ID: identification number of the process.
OS assign this ~~pr~~ number to process to distinguish it from other processes.

* priority: diff process can have different priority

* program counter: The address of the next instruction to be executed for the process.
It helps resume the process when the CPU is reassigned to it.

* Registers: It include general purpose registers, index registers, stack pointers and accumulator etc.

* Accounting: information for calculating the process's priority relative to other processes.

* memory allocations: This info may include the value base and limit register. It include paging, segmentation etc.

* event information: For a process in the blocked state this field contains information concerning events for ~~the~~ which the process is waiting

* list of open files: files opened by the process.

PAGE No.
DATE / /

Define threads and explain concepts of Multi-threading. Type of multithreading

Ans:

- A thread is a single sequence stream within a process

- threads are also called as lightweighted processes as it ~~process~~ posses. some of properties of process

- each threads belongs to exactly one process.

- Multithreading : is a programming technique where multiple threads run within a single process.

- These threads run in parallel, improving the application performances

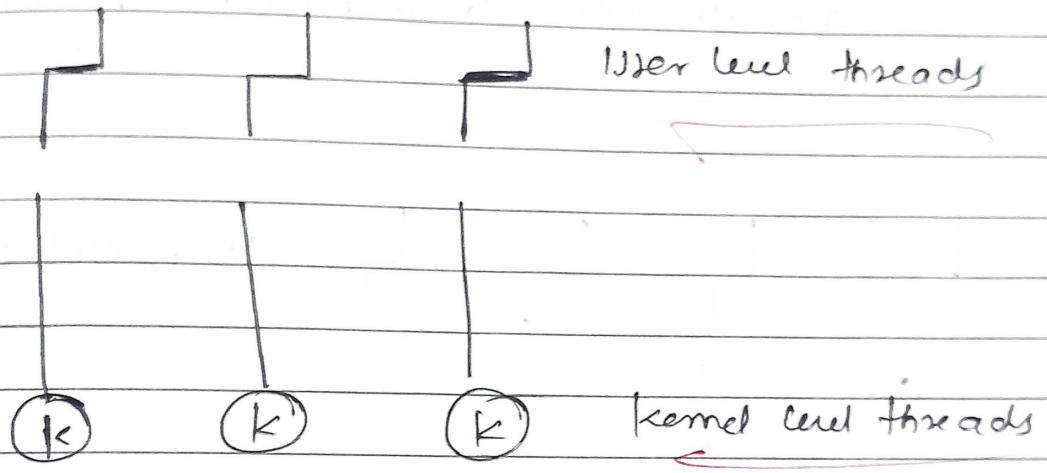
- considering the advantage of user level and kernel level threads, a hybrid threading model using both types of threads can be implemented

☆ Types of multi-threading model

- → one to one
- → many to one
- → Many to Many

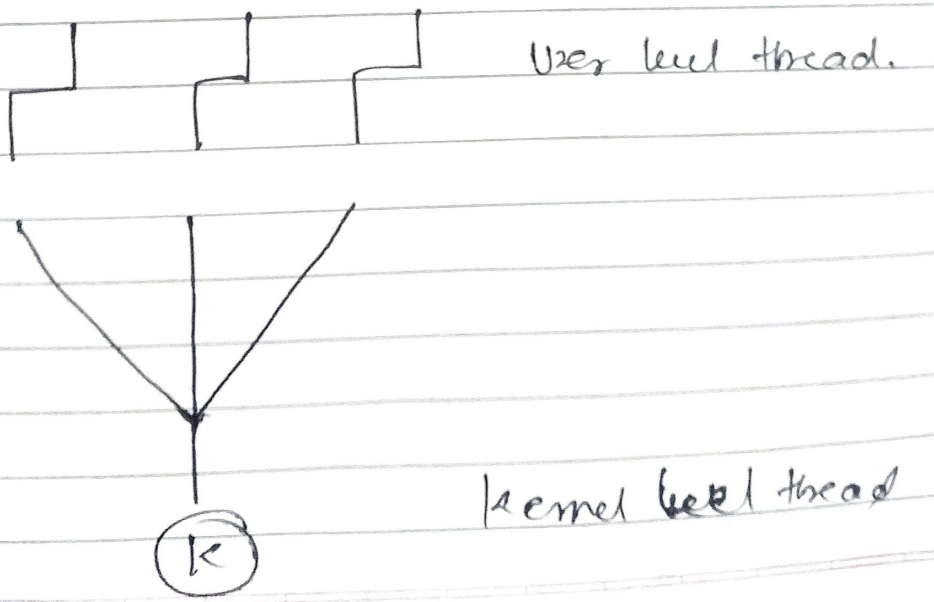
(1) one to one: in the one to one model, each user level thread is mapped to a kernel level thread

example: if you create 10 threads in your application, the kernel will create 10 corresponding threads



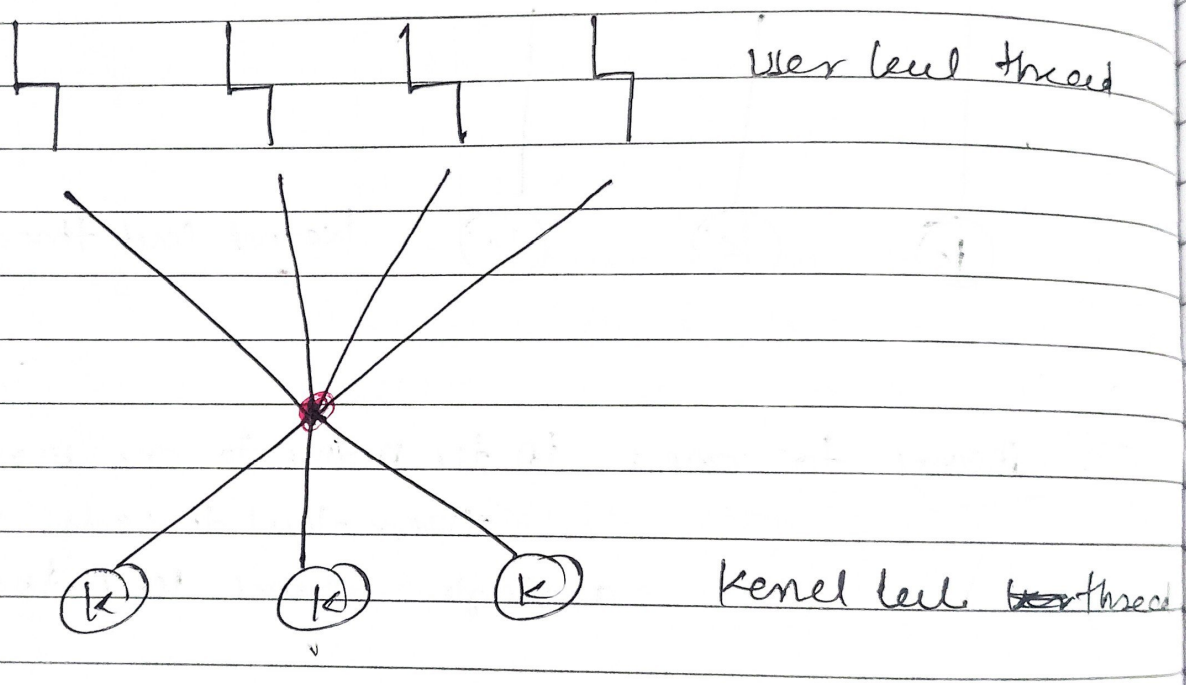
(2) many to one: in the many to one model, many user-level threads are mapped to a single kernel level threads

example: if you have 10 user threads, they all are scheduled as one kernel thread.



(3) Many to many: In the many to many model, many user-level threads are mapped to many kernel level threads

Example: If there are 100 user threads, they might be mapped 20 kernel threads, allowing more threads to manage by fewer kernel threads



Numerical

PAGE No.
DATE / /

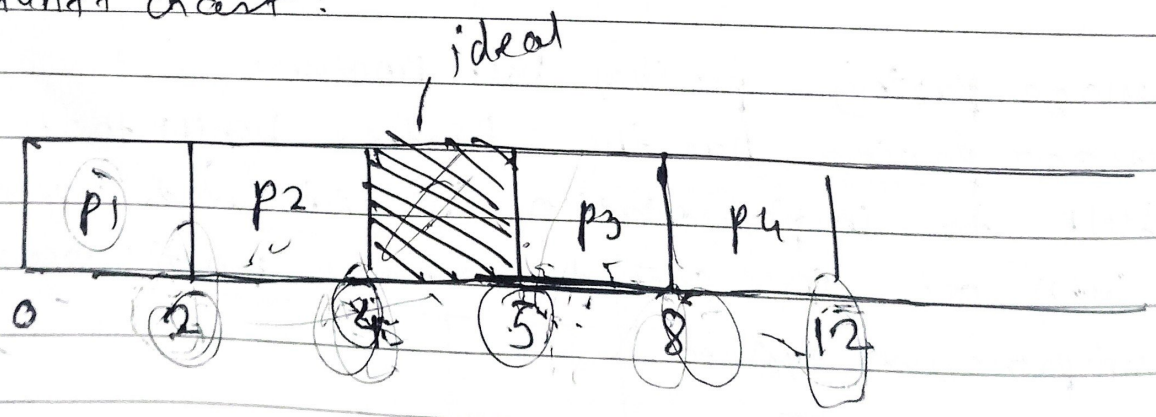
★ FCFS (First come, First serves)

Process No:	Arrival Time	Burst time (execution time)	Completion time	TAT	WT
P1	0	2	2	2	0
P2	1	2	4	3	1
P3	5	3	8	3	0
P4	6	4	12	6	2

Criteria: Arrival time

Mode: Non preemptive.

Gantt chart:



(TAT): Turn Around time : Complete Time - Arrival time
CT - AT

(WT) Waiting Time : TAT - BT
Turn Around time - Burst Time

(RT) Response Time : Gantt chart left - Arrival time

Q)

Process	Arrival Time	Burst time	completion time	TAT	WT
P1	2	6	17	15	9
P2	5	3	24	19	16
P3	1	8	11	10	2
P4	0	3	3	3	0
P5	4	4	21	17	13

avg waiting time = $\frac{9+16+2+0+13}{5} = \frac{40}{5} = 8$

P4	P3	P1	P5	P2

0 3 11 17 21 24

Criteria → Burst time

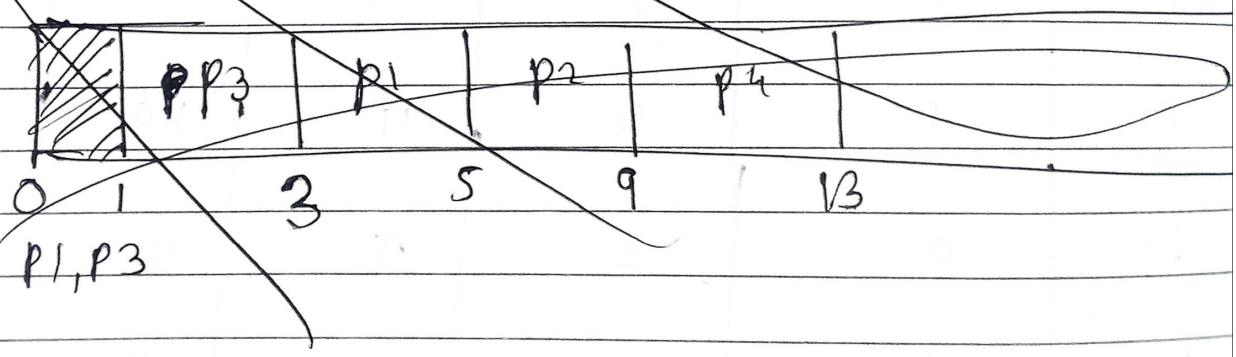
Mode → Non Preemptive

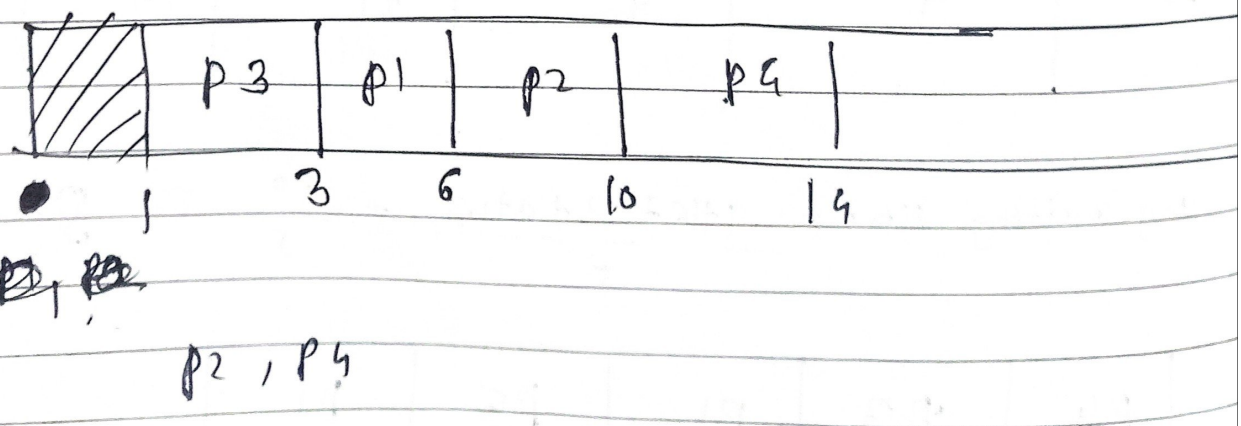
PAGE No.
DATE / /

☆ SJF (Shortest Job First)

process	Arrival time	Burst time	completion time	TAT	WT
P1	1	3	6	5	2
P2	2	4	10	8	4
P3	1	3	3	2	0
P4	4	4	14	10	6

Gantt chart





(3)

process	A.T	B.T	C.T	TAT	WT
P1	1	3	6	5	2
P2	2	4	10	8	4
P3	1	2	3	2	0
P4	4	4	14	10	6

	P3	P1	P2	P4
0	3	6	10	14

criteria → Burst time
mode → preventive → RT find karna pdega

PAGE No.
DATE / /

* SRTF (Shortest Remaing Time First)

process	A.T	B.T	C.T	TAT	WT	RT
P1	0	5 ~~5~~ ~~4~~ ~~3~~ ~~2~~	9	9	4	0
P2	1	3 ~~3~~ ~~2~~ ~~1~~	4	3	0	0
P3	2	4 ~~4~~	13	11	7	7
P4	4	1 ~~1~~	5	1	0	0

P1	P2	P3	P2	P4	P1	P1	P1	P1	P3

0 1 2 3 4 5 6 7 8 9

time → 0

RT = { (P1 first time − AT)

Criteria : Time quantum

mode : Preemitive

2 process

Round Robing

process	A.T	B.T	C.T	TAT	WT	RT
P1	0	~~8~~ ~~3~~ ~~1~~	12	12	7	0
P2	1	~~4~~ ~~2~~ 0	11	10	6	1
P3	2	~~2~~ 0	6	4	2	2
P4	4	~~4~~	9	5	4	4

Given TQ. = 2

Sequence of process in Ready queue

Ready queue

~~P1~~	~~P2~~	~~P3~~	~~P1~~	~~P4~~	~~P2~~	~~P1~~

0

Running queue

P1	P2	P3	P1	P4	P2	P1

0 2 4 6 8 9 11 12

(gantt chart)

context switch

Criteria: Priority

Mode: Preemitive

Priority Scheduling

Priority	Process	A.T	B.T	C.T	TAT	WT
10	P1	0	~~8~~ ~~4~~ 0	12	12	7
20	P2	1	~~4~~ ~~3~~ ~~2~~ 1	8	7	3
30	P3	2	~~2~~ ~~1~~ 0	4	2	0
40	P4	4	~~1~~ 0	5	1	0

P1	P2	P3	P3	P4	P2	P2	P2	P1

0 1 2 3 4 5 6 7 8 12

higher the no.

higher the priority

OS UT-2

PAGE No.
DATE / /

Q) explain principle of concurrency.

Ans:

Concurrency means that multiple process or tasks can be in progress at a same times. The OS manage them so they appear to run together, even if there's only one CPU.

In a single processor multiprogramming system, processes are not executed concurrently, In order to get the appearance of concurrent execution, a fixed timeslot is allocated to each process.

After utilization of this time slot, CPU gets allocated to other processes. such switching of CPU is called as context switching.

Advantage: - Better CPU utilization
- Faster program execution

eg: A computer is scanning the viruses while you are browsing the internet both tasks run ~~simuta~~ concurrently.

Q) What is process synchronization.

Ans:

Processes synchronization is a technique used in OS to ensure that multiple processes or threads can execute concurrently while safely sharing resources without leading to data inconsistent or race condition.

- Why it's needed?
- to avoid race conditions
- to maintain data consistency
- to prevent dead lock and confusion in output.

example: Two threads updating a bank account balance synchronization make sure that the final amount is correct no matter which runs first.

What is external fragmentation

Ans:

external fragmentation happens when free memory is split into pieces scattered throughout the system, making it hard to find large enough block of memory for new memory, even if the total free memory is enough.

example:

Imagine a system with 1 GB free memory.

but it's split into blocks of 100 MB, 50 MB and 200 MB. even though there's enough free space memory (1 GB) there's no single block large enough to fit 300 MB processes, causing the external fragmentation.

explain TSL (Test and Set Lock)

Ans:

TSL (Test and Set Lock) is an atomic operation that checks if a resource is locked (value = 1) or available (value = 0). if it's available, it set the lock to 1 indicating that the resource is now locked, if it's already locked, the process waits and retries.

How it's work?

- Test: check the value of memory location (the lock)
- Set: if the lock is not set (value is 0), set it to 1 to indicate the resource is now locked.

if the value was already 1 (lock) then the process know that the resource is busy and has to retry.
wait to

This operation is atomic. meaning it is completed in one single step without any interruption, ensuring proper synchronization b/w multiple processes.

Example: Imagine a single bathroom in a shared office.

① Person A. Checks if the bathroom is occupied (lock is 1) if it's free (lock 0) they lock it (set to 1) and use it.

② Person B check bathroom. If it's locked (1) they wait until person A is done.

③ once person A leaves, they unlock it set (0) and person B can use it.

PAGE No.
DATE / /

Q) What is deadlock & sol^n of deadlock?

Ans:

Deadlock is a situation in computer system where two or more processes are ~~stud~~ stuck in a way that they cannot proceed because each processes are waiting for a resources that the ~~hold~~ other holds.

This lead to a circular wait, where none of the processes can move forward.

* cond^n of deadlock?

- Mutual exclusion
- Hold and wait
- circular wait.
- No prevention

* Solution / prevention of Deadlock.

① deadlock prevention: ensure at least one deadlock condition doesn't happen

② avoidance: try to avoid these condition to make sure deadlock won't happen

③ deadlock detection & recovery: let it happen and then found out and fix it

④ deadlock ignorance: do nothing and hope it doesn't happen.

PAGE No.
DATE / /

Q. explain deadlock avoidance with an algorithm or example?

Ans:

Deadlock avoidance: deadlock avoidance ensure that the system will never enter a deadlock state by carefully analyzing each resource allocation ~~requ~~ request.

It only allow a process if it's safe to do so, meaning there is no risk of entering a deadlock situation.

★ Banker's Algorithm:

(1) Request resources: A process request resources

(2) Check safe state: simulate the allocation, check if the system remain in a safe state.

(3) Grant or Deny:
if safe, grant the request.
if unsafe, deny the request and wait

(4) Repeat: Repeat for every resource request.

PAGE No.
DATE / /

Q. Explain Resource allocation graph for deadlock detection?

Ans.

A Resource Allocation graph (RAG) is a visual tool used to detect deadlocks in a system. It shows the relationship between processes and resources using nodes and arrows.

- Components of RAG:

① Processes : Represented as circle.

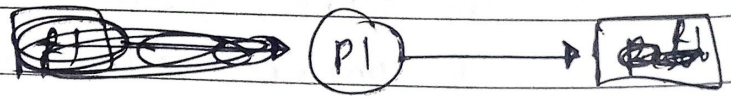
eg: (P1) (P2)

② Resources : Represented as Square

eg: [R1] [R2]

③ Request edge : Represented as →

process to Resource

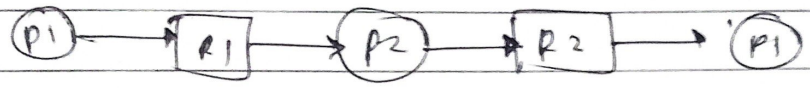


④ Assignment edge (←) : Resource to Process

How deadlock is detected?

- If the graph has a cycle
- if each resources has only one instances, a cycle means deadlock
- if resources have multiple instances, a cycle may or may not mean deadlock.

example:



P1 is waiting for R1
R1 is held by P2
P2 is waiting for R2
R2 is held by P1

This form a cycle, so the system is in deadlock

Wait for graph (WFG) used to determine presence of a deadlock

→ A wait for graph is simplified version of RAG used only for deadlock detection.

* Components: process only (no resource node)
edge (P1) → (P2) means P1 is waiting for a resource to hold by P2

* Deadlock detection using WFG:
if the wait-for graph contain cycle, then deadlock exists

— Used when each resource has only one instances

Q) explain Banker's Algorithm

Ans:

Banker's Algorithm is a deadlock avoidance algorithm that checks whether it's safe to grant resources to a process before actually allocating them.

It's called "bankers" because it works like a banker deciding whether to approve a loan — only if they're sure all customers (processes) can eventually be satisfied.

Algorithm:

① Request Resources : A process request ~~Resource~~ Resources

② check safe state: simulate the allocation, check if the system remains in a safe state.

③ grant or deny:
If safe, grant the request
if unsafe, deny and make the process wait.

④ Repeat : Repeat for every ~~process~~ request resources

Purpose: To make sure the system never enters a deadlock by only allowing resource allocation when it's safe.

Explain file access methods.

Ans:

File accessing methods are ways to read or write data in file, either in order or by jumping to specific parts.

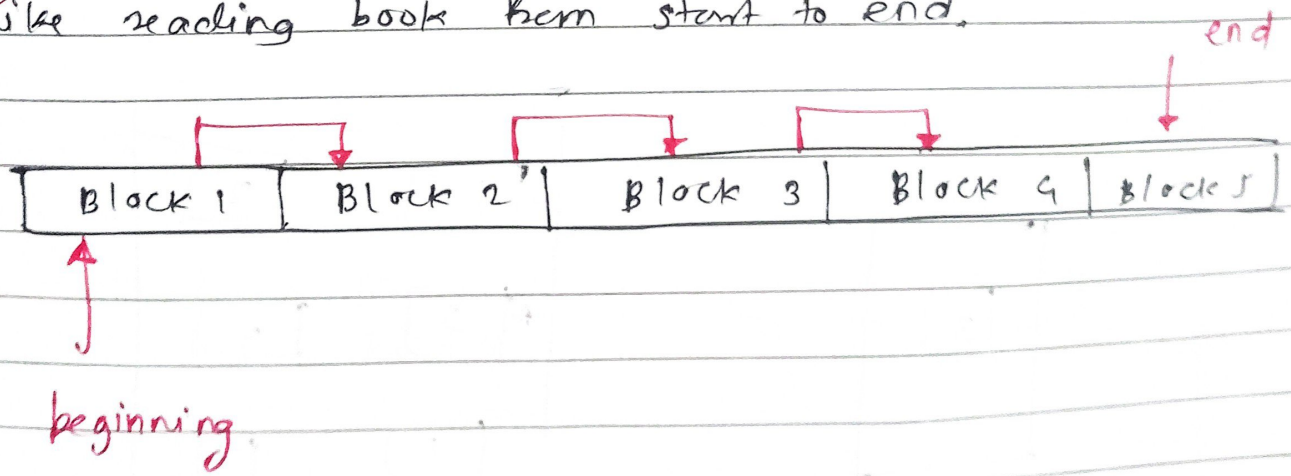
There are three types of file access

~~(1) Sequen~~

- sequential
- direct (or Random)
- Indexed access

(1) Sequential Access:

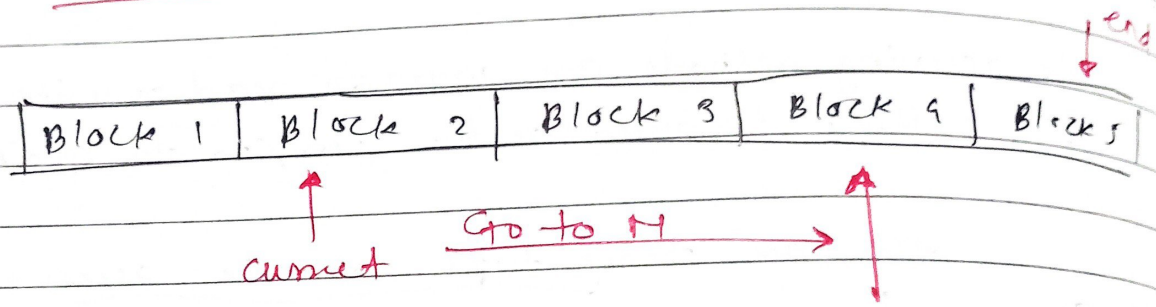
- data is accessed in order, one record after another
- most common method
- like reading book from start to end.



example: Reading a text file line by line.

(2) Direct (or Random) Access:

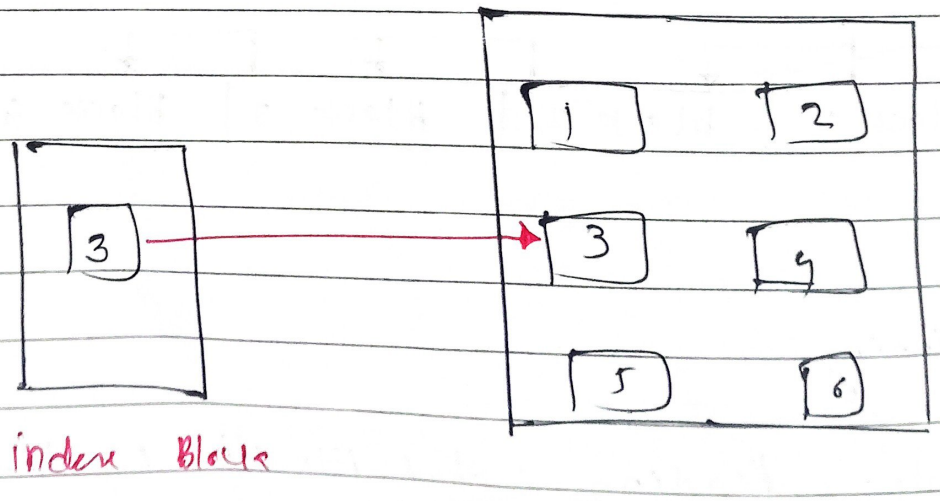
- data can be accessed directly at any position
- files are divided into fixed size block or records
- like jumping to page 50 of a book directly.



example: accessing a specific entry in database

(3) Indexed Access:

- uses a index (like a table of content) to access data quickly.
- index stores pointers to different blocks of files
- efficient for large files



Q) Technique of deadlock prevention.

Ans:

deadlock can happen when the processes get stuck waiting for each other's resources.
To prevent this we make sure to prevent the 4 main reason for deadlock and we can stop them like this

(1) Mutual exclusion: Avoid using exclusive access if possible

example: Allow shared access to read only resources

(2) Hold and wait: Make processes request all resources at once, or relate or hold resources before requesting new ones.

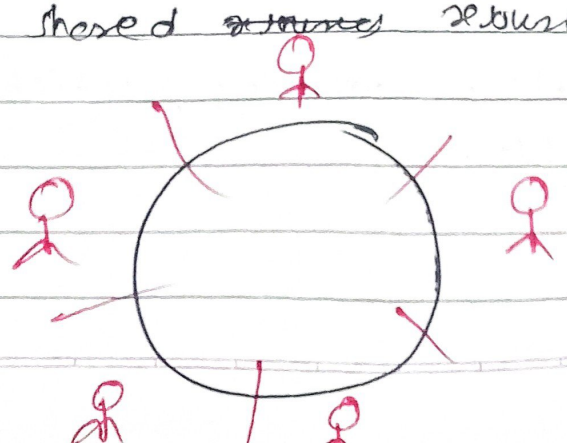
(3) No preemption: Allow system to take resources back from a process if needed.

(4) Circular wait: Impose a fixed order for resource allocation to prevent circular chains.

Q) explain dining philosopher problem

Ans:

The dining philosopher problem is a concurrency problem that shows how processes or thread compete for limited shared ~~resources~~ resources.



PAGE No.
DATE / /

- imagine 5 philosophers sitting at a table
- there is 1 fork b/w each philosopher (total 5 forks)
- To eat, a philosopher needs 2 fork (left and right)
- philosopher keep thinking and eating.

The problem: if all philosopher pick up the one fork at the same time, they wait forever for the second fork → this causes a deadlock

Solution:

1. Allow 4 philosopher to try to pick up forks at same time
2. if a philosopher can't pick up both forks, they put down the one they picked and wait
3. Use a semaphore or counter to limit access forks

Since there are 5 philosophers and 5 forks, if only 4 try ~~and~~ to pick forks at least one fork always free.

This prevent circular waiting, which is one of the cause of deadlock.

Q: Four necessary condition for deadlock (Confirm conditions)

Ans:

for a deadlock to occur, all four of the following must be true simultaneously.

(1) Mutual Exclusion:

At least one resource must be non-sharable. (only one process can use it at a time)

(2) Hold and wait:

A process is holding at least one resource and waiting to acquire additional resource held by others.

(3) No preemption:

A resource can not be ~~forcibly~~ forcibly taken from a process holding it - it must be released

(4) Circular wait: There exists a circular chain of processes, where each process is waiting for a resources held by the next

PAGE No.
DATE / /

explain memory management requirement.

Ans:

Memory management is responsible for efficient handling computer memory during program execution.

The main requirements are:

① Relocation: → program may be moved in memory during execution

- memory management must track and adjust the address.

② Protection: → one process should not access another process's memory.

- ensure data safety and process isolation.

③ Sharing: - Sometimes, processes need to share memory (eg: shared libraries)

- memory manage must support controlled sharing.

④ Logical organization: - programs are divided into modules (code, data, stack etc)

- memory system must handle modular programming.

⑤ physical organization: - Memory is split into main memory (RAM) and secondary memory storage (disk)

- memory manager handles data movement b/w them

PAGE NO.
DATE / /

Q) explain memory partitioning techniques with an example.
(Best, worst, first fit)

Ans:

- memory partitioning is a technique used by the OS to divide the main memory into sections, or partitions to allocate memory to processes.

- It helps in managing how memory is distributed and reused efficiently.

(1) First Fit : " First block that fit "

Allocate the first available block of memory that is large ~~enough~~ enough for the process

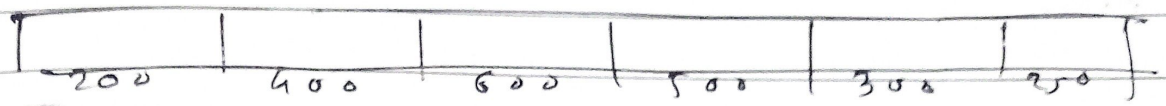
(2) Best Fit : " smallest suitable block "

Allocate the smallest block that is just large enough to satisfy the request.

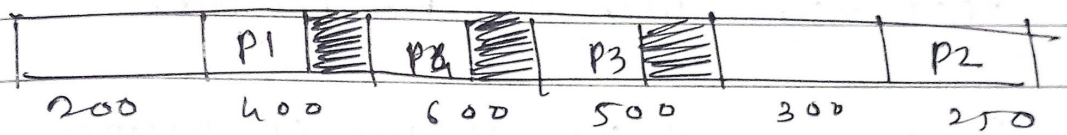
(3) Worst Fit : " largest available block ".

Allocate the largest available block, thinking the larger leftover part can be reused better.

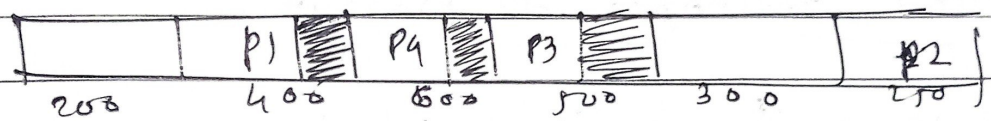
example: P1 = 310 P2 = 250 P3 = 480, P4 = 420



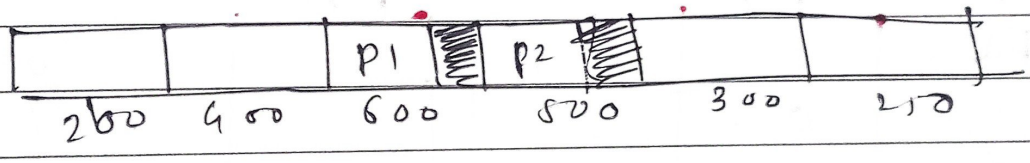
First fit:



Best fit:



Worst fit:



DATE / /

(P) explain the concept of demand paging

Ans:

Demand paging is a memory management technique where page of program are loaded into RAM only when they ~~need~~ needed.

if a required page is not in memory, the system causes a page fault, loads the page from disk, and continues the program.

In short only load what you need, when you need it.

	F	P/A (preset – abset)
0	F.L	1
1	–	0
2	–	0
3	F2	1
4	–	0
5	F3	1
6	–	0
7	–	0
8	–	0
9	F4	1
10	F5	1

page table

	Pages
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

process p1

	m.m
0	
1	p(10)
2	p(23)
3	p(35)
4	p(9)
5	p(1)
6	
7	
8	
9	
10	

page fault = 6

11 - 5 = 6

PAGE No.
DATE / /

- How it's work

① When a program starts, only essential page are loaded into RAM

② if the program tries ~~and~~ to access a page is not in memory page fault occurs.

* Advantage	* disadvantage
- save memory space	can cause delay due to page fault.
- faster startup time	
- only needed parts of the program are loaded	

PAGE NO.
DATE / /

Q: diff b/w segmentation and paging.
Ans:

NO:	Segmentation	Paging
1	divides memory into variable-sized segmentation	divides memory into fixed size pages
2	segments are of different size	pages are all same sizes
3	external fragmentation may occur	doesn't occur
4	internal fragmentation occur rare or less	can occur
5	Addressing → use segment number + offset	Addressing → page number + offset
6	used in more suitable for modular program	used in more suitable for efficient memory use
7	based on logical units	based on physical memory blocks
8	example: Code segment, data — 1 —, stack — 1 —	example: page 0 — 1 — 1 — 2

PAGE No.
DATE / /

Q. What is TLB?

Ans:

- TLB stand for translation lookaside Buffer.

- TLB is a high speed cache memory in the CPU that stores recent page table entries (PTEs)

- It helps speed up virtual to physical address translation by reducing access time to the page table stored in main memory

- It reduce memory access time by avoiding frequent lookup in the page table.

Benifits of TLB

- Fast memory access
- Improves performance of paging system

Explain page replacement algorithm with an example.

Ans:

- When memory (RAM) is full and new page is needed the system must replace an existing page.

- Page replacement Algorithms decide which page to remove.

- Used in demand paging system to reduce page fault.

- Common page replacement algorithm

① FIFO (First in, First out)
② LRU (Last Recently used)
③ Optimal

① FIFO: First in, First out

- The simplest page replacement algo and the work on the basis of First in, First out.
- This algo always choose oldest page for replacement

eg: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 F = 3

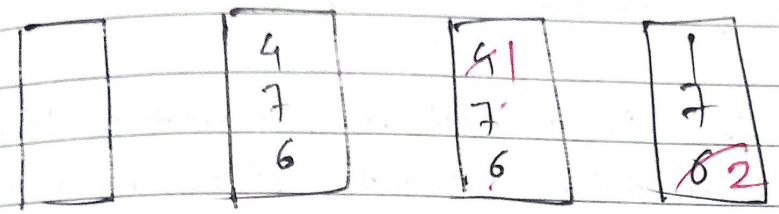
	4	~~4~~ 1	1	1	1
	7	7	~~7~~ 2	2	2
	6	6	6	~~6~~ 7	7
0	3	1	1	1	

PF = 6

② Optimal: The complex to all other ~~pla~~ page replacement algo.
this algo has lowest page fault rate.

This algo replace the page which will be not be used for the longest period of time.

eg: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 F = 3



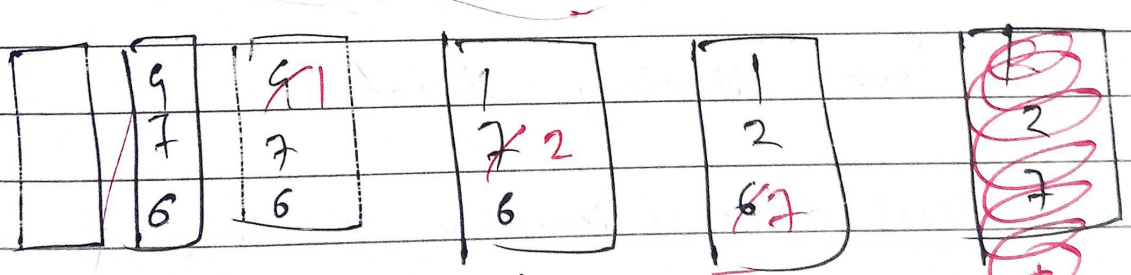
0 3 1 1 PF = 5

(3) LRU: least recently used.

- This algo choose the page those was not used for longest period of time in the past.

- This algo used backward in the time. While optimal page replacement algo looks forward in the time.

eg: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 F = 3



3 1 1 1

PF = 6

PAGE No.
DATE / /

Q. Explain files directories and it's types

Ans:

- Directories are keep tracks of files

- A file directory is a system maintained structure that stores information about files on a storage devices.

- It keep track of files names, locations, sizes, types, permissions and ~~dat~~ meta data enabeling efficient file management.

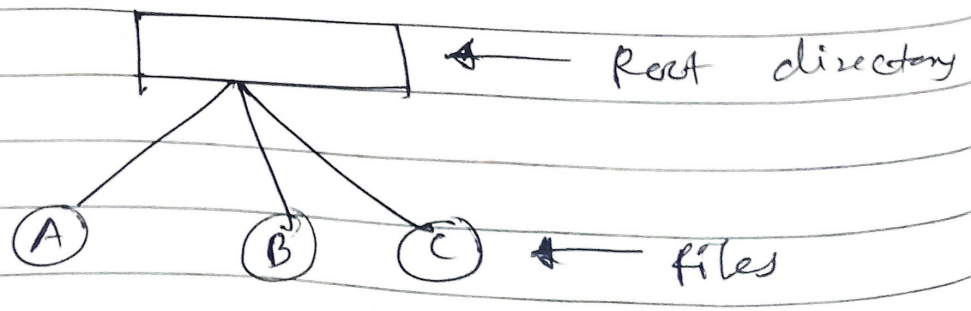
- Types of file directory structures

① Single-level directory structure
② Two-level — | —
③ Hierarchy — | —

(1) Single level directory structure.

- Single level directory structure is simplest directory structure.

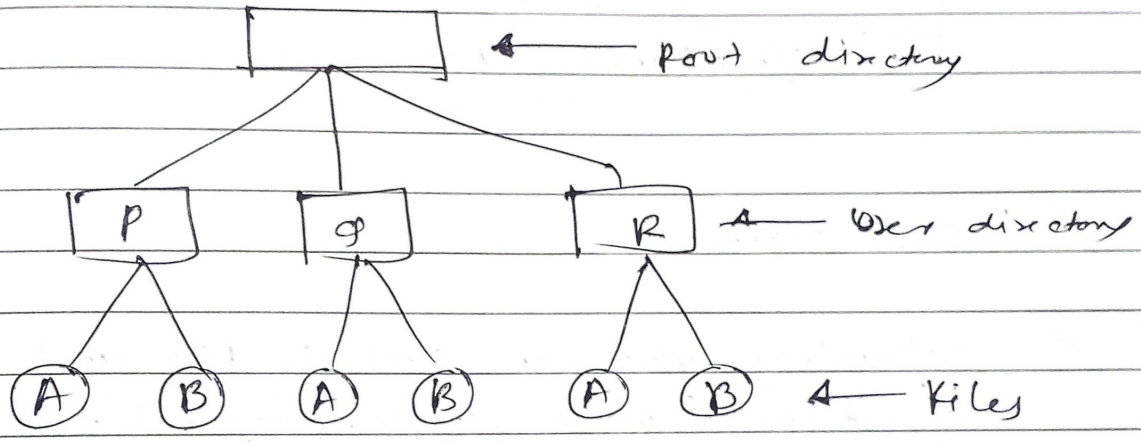
- In this ~~dir~~ directory structure, one directory contains all the files



(2) Two level directory structure.

- two level directory system / structure, the system maintains a separate directory for each user.

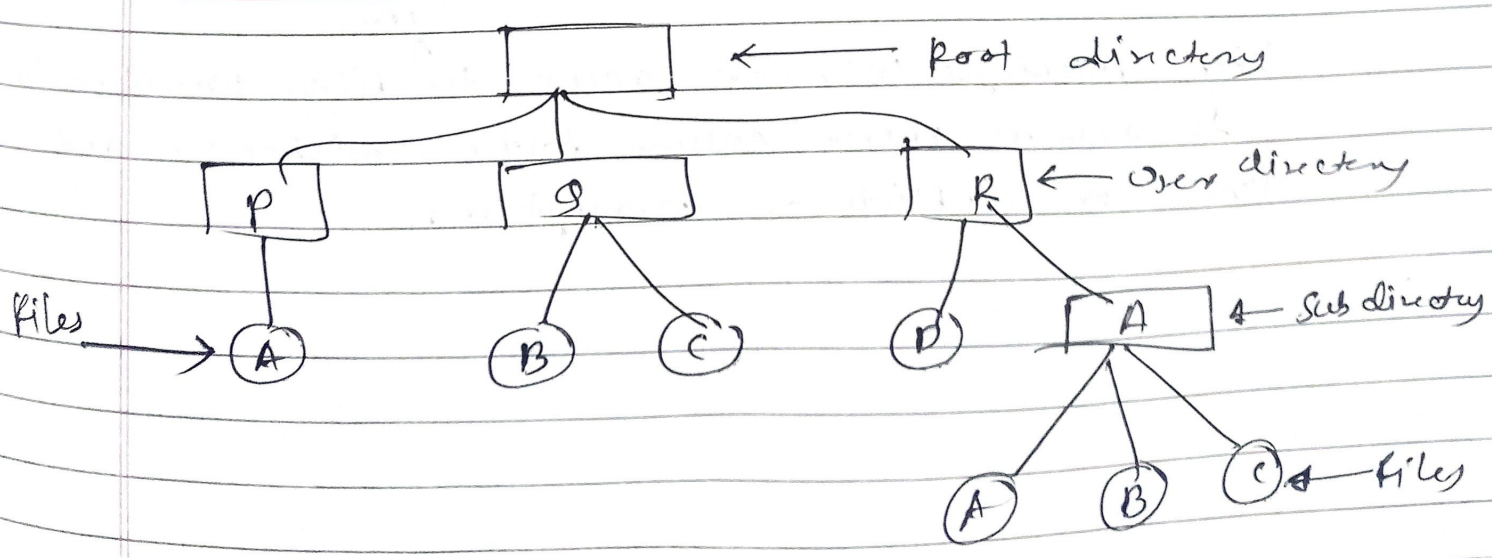
- each user's directory contains only their files



(3) Hierarchical level directory structure.

- This is multi-level directory structure, where each user can create subdirectories inside their own directory

- it look like a trees with branches



PAGE No.
DATE / /

Q) methods of file sharing:

Ans:

- It is necessary to share the file among the multiple user in multiple system

- It needs to deal with two issue.

→ access rights
→ simultaneous Access.

① access rights:

- determines who can access the file and what operations (read, write, execute) they are allowed to perform.

- this prevents unauthorized access.

② Simultaneous Access:

When multiple user can access the same file simultaneously the system must ensure data consistency and avoid conflicts or corruptions.

PAGE No.
DATE / /

Methods of File sharing:

1. File share by Name (logical sharing)
2. File share by Reference (physical sharing)
3. Remote File sharing (Network-based sharing)
4. concurrent File sharing
5. Peer to Peer (P2P) File sharing
6. cloud-Based File Sharing (eg: google drive)
7. permission based sharing (based on user access rights)

Q) explain disk scheduling algo with an example:

FCFS, SSTF, SCAN, CSCAN, Look, C-Look.

Ans:

Disk scheduling is a method used by the OS to decide which disk I/O request to serve next when multiple requests are waiting in the queue.

- The algos are.

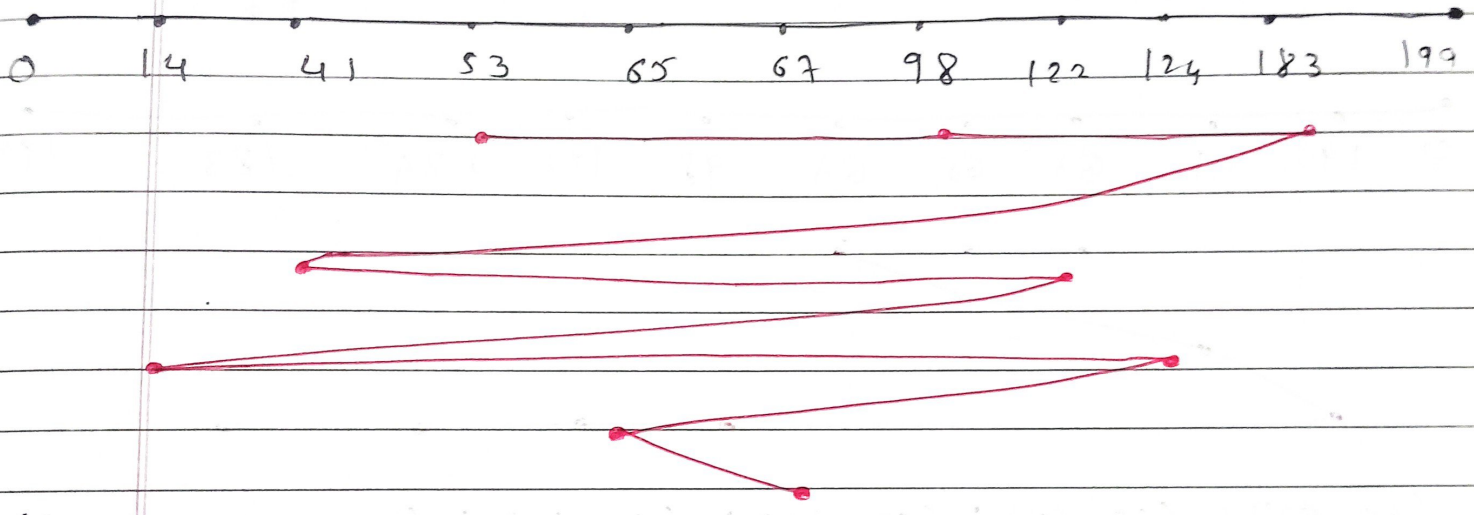
1. FCFS
2. SJTF
3. SCAN
4. CSCAN
5. Look
6. C - Look.

(1) FCFS (First come, First serve)

- it's first come, first serve algorithm
- simple disk scheduling algorithm
- request are apply in the order they arrived

eg: 98, 183, 41, 122, 14, 124, 65, 67

R(w) = 53



$(98-53) + (183-98) + (183-41) + (122-41) + (122-14)$
$+ (124-14) + (124-65) + (67-65)$

$= 632$

• Advantage

- simple
- no starvation

• disadvantage

- no seek time.

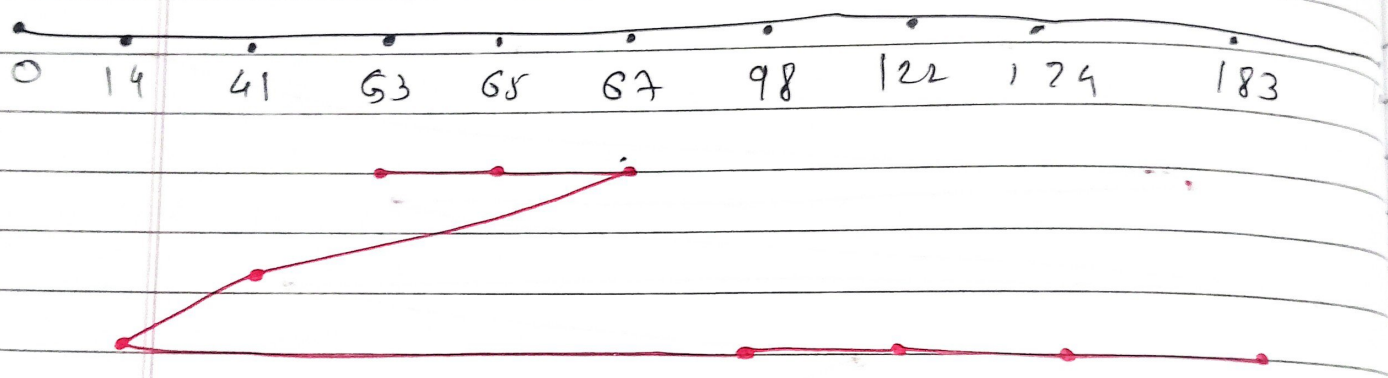
hint:
{closest delchnetka}

(2) SSTF (shortest seek time first)

- Services that required which is closet to the amount position of the read/write head.

- tie is broken is direction of head movement.

eg: 98, 183, 41, 122, 14, 124, 65, 67 head = 53



$(65-53)+(67-65)+(67-41)+(41-14)+(98-14)+(122-98)+(124-122)+(183-124)$

$= 236$

• Advantage

- very efficient in seek time
- less avg response time

• dis advantage

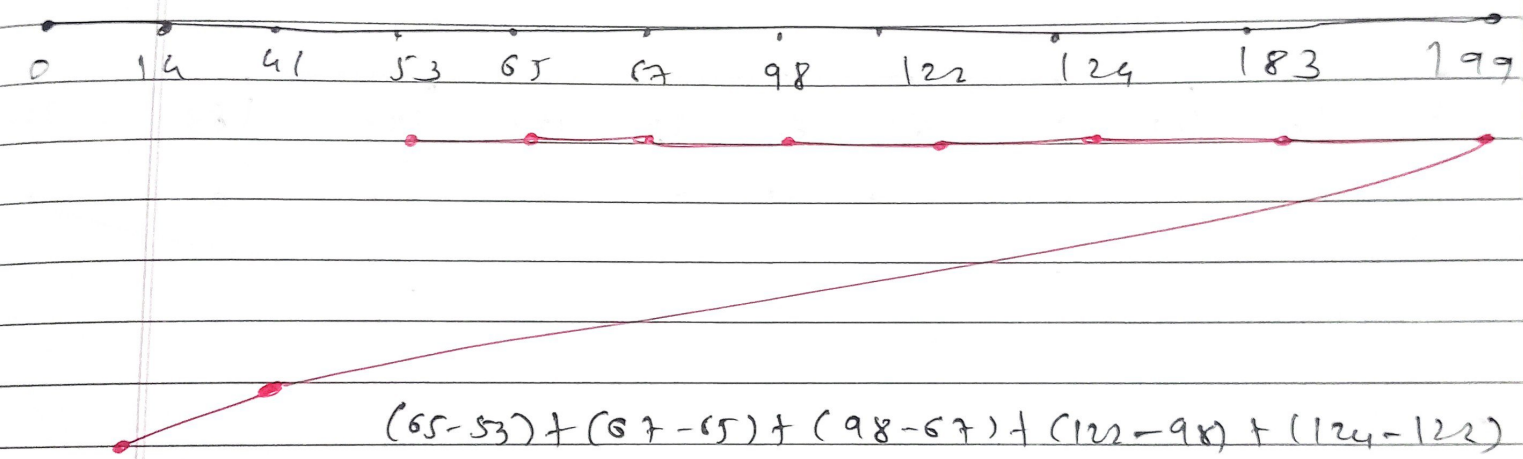
- request which are far from the head will start
- high variance in response time or waiting time

Start se end tak jayega fir piche jayega.

(3) Scan disk scheduling algorithm:

- head start at one end of the disk and moves towards the other hand
- and then direction of the head is reversed and processes continue, head continously scan back and forth across.

eg: 98, 183, 41, 122, 14, 124, 65, 67 R/W = 53



$(65-53)+(67-65)+(98-67)+(122-98)+(124-122)$
$+(183-124)+(199-183)+(41-14)+(199-41)$

$= 331$

* Advantage

- simple, easy to understand and implement
- low variance and avg waiting time.

* disadvantage

- long waiting time for location just visited by head.
- unnecessary move till the end of disk.

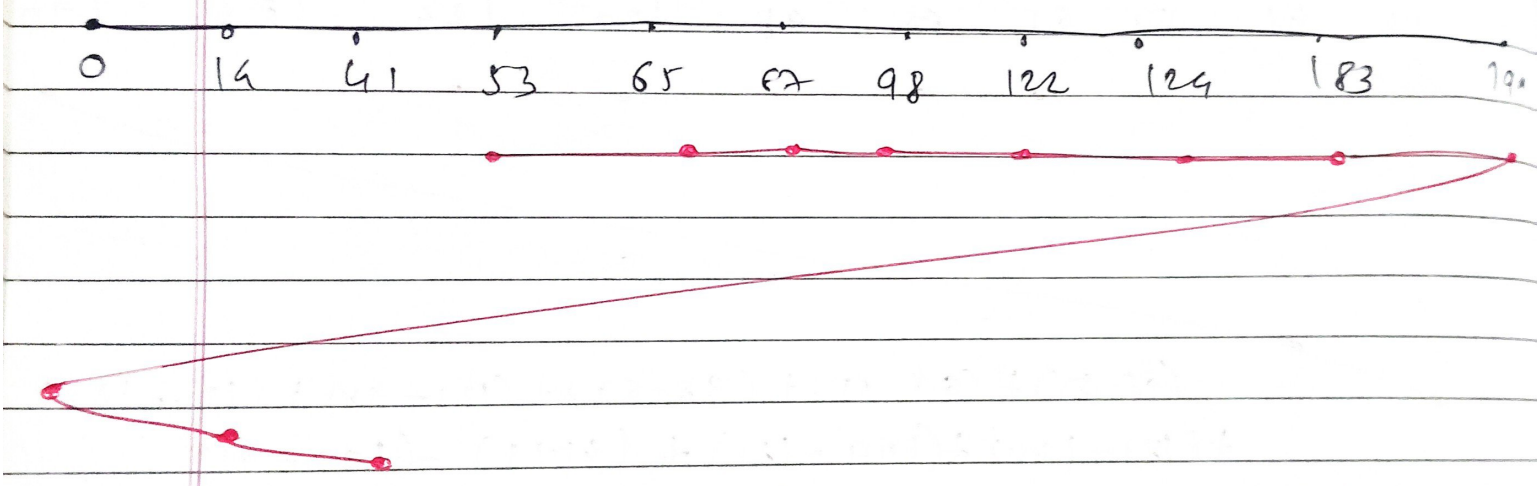
Jab time sab lega
aate waqt zero pe jayega then baki ko lega.

PAGE No.
DATE / /

(4) C-Scan (Circular Scan)

- head at ~~time~~ one end of the disk and move towards the other end and servicing the request in b/w and reaches the other end.
- and the direction of the head is reverse and head reaches first end without satisfying any request.

eg: 98, 183, 41, 122, 14, 124, 65, 67 p/w = 53



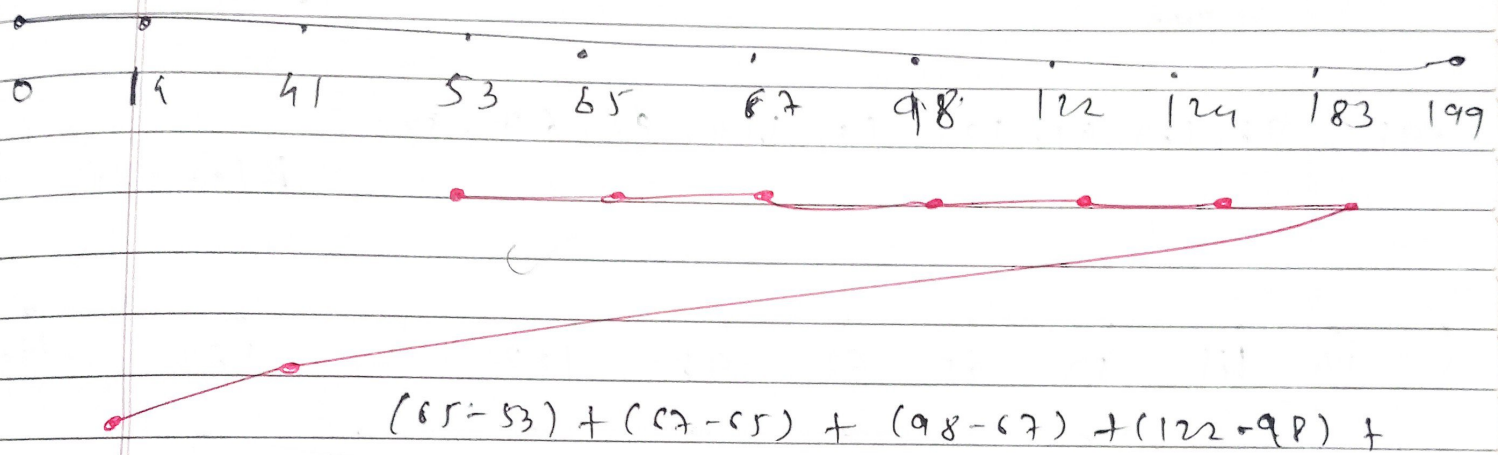
$(65-53) + (67-65) + (98-67) + (122-98) + (124-122)$
$+ (183-124) + (183-124) + (199-183) + (199-0) +$
$(14-0) + (41-14)$

$= 386$

(5) Look disk scheduling algo.

- it's same as scan algorithm but instead of going till last track, we go till second last request then change the direction

eg: 98, 183, 41, 122, 14, 124, 65, 67 $\qquad$ $P_{100} = 53$!



$(65-53) + (67-65) + (98-67) + (122-98) +$
$(124-122) + (183-124) + (183-41) + (41-14)$

$= 299$

- advantage

- better performance compare to scan algo

- it should be used in less load

- disadvantage

- it can't be used when the request is high

Sab leg a fate wait
then o pe jayega or remaining leg a

PAGE No.
DATE / /

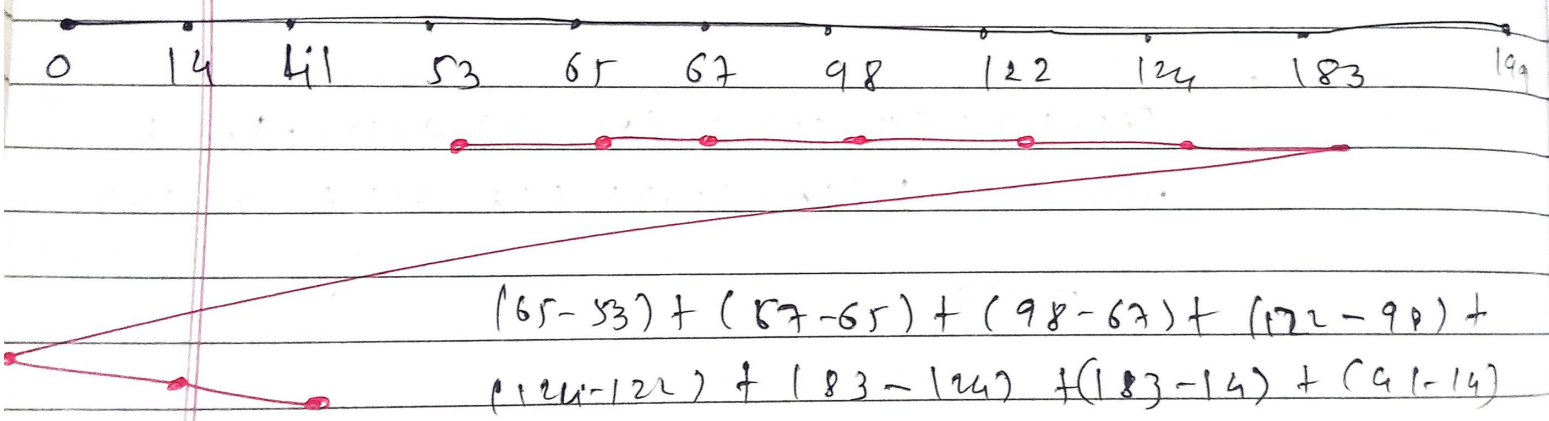
(6) C-LOOK :

- it takes the advantage of both look and C-scan also

- only one direction can have to go and take a the request and reverse back without taking any request and he move back and take the request.

eg: 98, 183, 41, 122, 14, 124, 65, 67, 53

R/W = 53



$(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-14)+(41-14)$

$= 326$

★ advantage

more uniform waiting time
Compare to look scheduling algo

★ disadvantage

- more overload in calculation

- it should not be used in case of no request.

PAGE No.
DATE / /

Q) diff b/w process and Thread

Ans:

Sr. NO	Process	Threads.
(1)	program in execution called as process. it is heavy weight process	Thread is part of process it is also called as light weight process.
(2)	A process take more time to terminate	A thread takes less time to terminate.
(3)	it take more time to creation	it takes less time for creation
(4)	it also take more time for context switching	it takes less time for context switching.
(5)	A process is less efficient in terms of communication	Thread is more efficient in term of communication.
(6)	every process runs in own memory	Threads share memory
(7)	A process does not share data with each other	Thread share data with each other
(8)	one process crash doesn't effect other	one thread crash can effect the process.
(9)	eg: Chrome tabs as processes	eg: file download and UI update as thread

PAGE No.
DATE / /

Q) diff b/w User Level thread (ULT) and kernel level thread (KLT)

Ans:

Sr. No	User Level Thread (ULT)	Kernel Level Thread (KLT)
(1)	User threads are implemented by user level libraries	Kernel level thread are implemented by Operating system.
(2)	The OS doesn't recognize user level thread directly	Kernel threads are recognized by Operating system
(3)	Implemetation of ULT is easy	implemetation of KLT is complicated.
(4)	Context switching time is less	Context switching time is more.
(5)	No hardware support is required for context switching	Hardware support is required.
(6)	ULT can be created and managed more quickly	KLT take more time to create and manage.
(7)	ULT are also called many to one mapping thread	KLT support one to one thread mapping
(8)	eg: ~~Java thread~~, ~~Mach-C thread~~. green threads in Java	eg: ~~Windows Java threads~~ windows threads

PAGE No.
DATE / /

5) explain diff types of threads in O.S

Ans:

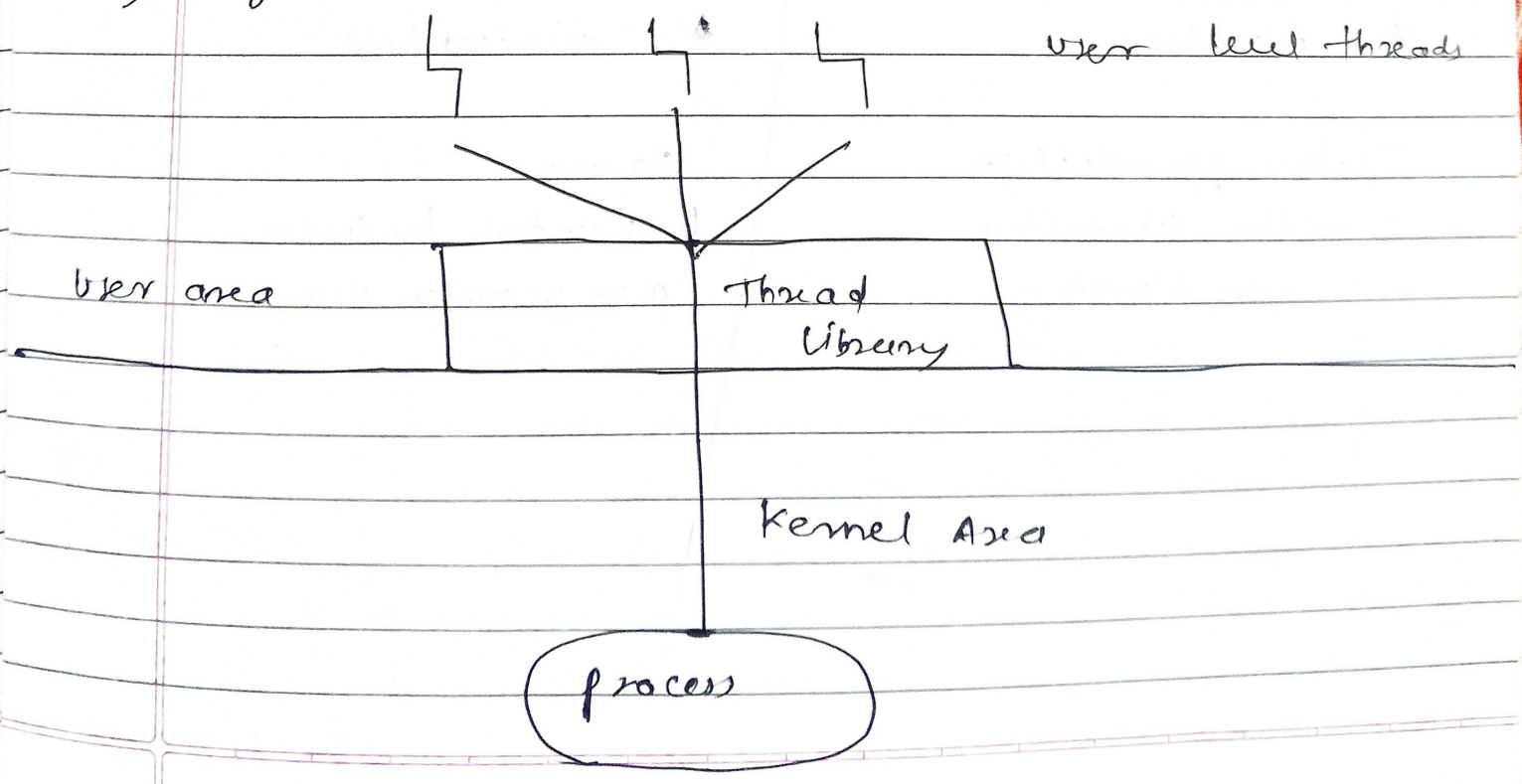
Threads can be implemented in two ways.

Types of threads
- → User level threads
- → Kernel level threads.

(1) User Level threads:

- Managed by user-level libraries, not the kernel
- The O.S is not aware of these threads
- All thread operation (creation, switching) are done in user space
- fast and efficient but if one thread blocks the whole process blocks.

eg: green threads in Java.



— ULT diagram —

★ Advantage	★ Disadvantage
- fast thread operations - portable - efficient resource usage	- blocking problem - Not suitable for CPU bound multi-threading - No true ~~path~~ parallelism

(2) Kernel level thread:

- Managed directly by the O.S kernel
- The kernel is aware of and schedule each thread individualy.
- threads can run truly in parallel on multicore system
- Slower than ULT due to system call overhead.

eg: windows threads, Linux kernel thread.

★ Advantage	★ disadvantage.
- true parallelism - better scheduling - non-blocking	slower less control by user more resource usage

PAGE No.
DATE / /

Q What is critical section problem? mention three condⁿ that must be satisfied by its solution

Ans

The critical section problem in an operating system arises when multiple processes access shared resources simultaneously. if multiple process enter their critical section at same time. it can lead to race condition and unexpected results.

Three necessary condⁿ of critical section problem

- → Mutual exclusion
- → Progress
- → Bounded waiting

(1) Mutual Exclusion:

- No two process should be present inside critical section at the same time,

- only one process is allowed in the critical section at a instance of time.

(2) progress: if no process in the critical section and some processes want to enter then only we can allowed the process to enter its critical section, without unnecessary delay.

(3) Bounded waiting: After a process request to enter its critical section, there must be a limit on the number of times other processes are allowed to enter before it's gets its turn.

- ensure no starvation.

Q* What is producer consumer problem? provide a soln to problem using semaphore.

Ans:

The producer-consumer problem is a classic synchronization problem in operating system where;

- The producer generates the data and puts it into a shared buffer

- The consumer takes data from the buffer and processes it.

* The challenge to prevent:

(1) producer from adding data when the buffer is full

(2) consumer from removing data when the buffer is empty

(3) Race condition when both access the buffer at the same time.

- Solution using semaphores?

- We use three semaphores:

- mutex — ensure mutual exclusion
- empty — count to no. of empty slots
- full — —— | —— filled slots

initialization: semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;

producer code

```
do {
    wait (empty);
    wait (mutex);

    signal (mutex);
    signal (full);

} while (true);
```

consumer code.

```
do{
    wait (full);
    wait (mutex);

    signal (mutex);
    signal (empty);

} while (true);
```

PAGE No.
DATE / /

Q. explain the term "Busy waiting". Give a solution to this problem using semaphore.

Ans:

Busy waiting is a situation where a process continuously check a condition in a loop without releasing the CPU.

it wastes CPU cycles because the process does no work while waiting.

eg:
```
while (lock == 1){
        //keep checking  - this is busy waiting
}
```

* Solution using semaphores (No busy waiting)

```
Semaphore mutex = 1;

Process A:
    wait (mutex);
    signal (mutex);

Process B:
    wait (mutex);
    signal (mutex);
```

* wait() → decrements semaphore.
* if value become negative, the process is block
* signal() → increment the semaphore, and wake up a blocked process if any.

or process scheduling

both are same

Q) Explain CPU scheduling criteria:

Ans:

CPU scheduling is the process of selecting a process from the ready queue to be executed by the CPU, to evaluate and compare different scheduling algorithm, we use several criteria.

(1) CPU Utilization:

it is amount of time CPU remain busy

(2) Throughput: No. of jobs processed per unit time

(3) Turn Around time: total time taken from submission to completion of a process.

Turn Around Time = Completion time − Arrival time

(4) Waiting Time: Time a process spends in the ready queue (not being executed)

Waiting Time = Turn Around time − Burst Time

(5) Response Time: Time from submission of a process to the first it gets the CPU

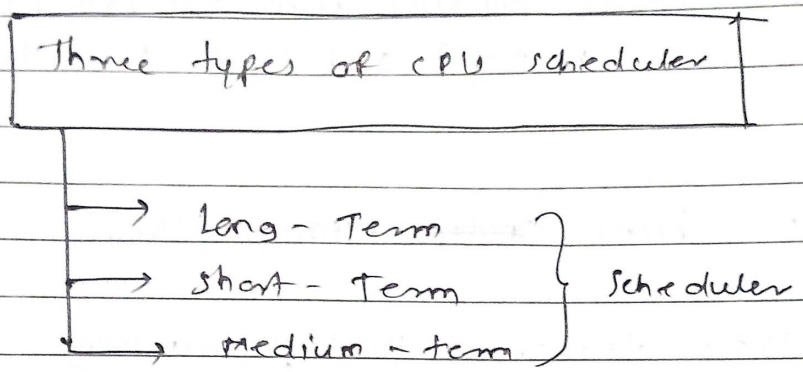
(6) fairness: every process should get fair share of the CPU time.

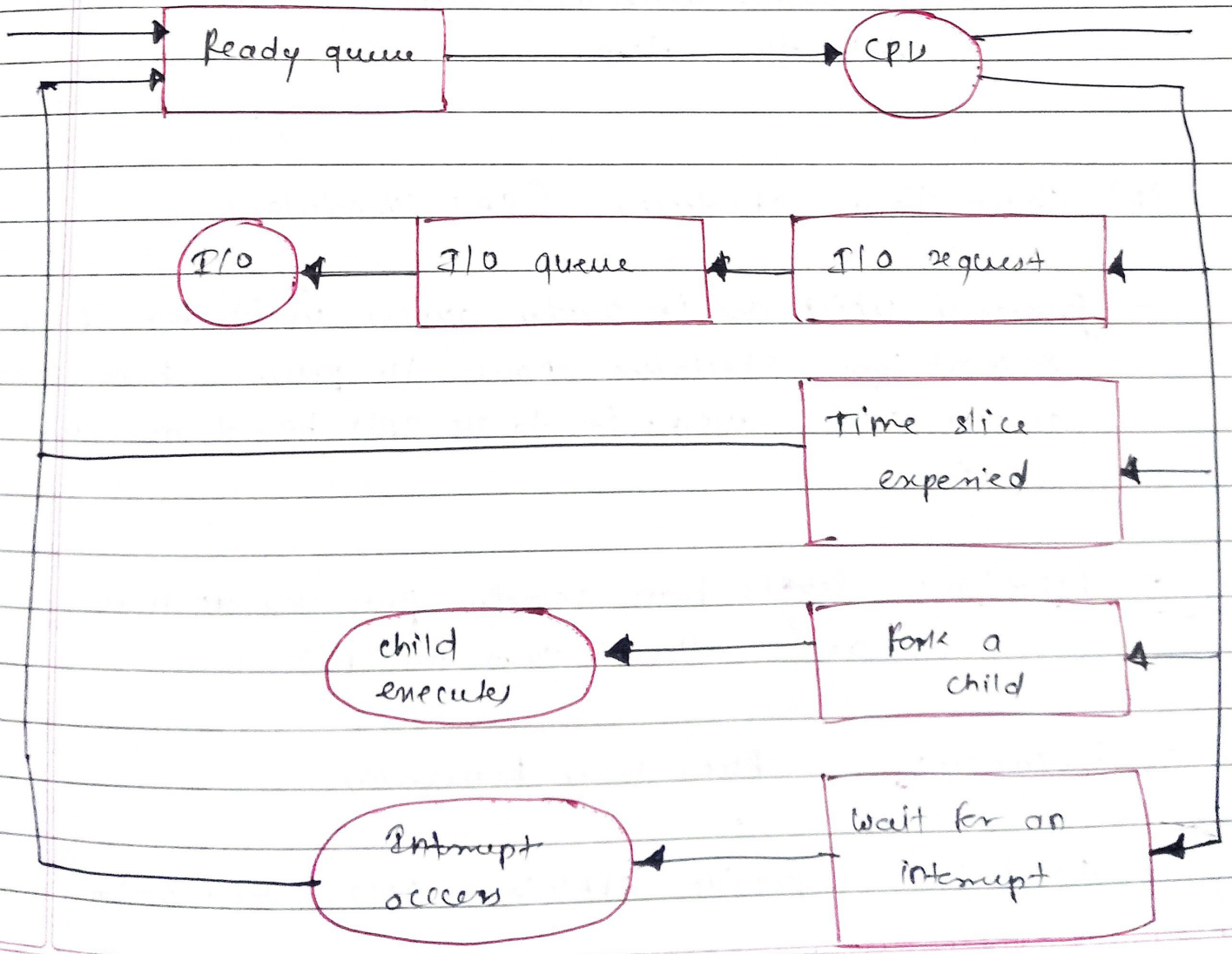
or Three CPU scheduler diagram

Q) Explain brief CPU scheduler with diagram.

Ans:

The CPU scheduler is a component of operating system that selects one process from the ready queue and assign the CPU to it for execution.





PAGE No.
DATE / /

(A) long-term scheduler (job scheduler)

- when program are submitted to the system for the purpose of processing, long term scheduler comes to know about it

- function: selects processes from the job pool (secondary memory) and loads them into main memory

- frequency: Runs ~~[illegible]~~ infrequently.

- Goal: control degree of multiprogramming

- example: decide which user-submitted jobs to admit into the system.

(B) Short-Term scheduler (CPU scheduler)

- processes which are in ready queue wait for CPU. A short term scheduler choose the process from ready queue and assigns it to the CPU based on some policy

- function: selects from ready processes in memory and allocate them to CPU

- frequency: Runs very frequently

- Goal: provide efficient CPU utilization

- example: decide which ready process to run next on the CPU

PAGE No.
DATE / /

(c) Medium-Term Scheduler (Swapper):

- if the degree of multiprogramming increases, medium-term scheduler swap out the processes from main memory

- Function: Moves process in and out of memory (swapping)

- Goal: Improve process mix or reduce memory pressure

- example: suspends background jobs to make room for high-priority ones.

Q) Explain MFT (Multiprogramming with fixed number of task) with example.

Ans:

MFT is a memory management technique used in early operating system to allow multiprogramming by dividing memory into fixed-sized partition.

each partition can hold exactly one process, once loaded, the process stays in its assigned partition until completion.

internal fragmentation occur when a process doesn't use the entire space of its partition. The remaining unused space is wasted.

example: suppose the total main memory is 16KB. and it's divided into 4 fixed partition of 4KB each

process	size
P1	2 KB
P2	3.5 KB
P3	4 KB
P4	1 KB

memory allocation:

partition	size	process	internal fragmentation
P1	4KB	P1	2KB (4-2)
P2	4KB	P2	0.5 KB (4-3.5)
P3	4KB	P3	0 KB (4-4)
P4	4KB	P4	3KB (4-1)

total internal fragmentation: 2+0.5+0+3 = 5.5 KB

PAGE No.
DATE / /

Q: explain the concept of segmentation.

Ans:

- Segmentation is a memory management technique in which the logical memory is divided into variable-sized segmennents,

based on the logical operation in a program like:

- Code (text segment)
- Data
- Stack
- Heap

each segment is a logical unit and can grow independently.

- Segmentation Table :

Field	Description
- Base Address	Starting physical address of the segment
- Limit	length of the segment.

Segment 0	Segment 1
size = 800	size = 1200

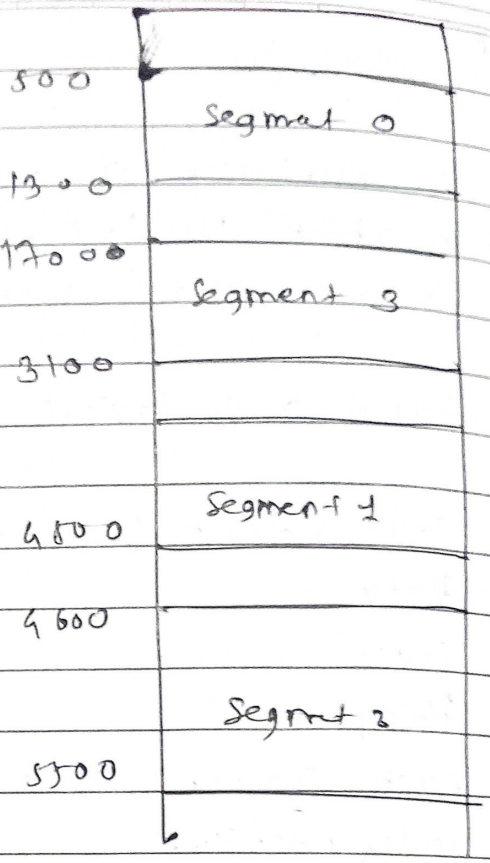
Segment 2	Segment 3
size = 900	size = 1400

Logical Address space.

Physical memory

	Limit	Base
0	800	500
1	1200	3300
2	900	4600
3	1400	1700

Segment table



★ Advantage

- No internal Fragmentation
- protection & independet growth
- logical structure

★ Disadvantage

- cause external fragmentation
- complex memory management
- slightly slower than paging.

PAGE No.
DATE / /

eg:

Segment	Size	Base Add
code	1KB	2000
data	2 KB	4000
stack	1 KB	7000

Now logical Address (1500)
Segment 1 = data → base = 4000.
offset = 500
So physical address = 4000 + 500 = 4500

Q) explain the concept of paging.

Ans:

Paging is a memory management technique that eliminates external fragmentation by dividing:

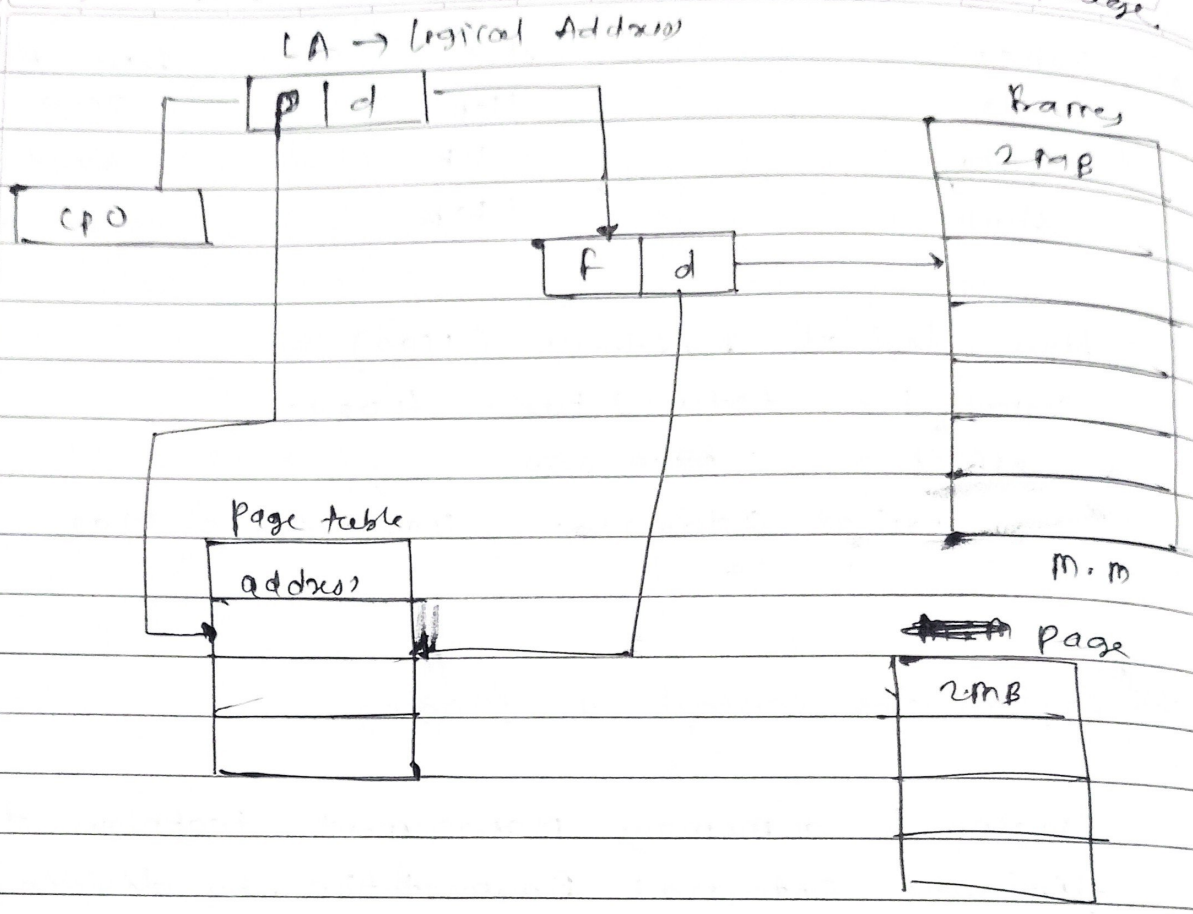
- logical memory into fixed-size block called page
- physical memory into fixed-size block called frame

The size of a page = size of frame.

* Key concepts:

- Logical Address = page number + offset
- OS maintain a page table that maps page to frame
- Paging is transparent to the user / programmer.

Initially all page remains in secondary memory storage, when process is needed to be executed it's page are loaded into any available memory frame from secondary storage.



* following basic operation of paging

(1) Page Number : index of page in process's logical memory

(2) Page offset : distance from the start of the page to a specific byte

(3) frame number : index of a frame in physical memory

(4) frame offset : same as page offset

(5) page table : A data structure maintained by the o that map each page number to a frame n

PAGE No.
DATE / /

$$\text{No. of page} = \frac{\text{Logical Address space}}{\text{Page size}}$$

$$\text{No. of frames} = \frac{\text{Physical Address space}}{\text{Frame size}}$$

eg: Page size = Frame size = 1 KB

Logical Address : (page 2 , offset 200)

Page table : page 2 → Frame 5

physical address : Start from 5 + 200 = (5 × 1024) + 200

= 5320

★ Advantage	★ disadvantage
- No external fragmentation	- Cause internal fragmentation
- simple to implement	- Overhead of maintaing page table
- virtual memory enables	- slightly ~~more~~ ~~to~~ slower

Q) diff b/w external fragmentation vs Internal fragmentation

Ans:

Sr.No	Internal fragmentation	External fragmentation
(1)	wasted space within an allocated memory block	wasted space outside the allocated block.
(2)	when allocated memory is larger than required	when free memory is scattered in small blocks
(3)	occur with paging	occur with segmentation
(4)	it's fixed size memory	it's variable size memory
(5)	~~Solution is best fit block~~	~~Solution is complicated~~
(6)	it occur when memory is divided into fixed size problem	it occur when memory is divided into variable size partitions
(7)	Use exact or dynamic-size allocation for solution	solution - use compaction, paging or segmentation
(8)	eg: Allocated 4KB, use only 2KB → 2KB wasted	eg: 3 blocks of 1KB free but can't fit 2KB process

PAGE No.
DATE / /

Q) explain diff file organization method

Ans:

Five fundamental organization of file

- → the pile
- → the sequential file
- → The indexed sequential file
- → the indexed file
- → the direct or hashed file.

(1) The pile:

- it is the simplest form of file organization
- in this file data is stored in the order of arrival.
- each record in this file comprises one burst of data

The idea behind the pile is simply to collect the mass of data and save it.

(2) Sequential file:

- it is most commonly used structures and record have fixed format.
- Record are stored in linear order, one after another.
- each and every record in file has same length
- Access start from the beginning and goes in order.

(3) The indexed sequential file:

- The indexed sequential file eliminate drawback of sequential file.
- Similar to sequential file, records are organized in logical order of a key field.
- The indexed sequential file contains index to the file to support random access. an overflow file.

(4) The indexed file:

- The sequential file and the indexed sequential file can search the record based on a single field of the file
- Apart from using key field, they cannot search record using other attribute of record.
- The ~~record~~ indexed file, records are accessed by only through their indexes.

(5) The direct or hashed file:

- This type of file make use of the ability found on disk to access straight any block of a known address.
- A hashing operation is performed on key value using hash function.
- direct file are frequently used where very fast access is necessary.

Q) explain file allocation methods in details

Ans:

File allocation is how OS stores files data blocks on disk. A good method ensure efficient space use and quick access.

File Allocation Methods

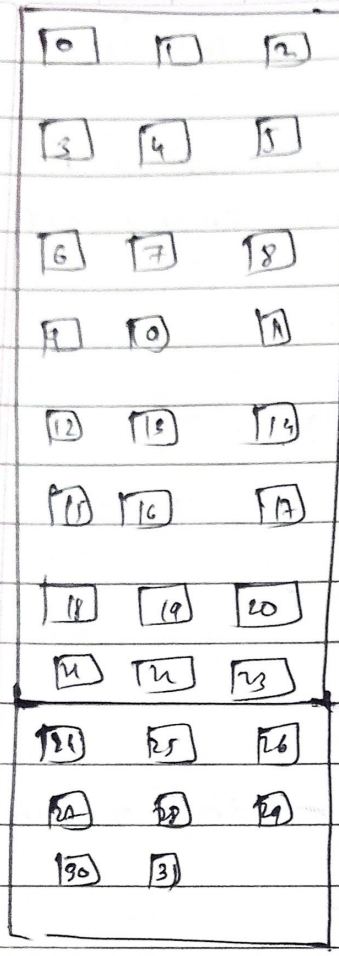
- → Contiguous Allocation
- → Linked List Allocation
- → Linked List Allocation using a table in memory
- → Indexed Allocation
- → I - nodes

(1) Contiguous Allocation:

- All file blocks are stored together in a sequence
- Support fast random and sequential access
- Need to know the file size in Advance

* Advantage	* disadvantage
- fast access	- causes external fragmentation
- Simple to implement	- difficult to grow file after allocation.

eg: if a file needs 5 blocks, allocate blocks 10 to 14



disk space

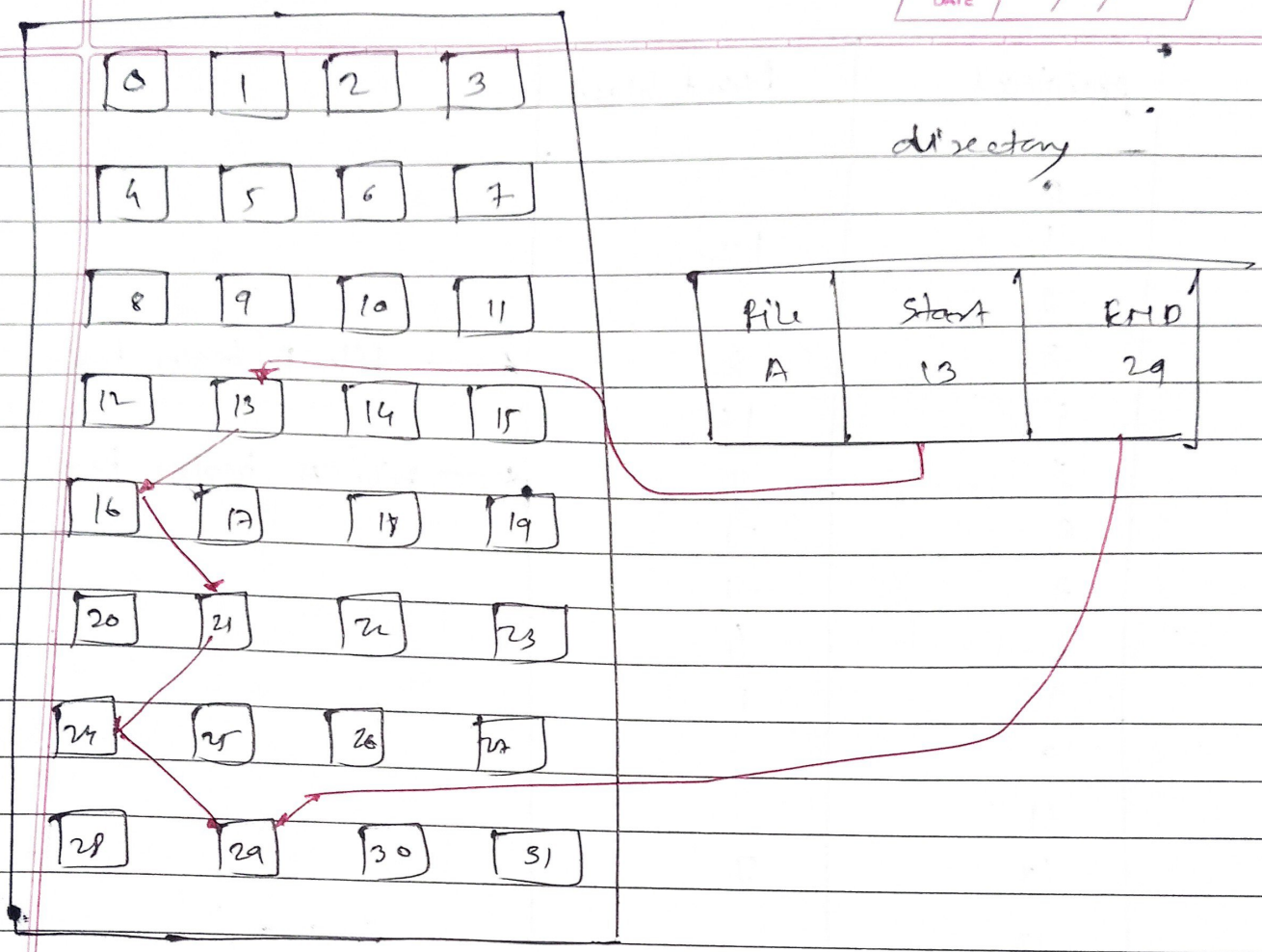
directory

File	Start block	length
A	0	3
B	3	5
C	14	8

(2) Linked List Allocation

- each file block contain a pointer to the next block
- Blocks may be scattered anywhere on the disk
- no need to know the file size in advance.

★ Advantage	★ Disadvantage
- No external fragmentation	- slow direct access
- Easy to grow files	



disk space

(3) Linked List using Table in memory

- All pointer are stored in separate table in memory
- File blocks still scattered, but no need to store pointers on disk.
- easy to traverse and update

★ Advantage	★ disadvantage
- Faster than pure LL	- Consume space
- easy to manage	

File A → 5, 9, 1, 12, 7

File B → 3, 8, 4, 14, 6

physical	Next block	
0		
1	12	
2		
3	8	← File B begins here
4	14	
5	9	← File A begins here
6	-1	
7	-1	
8	4	
9	1	
10		
11		
12	7	
13		
14	6	
15		

(4) Indexed Allocation:

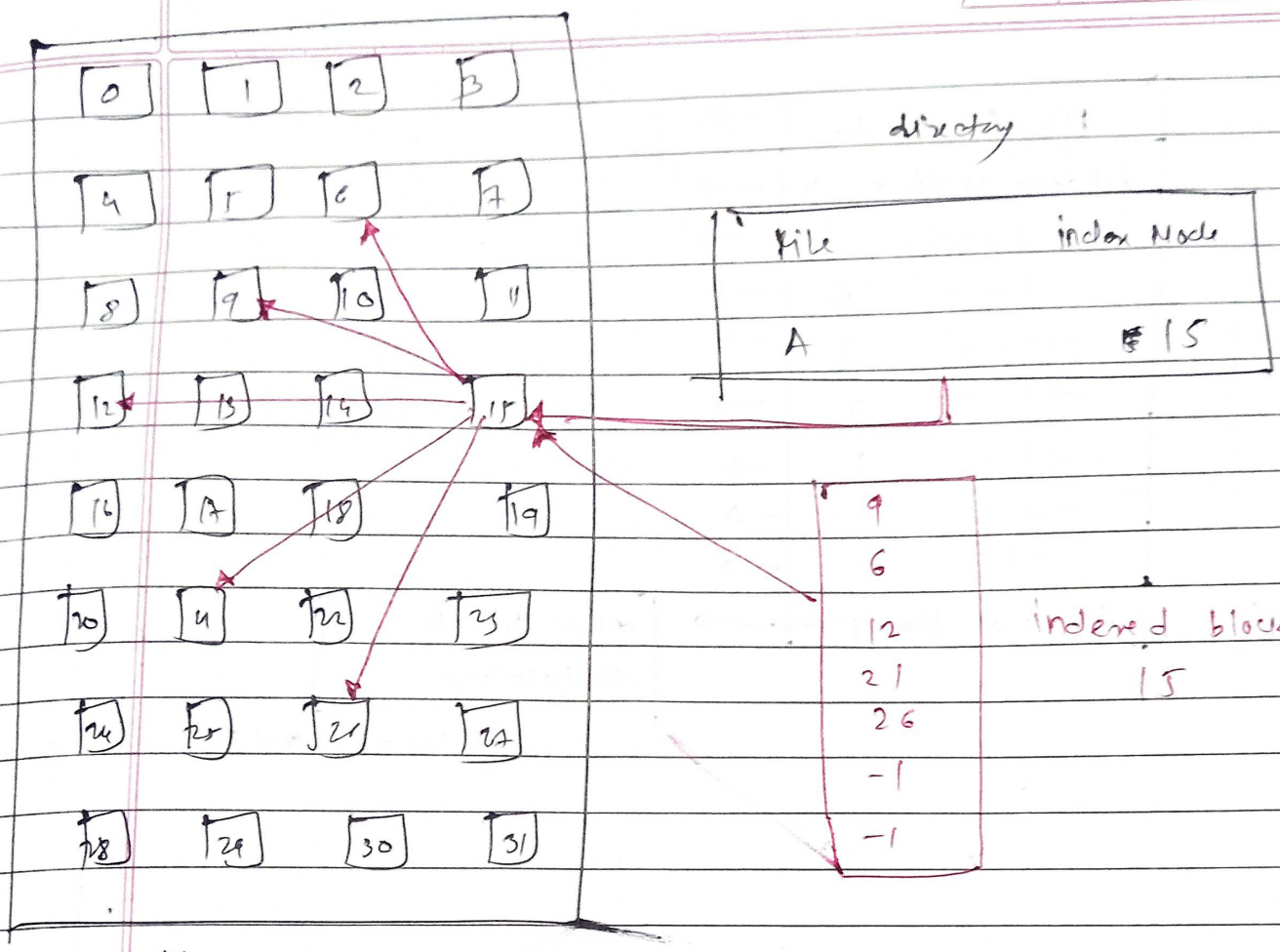
- each file has indexed block that store the address of all data blocks
- Direct access becomes very efficient
- if a file is large, multiple index block can be used

★ Advantage

- support direct access
- No fragmentation

★ disAdvantage

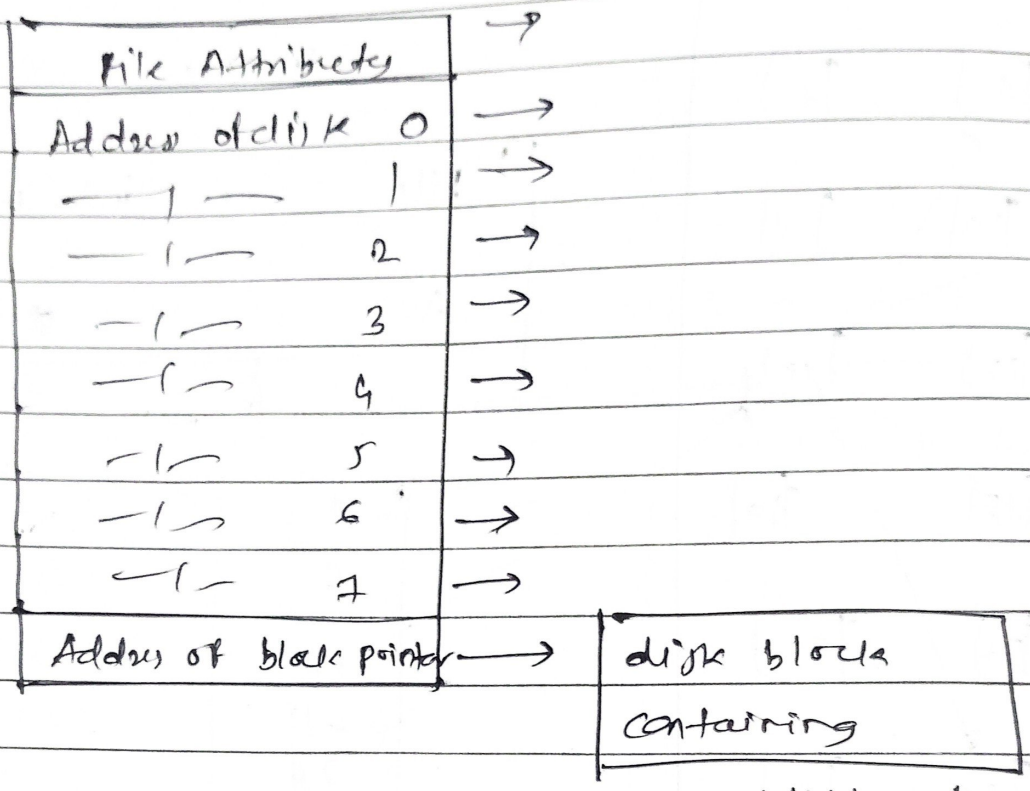
- limited file size unless multiple indexing used.



(5) I-Nodes

- each file has an i-node (index node) containing:
  - file metadata (size, owner etc)
  - pointers to data blocks
- efficient support small files via direct pointers

Advantage	disadvantage
- very flexible and powerful	- Accessing indirect or doubly indirect blocks adds overhead
- efficient for small and large files	


File Attributes
Address of disk 0
— " — 1
— " — 2
— " — 3
— " — 4
— " — 5
— " — 6
— " — 7
Address of block pointer
disk block containing additional disk address

PAGE No.
DATE / /

★ different structure of OS

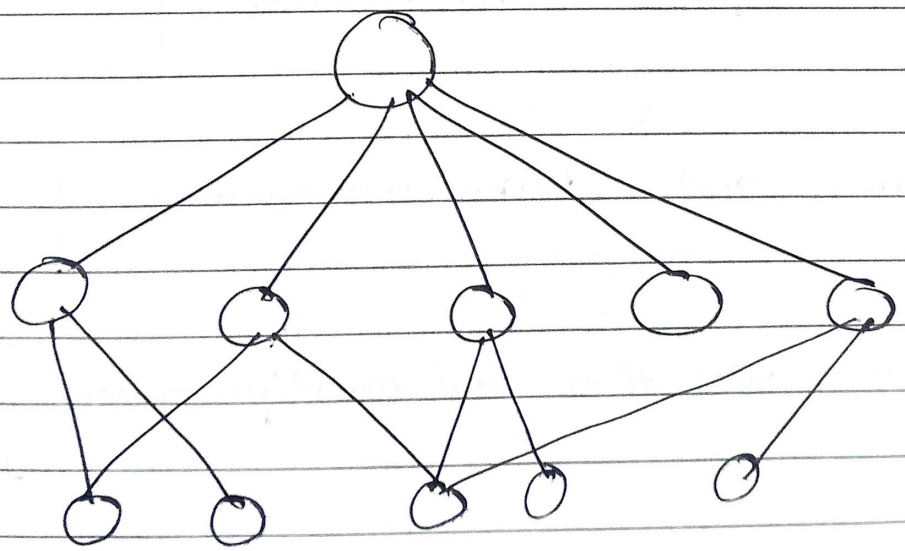
Ans:

Operating system structures

- → Monolithic
- → layered
- → Virtual machines
- → client-server model

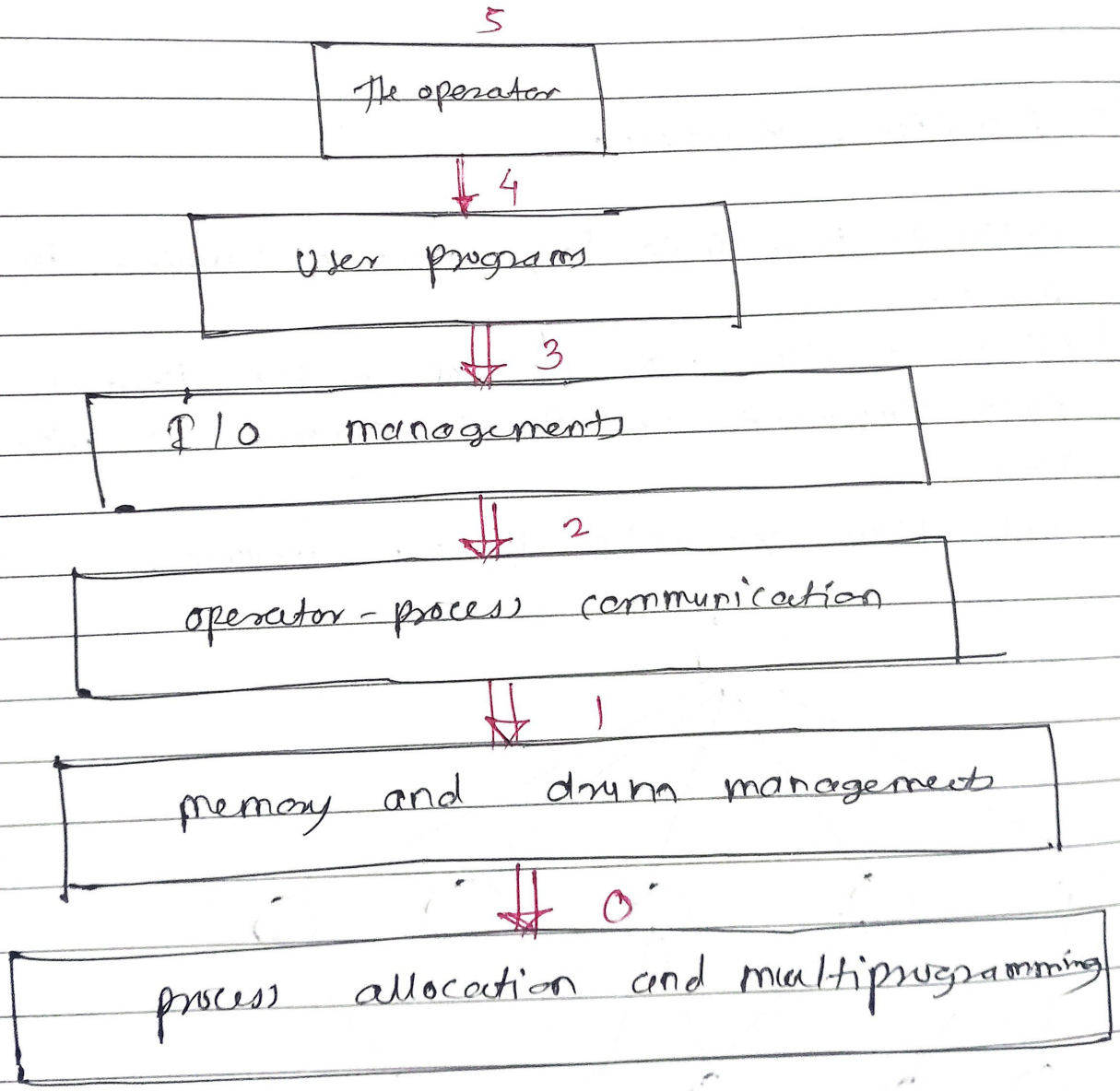
(1) Monolithic structure:

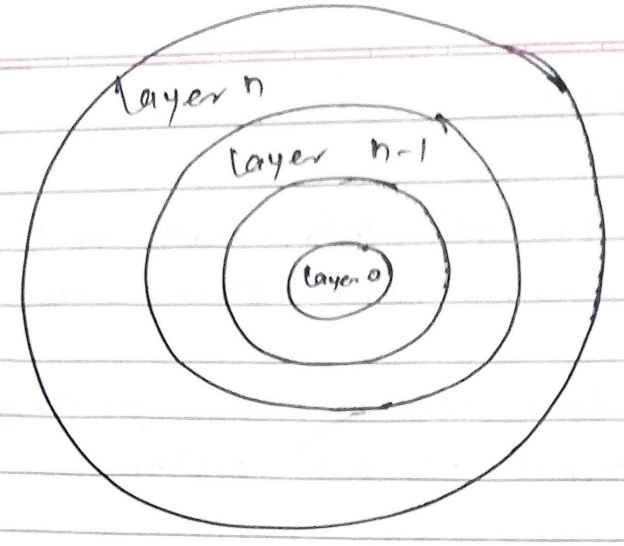
- Entire OS is a single large program running in kernel mode.
- All OS services are part of one big kernel
- each component can directly communicate with any other
- No clear separation between modules.



(2) Layered structure :

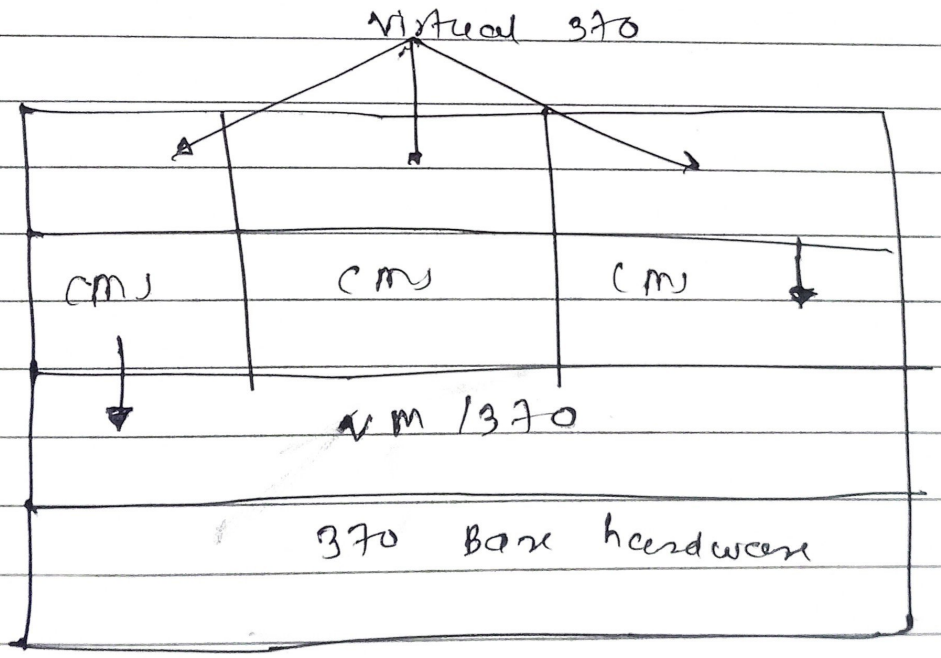
- OS is divided into multiple layers stacked over each other
- each layer use services of the layer below and provides services to the layer above.
- top layer is user interface, bottom layer interacts with hardware.
- ensure clear separation of functionalities.





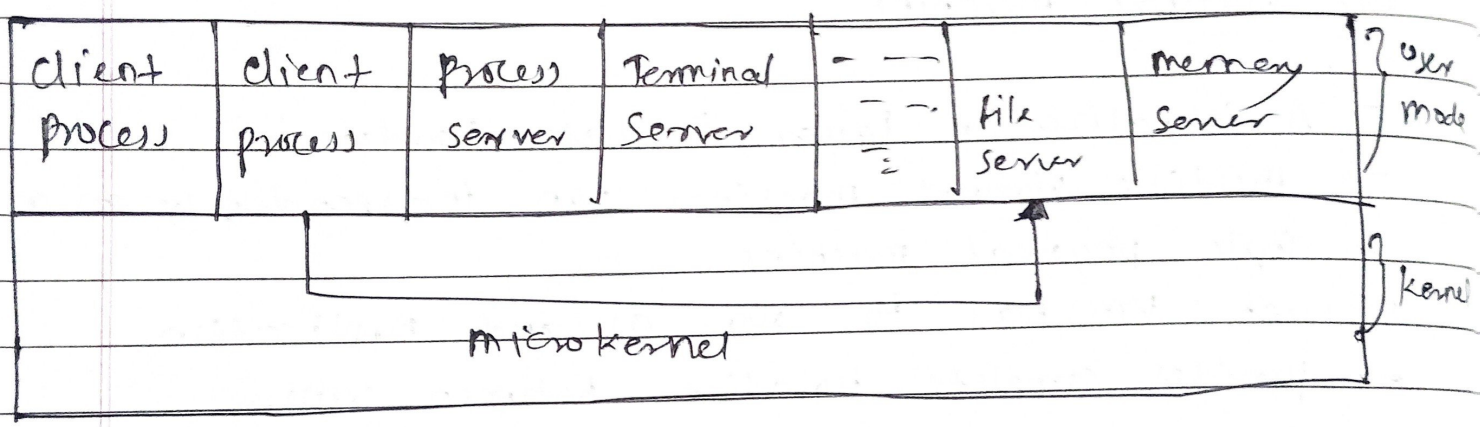
(B) Virtual Machine

- A virtualization layer emulates hardware
- multiple virtual machines run independently on a single physical machine.
- each VM has its own OS and Application
- provides complete isolation between VMs.



(9) Client - Server Model

- OS functionalities are split into client and servers
- client request services, servers process and ~~respond~~ respond
- communication is done via message passing
- common in microkernel - based system.



PAGE No.
DATE / /

Q) Disk organization within the OS

Ans:

Disk organization refers to how data is physically and logically arranged on a disk so that the OS can efficiently store, access, and manage files.

(1) physical structure of disk :

- disk is divided into platters
- each platters has two sides and spine around a central spindle
- Read/write heads move across platters to access data

(2) Disk components :

- platters : circular disk where data is stored using magnetic or electronic methods
- Tracks : Concentric circles on a platters that store data
- Sectors : Small section within a track that represent the smallest data unit.
- Clusters : Group of sectors used to allocate files
- Cylinders : A set of tracks aligned vertically across multiple platters.

- Read / Write Head : Move across the disk surface to read or write data.

- Disk controller : Manages communication between the operating system and disk hardware

(3) Disk formatting :

- low-level formatting : divides disk into sector and tracks

- Logical formatting : creates files system structure

- Partitioning : disk into logical units to install different OS or separate data

(4) File system organization :

- OS uses file system to manage files and directories on disk
- Maintains metadata, free space info and file allocation data.

(5) Disk access techniques :

- OS use disk scheduling algorithm (Name All)

- Minimize seek time and maximize performance

PAGE No.
DATE / /

(6) Buffering and caching:

- Data is temporarily held in buffers or cache to reduce direct disk access

- Improves speed and efficiency

(7) Logical view vs Physical view

- Logical view: How user and program see data

- Physical view: Actual arrangement of bits on the disk platters.