13/2/24

ASSIGNMENT 1

Unit-1&2 Introduction to Operating System & Process and Threads Management.

Q.1 What is Operating System? What are the various services provided by the operating system?

→ An Operating System is a software that acts as an intermediary between computer hardware and the user. It manages computer hardware resources and provides common services for computer programs.

i) Program Development.

It typically provide various tools and utilities to aid in program development, that includes compilers, debuggers, text editors that facilitate the creation, testing and debugging of programs

ii) Program Execution

It loads programs into memory, schedules them for execution on the CPU and provides the necessary runtime support for programs to run correctly.

iii) Access to I/O Devices (Resource Allocation).

It manages access to input/output devices such as keyboards, mice, displays, disk drivers. They allocate these resources to different programs and ensure that multiple programs can use them concurrently without conflicts.

iv) Memory Management

It manages computer memory, which includes RAM and secondary storage. They allocate memory to running programs, handle memory swapping and paging.

v) Controlled Access to Files.

It provides a file system that organizes and manages data stored on disk. They enforce access controls to files and directories, ensuring that only authorized users can access or modify.

vi) Communication

It provides mechanisms for inter-process communication (IPC) to allow communication and data exchange between different programs running on same system

vii) Error Detection and Response

It monitors the system for errors and exceptions such as hardware faults, software crashes. They provide mechanisms for error detection, handling and recovering recovery to prevent system failures and ensure the reliability of the system.

viii) Accounting

It may include accounting mechanisms to track resource usage by different users,

which includes monitoring CPU usage, memory usage, disk I/O.

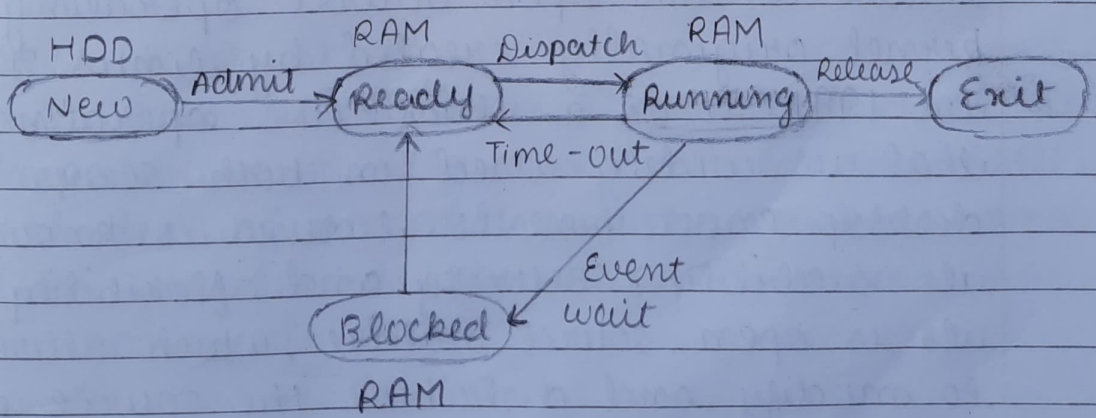
ix) Protection and security

It enforces security policies to protect the system and its resources from unauthorized access, malicious software, This ~~which~~ includes user authentication, encryption access control mechanisms.

Q.2 What is process? Explain five state process model and explain each state transition?

-> A process is an instance of a program that is being executed. It consists of the program code, ~~be in during~~ its current activity and set of ~~an~~ system resources allocated to it.

-> The Five-State Process Model is a conceptual model used to describe the various states that a process can be in during its lifecycle.



- State Transitions :-

1. New → Ready : Occurs when the process is initialized and ready to execute, waiting for CPU allocation.

2. Ready → Running : Occurs when the scheduler allocates CPU times to the process.

3. Running → Blocked : Occurs when the process needs to wait for an event such as I/O completion or user input.

4. Running → Ready : Occurs when the event, the process waiting for happens and it becomes to ready to execute again.

5. Running → Exit : Occurs when the process completes its execution or is terminated.

Q.3 What is linux OS ? What is shell ? What are the different types of shell in linux ?

—ᵛ Linux is an open-source operating system kernel originally created by Linus Torvalds in 1991. It is a Unix-like operating system that is widely used in both server and desktop environments. Linux is known for its stability, security and flexibility. Also its is open-source nature, which allows users to modify and distribute the source code.

→ A shell is a command-line interpreter that allows users to interact with the operating system by typing commands. It provides a text-based interface for executing commands, managing files and performing various system tasks.

→ Different types of shells are:

i) Bash (Bourne Again Shell)

It is the default shell on most linux distributions. It is a powerful and versatile shell that is backward-compatible with Original Unix Shell (sh) and includes features like command-line editing, history manipulation and job control.

ii) Csh (~~C~~ C Shell)

Csh is a shell designed for interactive use and includes features such as command-line editing and history substitution. It has a syntax similar to C programming language.

iii) ~~K~~Ksh (Korn Shell)

Ksh is another Unix shell that is compatible with the original Bourne Shell. It includes advanced scripting features and is often used in commercial Unix environments.

Q.4 What is thread? Explain the difference between process and thread.

→ A Thread is the smallest unit of execution within a process. It represents a single sequence of instructions that can be executed independently alongside other threads within the same process. They share the same memory space and resources of the process for which they belong, allowing them to efficiently perform parallel tasks.

→

Process	Thread
A Process is an instance of a program in execution which contains the program code, data & resources.	A Thread is a single sequence of execution within a process.
Each process has its own memory space, file handles and other resources.	Threads within the same process share the same memory space and resources.
Inter-process communication (IPC) mechanism are required for communication between processes.	Threads can communicate directly through shared variables or message passing.

› Processes are independent entities, and failure of one process does not affect others.	Threads within the same process are dependent and a failure in one thread can affect the entire process.
› Creating a new process is typically more resource-intensive and slower.	Creating a new thread within an existing process is faster and requires fewer resources.

Q.5 What are the user level threads and kernel level threads ?

→ i) User-Level Threads (ULTs)

› They are managed entirely by user-level libraries or applications without kernel support.

› Thread management functions such as thread creation, scheduling and synchronization are implemented using user-level libraries or runtime environments.

› ULTs are typically faster to create and manage because they do not involve kernel ~~involvement~~ for thread operations.

› The Operating system is unaware of the existence of user-level threads and treats the entire process as a single entity.

iii) Kernel - Level Threads (KLTs)

> They are managed directly by the operating system kernel.
> Thread management functions are implemented within the kernel, allowing for more efficient scheduling, resource management and synchronization.
> Creating and managing kernel - level threads typically incur higher overhead due to involvement of kernel operations.
> The kernel schedules threads independently, allowing for better utilization of multi-processor systems and providing finer-grained control over thread execution.

Q.6 What is the use of process control block? Discuss the content of PCB. Discuss how the PCB's are chained together to form a list of ready processes.

→ The Process Control Block (PCB) is a data structure used by the ~~kernel~~ operating system to manage information about each process in the system.

Usage of Process Control Block :-

i) Process Management : It store information about each process, including its state, program counter, CPU registers which facilitates process creation, execution, and termination

ii) Scheduling : It stores scheduling information such as process priority, scheduling tasks status and CPU scheduling parameters.

iii) Context Switching : When the OS switches from executing one process to another, it saves the state of the currently running process in its PCB and loads the state of the next process from its PCB.

* Other usages are Resource Management, interprocess communication.

→ Contents of PCB are :-

i) Process Identifier (PID) : Unique identifier assigned to each process.

ii) Process state : Indicates whether the process is ready, running, blocked or Terminated

iii) Accounting Information : Statistics related to CPU usage, execution time, etc.

iv) Open File Pointers : Pointers to files opened by the process.

v) Process Priority : Priority level assigned to the process for scheduling purposes.

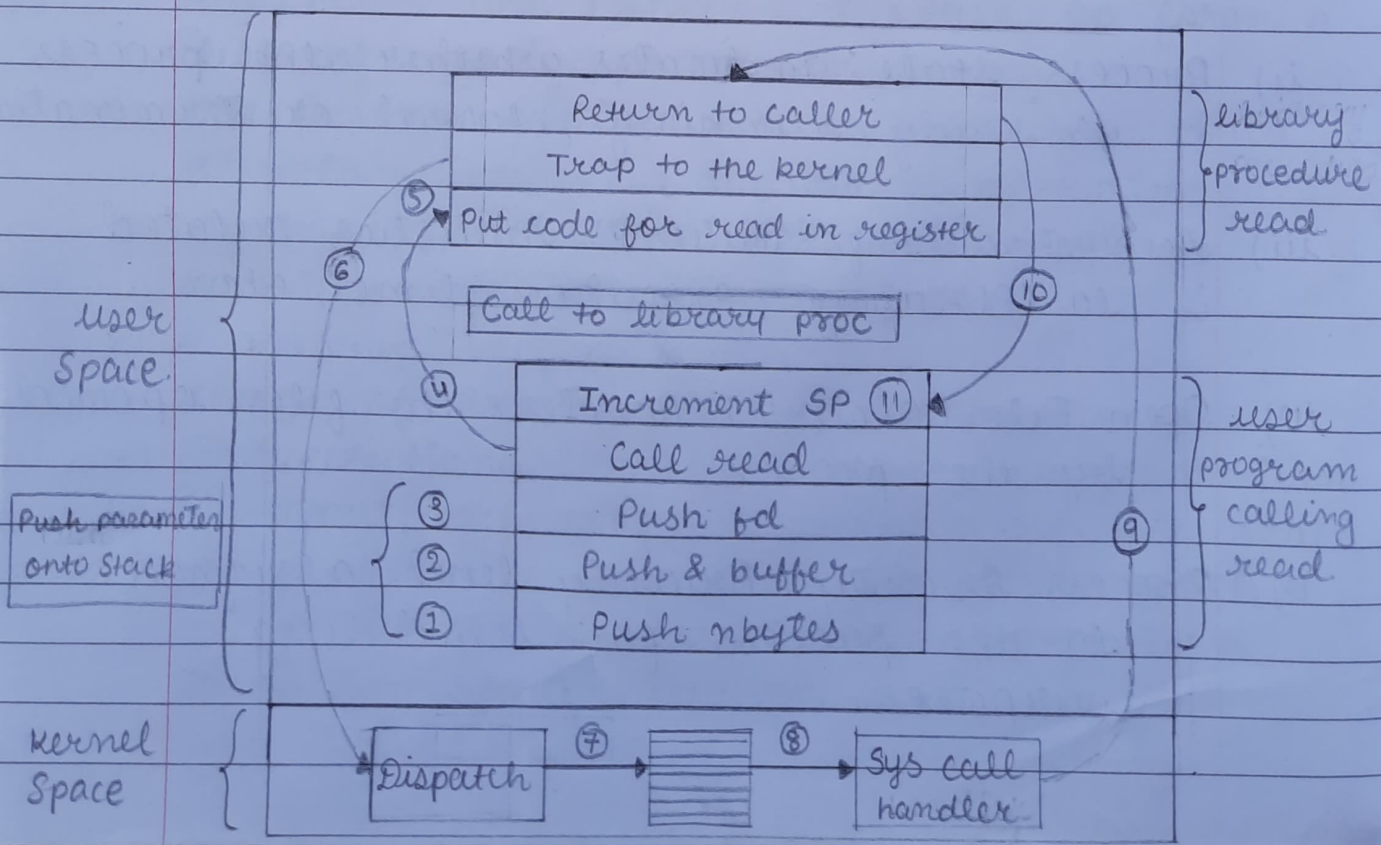
vi) Program Counter (PC): Address of the next instruction to be executed.

vii) Memory Management Information: Details about the memory allocated to the process.

—↓ The PCBs of ready processes are often organized into a list or queue known as the "Ready Queue". This ready queue helps the operating system efficiently in managing and scheduling processes for execution on the CPU. The PCBs are chained together within this queue using pointers.

Q.7 Explain with the help of diagram how System call will be is generated.

—↓



1-3: Calling program pushes the parameters onto the stack. The first and third parameters are called by value but the second one is called by its address as denoted by '&' symbol.

4: Actual call to the library procedure is made. This instruction is the normal procedure call instruction used to call all procedures.

5: Library procedure places the system call number in a place where the operating system expects it, such as a register.

6: Library procedure executes a TRAP instruction to switch from user mode to kernel mode and start execution at a fixed address within the kernel.

7: Kernel examines the system call number and then dispatches it to the correct system call handler.

8: System call handler runs.

9: Operation is completed and user is given back control once the TRAP instruction is set.

10: Procedure returns to user program

11: Operating system has to clear the stack. so it increments it enough so that it is empty.

Q.8 Describe microkernel and monolithic kernel structure.

—4 Microkernel and Monolithic kernel are two different approaches to designing the architecture of an operating system kernel.

i) Microkernel

Here, the kernel is kept minimalistic, and only essential services such as process management, memory management and inter-process communication (IPC) are implemented within the kernel, while other services run as user-space processes.

Communication between different components of the operating system is done through message passing, which allows for better modularity and isolation.

They are also more robust since a failure in one component does not necessarily crash the entire system. ~~However, they can suffer~~

However, they can suffer from lower performance due to the overhead of message passing between user-space servers and the kernel.

ii) Monolithic Kernel

Here, the entire operating system is implemented as a single large kernel running in a privileged mode. All kernel

services, including process management, memory management, device drivers, file systems are implemented as part of the kernel.

Communication between different parts of the kernel is typically done through direct function calls and shared data structures.

They are relatively simple in design and provide fast system call performance since there is minimal overhead for inter-module communication.

However, they can be large and complex, making them harder to maintain and extend. Additionaly, a bug or error in one part of the kernel can potentially crash the entire system.

Q.9 What is the need of thread? Describe any four advantages of multithreading model. -4

A Thread is the smallest unit of execution within a process i.e. a sequence of instructions that can be executed independently by the CPU.

They are essential for achieving concurrency within a program, allowing multiple tasks to run simultaneously within a single process. Also enables efficient utilization of CPU resources, facilitate responsiveness to user input.

- Advantages of Multithreading Model :-

i) Increased Throughput

By leveraging multiple threads to execute tasks concurrently, multithreaded applications can achieve higher throughput and better utilization of available CPU resources.

ii) Resource Sharing

Threads within the same process can share resources such as memory, file handles and network connections, reducing the overhead of resource allocation and context switching.

iii) Improved Responsiveness.

Multithreading allows an application to remain responsive to user input or external events while performing other tasks in the background.

iv) Simplified Programming

Multithreading simplifies the development of complex applications by breaking them into smaller, more manageable units of execution. Each thread can focus on a specific task, making the overall program more modular and easier to understand.

Q.10 Write short note on
i) Multiprogramming OS.

→ Multiprogramming Operating System refers to a technique where multiple programs reside in memory simultaneously and ~~enchan~~ enhances CPU utilization by keeping it busy with tasks during idle times. The CPU switches rapidly between executing different programs, giving the illusion that they are all running simultaneously.

- Memory Management : It manages the memory allocation for each program, ensuring that programs do not interfere with each other's memory space.
- Process Management : It oversees the creation, scheduling and termination of processes. Its allocates CPU time to each process, ensuring fair and efficient execution.
- Resource Sharing : It facilitates the sharing of system resources among multiple processes which includes CPU time, memory and I/O devices.
- Fault Tolerence : It include mechanisms for fault tolerence, such as process isolation and error recovery, to ensure system stability and reliability in the face of hardware failures or software errors.

ii) Priority Based Process Scheduling

→ Priority based process scheduling is a scheduling algorithm to determine the order in which processes are executed on the CPU based on their priority levels. Each process is assigned a priority value that indicates its relative importance.

- Scheduling Policy Priority levels: The scheduler selects the next process to execute based on its priority levels.
- Preemption: Priority-based scheduling may be preemptive or non-preemptive. In preemptive scheduling, the scheduler can interrupt the execution of a lower priority process to allocate CPU time to higher priority process. In Non-preemptive scheduling, ~~the~~ a process continues to execute until it completes its execution.
- Starvation: Priority levels should be managed to prevent starvation, where low-priority processes may never get the chance to execute if higher-priority processes continually monopolize the CPU.
- Dynamic or Static Assignment: Priority levels can be assigned to processes either statically or dynamically.

Q.11 Consider Five Processes P1 to P5 arrived at same time. They have estimated running time 10, 2, 6, 8 and 4 seconds, respectively. Their priorities are 3, 2, 5, 4 and 1, respectively with 5 being highest priority. Find the average turnaround time and average waiting time for Round Robin (quantum time = 3) and priority scheduling algorithm.

Ans

i) Round Robin

CT-AT TAT-BT

Process ID	B.T.	C.T.	T.A.T	W.T	
P1	10	30	30	20	10, 7, 4, 1, 0
P2	2	5	5	3	2, 0
P3	6	20	20	14	6, 3, 0
P4	8	29	29	21	8, 5, 2, 0
P5	4	24	24	20	4, 1, 0

∴ Ready Queue :- [TQ = 3]

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

∴ Gantt Chart :-

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

0 3 5 8 11 14 17 20 23 24 27 29 30

∴ Average Turnaround Time : (30+5+20+29+24)/5 = 21.6 ms

∴ Average waiting Time : (20+3+14+21+20)/5 = 15.6 ms

ii) Priority Scheduling Algorithm

CT-AT TAT-BT

Process ID	B.T.	Priority	C.T.	T.A.T	W.T
P1	10	3	24	24	14
P2	2	2	26	26	24
P3	6	5	6	6	0
P4	8	4	14	14	6
P5	4	1	30	30	26

Gantt Chart :-

P3	P4	P1	P2	P5
0	6	14	24	26	30

Avg. TAT : (24+26+6+14+30)/5
= 20 ms

Avg. WT : (14+24+0+6+26)/5
= 14 ms

Q.12 Consider the processes P1, P2, P3, P4 with burst time is 21, 3, 6 and 2 respectively, arrives for execution in the same order, with arrival time 0, draw GANTT chart and find the average waiting time using the FCFS and SJF scheduling algorithm.

→ i) FCFS (First Come First Serve)

CT-AT TAT-BT

Process ID	B.T.	C.T.	T.A.T	W.T
P1	21	21	21	0
P2	3	24	24	21
P3	6	30	30	24
P4	2	32	32	30

Gantt Chart :-

P1	P2	P3	P4
0	21	24	30	32

Avg. TAT : (21+24+30+32)/4 = 26.75 ms

Avg. WT : (0+21+24+30)/4 = 18.75 ms

ii) SJF (Shortest Job First)

CT-AT TAT-BT

Process ID	B.T.	C.T.	T.A.T	W.T
P1	21	32	32	11
P2	3	5	5	2
P3	6	11	11	5
P4	2	2	2	0

Gantt Chart :-

Avg. TAT : (32+5+11+2)/4 = 12.5 ms

P4	P2	P3	P1

0 2 5 11 32

Avg. WT : (11+2+5+0) = 4.5 ms

Q.13 Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 & 8 time units. All processes arrive at time zero. Consider the longest Remaining Time First (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. What is average turn around time and waiting time?

-4 LRTF (Longest Remaining Time First)

CT-AT TAT-BT

Process ID	B.T.	C.T.	T.A.T	W.T	
P0	2	12	12	10	2,1,0
P1	4	13	13	9	4,3,2,1,0
P2	8	14	14	6	8,7,6,5,4,3,2,1,0

Gantt Chart :-

Avg. TAT : (12+13+14)/3 = 13 ms

Avg. WT : (10+9+6)/3 = 8.3 ms

P2	P2	P2	P2	P1	P2	P1	P2	P0	P1	P2	P0	P1	P2

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

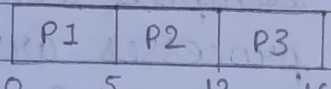
Q.14 Consider the 3 processes P1, P2 and P3 shown in the table. The completion order of the 3 processes under the policies FCFS, SJF and RR2 (round robin scheduling with CPU quantum time of 2 units). Calculate average response time ATAT, AWT?

—4 i) FCFS (First Come First Serve)

CT-AT, TAT-BT RQ-AT

Process	A.T	B.T	C.T.	T.A.T	W.T	R.T.
P1	0	5	5	5	0	0
P2	1	7	12	11	4	4
P3	3	4	16	13	9	9

• Gantt Chart :-



Avg. TAT : (5+11+13)/3 = 9.6 ms

Avg. WT : (0+4+9)/3 = 4.3 ms

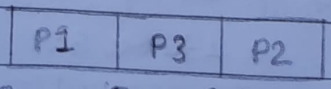
Avg. RT : (0+4+9)/3 = 4.3 ms

ii) SJF (Shortest Job First)

CT-AT. TAT-BT RQ-AT

Process	A.T.	B.T.	C.T.	T.A.T	W.T	R.T.
P1	0	5	5	5	0	0
P2	1	7	16	15	8	8
P3	3	4	9	6	2	2

• Gantt Chart :-



Avg. TAT : (5+15+6) = 8.6 ms

Avg. WT : (0+8+2) = 3.3 ms

Avg. RT : (0+8+2) = 3.3 ms

iii) RR2 (Round Robin)

CT-AT TAT-BT RQ-AT

Process	A.T.	B.T.	C.T.	T.A.T	W.T	R.T
P1	0	5	11	11	6	0
P2	1	7	16	15	8	1
P3	3	4	13	10	6	3

Ready Queue :- [TQ = 2]

P1	P2	P1	P3	P2	P1	P3	P2	P2

P1 → 5, 3, 2, 0
P2 → 7, 5, 3
P3 → 4, 2, 0

Gantt Chart :-

P1	P2	P1	P3	P2	P1	P3	P2	P2

0 2 4 6 8 10 11 13 15 16

Avg. TAT : (11+15+10)/3 = 12 ms

Avg. WT : (6+8+6)/3 = 6.6 ms

Avg. RT : (0+1+3)/3 = 1.3 ms

Q.15 Explain Structure of Operating System.

—ϕ The structure of an operating system can be conceptualized in different ways depending on the level of abstraction and the specific components being focused on.

i) Monolithic Systems

Here, all operating system services such as process & memory management, file system handling and device drivers are packaged into a single cohesive unit called the kernel. The kernel directly interacts with hardware and provides a wide range of functionalities.

ii) Layered Systems

Layered operating systems are organized into a hierarchy of layers, each responsible for a specific set of functions. These layers interact with each other in a sequential manner, with higher layers calling upon the services of lower layers. This modular approach enhances maintainability and flexibility.

iii) Microkernel

A Microkernel architecture keeps the kernel as small as possible, delegating most tasks such as device drivers and file system management, to user space processes. It primarily focuses on providing essential services like IPC, thread management and memory allocation.

iv) Client - Server Model

Here, the operating system is divided into two separate entities: the client and the server. The client handles user interface and application execution, while the server manages system resources and provide services like file access, printing and network communication.

v) Virtual Machines

They create multiple instances of a

virtual machine monitor (VMM), also known as a hypervisor, on top of physical hardware. The VMM abstracts and manages physical resources, allowing multiple operating systems to run concurrently on the same physical hardware.

(vi) Exokernels.

Exokernels take the microkernel concept further by providing minimal abstractions over hardware resources directly to applications. It enables application to directly manage hardware resources. This approach aims to improve performance and flexibility.

Q.16 Discuss role / functions of OS as a resource manager.

-4 As a resource manager, the operating system plays a crucial role in efficiently allocating and managing system resources to ensure optimal utilization, performance and reliability.

- Process Management: The OS creates and terminates processes, allocating resources and scheduling it for execution using various scheduling algorithm
- Memory Management: The OS allocates memory to processes, managing both physical

and virtual memory to provide each process with its memory space.

- File System Management: The OS organizes data into files and directories, managing file metadata, access permissions and directory structures.
- Device Management: The OS interacts with hardware devices through device drivers, which translate high-level commands from the OS into device-specific operation.
- Network Management: The OS provides networking protocols and services for communication between systems and devices over a network.

Q.17 Explain the different types of operating systems.

→ Different types of operating systems are:-

i) Mainframe Operating System

They are designed for large-scale computing environments, typically used by organizations for critical applications such as banking, airlines reservations. Example IBM's z/OS and UNISYS OS 2200

ii) Server Operating System

They are tailored for managing network resources and providing services

to clients over a network. They prioritize stability, security and scalability. Examples includes Windows Server, Linux and Unix variants like FreeBSD.

iii) Multiprocessor Operating Systems

These are optimized to manage multiple processors efficiently. They distributes tasks among processors, synchronize their activities and handles communication between them. Example Linux, Unix variants and windows server.

ii) Personal Computer Operating Systems.

These are designed for individual users and support a wide range of applications. They provide GUIS for ease of use. Example macOS, Microsoft Windows, Ubuntu, Fedora and Debian.

v) Handheld Computer Operating Systems.

They are optimized for small, portable devices like smartphones and tablets. They prioritize power efficiency, touch-based interfaces. Example iOS, Andraid.

vi) Embedded Operating Systems

They are built into devices and appliances to manage specific functions.

They are optimized for efficiency, reliability and real-time performance. Example Embedded Linux, FreeRTOS, VxWorks.

vii) Sensor Node Operating Systems.
They are designed for sensor networks, where numerous small devices gather data and communicate with each other. They prioritize energy efficiency, scalability and support for sensor-specific protocols. Example TinyOS, Contiki.

viii) Real-Time Operating Systems (RTOS)
They are used in systems where timely processing is critical like robotics, aerospace, automation. They prioritize quick and predictable response times to events. Example VxWorks, QNX, FreeRTOS.

ix) Smart Card Operating Systems.
They are optimized for smart cards, which are small, portable devices with embedded integrated circuits. They manage security features and facilitates interactions with card readers. Example Java Card OS and MULTOS.

Q.18 What is RAG? Explain briefly.
—4 RAG stands for Resource Allocation Graph, which is a data structure used in

operating systems to represent the resource allocation and resource request relationships between processes in a system.

- In a Resource Allocation Graph :-

1. Nodes represent two types of entities :
   - Processes : Represented by circles or rectangles.
   - Resources : Represented by squares or rectangles.

2. Edges represent two types of relationships :
   - Allocation Edge : Indicates that a process currently holds a resource.
   - Request Edge : Indicates that a process is requesting a resource.

- The structure of the graph reflects the current state of resource allocation and resource requests in the system. By analyzing the graph, it is possible to detect potential deadlock situations, where processes are waiting indefinitely ~~or~~ for resources that are held by other processes, forming a cycle in a graph.

Q.19 What is deadlock? Explain necessary and sufficient conditions for deadlock to occur.

→ Deadlock is a situation where two or

more processes are unable to proceed with their execution because each is waiting for a resource held by another process in the same set. ~~Necessary conditions~~

→ Necessary Conditions for Deadlock :-

i) Mutual ~~Execution~~ Exclusion

At least one resource must be held in a non-sharable mode, meaning only one process can use it ~~as~~ at a time and other processes must wait for it to be released.

ii) Hold and Wait

Processes must hold at least one resources and be waiting to acquire additional resources that are held by other processes. (They are granted earlier for the processe holding resources)

iii) No Preemption

Resources cannot be forcibly taken away from processes; they must be voluntarily released by the process holding them.

iv) Circular Wait

There must exist a circular chain

of two or more processes, each holding a resource that is requested by the next process in the chain.

- If all the necessary conditions are met, deadlock becomes a potential threat (but it does not guarantee it). But if any one condition is not satisfied, then deadlock cannot occur.

Q.20 Explain how deadlock can be prevented.
-4 Deadlock prevention involves designing systems and implementing strategies to eliminate one or more of the necessary conditions for deadlock to occur.

i) Attacking the Mutual Exclusion Condition.
Ensure that resources are designed to be shareable whenever possible. This eliminates the mutual exclusion condition by allowing multiple processes to access the same resources simultaneously.
Applicable where resources can be safely shared among processes without causing conflicts.

ii) Attacking Hold and Wait Condition.
Require processes to request and acquire all necessary resources before starting execution. If a process cannot

acquire all resources at once, it releases all currently held resources and retries the request later. Problem with this strategy is that a process may not know required resources at start of run. So resources will not be used optimally.

iii) Attacking the No Preemption Condition

Allow the operating system to preemptively revoke resources from processes if necessary. When a process requires additional resources that cannot be immediately allocated, it releases all currently held resources and waits until all required resources are available simultaneously. However, preemption must be carefully managed to avoid unfairness and potential data corruption issues.

iv) Attacking circular wait condition.

Implement a resource allocation hierarchy, where resources are assigned unique numerical ordered ranks. Processes are required to acquire resources in a predefined order, preventing circular dependencies and ensuring that deadlock cannot occur.

Q.21 Consider the following situation of system and Answer the following

questions using Banker's Algorithm.

i) What is the content of need matrix.

—↳

Process	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	4	2	0	0	0	0	0
P1	1	0	0	0	1	7	5	0	1	4	3	2	0	7	5	0
P2	1	3	5	4	2	3	5	6	2	7	8	6	1	0	0	2
P3	0	6	3	2	0	6	5	2	2	13	11	8	0	0	2	0
P4	0	0	1	4	0	6	5	6	2	13	12	12	0	6	4	2

* Total Resources : A = 3 B = 13 C = 12 D = 12

> Need[i] = Max[i] - Allocation[i]

—↳ Need for 'P0' : (0,0,1,2) - (0,0,1,2) = (0,0,0,0)
Need for 'P1' : (1,7,5,0) - (1,0,0,0) = (0,7,5,0)
Need for 'P2' : (2,3,5,6) - (1,3,5,4) = (1,0,0,2)
Need for 'P3' : (0,6,5,2) - (0,6,3,2) = (0,0,2,0)
Need for 'P4' : (0,6,5,6) - (0,0,1,4) = (0,6,4,2).

ii) Is the system in safe state ? Give the state sequence.

—↳ Safe State Sequence : P0 → P2 → P3 → P4 → P1

1. For Process 'P0',
Need ≤ Available : (0,0,0,0) ≤ (1,4,2,0)
Condition is TRUE, so we execute process 'P0'.
New Available = Available + Allocation
= (1,4,2,0) + (0,0,1,2) = (1,4,3,2)

* Process 'P4' continue at pg - 36.

2. For Process 'P1':
Need ≤ Available : (0,7,5,0) ≤ (1,4,3,2)
Condition is FALSE, so we examine process 'P2'.

3. For Process 'P2':
Need ≤ Available : (1,0,0,2) ≤ (1,4,3,2)
Condition is TRUE, so we execute process 'P2'.
∴ Providing Resources = Available - Need
= (1,4,3,2) - (1,0,0,2)
∴ left = (0,4,3,0)
After execution, resources are freed
∴ New Available = left + Max
= (0,4,3,0) + (2,3,5,6)

4. For Process 'P3':
Need ≤ Available : (0,0,2,0) ≤ (2,7,8,6)
Condition is TRUE, so we execute process 'P3'.
∴ Providing Resources = Available - Need
= (2,7,8,6) - (0,0,2,0)
∴ left = (2,7,6,6)
After execution, resources are freed
∴ New Available = left + Max
= (2,7,6,6) + (0,6,5,2)
= (2,13,11,8).

5. For Process 'P4':
Need ≤ Available : (0,6,4,2) ≤ (2,13,11,8)
Condition is TRUE, so we execute process 'P4'.

Q.22 Explain Deadlock recovery in brief.

—4 Deadlock recovery involves resolving situations where multiple processes are unable to proceed because each is waiting for the other to release a resource.

i) Recovery through Preemption.

Here, the operating system forcibly interrupts one or more processes to break the deadlock. It involves suspending one or more processes temporarily to release the resources they hold, allowing other processes to proceed. Once the deadlock is resolved, the preempted processes can be restarted. Recovering this way is frequently difficult or impossible.

ii) Recovery through Rollback.

This method involves rolling back the progress of one or more processes to a previous state where deadlock did not occur. It typically requires maintaining checkpoints or transaction logs to facilitate the rollback process.

iii) Recovery through Killing Processes.

Here, the operating system terminates one or more processes involved in the deadlock to break the deadlock. The decision of which processes to terminate may be based on priority, resource usage, etc. By killing selected processes,

the system frees up the resources they were holding, allowing other process to proceed without deadlock.

Q.21 → Continue for Process 'P4':

∴ Providing Resources = Available - Need
= (2,13,11,8) - (0,6,4,2)
∴ left = (2,7,7,6)

After execution, resources are freed.
∴ New Available = left + Max
= (2,7,7,6) + (0,6,5,6)
= (2,13,12,12)

6. Again For Process 'P1':
Need ≤ Available : (0,7,5,0) ≤ (2,13,12,12)
Condition is TRUE, so we execute process 'P1'.

∴ Providing Resources = Available - Need
= (2,13,12,12) - (0,7,5,0)
∴ left = (2,6,7,12)

After execution, resources are fixed
∴ New Available = left + Max
= (2,6,7,12) + (1,7,5,0)
= (3,13,12,12)

iii) If the request from process P1 arrives for (0,4,2,0) can it be granted immediately?

→ Yes; because Need ≤ Available, it can be
(0,4,2,0) (1,4,2,0) granted