Oprating system 2081/82

1. What is operating system ?explain its function in brief

_ operating system is a collection of set of software which manages all the gesources of the computer system it is an interediater between the user and hardware

Function of operating system

1. memory management

- memory management module performs the task of allocttion and dealloction of memory space to programs in need of this resources

2. processor management

- the operating system assigns processors to the different tasks that must be performed by the computer system

3. device management

- Operating system performs the task of allocation of the device

4. file management

- operating system manage all file related activities such as organization stoage retrieval naming sharing and protection of file

5. security management

Security management function of an operating system help in implementing mechanisms that secure and protect the computer system internally as well as externally

6. job scheduling

- job scheduling is the process of allocationg system resources to many different tasks by an operating system

7.time sharing

-It co-ordinates and assighs comilers assemblers utlity program and other software packages to various user working on computer system

2. what is producer consumer problem write the solution of producer consumer problem

- The producer-consumer problem (or bounded-buffer problem) is a classic operating system synchronization issue where a producer process generates data and stores it in a fixed-size buffer, while a consumer retrieves it. The core problem is preventing the producer from adding data when the buffer is full and ensuring the consumer does not read from an empty buffer, usually solved using semaphores (mutex, full, empty) to manage concurrent access and buffer state.

Key Components of the Problem:

- **Producer:** Produces data, waits if the buffer is full, and signals after producing.
- **Consumer:** Consumes data, waits if the buffer is empty, and signals after consuming.
- **Buffer:** A shared memory area of limited capacity.

Solution to Producer-Consumer Problem (Using Semaphores)
The most effective solution uses three semaphores to ensure proper synchronization and mutual exclusion:

1. **`mutex` (Binary Semaphore):** Ensures that only one process (producer or consumer) accesses the buffer at a time, initialized to 1
2. **`empty` (Counting Semaphore):** Tracks empty spaces in the buffer, initialized to the buffer size Ncap N
3. **`full` (Counting Semaphore):** Tracks filled slots in the buffer, initialized to 0

`3. define the term paging . explain different memory alloction strategies`

- **Paging** is a **memory management technique** in which the **logical address space** of a process is divided into fixed-size blocks called **pages**, and the **physical memory** is divided into fixed-size blocks called **frames**.

Pages are loaded into any available frames in main memory, and a **page table** is used to map pages to frames.

Advantages of Paging

- Eliminates **external fragmentation**
- Efficient use of memory
- Supports **virtual memory**
- Simplifies memory allocation

Memory Allocation Strategies

Memory allocation strategies decide **how memory is allocated to processes** in main memory.

1. First Fit Allocation

- Allocates the **first free memory block** that is large enough
- Stops searching once a suitable block is found

Advantages:

- Simple and fast
- Less searching time

Disadvantages:

- Causes **external fragmentation**

2. Best Fit Allocation

- Allocates the **smallest free block** that is sufficient for the process

Advantages:

- Reduces wasted memory

Disadvantages:

- Slow (searches entire memory list)
- Creates many small unusable fragments

3. Worst Fit Allocation

- Allocates the **largest available memory block**

Advantages:

- Leaves large blocks for future use

Disadvantages:

- Wastes large memory space
- Not efficient in practice

4. Next Fit Allocation

- Similar to First Fit

- Continues searching from the **last allocated position**

Advantages:

- Faster than First Fit

Disadvantages:

- Still causes fragmentation

5. Paging Allocation (Non-contiguous)

- Memory is allocated in **fixed-size frames**
- Pages can be stored **anywhere in memory**

Advantages:

- No external fragmentation
- Efficient memory utilization

Disadvantages:

- Internal fragmentation may occur

Difference: Paging vs Contiguous Allocation

Paging	Contiguous Allocation
Non-contiguous	Contiuous
no external fragmentation	External fragmentation
Uses page table	Simple base /limet

4. differentiate between internal and external fragmentation.

-Difference between Internal Fragmentation and External Fragmentation

Basis	**Internal Fragmentation**	**External Fragmentation**
Definition	Wasted memory **inside** an allocated memory block	Wasted memory **between** allocated memory blocks
Cause	Fixed-size memory allocation	Variable-size memory allocation
Occurs when	Allocated block is larger than required	Free memory is available but not contiguous
Memory wastage	Inside the block	Outside the blocks
Type of allocation	Paging, fixed partitioning	Segmentation, variable partitioning
Compaction	Not useful	Useful to reduce fragmentation
Example	Process needs 18 KB, allocated 20 KB → 2 KB wasted	Total free memory = 20 KB but in small pieces

5. Explain deadlock in os . explain necessary conditions for deadlock in os

-Deadlock in an operating system is a state where a set of processes are permanently blocked because each process

holds a resource while waiting for another resource held by another process in the set. It represents a circular dependency where no process can proceed, resulting in an indefinite wait.

Necessary Conditions for Deadlock (Coffman Conditions)
For a deadlock to occur, all four of the following conditions must hold simultaneously within a system:

- **Mutual Exclusion:** At least one resource must be held in a non-shareable mode; only one process can use the resource at any given time.
- **Hold and Wait:** A process must be holding at least one resource and is currently waiting to acquire additional resources that are currently being held by other processes.
- **No Preemption:** Resources cannot be forcibly taken away from a process; they must be released voluntarily by the process holding them.
- **Circular Wait:** A set of processes {P0,P1,…,Pn}the set cap P sub 0 comma cap P sub 1 comma … comma cap P sub n end-set{$P0$,$P1$,…,Pn}must exist such that P0cap P sub 0$P0$is waiting for a resource held by P1cap P sub 1$P1$ P1cap P sub 1$P1$ is waiting for P2cap P sub 2 $P2$ and Pncap P sub n Pn is waiting for a resource held by P0cap P sub 0$P0$

6. What is deadlock detection ? explain bankers algorithm with example

_ Deadlock detection is a technique used by an operating system to **check whether a deadlock has occurred** in the system.

In this approach:

- The system **allows deadlock to occur**
- It periodically runs a **deadlock detection algorithm**
- If deadlock is detected, the system takes action such as:
  - Process termination
  - Resource preemption

Banker's Algorithm

Banker's Algorithm is a **deadlock avoidance algorithm** developed by **Dijkstra**.
It ensures that the system **never enters an unsafe state** by checking resource allocation before granting a request.

The algorithm works like a banker who lends money only if it is safe to do so.

Data Structures Used

1. **Available** – number of available resources
2. **Max** – maximum resources required by each process
3. **Allocation** – resources currently allocated
4. **Need = Max − Allocation**

Banker's Algorithm – Example

Given:

Processes: P0, P1, P2
Resource types: A, B

A = 3 , B = 3

Process	A	B
P0	3	2
P1	1	2
P2	2	2

Allocation Matrix

Process	A	B
P0	1	0
P1	0	1
P2	1	1

Step 1: Calculate Need Matrix

Need = Max – Allocation

Process	A	B
P0	2	2
P1	1	1
P2	1	1

Step 2: Safety Check

Available = (3,3)

- P1 can execute → Available becomes (3,4)
- P2 can execute → Available becomes (4,5)
- P0 can execute → Available becomes (5,5)

7. Explain different file allocation techniques in os .

_ File allocation techniques in operating systems determine how disk blocks are assigned to files, directly impacting performance and storage efficiency. The three primary methods are **Contiguous** (sequential blocks), **Linked** (linked list of blocks),

and **Indexed** (index block tracking pointers), each balancing speed, fragmentation, and flexibility.

1. Contiguous Allocation

- **Description:** Each file occupies a contiguous set of blocks on the disk.
- **Advantages:** Excellent performance for both sequential and random access.
- **Disadvantages:** Suffers from external fragmentation (free space is broken into small, unusable pieces) and makes file growth difficult.
- **Use Case:** Ideal for read-only systems or where file size is known and fixed.

2. Linked Allocation

- **Description:** Files are stored as a linked list of disk blocks. The directory contains a pointer to the first and last blocks.
- **Advantages:** No external fragmentation; files can grow easily.
- **Disadvantages:** Poor random access performance (must traverse pointers) and wasted space for pointers.
- **Variants: File Allocation Table (FAT)**, used by MS-DOS, stores all pointers in a separate table, allowing faster random access.

3. Indexed Allocation

- **Description:** Brings all pointers together into one location called the **index block**. The directory entry points to this index block.
- **Advantages:** Supports direct (random) access efficiently; no external fragmentation.

- **Disadvantages:** High overhead for small files (an entire block is used for the index).
- **Variants:** Multi-level indexing (hierarchical index blocks) or combined schemes like Unix inodes.

8. Define cryptography . explain different security actacks.

_ Cryptography is the science of **protecting information** by converting it into an **unreadable form (cipher text)** so that only authorized users can access it.

It uses techniques like **encryption** and **decryption** to ensure:

- Confidentiality
- Integrity
- Authentication
- Non-repudiation

Security Attacks

A **security attack** is any attempt to **destroy, steal, modify, or gain unauthorized access** to information or system resources.

Types of Security Attacks

Security attacks are mainly classified into **two types**:

1. Passive Attacks

In passive attacks, the attacker **only monitors** the data and does not modify it.

(a) Eavesdropping

- Attacker secretly listens to communication
- Example: Listening to network traffic

(b) Traffic Analysis

- Attacker observes communication patterns
- Used to gather information even if data is encrypted

Characteristics:

- Difficult to detect
- Does not affect system resources

2. Active Attacks

In active attacks, the attacker **modifies, deletes, or disrupts data**.

(a) Masquerade

- Attacker pretends to be an authorized user
- Example: Using stolen login credentials

(b) Replay Attack

- Attacker captures data and **retransmits** it later
- Example: Reusing old authentication messages

(c) Modification of Message

- Data is altered during transmission
- Affects data integrity

(d) Denial of Service (DoS)

- Attacker overloads system resources
- Legitimate users cannot access services

9. write short notes

a. Race condition

- A race condition is a software bug where the system's outcome depends on the unpredictable timing or sequence of multiple, concurrent operations accessing shared resources, leading to inconsistent or incorrect results, data corruption, or security flaws. It happens when threads "race" to modify data, and the final state depends on who finishes first, often in "check-then-act" scenarios where another thread changes things between the check and the action. Synchronization mechanisms like locks prevent this by ensuring only one thread accesses the shared data at a time

b. multi programming

- Multiprogramming is an operating system technique that loads multiple programs into main memory simultaneously to maximize CPU utilization. By switching between jobs—especially when one is waiting for I/O—it minimizes idle time and improves system throughput. It is not true parallel execution but rather rapid context switching

c. system call

- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel. There are various situations where you must require system calls in the operating system. Following of the situations are as follows:

• It is must require when a file system wants to create or delete a file

. • Network connections require the system calls to sending and receiving data packets.

• If you want to read or write a file, you need to system calls.

• If you want to access hardware devices, including a printer, scanner, you need a system call.

• System calls are used to create and manage new processes

2080/81

1. What is operating system ? what are function of operating system ? explain

- operating system is a collection of set of software which manages all the gesources of the computer system it is an interediater between the user and hardware

Function of operating system

1. memory management

- memory management module performs the task of allocttion and dealloction of memory space to programs in need of this resources

2. processor management

- the operating system assigns processors to the different tasks that must be performed by the computer system

3. device management

- Operating system performs the task of allocation of the device

4. file management

- operating system manage all file related activities such as organization stoage retrieval naming sharing and protection of file

5. security management

Security management function of an operating system help in implementing mechanisms that secure and protect the computer system internally as well as externally

6. job scheduling

- job scheduling is the process of allocationg system resources to many different tasks by an operating system

7.time sharing

-It co-ordinates and assighs comilers assemblers utlity program and other software packages to various user working on computer system

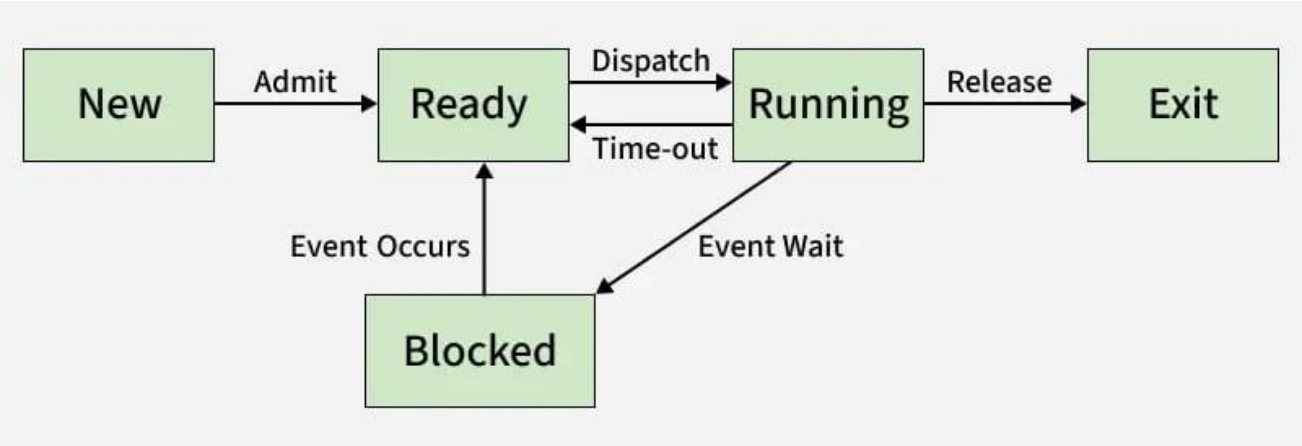
2. Define process and thread . describe different process states with necessary diagram.

- - **Process:** An executing instance of a program, known as a "program in execution." It includes the code, data, heap, and stack memory, along with resources like open file handles. Processes are independent and have high overhead for creation and context switching.
- **Thread:** A basic unit of CPU utilization, or a "lightweight process," that executes within a process. Threads share the same address space (code and heap) with other threads in the same process but have their own stack and program counter.

Process States and Diagram

A process transitions through different states in its lifetime, managed by the Process Control Block (PCB):

- **New:** The process is being created.
- **Ready:** The process is loaded into main memory and is waiting for CPU time to be scheduled.
- **Running:** The CPU is actively executing the process's instructions.
- **Blocked/Waiting:** The process cannot run, as it is waiting for an event (e.g., I/O completion or resource allocation).
- **Terminated/Exit:** The process has finished execution or was killed, and the OS is reclaiming its resources.



3. What is inter process communication ?explain petersonls solution to mutual exclusion with busy waiting .

- Inter-process communication (IPC) is a mechanism that allows multiple processes to coordinate and exchange data simultaneously [1]. It encompasses methods such as shared memory, message passing, pipes, and sockets, all of which are

essential for cooperative processing and system efficiency [1, 2].

Peterson's Solution to Mutual Exclusion with Busy Waiting

Peterson's solution is a classic software-based algorithm for achieving **mutual exclusion** between two processes, ensuring that only one process can enter a critical section (a code segment that accesses shared resources) at a time [1, 3]. It uses **busy waiting**, meaning processes waiting to enter the critical section continuously check a condition in a loop rather than being blocked by the operating system [3].
The solution requires two shared variables:

- `flag[2]`: A boolean array, initialized to `false`, where `flag[i] = true` indicates that process Picap P sub I *Pi*wants to enter the critical section [3].
- `turn`: An integer variable, initialized to either 0 or 1, indicating whose turn it is to enter the critical section if both want to [3].

5. What are the necessary conditions for deadlock ? differentiate between preemptive and non- preemptive scheduling

- Necessary Conditions for Deadlock

The four conditions necessary for a deadlock are:

1. **Mutual Exclusion:** Resources cannot be shared; only one process can use a resource at a time.
2. **Hold and Wait:** A process holds at least one resource and waits for another held by a different process.

3. **No Preemption:** Resources cannot be forcibly taken from a process.
4. **Circular Wait:** A set of processes are waiting for resources held by other processes in the set.

Difference Between Preemptive and Non-Preemptive Scheduling

Basis	**Preemptive Scheduling**	**Non-Preemptive Scheduling**
CPU control	CPU can be taken from a running process	CPU cannot be taken forcibly
Process switching	Can occur anytime	Occurs only when process finishes or blocks
Response time	Faster response	Slower response
Priority handling	High-priority process can interrupt	High-priority process must wait
CPU utilization	Better	Less efficient
Complexity	More complex	Simple
Starvation	Possible	Less likely
Examples	Round Robin, SJF (Preemptive), Priority (Preemptive)	FCFS, SJF (Non-preemptive), Priority (Non-preemptive)

6. Explain about segmentation with its importance and drawbacks.

- **Segmentation** is a memory management technique in which a process is divided into **logical, variable-sized segments** like code, data, and stack.
Each logical address consists of a **segment number and an offset**, matching the programmer's view of memory.

Importance of Segmentation

- **Logical Organization:** It maps to how a programmer views a program, organizing code into logical units (e.g., functions, arrays, stack) rather than rigid, arbitrary pages.
- **Elimination of Internal Fragmentation:** Because segments are allocated based on the exact size of the module, memory is not wasted inside a segment, solving the internal fragmentation issue found in paging.
- **Protection and Security:** Specific access rights (read, write, execute) can be assigned to individual segments, enhancing system security.
- **Sharing:** It makes it easier to share code or data between different processes, as entire functional modules can be shared.
- **Smaller Table Size:** Segment tables often require less space compared to the page tables used in paging.

Drawbacks of Segmentation

- **External Fragmentation:** As segments are loaded and removed from memory, free space gets divided into small, scattered blocks, creating external fragmentation.

- **Complex Memory Management:** Managing variable-sized segments is more complex for the operating system than managing fixed-size pages.
- **Higher Overhead:** Maintaining the segment table for each process requires extra memory and management, which can lead to performance overhead.
- **Slower Access Time:** Accessing memory requires two lookups (one for the segment table, one for the actual memory), increasing the time to fetch instructions.
- **Segmentation Faults:** Errors can occur if a program attempts to access memory outside its assigned segment boundary.

7. What is page fault ? describe not recently used page replacement algorithm with sutable example .

- Page Faults A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory

. Page Replacement Algorithms Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Suitable Example

Assume:

- **Number of frames = 3**

- **Page reference string:**
  1 2 3 4 1 2 5

Using FIFO Page Replacement Algorithm

Page Reference	Frames (after replacement)	Page Fault
1	1 – –	**Yes**
2	1 2 –	**Yes**
3	**123**	**Yes**
4	**423**	**Yes (1 replaced)**
1	**413**	**Yes (2 replaced)**
2	**412**	**Yes (3 replaced)**
5	**512**	**Yes (4 replaced)**

8**. Explain contiguous and linked list allocation in file system**.

- Contiguous Allocation In this method

, • Every file users a contiguous address space on memory.

• Here, the OS assigns disk address is in linear order

• In the contiguous allocation method, external fragmentation is the biggest issue.

Linked Allocation In this method,

• Every file includes a list of links.

• The directory contains a link or pointer in the first block of a file

. • With this method, there is no external fragmentation

• This File allocation method is used for sequential access files

. • This method is not ideal for a direct access file.

9. compare memory mapped IO and IO mapped IO explain first come first service disk scheduling algorithm with sutable example .

- 1. **Memory-Mapped I/O vs I/O-Mapped I/O**

Feature	**Memory-Mapped I/O**	**I/O-Mapped I/O (Port-Mapped I/O)**
Addressing	I/O devices are treated as **memory locations**	I/O devices have **separate address space (ports)**.
Instructions	Can use **normal memory instructions** (load, store).	Requires **special I/O instructions** (IN, OUT).
Speed	Faster, as memory instructions are used	Slightly slower due to special I/O instructions
Address space	Shares CPU memory address space with RAM.	Separate I/O address space, does not use memory addresses.
Complexity	Simpler in terms of instructions.	More hardware complexity to decode port addresses.

Example	Accessing video memory, RAM-mapped keyboard buffer	Keyboard controller using IN/OUT instructions

Part 2: First-Come-First-Serve (FCFS) Disk Scheduling Algorithm

FCFS (First-Come-First-Serve) is the simplest disk scheduling algorithm. As the name suggests, it services I/O requests in the exact order they arrive in the disk queue. It is similar to a queue at a ticket counter—the first person in line is served first.

Example

Scenario: A disk has 200 tracks (0-199).
Request Queue: 82, 170, 43, 140, 24, 16, 190
Initial Head Position: 50
Goal: Calculate total head movement (total cylinders moved).
Calculation steps:

1. **50 to 82:**

   $|82-50|=32$the absolute value of 82 minus 50 end-absolute-value equals 32

   $|82-50|=32$

2. **82 to 170:**

   $|170-82|=88$the absolute value of 170 minus 82 end-absolute-value equals 88

   $|170-82|=88$

3. **170 to 43:**

   |43−170|=127the absolute value of 43 minus 170 end-absolute-value equals 127

   |43−170|=127

4. **43 to 140:**

   |140−43|=97the absolute value of 140 minus 43 end-absolute-value equals 97

   |140−43|=97

5. **140 to 24:**

   |24−140|=116the absolute value of 24 minus 140 end-absolute-value equals 116

   |24−140|=116

6. **24 to 16:**

   |16−24|=8the absolute value of 16 minus 24 end-absolute-value equals 8

   |16−24|=8

7. **16 to 190:**

   |190−16|=174the absolute value of 190 minus 16 end-absolute-value equals 174

   |190−16|=17

Total Head Movement:

32+88+127+97+116+8+174=64232 plus 88 plus 127 plus 97 plus 116 plus 8 plus 174 equals 642
32+88+127+97+116+8+174=**642**
units.
Result: The disk head moves a total of 642 tracks to serve all requests in the order they arrived
10. Write short notes

A. Bankers algorithm

- The Banker's algorithm, developed by Edsger Dijkstra, is a resource allocation and deadlock avoidance algorithm that determines if allocating resources to a process maintains a safe state. It works by simulating the allocation, checking if all processes can complete, and only granting requests that avoid potential deadlocks. Key data structures include available resources, max demand, current allocation, and remaining needs

b. Types of operating system

Key Types of Operating Systems:

- **Batch Operating System:** Users do not interact directly with the computer; operators sort similar jobs into batches to speed up processing (e.g., payroll systems).
- **Time-Sharing (Multitasking) Operating System:** Enables multiple users to access a single system simultaneously, sharing CPU time to perform tasks efficiently.
- **Distributed Operating System:** Manages a group of distinct computers connected over a network to appear as a single, centralized system.

- **Network Operating System (NOS):** Runs on a server and provides the capability to manage data, users, groups, and security over a network.
- **Real-Time Operating System (RTOS):** Used when rigid time constraints are required, such as in scientific, medical imaging, or industrial control systems.

c. memory hierarchy

- The memory hierarchy is a structured organization of computer storage that balances the trade-offs between access speed, capacity, and cost by using multiple levels of memory. It is designed to minimize average access time by storing frequently used data in faster, smaller memory tiers closer to the CPU

Levels of the Memory Hierarchy

Level 1: Registers

Level 2: Cache Memory (L1, L2, L3)

Level 3: Main Memory (RAM)
Level 4: Secondary Storage

Level 5: Tertiary/Off-line Storage

d. FIFO scheduling

- First-In-First-Out (FIFO) scheduling, often referred to as First-Come, First-Served (FCFS), is a straightforward, non-preemptive algorithm that handles processes in the order of their arrival in the ready queue. It is a work-conserving algorithm, meaning it does not introduce idle times when work is pending.

Key Characteristics and Components

- **Order of Execution:** The process that arrives first is executed first.
- **Non-Preemptive:** Once a process receives the CPU, it continues to run until it completes its CPU burst or voluntarily relinquishes control.
- **Implementation:** Typically implemented using a FIFO queue data structure, where incoming processes are added to the back (enqueue) and removed from the front (dequeue).
- **Criteria:** The sole criterion for selection is the arrival time.

Advantages

- **Simplicity:** Extremely easy to understand and implement.
- **Fairness:** Every process gets a turn based on its arrival, ensuring no starvation.
- **Low Overhead:** Minimal scheduling decision-making required.

Disadvantages

- **Convoy Effect:** Long, CPU-bound processes can hold the CPU for a long time, causing shorter processes to wait in a "convoy".
- **Poor Performance:** Often results in high average waiting times and low throughput.
- **Not Suitable for Real-Time:** Ineffective for systems with tight, hard deadlines