OS Notes by Bhavy Sharma

Q1. What is Real Time Operating System?
Ans.
Real-Time Operating System (RTOS)

A **Real-Time Operating System (RTOS)** is an operating system designed to process data and execute tasks within a strict time constraint. RTOS ensures that critical tasks are completed within a defined deadline.

Types of Real-Time Operating Systems

1. **Hard RTOS** (Real Time Operating System) –
    - Strict deadlines must always be met.
    - Failure to meet deadlines can lead to system failure.
    - Example: Airbag systems in cars, medical life-support systems.
2. **Soft RTOS** (Real Time Operating System) –
    - Deadlines are important but not critical.
    - Slight delays are acceptable but should be minimized.
    - Example: Video streaming, online gaming.

Advantages of RTOS

- Work Done With in Deadline
- Full Speed Task Shifting
- Most of the Time RTOS are Error Free
- RTOS in ES(Embedded System)

Function of RTOS

- Task Management
- Scheduling
- Resource Allocation
- Interrupt Handling

Q2. What is Hard & Soft Semaphore?
Ans.

Hard & Soft Semaphore in Operating Systems

A **semaphore** is a synchronization mechanism used in operating systems to manage access to shared resources in a concurrent environment. Semaphores help prevent race conditions and ensure process synchronization.

1. Hard Semaphore

A **hard semaphore** is implemented in hardware and provides low-level synchronization, often used in **real-time systems** where fast response and strict timing constraints are required.

Characteristics of Hard Semaphore

- Implemented at the **hardware level**.
- Provides **low-latency** synchronization.
- Mostly used in **real-time operating systems (RTOS)**.
- Directly interacting with processors and memory.
- Used in **critical applications** like medical systems and industrial automation.

---

2. Soft Semaphore

A **soft semaphore** is implemented at the **software level** using operating system mechanisms and does not require special hardware. It is used in **general-purpose operating systems** for resource sharing and process synchronization.

Characteristics of Soft Semaphore

- Implemented in **software** (handled by the OS).
- Slightly **slower** than hard semaphores due to software-level execution.
- Used in **non-real-time systems** like Windows, Linux, and MacOS.
- Works at the **application level**

Q3. What is preemptive & non-preemptive Scheduling?
Ans.

Preemptive & Non-Preemptive Scheduling in Operating Systems

CPU scheduling is the process of selecting a process from the ready queue and allocating the CPU to it. Scheduling can be classified into two types:

1. **Preemptive Scheduling**
2. **Non-Preemptive Scheduling**

---

1. Preemptive Scheduling

In **preemptive scheduling**, the CPU can be taken away from a running process if a higher-priority process arrives or the current process exceeds its allowed time slice.

Characteristics of Preemptive Scheduling:

- Allows **better CPU utilization** by preventing long-running processes from blocking others.

- **Faster response time** for high-priority processes.
- Used in **real-time and multitasking systems**.

---

2. Non-Preemptive Scheduling

In **non-preemptive scheduling**, once a process starts execution, it **cannot be interrupted** until it completes or voluntarily releases the CPU.

Characteristics of Non-Preemptive Scheduling:

- The CPU is **never taken away** from a running process.
- Ensures **predictable execution** but may cause **starvation** for lower-priority processes.
- **Used in batch processing systems**.

Q4. What are the necessary conditions for a deadlock to occur? Explain different methods for deadlock prevention and avoidance. Under what conditions can a deadlock situation arise?

Ans.

Deadlock in Operating Systems

A **deadlock** is a situation in which two or more processes are **waiting indefinitely** for resources that are being held by each other, preventing further execution.

Necessary Conditions for Deadlock (Coffman's Conditions)

A deadlock occurs when the following **four conditions** hold simultaneously:

1. **Mutual Exclusion:**
    - A resource can only be held by **one process at a time**.
    - If another process requests it, it must wait.
2. **Hold and Wait:**
    - A process is **holding at least one resource** while waiting for additional resources.
3. **No Preemption:**
    - A resource **cannot be forcibly taken** from a process.
    - It must be released voluntarily by the process holding it.
4. **Circular Wait:**
    - A set of processes exist where **each process is waiting for a resource held by the next process in the chain**, forming a circular dependency.

---

2. Deadlock Avoidance

Deadlock avoidance allows the system to **check** resource allocation before granting requests, ensuring a deadlock never occurs. The **Banker's Algorithm** is a common approach.

Banker's Algorithm (for Deadlock Avoidance)

- Each process declares **maximum resource needs** in advance.
- The OS checks if granting a resource keeps the system in a **safe state**.
- If yes → Grant the resource.
- If no → Make the process wait.

Q5. What is the Purpose of system call?

Ans.

Purpose of System Calls in an Operating System

A **system call** is a mechanism that allows user-level programs to **request services** from the operating system (OS). Since user programs cannot directly access kernel resources (such as hardware or memory), system calls act as an interface between the **user space** and the **kernel space**.

- **Process Control**
- **File Management**
- **Device Management**
- **Memory Management**
- **Networking**

Q6. Explain how memory can dynamically allocate using first fit, best fit and worst fit.??

Ans.

Dynamic Memory Allocation in Operating Systems

Dynamic memory allocation refers to the process where memory is allocated to processes at runtime based on their requirements. The **First Fit, Best Fit, and Worst Fit** memory allocation strategies help manage free memory blocks efficiently in a system.

---

1. First Fit Allocation

Concept:

- The OS scans the memory from the beginning and assigns the **first available** block that is large enough to accommodate the

process.

Advantages:

1. Fast and simple to implement.
2. Reduces searching time since it stops after finding the first suitable block.

Disadvantages:

1. Can lead to fragmentation as small gaps are left in memory.

Example:

Available memory blocks: **100 KB, 500 KB, 200 KB, 300 KB, 600 KB**
Process size: **250 KB**

👉 **First Fit allocates 500 KB** (since it is the first available block that fits).

---

2. Best Fit Allocation

Concept:

- The OS searches the entire memory and allocates the **smallest block** that is large enough to fit the process.

Advantages:

✔️ Reduces **wastage** by using the smallest possible block.
✔️ Leads to better memory utilization.

Disadvantages:

❌ Searching for the smallest available block takes more time.
❌ May still cause fragmentation if very small holes are left.

Example:

Available memory blocks: **100 KB, 500 KB, 200 KB, 300 KB, 600 KB**
Process size: **250 KB**

👉 **Best Fit allocates 300 KB** (as it is the smallest block that can fit the process).

---

3. Worst Fit Allocation

Concept:

- The OS searches for the **largest available** block and assigns it to the process.

Advantages:

✔️ Leaves larger remaining memory blocks, which can accommodate future large processes.

Disadvantages:

❌ Leads to inefficient utilization since large memory blocks are broken into smaller, unusable pieces.
❌ Can increase external fragmentation.

Example:

Available memory blocks: **100 KB, 500 KB, 200 KB, 300 KB, 600 KB**
Process size: **250 KB**

👉 **Worst Fit allocates 600 KB** (as it is the largest available block).

Q7. Describe the critical Segmentation Problem with suitable Example?

Ans.

Critical Segmentation Problem in Operating Systems

What is Segmentation?

Segmentation is a **memory management technique** in which a process is divided into different **segments**, each representing a **logical unit** such as code, data, stack, etc.

Critical Segmentation Problem

Despite its advantages, segmentation has several critical issues:

1. External Fragmentation

- **Problem:** When processes are allocated and deallocated dynamically, small free memory spaces (holes) are created between allocated segments, making it **impossible to allocate new processes** even if there is enough total free memory.

- **Example:**

    - Assume we have **total free memory = 500 KB**, but it is divided into **scattered small blocks** (e.g., 100 KB, 50 KB, 150 KB, 200 KB).

    - A process requiring **300 KB cannot be allocated** despite sufficient total memory because it is not **contiguous**.

---

2. Segmentation Fault

- **Problem:** A **segmentation fault** occurs when a process tries to **access an invalid memory segment** (e.g., accessing memory outside its allocated range).

---

3. Memory Wastage Due to Variable Segment Size

- **Problem:** Since segments are of **different sizes**, memory management becomes difficult, leading to inefficient allocation.

- **Example:**

    - A program may have **Code = 300 KB, Data = 200 KB, Stack = 100 KB**, but available free memory is **350 KB in one block and 250 KB in another**.

    - Since segments must be **contiguous**, the process **cannot be allocated**, causing memory wastage.

Q8. What is the Fragmentation Problem? Describe the External and Internal Fragmentation.

Ans.

Fragmentation Problem in Operating Systems

What is Fragmentation?

Fragmentation is a memory management issue that occurs when free memory blocks are divided into small, non-contiguous parts, making them inefficient for allocation. It leads to **wasted memory** and reduces system performance.

There are two main types of fragmentation:

1. **External Fragmentation**

2. **Internal Fragmentation**

---

1. External Fragmentation

Definition:

- External fragmentation occurs when **free memory is available**, but it is **scattered in small, non-contiguous blocks**, preventing processes from being allocated even though enough memory exists.

Example:

Assume a system has **total free memory of 400 KB**, but it is divided into **non-contiguous blocks**:

- **100 KB free**
- **50 KB free**
- **150 KB free**
- **100 KB free**

Now, a process requiring **250 KB** cannot be allocated, even though the total free memory is 400 KB, because no single **continuous block** of 250 KB is available.

---

2. Internal Fragmentation

Definition:

- Internal fragmentation occurs when **allocated memory is larger than required**, leading to **unused space within allocated blocks**.

Example:

If a system allocates **fixed memory blocks of 100 KB** for processes:

- Process A (Needs 80 KB) → Allocated 100 KB → **20 KB wasted**
- Process B (Needs 60 KB) → Allocated 100 KB → **40 KB wasted**

Even though processes don't use the full allocated memory, the unused portion cannot be assigned to another process, leading to **internal fragmentation**.

Q9. **Write short notes on the following:**

➔ File allocate Method
➔ Swapping
➔ Threads
➔ Disk Structure
➔ Disk Scheduling
➔ Paging

Ans.

1. File Allocation Methods

File allocation refers to how files are stored on disk. The three main methods are:

- **Contiguous Allocation:** Stores file blocks **consecutively** in memory for fast access but can cause fragmentation.
- **Linked Allocation:** Uses a **linked list** where each file block contains a pointer to the next block, avoiding fragmentation but slower in access.
- **Indexed Allocation:** Uses an **index block** to store addresses of all file blocks, allowing **random access** to file parts.

---

2. Swapping

Swapping is a **memory management technique** where a process is temporarily moved from **main memory (RAM) to secondary storage (disk)** and later brought back for execution.

- Increases **multiprogramming** by freeing up memory.
- Can cause **swapping overhead**, leading to performance delays.
- Used in **Virtual Memory** systems when RAM is full.

---

3. Threads

A **thread** is the smallest unit of CPU execution within a process.

- **Single-threaded process:** Has only one execution path.
- **Multi-threaded process:** Allows multiple tasks to run concurrently within the same process.
- Threads **share resources** like memory and files but have independent execution paths.
- Example: A web browser runs multiple threads for rendering, downloading, and user interactions.

---

4. Disk Structure

Disk structure refers to how data is organized and managed on a **hard disk**. It includes:

- **Platters:** Rotating magnetic disks that store data.

- **Tracks & Sectors:** Concentric circles and smaller divisions where data is written.

---

5. Disk Scheduling

Disk scheduling determines the order in which disk I/O requests are handled to minimize seek time. Common algorithms include:

- **FCFS (First Come, First Served):** Executes requests in arrival order.

- **SSTF (Shortest Seek Time First):** Selects the request closest to the current head position.

---

6. Paging

Paging is a memory management technique where the logical address is divided into fixed–size pages, and physical memory is divided into page frames of the same size. A Page table maps pages to frames, allowing non-contiguous memory allocation and eliminating external fragmentation.

Advantages:

Eliminates external fragmentation

Efficient Memory Utilization

Allows easy process swapping

Q10. What is Process Synchronization? What are the mechanisms to control access to critical sections of the process? What are the hardware supported solutions for process synchronization?

Ans.

Process Synchronization

Process synchronization is a mechanism used in operating systems to coordinate processes that **share resources** and execute concurrently, ensuring **data consistency and avoiding race conditions**.

- It prevents **multiple processes from modifying shared data.**
- It ensures that **critical sections** (code sections where shared resources are accessed) are executed **one at a time**.

Mechanisms to Control Access to Critical Sections

To prevent race conditions, the following synchronization mechanisms are used:

1. **Mutual Exclusion (Mutex)**
    - Ensures that **only one process** can access a critical section at a time.
2. **Semaphores**
    - Integer variables used to control access to resources.
    - Two types:
        - **Binary Semaphore (0 or 1, acts like a lock).**
        - **Counting Semaphore (allows multiple accesses up to a limit).**

3. **Monitors**
    - A higher-level synchronization construct that encapsulates shared resources and provides **automatic synchronization**.

Hardware-Supported Solutions for Process Synchronization

1. **Compare-and-Swap (CAS) Instruction**
    - Compares a memory location with an expected value and swaps it **if it matches**.
    - Used in **lock-free synchronization** in multi-core processors.
2. **Disable Interrupts**
    - The OS disables **CPU interrupts** while a process is in its critical section.
    - Not suitable for multi-processor systems.
3. **Atomic Operations**
    - Ensures that specific operations (e.g., incrementing a counter) **execute as a single unit** without interference.

Q11. What are the basic functions an operating system performs as a resource manager?

Ans.

Basic Functions of an Operating System as a Resource Manager

An **Operating System (OS)** acts as a **resource manager** by efficiently managing the hardware and software resources of a computer system. It

ensures that multiple programs can run **simultaneously without conflicts** by allocating and deallocating resources as needed.\

Functions of an OS as a Resource Manager

1. Process Management

- Manages **creation, execution, and termination** of processes.
- Provides **interprocess communication (IPC)** and synchronization.

2. Memory Management

- Allocates and deallocates **main memory (RAM)** for processes.
- Implements **paging, segmentation, and virtual memory** to optimize RAM usage.
- Prevents memory conflicts and ensures efficient utilization.

3. File System Management

- Organizes, stores, and manages **files and directories**.
- Provides operations like **create, read, write, delete, and modify** files.

4. Device Management

- Manages **input/output (I/O) devices** such as keyboards, printers, and hard disks.
- Uses **device drivers** for communication between hardware and software.

5. Disk & Storage Management

- Allocates and tracks **secondary storage** (HDD, SSD).

- Manages **swap space** for virtual memory operations.

Q12. What are different types of files? What are the task of the file management system? List some file system related commands like Unix.?

Ans.

1. Different Types of Files in an Operating System

Files are used to store data in an organized manner. The common types of files include:

A. Regular Files (Data Files)

- **Text Files (.txt, .doc, .pdf):** Contain readable characters, such as documents or logs.
- **Binary Files (.exe, .bin, .dat):** Contain machine-readable data, such as compiled programs or images.

B. Directory Files

- Contain file and directory information.
- Used to organize and store file locations in a hierarchical structure.

C. Executable Files

- Contain compiled code that can be executed by the OS (e.g., `.exe`, `.sh`, `.bat`).

D. System Files

- Files that the OS needs for booting and execution (e.g., `.dll`, `.sys`, `.cfg`).

---

2. Tasks of the File Management System

A **File Management System (FMS)** is responsible for:

✅ **File Creation & Deletion:** Allows users to create and remove files.
✅ **Directory Management:** Organizes files into folders/directories.
✅ **File Access & Permissions:** Controls who can read, write, and execute files.
✅ **Storage Allocation:** Allocates disk space for file storage efficiently.
✅ **File Protection & Security:** Prevents unauthorized access and corruption.
✅ **Backup & Recovery:** Provides mechanisms to recover lost or corrupted files.

Q13. Explain the Demand Paging and cache memory.

Ans Demand Paging : It is a memory management technique used in virtual memory systems where pages of a process are loaded into RAM only when they are needed , rather than loading the entire process at once.

Advantages of Demand Paging

1.Efficient memory usage

2.Faster program execution

3.Allows large Programs

Disadvantages of Demand Paging

1.**Page Fault overhead** : too many faults can slow down performance.

2.**Slower Response Time** : If frequently used pages are not in RAM, performance drops.

Cache Memory :- Cache memory is a small, high-speed memory located close to the CPU that stores frequently used data & instructions to improve processing speed.

Advantages of Cache Memory

1. Faster Processing
2. Reduces RAM Bottlenecks

Disadvantages :-

Limited Size

Higher Cost (costly than RAM and storages devices)

Cache Memory speeds up CPU performance by storing frequently used data for quick access.

Ques14. What is page frame and page fault?

Ans.

Page frame

A page frame is a fixed-size block of physical memory (RAM) that stores a page from a process's virtual memory. When an operating system uses paging for memory management, it divides both physical memory and virtual memory into equal sized block:

***Page (in virtual memory)**

***Page Frame (in physical memory)**

When a process runs, its pages are loaded into available page frames in RAM

Page Fault

A page fault occurs when a process tries to access a page that is not currently in RAM, The OS then:

1. Pauses the process

2. Fetches the required page from disk.

3. Loads it into a page frame (or replace an existing one if needed)

4. Resumes the process

If the required page is found in RAM, it is a page hit (not a page fault).Frequent page faults can slow down performance due to excessive disk access.

Q15. **Difference between physical address and logical address.**

Feature	**Logical Address**	**Physical Address**
Definition	Address generated by CPU during program execution.	Actual Address in memory where data is stored.
Address Space	It is also called virtual memory, it does not exist physically.It exists in the user's view.	Exists in actual hardware (RAM) and manages by OS
Mapping	Converted to a physical address by Memory Management(MMU).	Directly used by hardware for accessing memory.

Visibility	Users & programs see only logical addresses.	OS & hardware deal with Physical address.
Dependency	Independent of physical memory size.	Dependent on physical memory size.