Operating Systems-Interview prep

1. Introduction to Operating Systems

Definition

An Operating System (OS) is system software that acts as an intermediary between computer hardware and the user. It manages hardware resources, provides a user interface, and offers services to applications software.

Types of Operating Systems

Batch Operating Systems: Executes jobs in batches without user interaction. Jobs are collected, grouped, and processed sequentially.

Time-Sharing Operating Systems: Allows multiple users to share system resources simultaneously, providing the illusion of direct access to the system.

Distributed Operating Systems: Manages a group of independent computers and makes them appear to be a single computer.

Embedded Operating Systems: Designed for specific control functions within devices with limited resources.

Real-time Operating Systems: Guarantees a response within a specified time frame. Used in critical systems like medical devices, automotive systems, and industrial controls.

Key Functions of an Operating System

Process Management: Controls the execution of processes, including scheduling, creation, and termination.

Memory Management: Manages the allocation and deallocation of memory space as needed by various programs.

File System Management: Manages the creation, deletion, reading, and writing of files and directories.

Device Management: Manages device communication via their respective drivers.

Security and Access Control: Protects data and resources from unauthorized access and ensures secure operations.

User Interface: Provides a means for users to interact with the computer, typically through a graphical user interface (GUI) or command-line interface (CLI).

2. Process Management

Process Concepts

- **Process**: An active program, including the current activity, contents of registers, program counter, and variables.
- **Process States**: Different stages in the lifecycle of a process.
- **New:** The process is being created.
- **Ready:** The process is waiting to be assigned to a processor.
- **Running**: Instructions are being executed.
- **Waiting:** The process is waiting for some event to occur.
- **Terminated**: The process has finished execution.

Process Control Block (PCB): A data structure containing all information about a process, including process state, program counter, CPU registers, memory management information, and accounting information.

Threads

Thread: The smallest sequence of programmed instructions that can be managed independently by a scheduler.
Multithreading: Allows multiple threads to exist within the context of a single process, sharing resources but executing independently.

Process Scheduling:

Schedulers: Components that decide which process to execute next.
Long-term Scheduler: Decides which processes are admitted to the system for processing.
Short-term Scheduler: Selects which process should be executed next and allocates CPU.
Medium-term Scheduler: Temporarily removes processes from main memory and reintroduces them to improve process mix.

Scheduling Algorithms:

First-Come, First-Served (FCFS): Processes are attended to in the order they arrive.

Shortest Job Next (SJN): Selects the process with the smallest execution time.

Priority Scheduling: Executes processes based on priority.

Round Robin (RR): Each process is assigned a fixed time slice (quantum) in rotation.

Multilevel Queue Scheduling: Multiple queues with different priority levels.

Context Switching

Context Switching: The process of storing the state of a currently running process and restoring the state of the next process to be executed.

3. Memory Management

Memory Allocation

Contiguous Memory Allocation: Each process is allocated a single contiguous section of memory.

Non-contiguous Memory Allocation: Memory is divided into several sections that can be assigned to a process.

Paging: Divides memory into fixed-sized pages.

Segmentation: Divides memory into variable-sized segments based on logical divisions.

Virtual Memory

Paging: Maps logical addresses to physical memory via pages and page tables. Uses page replacement algorithms when memory is full.

Segmentation: Each segment can be different sizes, improving the fit between the memory and the process requirements.

Page Replacement Algorithms:

FIFO (First-In-First-Out): The oldest page in memory is replaced first.

LRU (Least Recently Used): The least recently used page is replaced.

Optimal Page Replacement: Replaces the page that will not be used for the longest time in the future.

Memory Fragmentation

External Fragmentation: Occurs when there is enough total memory space to satisfy a request, but it is not contiguous.

Internal Fragmentation: Occurs when allocated memory may be slightly larger than requested memory, causing the space inside a partition to go unused.

4. File System Management

File Concepts

File Attributes: Name, identifier, type, location, size, protection, time, date, and user identification.

File Operations: Operations such as create, delete, open, close, read, write, and append.

File Types: Regular files, directories, character special files, and block special files.

File System Structure

Directory Structure: Organizes files and directories within the file system.
Single-level Directory: All files are contained in a single directory.
Two-level Directory: Each user has their own directory.
Tree-Structured Directory: Hierarchical directory structure.
Acyclic-Graph Directory: Allows sharing of files and directories.
General Graph Directory: Contains cycles, making it complex.

File Allocation Methods

Contiguous Allocation: Each file occupies a contiguous set of blocks.
Linked Allocation: Each file is a linked list of disk blocks.
Indexed Allocation: Each file has its own index block of pointers to its data blocks.

Access Methods

Sequential Access: Information in the file is processed in order, one record after the other.
Direct Access: File is viewed as a numbered sequence of blocks or records, allowing access in any order.

5. Device Management

I/O Hardware

Polling: CPU continuously checks the status of a device.
Interrupts: Device sends an interrupt signal to the CPU to indicate that it needs attention.
Direct Memory Access (DMA): Allows devices to transfer data to/from memory without CPU intervention.

I/O Software Layers

Device Drivers: Specialised programs that control particular devices.
Interrupt Handlers: Handles the interrupt signals from devices.
Device-Independent I/O Software: Provides a uniform interface to interact with different devices.
User-Space I/O Software: Provides the APIs that user applications use to perform I/O operations.

6. Security and Access Control

Security Mechanisms

Authentication: Verifying the identity of a user or process.
Authorization: Granting permission to a user or process to access resources.
Encryption: Protecting data by transforming it into an unreadable format, only readable by someone who has the decryption key.

Access Control Models

Discretionary Access Control (DAC): Access rights are determined by the owner of the resource.
Mandatory Access Control (MAC): Access rights are regulated by a central authority based on multiple levels of security.
Role-Based Access Control (RBAC): Access rights are assigned based on roles within an organization.

7. Important Operating Systems Concepts

Deadlocks

Necessary Condition:

Mutual Exclusion: Only one process can use a resource at a time.
Hold and Wait: A process holding at least one resource is waiting to acquire additional resources held by other processes.
No Preemption: Resources cannot be forcibly taken from processes holding them.
Circular Wait: There exists a set of processes such that each process is waiting for a resource held by the next process in the set.
Deadlock Prevention: Ensure that at least one of the necessary conditions cannot hold.
Deadlock Avoidance: Use algorithms to ensure the system never enters an unsafe state.
Deadlock Detection and Recovery: Allow the system to enter a deadlock state, detect it, and recover.

Concurrency:

Race Condition: The behaviour of the software depends on the sequence or timing of uncontrollable events.

Critical Section: A section of code that accesses a shared resource and must not be concurrently accessed by more than one thread.
Mutual Exclusion: Ensures that only one process or thread accesses the critical section at a time.
Synchronization Mechanisms: Tools to manage access to resources in concurrent programming.
Semaphores: A signaling mechanism to control access, can be binary (0 or 1) or counting.
Mutexes: A mutual exclusion object that prevents multiple threads from accessing a resource simultaneously.
Monitors: A high-level synchronization construct that allows threads to have both mutual exclusion and the ability to wait for a certain condition to become true.

8. Important Questions for Operating Systems Interviews

Memory Management

What is Virtual Memory?

Virtual memory is a memory management technique that provides an "idealized abstraction of the storage resources" to the user. It enables a computer to compensate for physical memory shortages by temporarily transferring data from random access memory (RAM) to disk storage. This process allows for larger programs to run on systems with limited RAM.

Explain Paging and Segmentation?

Paging:
Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. It divides the process's virtual memory into fixed-size blocks called pages, and the physical memory into blocks of the same size called frames. When a program needs to access data, the operating system translates virtual addresses to physical addresses using a page table.

Advantages: Simplifies memory allocation, avoids external fragmentation, and allows for efficient use of physical memory.
Disadvantages: Can suffer from internal fragmentation and overhead due to maintaining page tables.

Segmentation
Segmentation is a memory management technique that divides the process's memory into different segments based on the logical divisions, such as

functions, data, and stacks. Each segment has a name and a length, and segments are not necessarily of equal size.

Advantages: Simplifies handling of growing data structures and provides better protection and sharing by segment-level access control.
Disadvantages: Can suffer from external fragmentation and is more complex to implement than paging.

What is a Page Fault and How is it Handled?

A page fault occurs when a program tries to access a page that is not currently mapped to the physical memory. When a page fault happens, the operating system must take the following steps:

1. **Identify the Fault**: Determine that a page fault has occurred and identify the missing page.
2. **Locate the Page**: Find the location of the required page on the disk (secondary storage).
3. **Allocate a Frame**: Select a free frame or use a page replacement algorithm to free up a frame if none are available.
4. **Load the Page**: Transfer the required page from disk to the allocated frame in physical memory.
5.**Update Tables**: Update the page table and other relevant data structures to reflect the new location of the page.
6. **Resume Execution**: Restart the instruction that caused the page fault.

Compare and Contrast Different Page Replacement Algorithms

FIFO (First-In-First-Out)
Advantages: Simple to implement and understand.
Disadvantages: Does not consider the usage of pages, leading to suboptimal performance (Belady's Anomaly).

LRU (Least Recently Used):
Advantages: More efficient than FIFO as it considers the recent usage patterns.
Disadvantages: Requires additional hardware support or software overhead to keep track of the order of usage.

Optimal Page Replacement:
Advantages: Provides the best possible performance by replacing the page that will not be used for the longest time in the future.
Disadvantages: Not practical for implementation as it requires future knowledge of page references.

File Systems

Describe Different File Allocation Methods

Contiguous Allocation:

Advantages: Simple and allows direct access to files.
Disadvantages: Can lead to external fragmentation and is inflexible for file growth.

Linked Allocation:
Advantages: Solves the problem of fragmentation and allows files to grow easily.
Disadvantages: Slower direct access and requires additional space for pointers.

Indexed Allocation:
Advantages: Supports direct access and does not suffer from external fragmentation.
Disadvantages: Requires additional space for index blocks and can suffer from internal fragmentation.

How Does the Operating System Manage File Systems?
The operating system manages file systems through several key components and tasks:

File Control Blocks (FCBs): Structures that store information about files, such as size, permissions, and location.
Directory Structure: Organizes files in a hierarchical manner for easy navigation and access.
File System Mounting: Integrates different file systems into a single namespace.
File Operations: Handles file creation, deletion, reading, writing, and other operations.
Access Control: Manages user permissions to ensure secure file access.

Explain the Concept of Inodes in Unix/Linux File Systems
In Unix/Linux file systems, an inode (index node) is a data structure that stores metadata about a file. This includes:

File Attributes: Permissions, owner, group, size, timestamps.
Disk Block Addresses: Pointers to the data blocks on the disk where the file's contents are stored.
Link Count: Number of hard links pointing to the inode.

Inodes do not store file names; instead, directories contain the mappings between file names and inodes.

Device Management

What is Direct Memory Access (DMA)?

Direct Memory Access (DMA) is a feature that allows peripheral devices to transfer data directly to/from memory without involving the CPU. This frees up the CPU to perform other tasks while the data transfer is handled by the DMA controller.

Explain the Role of Device Drivers in the Operating System

Device drivers are specialized programs that allow the operating system to communicate with hardware devices. They provide a standardized interface between the OS and the hardware, translating high-level commands into device-specific operations. Drivers handle the intricacies of device communication, enabling the OS and applications to use hardware without needing to know the details of its operation.

Describe the Difference Between Polling and Interrupts

Polling: The CPU continuously checks the status of a device to see if it needs attention. This can waste CPU resources as it repeatedly queries devices.
Interrupts: Devices send an interrupt signal to the CPU when they need attention. The CPU then pauses its current tasks, handles the interrupt, and resumes normal operations. This is more efficient than polling as it allows the CPU to perform other tasks until an interrupt occurs.

Security

What are the Differences Between Authentication and Authorization?

Authentication: The process of verifying the identity of a user or entity. Common methods include passwords, biometrics, and security tokens.
Authorization: The process of determining whether an authenticated user or entity has permission to access specific resources or perform certain actions.

Describe Different Access Control Models

Discretionary Access Control (DAC): Access rights are determined by the resource owner. Users can control access to their resources.
Mandatory Access Control (MAC): Access rights are regulated by a central authority based on multiple levels of security. Users cannot change access rights.
Role-Based Access Control (RBAC): Access rights are assigned based on user roles within an organization. Users inherit permissions based on their role.

Explain How Encryption is Used in Operating Systems for Security
Encryption is used to protect data by transforming it into an unreadable format, ensuring that only authorized parties can access it. Operating systems use encryption for:

File and Disk Encryption: Protects data at rest on storage devices.
Network Encryption: Secures data in transit over networks.
Password Storage: Encrypts passwords to prevent unauthorized access.

Encryption algorithms include symmetric (same key for encryption and decryption) and asymmetric (public and private keys) methods.

Deadlocks and Concurrency

What are the Necessary Conditions for a Deadlock to Occur?

1. **Mutual Exclusion**: Only one process can use a resource at a time.
2.**Hold and Wait:** A process holding at least one resource is waiting to acquire additional resources held by other processes.
3.**No Preemption:** Resources cannot be forcibly taken from processes holding them.
4. **Circular Wait**: There exists a set of processes such that each process is waiting for a resource held by the next process in the set.

How Can Deadlocks be Prevented or Avoided?

Deadlock Prevention: Ensure that at least one of the necessary conditions for deadlock cannot hold. For example, by allocating all resources to a process before it starts execution (eliminating hold and wait).
Deadlock Avoidance: Use algorithms like Banker's algorithm to ensure the system never enters an unsafe state.
Deadlock Detection and Recovery: Allow the system to enter a deadlock state, detect it, and recover by terminating processes or preempting resources.

Explain the Concept of a Critical Section and How Mutual Exclusion is Ensured?

A critical section is a part of the code that accesses shared resources and must not be concurrently accessed by more than one thread or process. Mutual exclusion is ensured using synchronization mechanisms like:

Semaphores: A signaling mechanism to control access to the critical section.
Mutexes: A mutual exclusion object that allows only one thread to access the critical section at a time.
Monitors: High-level synchronization constructs that provide mutual exclusion and condition variables for managing complex synchronizations.

Recommended Playlists for OS Interviews

YouTube Playlists
1. Operating Systems - Neso Academy
2. Operating Systems - Gate Smashers

Here are some coding questions on operating systems with links:

1. Implement a simple semaphore
2. Implement the producer
3. Implement the readers-writers problem
4. Implement a bounded blocking queue
5. Implement a thread-safe singleton class
6. Implement a simple thread pool
7. Simulate a page replacement algorithm
8. Implement the sleeping barber problem
9. Implement a bounded buffer
10. Simulate a disk scheduling algorithm

These questions should help you practice important operating system concepts through coding.