OPERATING SYSTEM

Made by- @rajayush2167©

25 Thrashing (IMP)

26 Page Replacement Algorithm

27 Virtual Memory (IMP)

28 Disk Scheduling Algorithm

29 Linker & Loader Link

30 Most Asked Interview Questions

Here is a detailed breakdown of the topics you've asked about:

---

1. What is an Operating System (OS)?

An **Operating System (OS)** is software that manages hardware resources and provides services for computer programs. It acts as an intermediary between users and the computer hardware. The primary functions of an OS include managing system resources (CPU, memory, input/output devices), providing an interface for user interaction, handling system security, managing files, and ensuring multitasking.

---

2. Multiprogramming vs Multitasking

- **Multiprogramming**: Refers to the capability of an OS to run multiple programs at the same time by managing the CPU's time effectively. It keeps multiple programs loaded in memory and switches between them, but only one program is actively running at any time.
- **Multitasking**: Is an extension of multiprogramming where the OS runs multiple tasks (processes) concurrently. It gives the illusion of simultaneous execution by quickly switching between tasks. In modern systems, multitasking can be preemptive or cooperative.

---

3. Types of OS

- **Batch OS**: Executes jobs in batches without interaction with the user.
- **Time-Sharing OS**: Allows multiple users to interact with the system simultaneously.
- **Real-Time OS**: Used for time-sensitive applications where the response time must be guaranteed.
- **Distributed OS**: Manages a group of independent computers and makes them appear as a single system to users.

- **Network OS**: Provides features for networking, such as file sharing and communication over a network.
- **Mobile OS**: Designed for mobile devices with constraints on power, processing, and memory (e.g., Android, iOS).

---

4. Schedulers

Schedulers are responsible for determining the order in which processes are executed on the CPU. They can be:

- **Long-term Scheduler**: Decides which processes are admitted into the ready queue (controls the degree of multiprogramming).
- **Short-term Scheduler**: Selects which process will be executed next.
- **Medium-term Scheduler**: Moves processes in and out of the ready queue, swapping processes in memory.

---

5. Common Linux Commands

- `ls`: List directory contents.
- `pwd`: Print the current working directory.
- `cd`: Change the directory.
- `cp`: Copy files or directories.
- `mv`: Move or rename files or directories.
- `rm`: Remove files or directories.
- `chmod`: Change file permissions.
- `ps`: Display active processes.
- `top`: View system performance.
- `kill`: Terminate processes.

---

6. System Calls

A **system call** is an interface between user programs and the OS. It allows programs to request services from the OS, such as file management, process control, communication, and memory allocation.

---

7. Fork System Call

The **fork** system call creates a new process by duplicating the calling process. The child process is identical to the parent, but they have different process IDs. It is commonly used to create new processes.

---

8. Process vs Threads (Important)

- **Process**: An independent unit of execution with its own address space. Each process has its own memory, resources, and execution state.
- **Thread**: A smaller unit of execution within a process. Multiple threads can exist within the same process and share the same resources, but each thread has its own execution stack.

---

9. User Level vs Kernel Level Thread

- **User-Level Thread**: Managed entirely by user-space libraries without OS intervention. Faster context switching but lacks certain OS features.
- **Kernel-Level Thread**: Managed by the OS kernel, which allows it to take advantage of features such as scheduling and synchronization. It is slower than user-level threads but is more efficient in utilizing system resources.

---

10. Process Scheduling Algorithms

- **First Come First Serve (FCFS)**, **Shortest Job First (SJF)**, **Shortest Remaining Time First (SRTF)**, **Round Robin (RR)**, and others are used to decide which process gets CPU time based on different criteria (arrival time, burst time, etc.).

---

11. CPU Scheduling

CPU scheduling is the process by which the OS determines which of the ready processes will be executed by the CPU. It aims to maximize CPU utilization and process throughput, while minimizing turnaround time and waiting time.

---

12. FCFS (First Come First Serve) Algorithm (Important)

- **FCFS** schedules processes in the order they arrive. The first process that arrives is executed first. It is simple but can lead to long waiting times, especially when a long job arrives before shorter ones (known as the **convoy effect**).

---

13. SJF (Shortest Job First) Algorithm (Important)

- **SJF** selects the process with the shortest burst time for execution next. It minimizes average waiting time but requires knowing the burst time of processes ahead of time, which is often impractical.

---

14. SRTF (Shortest Remaining Time First) Algorithm

- **SRTF** is a preemptive version of SJF, where the process with the shortest remaining time is selected next. It minimizes average waiting time, but processes with longer burst times can be preempted by processes with shorter times.

---

15. RR (Round Robin) Algorithm (Important)

- **Round Robin (RR)** is a preemptive scheduling algorithm where each process gets a fixed time slice (quantum) to execute. If it doesn't finish within that time, it is moved to the end of the queue, and the next process is executed.

---

16. PEP (Priority-Exponential Scheduling) Algorithm

- **PEP** is a scheduling algorithm where processes are given priorities. The system assigns priorities based on different parameters like process duration, urgency, etc., and selects the process with the highest priority.

---

17. Process Synchronization

Process synchronization is the coordination of multiple processes to ensure that they do not interfere with each other while sharing resources. Common issues include race conditions, deadlocks, and resource conflicts.

---

18. Semaphores (Important)

A **semaphore** is a synchronization primitive used to manage access to shared resources. It can be either a **binary semaphore** (0 or 1) or a **counting semaphore** (integer values). Semaphores are used to solve problems like mutual exclusion and process synchronization.

---

19. Deadlock (Important)

Deadlock occurs when two or more processes are blocked because they are waiting for each other to release resources, causing the system to freeze. Deadlock can be avoided, detected, and resolved using different strategies.

---

20. Deadlock Handling and Prevention

Deadlock handling involves:

- **Prevention**: Preventing one or more of the necessary conditions for deadlock.
- **Avoidance**: Using algorithms like the **Banker's algorithm** to avoid entering a deadlock state.
- **Detection**: Identifying deadlocks and taking corrective actions.
- **Recovery**: Terminating or rolling back processes to resolve deadlocks.

---

21. Deadlock Avoidance: Banker's Algorithm (Important)

The **Banker's algorithm** is used to avoid deadlock by ensuring that resources are allocated in a way that the system can always remain in a safe state. It checks if a process can obtain all the resources it needs without causing a deadlock.

---

22. Memory Management Techniques

Memory management ensures that each process gets enough memory while avoiding conflicts. Common techniques include:

- **Paging**: Dividing memory into fixed-sized pages.
- **Segmentation**: Dividing memory into segments based on logical divisions (e.g., code, data).
- **Virtual Memory**: Using disk storage as an extension of RAM to provide the illusion of a larger memory space.

- **Dynamic Memory Allocation**: Allocating and deallocating memory dynamically as required by the processes.

Let's go over the remaining topics you've asked about in detail:

---

3. Segmentation (Important)

- **Segmentation** is a memory management scheme that divides a program's memory into different segments, such as code, data, stack, and heap. Unlike paging, which divides memory into fixed-size blocks, segmentation uses variable-sized chunks. This method is more natural as it aligns with the program's logical structure.
- **Advantages**: It provides a more intuitive way to allocate memory for programs, as each segment corresponds to a logical unit (e.g., a function, array, or stack). It also allows easier sharing of code and data segments.
- **Disadvantages**: It can lead to **external fragmentation**, where free memory is scattered in small blocks, making it difficult to allocate large contiguous spaces.

---

24. Paging (Important)

- **Paging** is a memory management scheme that divides both physical memory (RAM) and logical memory (processes) into fixed-sized blocks called **pages** (for processes) and **page frames** (for physical memory).
- **Advantages**: It eliminates external fragmentation and makes it easier to allocate memory. It allows for **non-contiguous memory allocation**, which means a process's memory doesn't need to be stored in a single, contiguous block.
- **Disadvantages**: **Internal fragmentation** can occur because a process may not always perfectly fit into the allocated pages. Additionally, there is overhead associated with maintaining page tables.
- **Page Tables**: The OS maintains a page table that maps virtual pages to physical memory locations.

---

25. Thrashing (Important)

- **Thrashing** occurs when a system spends more time swapping data in and out of memory than executing actual processes. This can happen when the OS is over-committed to running too many processes, causing excessive paging.
- **Symptoms**: A system experiencing thrashing is very slow, as the CPU is constantly waiting for memory pages to be swapped in and out.

- **Causes**: Thrashing usually happens when the system's total memory requirements exceed the available physical memory, causing the operating system to rely heavily on swapping.
- **Solution**: To avoid thrashing, the OS may use **working set management** (ensuring that each process has a set of pages in memory) or use a more efficient **page replacement algorithm**.

---

26. Page Replacement Algorithm

- **Page replacement algorithms** determine how the OS decides which pages to swap out of memory when new pages need to be loaded. Some common algorithms include:
    - **FIFO (First In, First Out)**: The oldest page in memory is replaced.
    - **LRU (Least Recently Used)**: The page that hasn't been used for the longest period is replaced.
    - **Optimal (MIN)**: The page that will not be used for the longest time in the future is replaced. This is the best but not implementable in practice.
    - **Clock (Second-Chance)**: A more efficient approximation of LRU, where pages are arranged in a circular queue and given a second chance before being replaced.

---

27. Virtual Memory (Important)

- **Virtual Memory** is a memory management technique that gives the illusion of a larger memory space than what is physically available by using both physical memory (RAM) and disk storage. Virtual memory allows programs to access a large range of addresses without needing to fit entirely in physical memory.
- **How it works**: It uses paging or segmentation to break processes into smaller chunks (pages or segments) and keeps only the most frequently used chunks in physical memory, while the rest are stored on disk.
- **Advantages**: It allows larger programs to run on machines with limited RAM and helps in efficient memory utilization.
- **Disadvantages**: The main issue is the potential for thrashing if the system runs out of physical memory and has to swap too much data between RAM and disk.

---

28. Disk Scheduling Algorithm

- **Disk scheduling algorithms** are used to decide the order in which disk I/O requests are processed. The goal is to minimize the time it takes to complete disk operations, especially with random access patterns.
    - **FCFS (First Come First Serve)**: Processes requests in the order they arrive, but this can cause high waiting time.

- **SSTF (Shortest Seek Time First)**: Chooses the request closest to the current head position, minimizing seek time.
- **SCAN (Elevator Algorithm)**: The disk arm moves in one direction, serving requests, and when it reaches the end, it reverses direction.
- **C-SCAN (Circular SCAN)**: Similar to SCAN, but when the disk arm reaches the end, it returns to the beginning without serving requests in the reverse direction.
- **LOOK**: Similar to SCAN but the arm only moves as far as the furthest request in the current direction before reversing.
- **C-LOOK**: Like C-SCAN, but only goes to the farthest request in one direction before turning around.

---

29. Linker & Loader

- **Linker**: A linker combines different object files (usually compiled source code) into a single executable program. It resolves symbols (i.e., functions or variables) that were declared but not defined in each module and connects them together.
  - **Static Linking**: Linker resolves all the references at compile time.
  - **Dynamic Linking**: Links libraries during runtime, allowing for shared use of libraries.
- **Loader**: A loader is responsible for loading executable files into memory and preparing them for execution. It places code and data into memory, sets up the stack, heap, and registers, and starts the program execution.
  - **Types of Loaders**: Absolute loaders, Relocating loaders, and Dynamic loaders.

---

30. Most Asked Interview Questions (Important)

Here are some common operating system-related interview questions:

- **What is the difference between process and thread?**
  - Process: Independent execution unit with its own memory space.
  - Thread: Smaller unit of execution within a process, sharing memory with other threads of the same process.
- **What is the role of the operating system in process synchronization?**
  - The OS ensures that processes do not interfere with each other when accessing shared resources. Mechanisms like **semaphores, mutexes**, and **monitors** are used to achieve synchronization.
- **What is deadlock, and how can it be prevented?**
  - **Deadlock** is a situation where two or more processes are unable to proceed because each is waiting for the other to release resources. Prevention involves breaking one or more of the necessary conditions for deadlock.
- **What are the different types of scheduling algorithms?**

    - **FCFS**, **SJF**, **Round Robin**, **Priority Scheduling**, etc., each with its advantages and disadvantages depending on system requirements.
- **What is the difference between paging and segmentation?**
    - **Paging** divides memory into fixed-size blocks (pages), while **segmentation** divides memory into variable-sized segments based on logical divisions (e.g., code, stack).
- **What is virtual memory, and how does it work?**
    - Virtual memory is a technique that uses both RAM and disk storage to create the illusion of a larger memory pool, allowing programs to run even if they don't fit entirely in RAM.
- **What is thrashing? How can it be prevented?**
    - **Thrashing** happens when the system spends more time swapping data than executing processes. It can be prevented by ensuring that the system is not overcommitted and that the working set of processes is properly managed.
- **Explain the concept of semaphores.**
    - A **semaphore** is a synchronization tool used to manage access to shared resources. It can be binary (like a lock) or counting, allowing for more complex synchronization patterns.
- **What is a system call?**
    - A **system call** is a request made by a user-level process to the OS kernel to perform low-level tasks, such as file operations, memory allocation, or process control.

If you need further clarification or more details on any topic, feel free to ask!