1. Define the types of Operating System?

2. Explain DHCP?

3. Explain DNS?

4. Explain paging?

5. Explain segmentation?

6. Explain memory management?

7. Explain the function of OS?

8. Explain a kernel? Its architecture and working?

9. Explain a shell script?

10. Explain a page fault?

11. Explain a deadlock?

12. Define the necessary conditions for deadlock?

13. Explain a semaphore?

14. Explain a mutex?

15. Difference among kernel space and user space.

16. Write in brief the ping command.

17. Explain UNIX?

18. Explain grep?

19. Explain pipe?

20. Difference among Thread & Process.

21. Explain a scheduling algorithm?

22. Explain pre-emptive and non-preemptive scheduling?

23. Define the different scheduling algorithms

24. Explain booting process?

25. Explain bias?

26. Explain the difference among static memory allocation and dynamic memory allocation?

27. UNIX commands like touch, sed, grep.

28. Explain a process and process table? Define different states of process?

29. Define the benefits of multithreaded programming?

30. Explain Thrashing?

31. Explain Belady’s Anomaly?

32. Explain starvation and aging?

33. Explain a trap and trapdoor?

34. Explain a daemon?

35. Which application software's executed on OS?

36. Define daemon objects and thread objects?

37. Give commands for finding process ID.

38. How to edit, rename and move file in Linux?

39. Give 5 commands in Linux with explanation

40. Which are deadlock handling situations?

**1. Types of Operating Systems**

**1.1 Batch Operating Systems:**

* **Definition:** These are designed to execute a series of jobs or tasks sequentially. Jobs are collected, batched together, and executed one after another without user interaction.
* **Example:** Early mainframes like IBM's OS/360.
* **Features:** Minimal user interaction, jobs are processed in the order they are submitted.

**1.2 Time-Sharing Operating Systems:**

* **Definition:** These systems enable multiple users to access the computer system simultaneously. The CPU's time is divided among users in short intervals, creating the illusion that each user has their own computer.
* **Example:** UNIX, Linux.
* **Features:** Efficient resource utilization, multi-user access, and interactive user experience.

**1.3 Real-Time Operating Systems (RTOS):**

* **Definition:** Designed to handle tasks within strict timing constraints. They ensure that critical tasks are processed within a guaranteed time frame.
* **Example:** FreeRTOS, VxWorks.
* **Features:** Predictable response times, high reliability for real-time applications like embedded systems and industrial controls.

**1.4 Distributed Operating Systems:**

* **Definition:** Manage a group of independent computers and present them as a unified system to users. These systems coordinate resources across a network.
* **Example:** Google's internal distributed systems, Hadoop.
* **Features:** Resource sharing, fault tolerance, and distributed computing capabilities.

**1.5 Network Operating Systems:**

* **Definition:** Provide services to manage network resources and support communication between computers in a network. They handle tasks related to networking.
* **Example:** Windows Server, Novell NetWare.
* **Features:** Network management, security, and file sharing.

**1.6 Embedded Operating Systems:**

* **Definition:** Designed for embedded systems with specific functionalities and limited resources. They are tailored to run on devices with constrained hardware.
* **Example:** Android (for smartphones), QNX (for automotive systems).
* **Features:** Efficient resource usage, specialized functionality, and real-time processing.

**2. DHCP (Dynamic Host Configuration Protocol)**

* **Definition:** DHCP is a network protocol used to automatically assign IP addresses and other network configuration parameters to devices on a network.
* **Process:**
  + **Discovery:** A device sends a broadcast message to discover DHCP servers.
  + **Offer:** DHCP servers respond with an offer containing an IP address and configuration information.
  + **Request:** The device requests the offered IP address.
  + **Acknowledgment:** The server confirms the assignment and provides the configuration details.
* **Benefits:** Simplifies network management by automating IP address assignment and reducing configuration errors.

**3. DNS (Domain Name System)**

* **Definition:** DNS translates human-readable domain names into IP addresses that computers use to identify each other on the network.
* **Structure:**
  + **Domain Names:** Organized hierarchically (e.g., .com, example.com, www.example.com).
  + **DNS Records:** Include information such as IP addresses (A records), mail servers (MX records), and name servers (NS records).
  + **DNS Servers:** Resolve domain names by querying different servers in the DNS hierarchy to find the corresponding IP address.
* **Process:** A DNS query involves several steps: resolution by local cache, querying root servers, querying TLD servers, and querying authoritative servers.

**4. Paging**

* **Definition:** Paging is a memory management technique that divides memory into fixed-size blocks called "pages" and maps them to physical memory frames.
* **Process:**
  + **Logical Address Space:** Divided into pages.
  + **Physical Address Space:** Divided into frames.
  + **Page Table:** Maintains the mapping between pages and frames.
  + **Page Fault:** Occurs when a page is not in physical memory, prompting the OS to load it from disk.
* **Advantages:** Simplifies memory allocation and eliminates fragmentation.

**5. Segmentation**

* **Definition:** Segmentation is a memory management technique where memory is divided into segments based on the logical division of processes.
* **Features:**
  + **Segments:** Represent different logical units such as code, data, stack, and heap.
  + **Segment Table:** Maintains the mapping of segments to physical memory locations.
  + **Flexibility:** Allows processes to grow dynamically and provides better alignment with logical divisions.
* **Advantages:** Reduces fragmentation and improves memory management by handling logical divisions.

**6. Memory Management**

* **Definition:** Memory management involves the allocation and deallocation of memory resources to processes and applications.
* **Key Functions:**
  + **Allocation:** Assigning memory to processes and managing both physical and virtual memory.
  + **Deallocation:** Reclaiming memory when processes terminate or no longer need it.
  + **Paging and Segmentation:** Techniques to manage memory efficiently and handle fragmentation.
  + **Virtual Memory:** Extends physical memory by using disk space, allowing for more efficient use of RAM and enabling larger applications to run.

**7. Functions of OS**

* **Process Management:** Controls process creation, scheduling, and termination. It ensures processes are executed efficiently and fairly.
* **Memory Management:** Allocates and deallocates memory, handles paging and segmentation, and manages virtual memory.
* **File System Management:** Manages file creation, deletion, organization, and access permissions. It provides a way to store and retrieve files.
* **Device Management:** Controls and coordinates hardware devices through device drivers. It manages input/output operations and handles device communication.
* **User Interface:** Provides interfaces for user interaction, such as command-line interfaces (CLI) or graphical user interfaces (GUI).
* **Security and Protection:** Ensures system security by managing access controls, user permissions, and protecting against unauthorized access and malware.

**8. Kernel**

* **Definition:** The kernel is the core component of an operating system, responsible for managing system resources and hardware interactions.
* **Architecture:**
  + **Monolithic Kernel:** Contains all OS services in a single large block of code running in kernel mode (e.g., Linux). It handles process management, memory management, device drivers, and file systems within the same address space.
  + **Microkernel:** Only essential services run in the kernel; other services (e.g., device drivers, file systems) run in user space (e.g., QNX). It aims to minimize the kernel's size and improve stability and security.
* **Working:**
  + **System Calls:** The kernel provides an interface for applications to request services such as file operations or process control.
  + **Resource Management:** Manages CPU scheduling, memory allocation, and device I/O.
  + **Inter-process Communication (IPC):** Facilitates communication between processes and manages process synchronization.

**9. Shell Script**

* **Definition:** A shell script is a text file containing a sequence of commands for a Unix-based shell (e.g., Bash). It automates repetitive tasks by executing a series of commands.
* **Features:**
  + **Variables:** Store and manipulate data.
  + **Loops:** Repeat commands for multiple iterations (e.g., for, while).
  + **Conditionals:** Execute commands based on conditions (e.g., if, else).
  + **Examples:** Automating backups, system monitoring, and file management.

**10. Page Fault**

* **Definition:** A page fault occurs when a process tries to access a page of memory that is not currently loaded in physical memory.
* **Handling a Page Fault:**
  + **Detection:** The memory management unit (MMU) triggers a page fault interrupt when an invalid access is detected.
  + **Page Fault Handler:** The OS identifies the location of the required page (typically on disk).
  + **Page Replacement:** If needed, the OS selects a page to evict from physical memory, loads the required page from disk, and updates the page table.
  + **Resume Execution:** The process resumes execution with the newly loaded page.

**11. Deadlock**

**Definition:** Deadlock is a situation in which two or more processes are unable to proceed because each is waiting for a resource held by the other(s). This results in a standstill where none of the involved processes can continue execution.

**Example:**

* **Process A** holds **Resource 1** and waits for **Resource 2**.
* **Process B** holds **Resource 2** and waits for **Resource 1**.

**Consequences:**

* **Resource Starvation:** Processes are stuck indefinitely.
* **System Performance:** Reduced efficiency due to halted processes.

**12. Necessary Conditions for Deadlock**

The following four conditions must be met simultaneously for a deadlock to occur:

1. **Mutual Exclusion:** Resources cannot be shared and are held exclusively by one process at a time.
2. **Hold and Wait:** Processes holding resources can request additional resources without releasing the ones they already hold.
3. **No Preemption:** Resources cannot be forcibly taken from processes; they must be released voluntarily.
4. **Circular Wait:** A circular chain of processes exists, where each process waits for a resource held by the next process in the chain.

**13. Semaphore**

**Definition:** A semaphore is a synchronization tool used to control access to shared resources in concurrent programming. It helps prevent race conditions by managing access to critical sections of code.

**Types:**

* **Binary Semaphore:** Can only take values 0 or 1. It’s used for mutual exclusion (similar to a mutex).
* **Counting Semaphore:** Can take any non-negative integer value, used to control access to a resource pool.

**Operations:**

* **P (Proberen) Operation:** Decreases the semaphore value; if the value is zero, the process waits.
* **V (Verhogen) Operation:** Increases the semaphore value; if processes are waiting, one is allowed to proceed.

**14. Mutex**

**Definition:** A mutex (mutual exclusion) is a synchronization primitive used to ensure that only one thread or process can access a shared resource at a time.

**Features:**

* **Lock/Unlock Operations:** Only one thread can lock the mutex at any given time, ensuring exclusive access to the resource.
* **Use Case:** Typically used to prevent race conditions and ensure data consistency when multiple threads access shared resources.

**15. Difference Between Kernel Space and User Space**

**Kernel Space:**

* **Definition:** The memory area where the OS kernel operates with full access to hardware and system resources.
* **Access:** Unrestricted; has direct access to hardware and system resources.
* **Responsibility:** Manages core functions such as process management, memory management, and device drivers.

**User Space:**

* **Definition:** The memory area where user applications run, with restricted access to prevent interference with kernel operations.
* **Access:** Restricted; cannot directly access hardware or kernel memory.
* **Responsibility:** Executes user applications and provides interfaces to interact with the system.

**16. ping Command**

**Definition:** The ping command is used to test the reachability of a network device or IP address. It measures the round-trip time for messages sent from the originating host to a destination computer and back.

**Function:**

* **Send Echo Request:** Sends ICMP Echo Request messages to the target.
* **Receive Echo Reply:** Measures the time taken for a reply.
* **Output:** Provides information about packet loss and round-trip time.

**17. UNIX**

**Definition:** UNIX is a powerful, multiuser, multitasking operating system known for its stability, portability, and security. It supports multiple users simultaneously and provides a rich set of utilities and tools for various tasks.

**Features:**

* **Multitasking:** Allows multiple processes to run simultaneously.
* **Multiuser:** Supports multiple users with separate permissions and configurations.
* **Portability:** Can be installed on various hardware platforms.
* **Security:** Provides strong access control and protection mechanisms.

**18. grep**

**Definition:** grep is a command-line utility used to search for specific patterns within files or input data. It prints lines that match the given pattern, which can be specified using regular expressions.

**Usage:**

* **Syntax:** grep [options] pattern [file...]
* **Example:** grep "error" logfile.txt searches for the word "error" in logfile.txt.

**19. Pipe**

**Definition:** A pipe (|) is a command-line operator used to connect the output of one command to the input of another. It allows chaining commands, enabling complex processing tasks to be performed in sequence.

**Usage:**

* **Syntax:** command1 | command2
* **Example:** ls | grep "text" lists files and filters the output to include only those containing "text."

**20. Difference Between Thread and Process**

**Process:**

* **Definition:** An independent program in execution with its own memory space and resources.
* **Isolation:** Processes are isolated from each other and communicate through inter-process communication (IPC).
* **Overhead:** Creating and managing processes involves more overhead compared to threads.

**Thread:**

* **Definition:** A lightweight unit of execution within a process that shares the same memory space and resources.
* **Sharing:** Threads within the same process share memory and resources, allowing for more efficient communication.
* **Overhead:** Threads have lower creation and management overhead compared to processes.

**21. Scheduling Algorithm**

**Definition:** A scheduling algorithm determines the order in which processes are executed by the CPU. It manages how processes are selected for execution based on various criteria like priority, execution time, and fairness.

**Types:**

* **First-Come-First-Served (FCFS):** Processes are executed in the order they arrive.
* **Shortest Job Next (SJN):** Selects the process with the shortest execution time next.
* **Round Robin (RR):** Allocates a fixed time slice to each process in a cyclic order.

**22. Pre-emptive vs. Non-Preemptive Scheduling**

**Pre-emptive Scheduling:**

* **Definition:** The operating system can interrupt a running process to assign CPU time to another process.
* **Advantage:** Improves system responsiveness and ensures fair allocation of CPU time.
* **Example:** Round Robin, Shortest Remaining Time First (SRTF).

**Non-Preemptive Scheduling:**

* **Definition:** Once a process starts executing, it runs to completion or until it voluntarily yields control.
* **Advantage:** Simplifies process management and reduces context-switching overhead.
* **Example:** First-Come-First-Served (FCFS), Shortest Job First (SJF).

**23. Different Scheduling Algorithms**

**First-Come-First-Served (FCFS):**

* **Description:** Processes are executed in the order they arrive. Simple but can lead to long wait times (convoy effect).

**Shortest Job Next (SJN):**

* **Description:** Chooses the process with the shortest execution time next. Reduces average waiting time but can cause starvation for longer processes.

**Round Robin (RR):**

* **Description:** Allocates a fixed time slice to each process in a cyclic order. Fair and simple, but can lead to high context switching overhead.

**Priority Scheduling:**

* **Description:** Processes are assigned priorities, and the process with the highest priority is selected next. Can be preemptive or non-preemptive.

**24. Booting Process**

**Definition:** The booting process is the sequence of steps that a computer system follows to load the operating system into memory and prepare the system for use.

**Steps:**

1. **Power-On Self-Test (POST):** Hardware components are checked for functionality.
2. **Bootloader Execution:** The bootloader (e.g., GRUB) is loaded, which initializes the OS loading process.
3. **Kernel Loading:** The OS kernel is loaded into memory and initialized.
4. **System Initialization:** System services and background processes are started.
5. **User Login:** The system presents a login prompt, and the user can start interacting with the OS.

**25. Bias**

**Definition:** Bias refers to a systematic deviation from the true value or expected outcome in various contexts.

**In Computing:**

* **Hardware Bias:** Differences in hardware design or configuration that affect performance or behaviour.
* **Algorithmic Bias:** Systematic favouring of certain outcomes or groups due to design choices in algorithms or software.

**In Statistics:**

* **Sampling Bias:** Occurs when certain groups are overrepresented or underrepresented in a sample, leading to inaccurate conclusions.

**26. Static vs. Dynamic Memory Allocation**

**Static Memory Allocation:**

* **Definition:** Memory is allocated at compile time before the program starts executing. The size and lifetime of the allocated memory are fixed.
* **Characteristics:**
  + **Fixed Size:** Memory size is predetermined and cannot be changed during execution.
  + **Lifetime:** The memory remains allocated for the entire duration of the program.
  + **Efficiency:** Faster access because the memory address is fixed and known at compile time.
  + **Example:** Global variables, static variables in C/C++.

**Dynamic Memory Allocation:**

* **Definition:** Memory is allocated at runtime as needed. The size and lifetime of the memory can vary during program execution.
* **Characteristics:**
  + **Flexible Size:** Memory can be requested and released as needed, allowing for dynamic sizing.
  + **Lifetime:** The memory remains allocated until explicitly deallocated or until the end of the process.
  + **Efficiency:** May involve overhead due to memory management, but provides flexibility.
  + **Example:** Memory allocated using functions like malloc(), calloc(), realloc() in C, or new in C++.

**27. UNIX Commands: touch, sed, grep**

**touch:**

* **Definition:** Creates an empty file or updates the access and modification timestamps of an existing file.
* **Usage:** touch filename
* **Example:** touch newfile.txt creates newfile.txt if it doesn’t exist or updates its timestamps if it does.

**sed:**

* **Definition:** Stream editor used for parsing and transforming text in a pipeline.
* **Usage:** sed [options] 'command' file
* **Example:** sed 's/old/new/g' file.txt replaces all occurrences of "old" with "new" in file.txt.

**grep:**

* **Definition:** Searches for patterns in files and outputs matching lines.
* **Usage:** grep [options] pattern [file...]
* **Example:** grep 'error' logfile.txt searches for lines containing "error" in logfile.txt.

**28. Process and Process Table**

**Process:**

* **Definition:** A process is an instance of a program in execution, including its code, data, and resources.
* **Components:** Process ID (PID), program counter, register values, memory allocation, and I/O status.

**Process Table:**

* **Definition:** A data structure maintained by the OS to track all active processes.
* **Contains:** Information such as PID, process state, priority, CPU usage, memory allocation, and pointers to the process's resources.

**States of a Process:**

1. **New:** The process is being created.
2. **Ready:** The process is waiting to be assigned to a CPU.
3. **Running:** The process is currently executing on the CPU.
4. **Waiting (Blocked):** The process is waiting for an event (e.g., I/O operation) to complete.
5. **Terminated:** The process has finished execution and is removed from the process table.

**29. Benefits of Multithreaded Programming**

* **Improved Performance:** Multithreading allows a program to perform multiple operations simultaneously, utilizing multiple CPU cores effectively.
* **Responsiveness:** In interactive applications, multithreading helps maintain responsiveness by performing background tasks without freezing the user interface.
* **Resource Sharing:** Threads within the same process share memory and resources, reducing overhead compared to multiple processes.
* **Scalability:** Applications can scale better with increased CPU cores, handling more tasks in parallel.

**30. Thrashing**

**Definition:** Thrashing occurs when a system spends more time swapping data between memory and disk (page faults) than executing processes. This results in poor performance and high I/O overhead.

**Causes:**

* **Insufficient Physical Memory:** Too many processes or high memory demands exceed available RAM.
* **High Page Fault Rate:** Excessive paging due to frequent page faults.

**Symptoms:**

* **Reduced System Performance:** Slow response and high disk activity.
* **Increased Swapping:** Excessive page swapping between disk and RAM.

**31. Belady’s Anomaly**

**Definition:** Belady's Anomaly refers to a situation in which increasing the number of page frames results in an increase in the number of page faults, contrary to what might be expected.

**Context:**

* **Virtual Memory:** This anomaly is observed in certain page replacement algorithms, particularly FIFO (First-In-First-Out).
* **Explanation:** Increasing memory may not always reduce page faults due to the algorithm's inefficiency in handling certain access patterns.

**32. Starvation and Aging**

**Starvation:**

* **Definition:** A situation where a process is perpetually denied necessary resources to proceed, often due to other processes receiving priority.
* **Example:** Low-priority processes might starve in a system where high-priority processes keep arriving.

**Aging:**

* **Definition:** A technique to prevent starvation by gradually increasing the priority of processes that have waited for a long time.
* **Mechanism:** As processes wait longer, their priority is increased, eventually allowing them to get CPU time.

**33. Trap and Trapdoor**

**Trap:**

* **Definition:** An exception or interrupt triggered by an exceptional condition during the execution of a program (e.g., division by zero, invalid memory access).
* **Usage:** Allows the OS to handle the exception and potentially recover from errors.

**Trapdoor:**

* **Definition:** A hidden entry point or method in a system that allows unauthorized access or manipulation.
* **Usage:** Often used in the context of security vulnerabilities or backdoors in software.

**34. Daemon**

**Definition:** A daemon is a background process that runs continuously and performs system or application-level tasks without direct user interaction.

**Characteristics:**

* **Background Operation:** Runs in the background, typically started during system boot.
* **Examples:** Web servers (e.g., Apache), database servers (e.g., MySQL), system monitoring tools.

**35. Application Software Executed on OS**

* **Definition:** Software applications designed to perform specific tasks or functions for users, running on top of an operating system.
* **Examples:** Web browsers (e.g., Chrome, Firefox), office suites (e.g., Microsoft Office, LibreOffice), media players (e.g., VLC), and development tools (e.g., IDEs like Eclipse).

**36. Daemon Objects and Thread Objects**

**Daemon Objects:**

* **Definition:** Objects or processes that run in the background to provide various services or functionalities.
* **Characteristics:** Typically not directly interacted with by users; perform routine tasks or handle system services.

**Thread Objects:**

* **Definition:** Threads represent a single path of execution within a process. Multiple threads can run concurrently within the same process, sharing memory and resources.
* **Characteristics:** Used to perform concurrent operations within a process, improving application performance and responsiveness.

**37. Commands for Finding Process ID**

* **ps:** Displays information about active processes.
  + **Example:** ps -e or ps aux shows all processes.
* **top:** Provides a real-time view of system processes.
  + **Example:** top displays process IDs along with other information.
* **pgrep:** Searches for processes based on name and other attributes.
  + **Example:** pgrep process\_name returns the PID of processes matching the name.

**38. Edit, Rename, and Move Files in Linux**

**Edit File:**

* **Command:** nano filename or vim filename
* **Function:** Opens the file in the specified text editor for editing.

**Rename File:**

* **Command:** mv old name newname
* **Function:** Renames a file or moves it to a different location.

**Move File:**

* **Command:** mv filename /path/to/destination/
* **Function:** Moves the file to the specified directory.

**39. Five Commands in Linux with Explanation**

**ls:**

* **Definition:** Lists directory contents.
* **Example:** ls -l shows detailed information about files and directories.

**cp:**

* **Definition:** Copies files or directories.
* **Example:** cp file1.txt file2.txt copies file1.txt to file2.txt.

**rm:**

* **Definition:** Removes files or directories.
* **Example:** rm file.txt deletes file.txt.

**mkdir:**

* **Definition:** Creates a new directory.
* **Example:** mkdir newdir creates a directory named newdir.

**chmod:**

* **Definition:** Changes file permissions.
* **Example:** chmod 755 file.sh sets the permissions of file.sh to rwxr-xr-x.

**40. Deadlock Handling Situations**

**Deadlock Prevention:**

* **Eliminate one of the necessary conditions for deadlock.** For example, using preemptive resource allocation to avoid circular wait.

**Deadlock Avoidance:**

* **Use algorithms to ensure that the system never enters a deadlock state.** The Banker's Algorithm is an example that checks resource requests for safety before allocation.

**Deadlock Detection:**

* **Detect deadlocks and take corrective actions.** Includes algorithms that periodically check for cycles in resource allocation graphs and initiate recovery.

**Deadlock Recovery:**

* **Recover from deadlock once detected.** Techniques include terminating processes, rolling back processes to a safe state, or preempting resources.