Operating System Interview Questions

1. Define the types of Operating System?

- Batch Operating System: Executes jobs in batches without manual intervention.

- Time-Sharing Operating System: Allows multiple users to use a system simultaneously by sharing time.

- Distributed Operating System: Manages a group of distinct computers and makes them appear to be a single computer.

- Real-Time Operating System: Processes data as it comes in, typically used in embedded systems.

- Network Operating System: Provides services to computers connected to a network.

- Mobile Operating System: Designed specifically for mobile devices.

2. Explain DHCP?

- DHCP (Dynamic Host Configuration Protocol) is a network management protocol used to automatically assign IP addresses and other communication parameters to devices on a network, enabling them to communicate.

3. Explain DNS?

- DNS (Domain Name System) translates human-readable domain names (like www.example.com) into IP addresses that computers use to identify each other on the network.

4. Explain paging?

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory by dividing the process's memory into fixed-sized pages and mapping them to physical frames.

5. Explain segmentation?

- Segmentation divides a program's memory into different segments (like code, data, stack) of varying lengths, improving flexibility and protection in memory management.

6. Explain memory management?

- Memory management involves efficiently allocating, managing, and deallocating memory space to processes while ensuring protection, speed, and fairness.

7. Explain the function of OS?

- Functions of OS include managing hardware resources, providing an interface between users and hardware, file management, process management, memory management, and ensuring security.

8. Explain a kernel? Its architecture and working?

- Kernel is the core component of an OS that manages system resources and communication between hardware and software. Architecture typically includes monolithic, microkernel, or hybrid structures. It manages processes, memory, device I/O, and system calls.

9. Explain a shell script?

- Shell script is a script written for the shell, or command-line interpreter, of an operating system. It automates tasks by executing commands in sequence.

10. Explain a page fault?

- Page fault occurs when a program tries to access a page that is not currently in physical memory, triggering the OS to retrieve the data from secondary storage (swap space).

11. Explain a deadlock?

- Deadlock is a situation in an OS where two or more processes are unable to proceed because each is waiting for the other to release resources.

12. Define the necessary conditions for deadlock?

- Mutual Exclusion: Only one process can use a resource at a time.

- Hold and Wait: A process holding a resource can request more resources.

- No Preemption: Resources cannot be forcibly taken from a process.

- Circular Wait: A closed chain of processes exists, with each holding a resource needed by the next.

13. Explain a semaphore?

- Semaphore is a synchronization primitive used to control access to a common resource in concurrent systems, typically using counters to signal availability.

14. Explain a mutex?

- Mutex (Mutual Exclusion Object) is a locking mechanism used to ensure that only one thread can access a resource at a time, preventing race conditions.

15. Difference among kernel space and user space.

- Kernel Space: Where the core of the operating system operates, with full access to hardware.

- User Space: Where user applications run, with restricted access to system resources for security and stability.

16. Write in brief the ping command.

- ping is a network utility used to test the reachability of a host on an IP network by sending ICMP echo request messages and waiting for a reply.

17. Explain UNIX?

- UNIX is a powerful, multiuser, multitasking operating system originally developed in the 1970s. It is known for its portability, efficiency, and the large number of utilities it provides.

18. Explain grep?

- grep is a command-line utility in UNIX/Linux that searches for a specified pattern in files or input and prints matching lines.

19. Explain pipe?

- Pipe (|) is a method of connecting multiple commands in UNIX/Linux, where the output of one command serves as the input to the next.

20. Difference among Thread & Process.

- Process: An independent program in execution with its own memory space.

- Thread: A lightweight process that shares memory and resources with other threads within the same process.

21. Explain a scheduling algorithm?

- Scheduling Algorithm determines the order in which processes run on the CPU. Examples include First-Come, First-Served (FCFS), Shortest Job Next (SJN), and Round Robin (RR).

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

- Pre-emptive Scheduling: The CPU can be taken away from a running process to allow another process to run.

- Non-preemptive Scheduling: Once a process starts running, it cannot be stopped until it finishes or voluntarily yields the CPU.

23. Define the different scheduling algorithms.

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

- SJN (Shortest Job Next): Processes with the shortest burst time are scheduled first.

- Priority Scheduling: Processes are scheduled based on priority.

- Round Robin (RR): Each process gets a fixed time slice in a cyclic order.

- Multilevel Queue Scheduling: Processes are classified into different queues based on priority or type.

24. Explain booting process?

- Booting is the process of starting a computer and loading the operating system into memory. It typically involves executing the BIOS/UEFI, loading the bootloader, and then loading the OS kernel.

25. Explain bias?

- Bias in an OS context can refer to unfair scheduling or resource allocation favoring certain processes over others, though it is more commonly discussed in the context of machine learning or algorithms.

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

- Static Memory Allocation: Memory is allocated at compile time and cannot be resized.
- Dynamic Memory Allocation: Memory is allocated at runtime and can be resized.

27. UNIX commands like touch, sed, grep.

- touch: Creates an empty file or updates the timestamp of an existing file.
- sed: A stream editor for filtering and transforming text.
- grep: Searches for patterns within files or input streams.

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

- Process: A program in execution.
- Process Table: A data structure maintained by the OS to track information about all processes.
- Process States: New, Ready, Running, Waiting, Terminated.

29. Define the benefits of multithreaded programming?

- Benefits: Increased efficiency, responsiveness, resource sharing, and parallelism.

30. Explain Thrashing?

- Thrashing occurs when excessive paging operations lead to a significant slowdown because the system spends more time swapping pages in and out of memory than executing processes.

31. Explain Belady's Anomaly?

- Belady's Anomaly refers to the counterintuitive situation where increasing the number of page frames results in an increase in the number of page faults in certain page replacement algorithms (notably FIFO).

32. Explain starvation and aging?

- Starvation: A process is perpetually denied necessary resources.

- Aging: A technique to gradually increase the priority of waiting processes to avoid starvation.

33. Explain a trap and trapdoor?

- Trap: A software-generated interrupt caused by an error or a specific request from a user program.

- Trapdoor: A hidden feature or functionality within a program used for unauthorized access or control.

34. Explain a daemon?

- Daemon is a background process that runs continuously and handles requests for services, such as web servers or print servers, in UNIX/Linux systems.

35. Which application software's executed on OS?

- Application Software: Examples include word processors, browsers, games, and multimedia editors. These run atop the OS, using its services to interact with hardware.

36. Define daemon objects and thread objects?

- Daemon Object: Refers to a daemon process in an OS, typically running in the background.

- Thread Object: Represents a thread in programming, encapsulating the thread's attributes and behavior.

37. Give commands for finding process ID.

- Commands:

  - `ps` (displays the current processes along with their IDs)

  - `pgrep` (searches for processes by name and returns their IDs)

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

- Edit: Use text editors like `vi`, `nano`, or `gedit`.

- Rename: Use `mv oldname newname`.

- Move: Use `mv filename /destination/directory/`.

39. Give 5 commands in Linux with explanation.

- `ls`: Lists directory contents.

- `cd`: Changes the current directory.

- `cp`: Copies files or directories.

- `rm`: Removes files or directories.

- `chmod`: Changes file permissions.

40. Which are deadlock handling situations?

- Deadlock Handling:

  - Prevention: Design the system in such a way that deadlocks cannot occur.

  - Avoidance: Use algorithms (like Banker's Algorithm) to avoid unsafe states.

  - Detection and Recovery: Detect deadlocks and recover by terminating processes or preempting resources.

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?

   * Operating System (OS): The OS is software that manages computer hardware and software resources and provides services for computer programs.

   * Primary Functions:

   - Process Management: Manages the execution of processes, including multitasking, process scheduling, and synchronization.

   - Memory Management: Handles memory allocation and deallocation, managing physical and virtual memory.

   - File System Management: Manages files on storage devices, including their creation, deletion, and access permissions.

   - Device Management: Manages hardware devices, providing interfaces for device communication.

   - Security and Access Control: Ensures data integrity and controls access to resources.

2. Explain the difference between process and thread.

   - Process: An independent execution unit with its own memory space and resources. Each process runs in its own address space, making it isolated from others.

- Thread: A smaller unit of a process that shares the process's resources but can execute independently. Multiple threads within the same process share memory and file descriptors, leading to faster communication but potential issues with data consistency.

3. What is virtual memory, and how does it work?

- Virtual Memory: An abstraction of physical memory that allows a computer to use more memory than is physically available by using disk space as an extension of RAM.

- How it Works: The OS divides virtual memory into pages, which can be mapped to physical memory (RAM) or stored on disk (swap space). When a program accesses memory not currently in RAM, a page fault occurs, and the OS retrieves the required page from disk.

4. Describe the difference between multiprogramming, multitasking, and multiprocessing.

- Multiprogramming: Multiple programs are loaded into memory, and the CPU executes parts of each program by switching between them to maximize resource utilization.

- Multitasking: A type of multiprogramming where multiple tasks (processes or threads) are executed concurrently, giving the appearance that they are running simultaneously.

- Multiprocessing: The use of multiple CPUs or cores to execute multiple processes simultaneously, improving overall system performance.

5. What is a file system, and what are its components?

- File System: A method of organizing and storing files on storage devices like hard drives.

- Components:

- Files: The smallest unit of data storage in a file system.

- Directories: Folders that organize files into a hierarchical structure.

- Inodes: Data structures that store information about files (e.g., permissions, size, and pointers to data blocks).

- Superblock: Contains metadata about the file system itself, such as its size and status.

6. What is a deadlock, and how can it be prevented?

- Deadlock: A situation where two or more processes are unable to proceed because each is waiting for the other to release resources.

- Prevention Techniques:

- Mutual Exclusion: Ensuring that resources are not shared if possible.

- Hold and Wait: Avoiding situations where a process holds a resource while waiting for another.

- No Preemption: Allowing the OS to forcibly release resources from a process if needed.

- Circular Wait: Preventing circular chains of resource requests by ordering resources and forcing processes to request them in order.

7. Explain the difference between a kernel and a shell.

- Kernel: The core of the OS, responsible for managing hardware, processes, memory, and system calls. It operates in kernel mode with unrestricted access to system resources.

- Shell: A user interface for accessing the OS services, typically via a command-line interface (CLI). It interprets user commands and sends them to the kernel for execution.

8. What is CPU scheduling, and why is it important?

- CPU Scheduling: The process of determining which process or thread should be executed by the CPU at any given time.

- Importance: Efficient CPU scheduling maximizes CPU utilization, minimizes waiting time, ensures fair process allocation, and improves system responsiveness.

9. How does a system call work?

- System Call: A mechanism that allows user-level programs to request services from the kernel. When a system call is made, the CPU switches from user mode to kernel mode, the kernel executes the requested service, and then control is returned to the user program.

10. What is the purpose of device drivers in an operating system?

- Device Drivers: Software components that allow the OS to communicate with hardware devices. They translate OS commands into device-specific instructions and manage device operations like input/output (I/O) and interrupt handling.

11. Explain the role of the page table in virtual memory management.

- Page Table: A data structure used in virtual memory systems to map virtual addresses to physical addresses. Each process has its own page table, which the OS uses to translate memory accesses and manage page faults.

12. What is thrashing, and how can it be avoided?

- Thrashing: A condition where excessive paging operations lead to a significant slowdown in system performance because the OS spends more time swapping pages in and out of memory than executing processes.
- Avoidance Techniques:
  - Increase Physical Memory: Reducing the need for swapping.
  - Adjust Paging Algorithms: Using more efficient algorithms for page replacement.
  - Limit Degree of Multiprogramming: Reducing the number of active processes to decrease page fault rates.

13. Describe the concept of a semaphore and its use in synchronization.

- Semaphore: A synchronization primitive used to manage access to shared resources by multiple processes or threads. It uses counters to signal when a resource is available or needs to be waited on.
- Use in Synchronization: Semaphores prevent race conditions by ensuring that only one process or thread accesses a resource at a time.

14. How does an operating system handle process synchronization?

- Process Synchronization: The OS uses mechanisms like semaphores, mutexes, monitors, and condition variables to coordinate the execution of processes so that they can access shared resources without conflicts.

15. What is the purpose of an interrupt in operating systems?

- Interrupt: A signal sent to the CPU that temporarily halts the current process to address an event (e.g., I/O completion, hardware malfunction). The OS handles the interrupt and then resumes the interrupted process.

16. Explain the concept of a file descriptor.

- File Descriptor: An integer value that uniquely identifies an open file or I/O resource in a process. File descriptors are used by the OS to keep track of files and resources that a process is interacting with.

17. How does a system recover from a system crash?

- System Recovery: The OS may use techniques such as journaling file systems, snapshots, and checkpointing to restore the system to a consistent state after a crash. Recovery processes may include replaying logs, restoring from backups, or repairing corrupted files.

18. Describe the difference between a monolithic kernel and a microkernel.

- Monolithic Kernel: A kernel architecture where the entire OS runs in kernel space, with all components (e.g., file system, drivers) tightly integrated.

- Microkernel: A smaller kernel that only includes essential functions (e.g., communication, basic I/O), with other services running in user space. This design improves modularity and reliability.

19. What is the difference between internal and external fragmentation?

- Internal Fragmentation: Occurs when allocated memory blocks are slightly larger than the requested memory, leaving small unused spaces inside them.

- External Fragmentation: Occurs when free memory is split into small, non-contiguous blocks, making it difficult to allocate large contiguous memory blocks.

20. How does an operating system manage I/O operations?

- I/O Management: The OS handles I/O operations using device drivers, buffering, and interrupt handling. It ensures data is transferred between memory and I/O devices efficiently, coordinating access to shared I/O resources.

21. Explain the difference between preemptive and non-preemptive scheduling.

- Preemptive Scheduling: The CPU can interrupt and reassign processes, allowing the OS to switch processes based on priority, time slices, or other criteria.

- Non-preemptive Scheduling: Once a process starts running, it cannot be interrupted until it finishes or voluntarily yields the CPU.

22. What is round-robin scheduling, and how does it work?

- Round-Robin Scheduling: A CPU scheduling algorithm where each process is assigned a fixed time slice (quantum) and is cycled through in order. If a process doesn't finish within its time slice, it is moved to the end of the queue, and the next process is scheduled.

23. Describe the priority scheduling algorithm. How is priority assigned to processes?

- Priority Scheduling: Processes are scheduled based on their priority, with higher-priority processes getting preference.

- Priority Assignment: Can be static (assigned at creation) or dynamic (changes over time based on criteria like aging, resource requirements, or user-defined settings).

24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?

- Shortest Job Next (SJN): A scheduling algorithm where the process with the shortest execution time is selected next.

- Usage: Ideal in batch processing systems where job lengths are known in advance, as it minimizes average waiting time.

25. Explain the concept of multilevel queue scheduling.

- Multilevel Queue Scheduling: Processes are grouped into different queues based on priority or type (e.g., system processes, interactive processes). Each queue has its own scheduling algorithm, and processes can be promoted or demoted between queues.

26. What is a process control block (PCB), and what information does it contain?

- Process Control Block (PCB): A data structure used by the OS to store information about a process, such as:

  - Process ID (PID): Unique identifier for the process.

  - Process State: Current state (e.g., running, waiting).

  - CPU Registers: Saved state of the CPU when the process is not executing.

  - Memory Management Info: Information on memory allocation, page tables.

- I/O Status: Information on I/O devices assigned to the process.
- Accounting Info: CPU usage, process priority, etc.

27. Describe the process state diagram and the transitions between different process states.

New State: In this step, the process is about to be created but not yet created. It is the program that is present in secondary memory that will be picked up by the OS to create the process.

Ready State: New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue called a ready queue for ready processes.

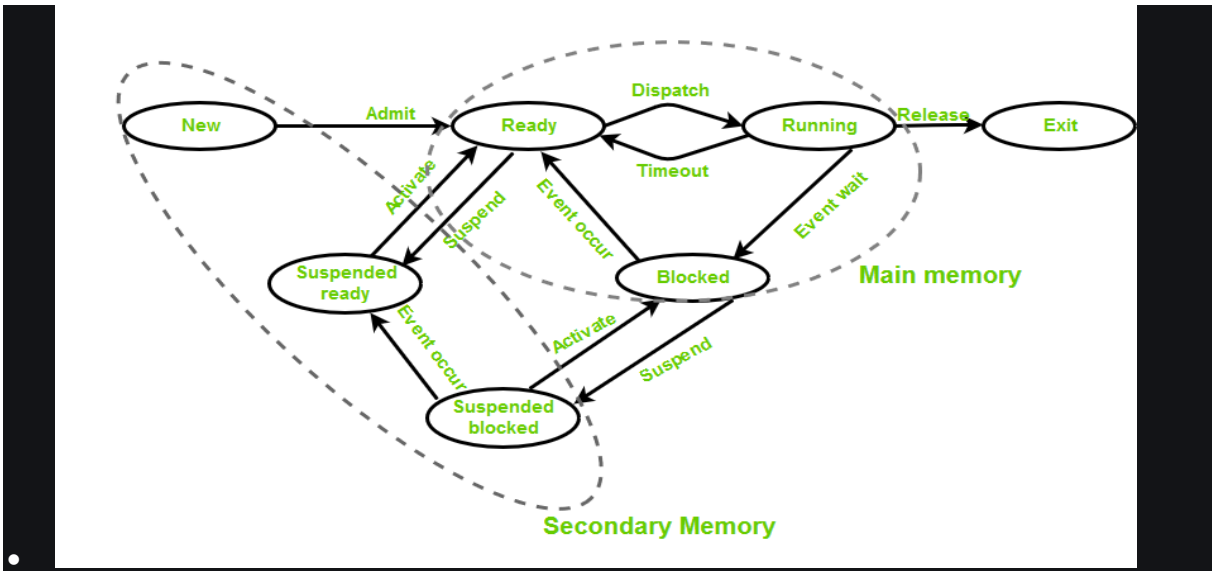
Run State: The process is chosen from the ready queue by the OS for execution and the instructions within the process are executed by any one of the available CPU cores.

Blocked or Wait State: Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or waits state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.

Terminated or Completed State: Process is killed as well as PCB is deleted. The resources allocated to the process will be released or deallocated.

Suspend Ready: Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought onto the main memory.

Suspend Wait or Suspend Blocked: Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.



CPU and I/O Bound Processes: If the process is intensive in terms of CPU operations, then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.

- Process State Diagram:
  - New: The process is being created.
  - Ready: The process is waiting to be assigned to a CPU.
  - Running: The process is currently being executed by the CPU.
  - Waiting: The process is waiting for some event (e.g., I/O completion).
  - Terminated: The process has finished execution.
- Transitions:
  - New to Ready: Process creation completes.
  - Ready to Running: Scheduler assigns CPU to the process.
  - Running to Waiting: Process requests I/O.
  - Running to Ready: Process is preempted by the scheduler.
  - Waiting to Ready: I/O operation completes.
  - Running to Terminated: Process completes execution.

28. How does a process communicate with another process in an operating system?

- Interprocess Communication (IPC): Processes communicate using mechanisms like:

- Pipes: Unidirectional communication channel.
- Message Queues: Store messages between processes.
- Shared Memory: A memory segment shared between processes.
- Semaphores: Synchronization tools to control access to shared resources.

29. What is process synchronization, and why is it important?

- Process Synchronization: Ensures that multiple processes or threads can safely access shared resources without conflicts, preventing race conditions and ensuring data consistency.

30. Explain the concept of a zombie process and how it is created.

- Zombie Process: A process that has completed execution but still has an entry in the process table, holding its exit status until the parent process retrieves it using `wait()`.
- Creation: Happens when a child process terminates, but the parent hasn't yet read its exit status.

31. Describe the difference between internal fragmentation and external fragmentation.

- Internal Fragmentation: Occurs when allocated memory blocks are slightly larger than the requested memory, leaving small unused spaces inside them.
- External Fragmentation: Occurs when free memory is split into small, non-contiguous blocks, making it difficult to allocate large contiguous memory blocks.

32. What is demand paging, and how does it improve memory management efficiency?

- Demand Paging: A type of virtual memory management where pages are loaded into memory only when they are needed, reducing the amount of physical memory required and minimizing I/O overhead.

33. Explain the role of the page table in virtual memory management.

- Page Table: A data structure used in virtual memory systems to map virtual addresses to physical addresses. Each process has its own page table, which the OS uses to translate memory accesses and manage page faults.

34. How does a memory management unit (MMU) work?

- Memory Management Unit (MMU): A hardware component that translates virtual addresses to physical addresses using the page table. It handles address translation, access control, and memory protection.

35. What is thrashing, and how can it be avoided in virtual memory systems?

Thrashing: A condition where excessive paging operations lead to a significant slowdown in system performance because the OS spends more time swapping pages in and out of memory than executing processes.

36. What is a system call, and how does it facilitate communication between user programs and the operating system?

System Call: A mechanism that allows user-level programs to request services from the kernel. When a system call is made, the CPU switches from user mode to kernel mode, the kernel executes the requested service, and then control is returned to the user program.

37. Describe the difference between a monolithic kernel and a microkernel.

- **Monolithic Kernel:** A kernel architecture where the entire OS runs in kernel space, with all components (e.g., file system, drivers) tightly integrated.
- **Microkernel:** A smaller kernel that only includes essential functions (e.g., communication, basic I/O), with other services running in user space. This design improves modularity and reliability.

38. How does an operating system handle I/O operations?

I/O Management: The OS handles I/O operations using device drivers, buffering, and interrupt handling. It ensures data is transferred between memory and I/O devices efficiently, coordinating access to shared I/O resources.

39. Explain the concept of a race condition and how it can be prevented.

- Race Condition: A situation where the behavior of software depends on the relative timing of events, such as process execution, leading to unpredictable outcomes.

- Prevention: Use synchronization mechanisms like locks, semaphores, and atomic operations to ensure that critical sections of code are executed by only one process at a time.

40. Describe the role of device drivers in an operating system.

- **Device Drivers:** Software components that allow the OS to communicate with hardware devices. They translate OS commands into device-specific instructions and manage device operations like input/output (I/O) and interrupt handling.

41. What is a zombie process, and how does it occur? How can a zombie process be prevented?

- **Zombie Process:** A process that has completed execution but still has an entry in the process table, holding its exit status until the parent process retrieves it using `wait()`.

- Prevention: Parent processes should use `wait()` or `waitpid()` to retrieve the exit status of their child processes, removing the zombie from the process table.

42. Explain the concept of an orphan process. How does an operating system handle orphan processes?

- Orphan Process: A process whose parent has terminated. The OS typically reassigns orphan processes to the `init` process (in Unix-like systems), which adopts them and cleans them up when they terminate.

43. What is the relationship between a parent process and a child process in the context of process management?

- Parent-Child Relationship: A parent process creates a child process using system calls like `fork()`. The child inherits attributes (e.g., environment, open file descriptors) from the parent but has a unique process ID.

44. How does the fork() system call work in creating a new process in Unix-like operating systems?

- fork(): The `fork()` system call creates a new process (the child) by duplicating the parent process. The child gets a copy of the parent's memory space, but the two processes execute independently.

45. Describe how a parent process can wait for a child process to finish execution.

- wait(): The parent process can use `wait()` or `waitpid()` to pause its execution until one of its child processes terminates. These calls also retrieve the child's exit status.

46. What is the significance of the exit status of a child process in the wait() system call?

- Exit Status: Indicates how a child process terminated (e.g., normal exit, error, or signal). The parent can use this information to decide how to proceed or to detect errors in the child process.

47. How can a parent process terminate a child process in Unix-like operating systems?

- Termination: The parent can send a signal (e.g., `SIGTERM`, `SIGKILL`) to the child process using the `kill()` system call to terminate it.

48. Explain the difference between a process group and a session in Unix-like operating systems.

- Process Group: A collection of related processes that can receive signals as a group, usually created by a shell when executing a pipeline of commands.

- Session: A collection of process groups, typically associated with a user login session. A session can have a controlling terminal and manage job control within a shell.

49. Describe how the exec() family of functions is used to replace the current process image with a new one.

- exec(): The `exec()` family of functions (e.g., `execl()`, `execvp()`) replaces the current process image with a new program. This means the process ID remains the same, but the code, data, and stack are replaced with those of the new program.

50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?

- waitpid(): Allows a parent process to wait for a specific child process to terminate, based on its process ID.

- Difference from wait(): `waitpid()` provides more control, allowing the parent to wait for a specific child or use options like `WNOHANG` to return immediately if no child has exited.

51. How does process termination occur in Unix-like operating systems?

- Process Termination: A process can terminate by:

  - Normal Exit: The process finishes execution and calls `exit()`.

  - Signal: The process is terminated by a signal, either from another process or from the OS.

  - Fatal Error: The process encounters a fatal error (e.g., illegal instruction).

  - The OS then cleans up resources associated with the process, possibly leaving a zombie process if the parent hasn't read its exit status.

52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?

- Long-Term Scheduler: Decides which processes should be admitted to the system for execution, controlling the degree of multiprogramming (number of processes in memory). It helps maintain a balance between CPU-bound and I/O-bound processes to optimize resource utilization.

53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?

- Short-Term Scheduler: Also known as the CPU scheduler, it selects which process should run next from the ready queue. It operates frequently (every few milliseconds) and makes decisions based on CPU burst time.

- Long-Term Scheduler: Decides which processes should be admitted to the system for execution, controlling the degree of multiprogramming (number of processes in memory). It helps maintain a balance between CPU-bound and I/O-bound processes to optimize resource utilization.

- Medium-Term Scheduler: Manages swapping (moving processes in and out of memory) to control the level of multiprogramming. It operates less frequently than the short-term scheduler.

54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

- **Scenario:** If the system is experiencing thrashing due to excessive page swapping, the medium-term scheduler may suspend some processes (swapping them out of memory) to reduce the load and improve performance.
- **Efficiency:** By reducing the number of processes in memory, the medium-term scheduler decreases competition for CPU and I/O resources, leading to more efficient execution of the remaining processes.