Part D

Common Interview Questions (Must know)

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

2. Explain the difference between process and thread.

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

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

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

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

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

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

9. How does a system call work?

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

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

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

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

14. How does an operating system handle process synchronization?

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

16. Explain the concept of a file descriptor.

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

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

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

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

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

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

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

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

25. Explain the concept of multilevel queue scheduling.

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

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

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

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

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

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

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

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

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

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

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

operating system?

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

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

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

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

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

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

processes?

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

management?

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

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

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

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

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

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

new one.

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

wait()?

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

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?

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?

54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps

manage system resources more efficiently.

1. **What is an operating system, and what are its primary functions?**
   * An operating system (OS) is software that acts as an intermediary between computer hardware and user applications. Its primary functions include:
     + Process management: Creation, scheduling, and termination of processes.
     + Memory management: Allocation and deallocation of memory resources.
     + File system management: Organization and manipulation of files and directories.
     + Device management: Control and coordination of peripheral devices.
     + User interface: Interaction between users and the computer system.
2. **Explain the difference between process and thread.**
   * A process is an instance of a running program, while a thread is a subset of a process. Multiple threads can exist within a single process and share resources such as memory.
3. **What is virtual memory, and how does it work?**
   * Virtual memory is a memory management technique that allows a computer to compensate for physical memory shortages by temporarily transferring data from RAM to disk storage. It provides the illusion of having more memory than is physically available.
4. **Describe the difference between multiprogramming, multitasking, and multiprocessing.**
   * Multiprogramming: Running multiple programs simultaneously by switching between them.
   * Multitasking: Performing multiple tasks (such as running applications) simultaneously within a single operating system.
   * Multiprocessing: Using multiple processors or cores to execute tasks concurrently.
5. **What is a file system, and what are its components?**
   * A file system organizes and manages data stored on a storage device. Its components include directories, files, and metadata such as file attributes and access permissions.
6. **What is a deadlock, and how can it be prevented?**
   * A deadlock occurs when two or more processes are unable to proceed because each is waiting for the other to release a resource. Deadlocks can be prevented by employing techniques such as resource allocation strategies, deadlock detection, and avoidance algorithms.
7. **Explain the difference between a kernel and a shell.**
   * The kernel is the core component of an operating system that manages hardware resources and provides essential services to user applications.
   * The shell is a command-line interface that allows users to interact with the operating system by entering commands.
8. **What is CPU scheduling, and why is it important?**
   * CPU scheduling is the process of determining which process gets to use the CPU and for how long. It is important for maximizing CPU utilization and overall system performance.
9. **How does a system call work?**
   * A system call is a request made by a user program to the operating system kernel for a service. It involves transitioning from user mode to kernel mode to execute privileged instructions.
10. **What is the purpose of device drivers in an operating system?**
    * Device drivers are software components that allow the operating system to communicate with hardware devices. They provide an abstraction layer that enables applications to interact with hardware in a uniform manner.
11. **Explain the role of the page table in virtual memory management.**
    * The page table is a data structure used by the operating system to map virtual addresses to physical addresses in virtual memory. It enables efficient memory management by allowing the system to allocate and access memory in smaller, manageable units called pages.
12. **What is thrashing, and how can it be avoided?**
    * Thrashing occurs when a system spends a significant amount of time swapping data between physical memory and disk storage due to excessive paging activity. It can be avoided by optimizing memory usage, adjusting system parameters, and implementing effective memory management strategies.
13. **Describe the concept of a semaphore and its use in synchronization.**
    * A semaphore is a synchronization primitive used to control access to shared resources by multiple processes or threads. It maintains a counter that can be incremented or decremented atomically to signal the availability of resources.
14. **How does an operating system handle process synchronization?**
    * Operating systems use various synchronization mechanisms such as semaphores, mutexes, and condition variables to ensure that concurrent processes or threads access shared resources safely and avoid race conditions or deadlocks.
15. **What is the purpose of an interrupt in operating systems?**
    * An interrupt is a signal generated by hardware or software to notify the CPU of an event that requires immediate attention. It allows the operating system to handle asynchronous events, such as I/O operations or hardware interrupts, efficiently.
16. **Explain the concept of a file descriptor.**
    * A file descriptor is a unique identifier used by the operating system to access files and other I/O resources. It represents an open file or stream and is used by applications to perform read, write, or seek operations on the file.
17. **How does a system recover from a system crash?**
    * System recovery mechanisms such as journaling file systems, system checkpoints, and system logs are used to recover from system crashes. These mechanisms help restore the system to a stable state and recover data that may have been lost due to the crash.
18. **Describe the difference between a monolithic kernel and a microkernel.**
    * A monolithic kernel incorporates all operating system functions, such as process management, memory management, and device drivers, into a single kernel image.
    * A microkernel delegates most operating system functions to user-space processes, keeping only essential functions, such as inter-process communication and memory management, in the kernel.
19. **What is the difference between internal and external fragmentation?**
    * Internal fragmentation occurs when allocated memory blocks are larger than necessary, resulting in wasted space within a memory segment.
    * External fragmentation occurs when free memory blocks are scattered throughout memory, making it challenging to allocate contiguous blocks of memory to processes.
20. **How does an operating system manage I/O operations?**
    * Operating systems use device drivers and I/O subsystems to manage I/O operations. They provide interfaces for applications to access devices, handle device interrupts, and implement I/O scheduling algorithms to optimize performance.
21. **Explain the difference between preemptive and non-preemptive scheduling.**
    * Preemptive scheduling allows the operating system to interrupt the execution of a process to allocate the CPU to another process with a higher priority.
    * Non-preemptive scheduling does not allow the operating system to interrupt the execution of a process voluntarily. Instead, the currently running process must relinquish the CPU voluntarily.
22. **What is round-robin scheduling, and how does it work?**
    * Round-robin scheduling is a CPU scheduling algorithm that allocates CPU time to processes in a fixed time-slice or quantum. Processes are scheduled in a circular queue, and each process is given equal time to execute before being preempted and placed back in the queue.
23. **Describe the priority scheduling algorithm. How is priority assigned to processes?**
    * Priority scheduling is a CPU scheduling algorithm that assigns priority levels to processes based on factors such as CPU burst time, deadline constraints, or user-defined priorities. Processes with higher priority levels are given preference in CPU allocation.
24. **What is the shortest job next (SJN) scheduling algorithm, and when is it used?**
    * The shortest job next (SJN) scheduling algorithm selects the process with the shortest CPU burst time for execution. It is also known as the shortest job first (SJF) scheduling algorithm and is used to minimize average waiting time and turnaround time.
25. **Explain the concept of multilevel queue scheduling.**

* Multilevel queue scheduling is a CPU scheduling algorithm that divides processes into multiple queues based on specific criteria such as priority or process type. Each queue has its own scheduling algorithm, allowing processes to be scheduled differently based on their characteristics.

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

* A process control block (PCB) is a data structure used by the operating system to manage information about each process. It contains essential information such as process state, program counter, CPU registers, memory allocation, and scheduling information.

1. **Describe the process state diagram and the transitions between different process states.**

* The process state diagram represents the various states that a process can be in, such as ready, running, blocked, or terminated. Transitions between these states occur based on events such as CPU scheduling, I/O operations, or process completion.

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

* Processes can communicate with each other through inter-process communication (IPC) mechanisms provided by the operating system, such as pipes, message queues, shared memory, or sockets.

1. **What is process synchronization, and why is it important?**

* Process synchronization is the coordination of multiple processes to ensure they operate correctly when accessing shared resources or communicating with each other. It is essential for preventing race conditions, deadlocks, and ensuring data integrity.

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

* A zombie process is a terminated process that has completed execution but still has an entry in the process table. It is created when a parent process fails to call the **wait()** system call to retrieve the exit status of its child process.

1. **Describe the difference between internal fragmentation and external fragmentation.**

* Internal fragmentation occurs when allocated memory blocks are larger than necessary, leading to wasted space within a memory segment. External fragmentation occurs when free memory blocks are scattered throughout memory, making it challenging to allocate contiguous blocks of memory to processes.

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

* Demand paging is a memory management technique that brings pages into memory only when they are needed. It improves memory management efficiency by reducing the amount of memory needed to store processes and allowing the system to handle larger processes.

1. **Explain the role of the page table in virtual memory management.**

* The page table is a data structure used by the operating system to map virtual addresses to physical addresses in virtual memory. It enables efficient memory management by allowing the system to allocate and access memory in smaller, manageable units called pages.

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

* The memory management unit (MMU) is a hardware component that translates virtual addresses generated by the CPU into physical addresses in memory. It provides memory protection, address translation, and caching functionality.

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

* Thrashing occurs when a system spends a significant amount of time swapping data between physical memory and disk storage due to excessive paging activity. It can be avoided by optimizing memory usage, adjusting system parameters, and implementing effective memory management strategies.

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

* A system call is a request made by a user program to the operating system kernel for a service or operation that requires privileged access. It facilitates communication between user programs and the operating system by providing a secure interface for accessing system resources.

1. **Describe the difference between a monolithic kernel and a microkernel.**

* A monolithic kernel incorporates all operating system functions, such as process management, memory management, and device drivers, into a single kernel image. A microkernel delegates most operating system functions to user-space processes, keeping only essential functions, such as inter-process communication and memory management, in the kernel.

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

* The operating system manages I/O operations by providing device drivers to communicate with hardware devices, implementing I/O scheduling algorithms to optimize performance, and providing interfaces for applications to access and control devices.

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

* A race condition occurs when the outcome of a system depends on the timing or sequence of events. It can be prevented by using synchronization techniques such as locks, semaphores, or mutexes to control access to shared resources and ensure consistency.

1. **Describe the role of device drivers in an operating system.**

* Device drivers are software components that allow the operating system to communicate with hardware devices such as printers, keyboards, and disk drives. They act as intermediaries between the hardware and the operating system, providing a standardized interface for the OS to control and access devices.

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

* A zombie process is a terminated process that still has an entry in the process table. It occurs when a parent process fails to call the **wait()** system call to retrieve the exit status of its child process. Zombie processes can be prevented by ensuring that parent processes properly handle the termination of their children.

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

* An orphan process is a process whose parent process has terminated or been killed unexpectedly. In Unix-like operating systems, orphan processes are adopted by the init process (usually with PID 1), which becomes their new parent process. The init process then waits for the orphan process to complete and cleans up its resources.

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

* In process management, a child process is created by a parent process. The parent process typically controls and supervises the execution of its child processes. Parent processes can communicate with their children, wait for them to finish execution, and receive status information about them.

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

* The **fork()** system call is used to create a new process in Unix-like operating systems. When **fork()** is called, the operating system creates a duplicate of the calling process, known as the child process. The child process inherits the memory space, file descriptors, and other attributes of the parent process.

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

* A parent process can wait for a child process to finish execution by calling the **wait()** or **waitpid()** system calls. These calls cause the parent process to block until the specified child process terminates. Once the child process exits, the parent process resumes execution.

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

* The exit status of a child process returned by the **wait()** system call indicates the outcome of the child process's execution. It typically includes information such as the exit code, termination status, and any error conditions encountered during execution.

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

* A parent process can terminate a child process by sending it a termination signal using the **kill()** system call with thew child process's PID. Alternatively, the parent process can call the **killpg()** system call to terminate all processes in a process group, including the child process.

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

* A process group is a collection of one or more processes that share the same process group ID (PGID) and can be controlled collectively by the operating system. A session is a collection of one or more process groups that share the same session ID (SID) and typically represent a user's login session.

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

* The **exec()** family of functions is used to replace the current process image with a new one in Unix-like operating systems. These functions load a new program into the current process's memory space, replacing the existing program code, data, and stack with those of the new program.

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

* The **waitpid()** system call is used to wait for the termination of a specific child process, whereas **wait()** waits for any child process to terminate. **waitpid()** provides more control over which child process to wait for by specifying its process ID and additional options.

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

* Process termination in Unix-like operating systems typically occurs when a process voluntarily calls the **exit()** system call to terminate its execution or when it receives a termination signal from the operating system or another process.

1. **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?**

* The long-term scheduler (also known as the admission scheduler) selects which processes from the job pool should be loaded into memory and brought into the ready queue for execution. It influences the degree of multiprogramming by controlling the number of processes in memory at any given time, balancing system throughput and responsiveness.

1. **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?**

* The short-term scheduler (also known as the CPU scheduler) selects which process from the ready queue should be executed next and has the highest frequency of execution. It makes decisions on a per-CPU basis and has a narrow scope, focusing on immediate scheduling decisions.

1. **Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.**
   * The medium-term scheduler is invoked when the system experiences memory pressure, such as when the available physical memory becomes low due to the presence of too many processes in memory. In such scenarios, the medium-term scheduler may decide to swap out some processes from memory to disk to free up memory resources. This helps manage system resources more efficiently by preventing thrashing and improving overall system performance.