OS INTERVIEW QUESTIONS

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

An OS is software that manages computer hardware and software resources, providing services like memory management, file handling, and process control.

2. Explain the difference between process and thread.

A process is an independent program in execution, while a thread is a smaller unit within a process that shares resources.

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

Virtual memory extends physical memory by using disk space, allowing programs to use more memory than what's available.

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

Multiprogramming runs multiple programs, multitasking switches between tasks, and multiprocessing uses multiple CPUs.

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

A file system manages how data is stored and retrieved; it includes directories, files, and metadata.

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

A deadlock is when processes can't proceed due to resource holding; it can be avoided by proper resource allocation.

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

The kernel is the core of the OS, managing resources; the shell is a command interface for users.

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

CPU scheduling decides the order of task execution, optimizing CPU use and system performance.

9. How does a system call work?

A system call allows a program to request services from the OS, like reading a file.

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

Device drivers act as translators between the OS and hardware devices.

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

The page table maps virtual addresses to physical memory addresses.

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

Thrashing happens when excessive paging reduces performance; it's avoided by managing memory use properly.

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

A semaphore controls access to shared resources, preventing conflicts in multi-threading.

14. How does an operating system handle process synchronization?

The OS uses locks, semaphores, and monitors to keep processes in sync.

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

Interrupts signal the CPU to handle urgent tasks, like hardware requests.

16. Explain the concept of a file descriptor.

A file descriptor is a number that represents an open file or I/O resource.

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

Recovery involves restarting services, restoring data, or rebooting the system.

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

Monolithic kernels have all services in one, while microkernels separate core functions.

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

Internal fragmentation wastes space inside blocks, while external leaves gaps between blocks.

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

It uses buffers, queues, and scheduling to handle input and output efficiently.

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

Preemptive allows interrupting tasks, while non-preemptive runs tasks to completion.

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

Round-robin assigns time slices to each process, cycling through them in order.

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

Processes are run based on priority levels, which can be set by the system or user.

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

SJN runs the shortest task first, used when short tasks frequently arrive.

25. Explain the concept of multilevel queue scheduling.

Multilevel queue scheduling divides tasks into different queues based on priority.

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

PCB stores process data like ID, state, and resources.

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

Processes move between states like ready, running, and waiting based on events.

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

Processes use IPC methods like pipes, sockets, or shared memory.

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

It ensures processes run in a coordinated way to avoid conflicts.

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

A zombie process is a completed process that still has an entry in the process table.

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

Internal happens inside allocated memory, external occurs between allocations.

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

Demand paging loads pages only when needed, saving memory.

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

The page table keeps track of virtual to physical memory mapping.

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

The MMU handles virtual address translation to physical addresses.

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

Thrashing is excessive paging; avoid it by tuning memory and processes.

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

System calls let programs request OS services, like file access.

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

Monolithic kernels have all services in one place; microkernels separate core parts.

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

The OS uses I/O scheduling, buffering, and interrupts to manage devices.

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

A race condition is a timing error; it's prevented using synchronization tools.

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

Device drivers connect the OS to hardware devices, translating commands.

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

A zombie process is a terminated process not cleaned up by the parent; it can be avoided by handling process termination properly.

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

An orphan process has no parent; the OS reassigns it to a system process like `init`.

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

A parent process creates and controls a child process.

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

`fork()` creates a new process by duplicating the calling process.

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

The parent uses `wait()` or `waitpid()` to pause until the child finishes.

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

The exit status tells the parent how the child process ended.

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

The parent can use `kill` with the child's PID to terminate it.

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

A process group manages related processes; a session groups related process groups.

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

`exec()` replaces the current process code with a new program.

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

`waitpid()` waits for specific processes, unlike `wait()` which waits for any child.

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

A process ends when it calls `exit()`, is killed, or finishes naturally.

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?

The long-term scheduler controls how many processes enter the system, managing load.

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?

The short-term scheduler runs most often, deciding which process runs next.

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

It swaps out processes to free up memory, improving performance when resources are tight.