OPERATING SYSTEM

Interview Question & Answers

PART-1

1.What is an operating system?

Ans: An operating system is a program that acts as an intermediary between the user and the computer hardware. The purpose of an OS is to provide a convenient environment in which user can execute programs in a convenient and efficient manner.It is a resource allocator responsible for allocating system resources and a control program which controls the operation of the computer h/w.

2.What are the various components of a computer system?

Ans:

1. The hardware
2. The operating system
3. The application programs
4. The users.

3.What is purpose of different operating systems?

Ans:The machine Purpose Workstation individual usability &Resources utilization Mainframe Optimize utilization of hardware PC Support complex games, business application Hand held PCs Easy interface & min. power consumption

4.What are the different operating systems?

Ans:

1. Batched operating systems
2. Multi-programmed operating systems
3. timesharing operating systems
4. Distributed operating systems
5. Real-time operating systems

6.What is a boot-strap program

Ans: A bootstrap program, or bootloader, is a small program that initializes the hardware and loads the operating system into memory when the computer is powered on. It prepares the system for operation by performing hardware checks and loading the main OS from storage

7.What is BIOS?

Ans: BIOS (Basic Input/Output System) is firmware stored on a motherboard that initializes and tests the hardware components of a computer during the boot process. It also provides a runtime environment for the operating system and manages data flow between the OS and connected devices.

8. Explain the concept of batched operating systems?

In batched operating systems, users give their jobs to the operator who sorts the programs according to their requirements and executes them. This is time-consuming but keeps the CPU busy all the time.

9. Explain the concept of multi-programmed operating systems?

A multi-programmed operating system can execute a number of programs concurrently. The operating system fetches a group of programs from the job pool in the secondary storage, which contains all the programs to be executed, and places them in the main memory. This process is called job scheduling. Then it chooses a program from the ready queue and gives it to the CPU to execute. When an executing program needs some I/O operation, the operating system fetches another program and hands it to the CPU for execution, thus keeping the CPU busy all the time.

10. Explain the concept of time-sharing operating systems?

Time-sharing operating systems are a logical extension of multi-programmed OS where users can interact with the program. The CPU executes multiple jobs by switching among them, but the switches occur so frequently that the user feels as if the operating system is running only their program.

11. Explain the concept of multi-processor systems or parallel systems?

Multi-processor systems contain multiple processors to increase execution speed, reliability, and economy. They are of two types:

1. Symmetric multiprocessing
2. Asymmetric multiprocessing
   In symmetric multiprocessing, each processor runs an identical copy of the OS, and these copies communicate with each other as needed. In asymmetric multiprocessing, each processor is assigned a specific task.

12. Explain the concept of distributed systems?
Distributed systems operate in a network and can share network resources and communicate with each other.

13. Explain the concept of real-time operating systems?
A real-time operating system is used when rigid time requirements are placed on the operation of a processor or the flow of data; thus, it is often used as a control device in dedicated applications. Here, sensors bring data to the computer. The computer must analyze the data and possibly adjust controls to modify the sensor input. They are of two types:

1. Hard real-time OS
2. Soft real-time OS
   Hard real-time OS has well-defined fixed time constraints, while soft real-time operating systems have less stringent timing constraints.

14. Define MULTICS.
MULTICS (Multiplexed Information and Computing Services) operating system was developed from 1965 to 1970 at the Massachusetts Institute of Technology as a computing utility. Many of the ideas used in MULTICS were subsequently adopted in UNIX.

15. What is SCSI?
SCSI stands for Small Computer Systems Interface.

16. What is a sector?
A sector is the smallest addressable portion of a disk.

17. What is cache-coherency?
In a multiprocessor system, there exist several caches, each of which may contain a copy of the same variable A. When a change occurs in one cache, it should immediately be reflected in all other caches. This process of maintaining the same value of data in all caches is called cache-coherency.

18. What are residence monitors?
Early operating systems were called residence monitors.

19. What is dual-mode operation?

In order to protect the operating systems and system programs from malfunctioning programs, two mode operations were evolved:

1. System mode.
2. User mode.
   In this setup, user programs cannot directly interact with system resources; instead, they request the operating system, which checks the request and performs the required tasks for the user programs. DOS was written for Intel 8088 and does not have dual-mode operation, while Pentium provides dual-mode operation.

20. What are the operating system components?

1. Process management
2. Main memory management
3. File management
4. I/O system management
5. Secondary storage management
6. Networking
7. Protection system
8. Command interpreter system

21. What are operating system services?

1. Program execution
2. I/O operations
3. File system manipulation
4. Communication
5. Error detection
6. Resource allocation
7. Accounting
8. Protection

22. What are system calls?

System calls provide the interface between a process and the operating system. System calls for modern Microsoft Windows platforms are part of the Win32 API, which is available for all compilers written for Microsoft Windows.

23. What is a layered approach and what is its advantage?

The layered approach is a step towards modularizing the system, in which the operating system is broken up into a number of layers (or levels), each built on top of the lower layer. The bottom layer is the hardware, and the topmost is the user interface. The main advantage of the layered approach is modularity. The layers are selected such that each uses the functions (operations) and services of only the lower layer. This approach simplifies debugging and system verification.

24. What is the microkernel approach and cite its advantages?

The microkernel approach is a step towards modularizing the operating system, where all non-essential components from the kernel are removed and implemented as system and user-level programs, making the kernel smaller. The benefits of the microkernel approach include the ease of extending the operating system. All new services are added to the user space and consequently do not require modification of the kernel. As the kernel is smaller, it is easier to upgrade it. This approach also provides more security and reliability since most services run as user processes rather than in the kernel, keeping the kernel intact.

25. What are virtual machines and cite their advantages?

Virtual machines are a concept by which an operating system can create the illusion that a process has its own processor with its own (virtual) memory. The operating system implements the virtual machine concept by using CPU scheduling and virtual memory.

1. The basic advantage is that it provides a robust level of security as each virtual machine is isolated from all other VMs, thus completely protecting system resources.
2. Another advantage is that system development can be done without disrupting normal operation. System programmers are given their own virtual machine, and system development occurs on the virtual machine instead of the actual physical machine.
3. Additionally, virtual machines solve compatibility problems. For example, Java supplied by Sun Microsystems provides a specification for the Java Virtual Machine.

26. What is a process?

A process is a program in execution, or it may also be called a unit of work. A process requires some system resources such as CPU time, memory, files, and I/O devices to accomplish the task. Each process is represented in the operating system by a Process Control Block (PCB) or Task Control Block. Processes are of two types:

1. Operating system processes
2. User processes

27. What are the states of a process?

1. New
2. Running
3. Waiting
4. Ready
5. Terminated

28. What are various scheduling queues?

1. Job queue
2. Ready queue
3. Device queue

29. What is a job queue?

When a process enters the system, it is placed in the job queue.

30. What is a ready queue?

The processes that are residing in the main memory and are ready and waiting to execute are kept on a list called the ready queue.

31. What is a device queue?

A list of processes waiting for a particular I/O device is called a device queue.

32. What is a long-term scheduler & short-term scheduler?

Long-term schedulers are the job schedulers that select processes from the job queue and load them into memory for execution. The short-term schedulers are the CPU schedulers that select a process from the ready queue and allocate the CPU to one of them

.

33. What is context switching?

Transferring control from one process to another requires saving the state of the old process and loading the saved state for the new process. This task is known as context switching.

34. What are the disadvantages of context switching?

The time taken for switching from one process to another is pure overhead since the system does no useful work while switching. One solution is to use threading whenever possible.

35. What are cooperating processes?

Cooperating processes share system resources as data among each other. They can communicate with each other via interprocess communication facilities, which are generally used in distributed systems. A good example is a chat program used on the web.

36. What is a thread?

A thread is a program line under execution. Sometimes called a light-weight process, it is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and a stack.

37. What are the benefits of multithreaded programming?

1. Responsiveness (needn't wait for a lengthy process)
2. Resource sharing
3. Economy (context switching between threads is easy)
4. Utilization of multiprocessor architectures (perfect utilization of the multiple processors).

38. What are the types of threads?

1. User thread
2. Kernel thread

User threads are easy to create and use, but the disadvantage is that if they perform a blocking system call, the kernel is engaged completely with the single user thread, blocking other processes. They are created in user space. Kernel threads are supported directly by the operating system. They are slower to create and manage. Most operating systems like Windows NT, Windows 2000, Solaris 2, BeOS, and Tru64 Unix support kernel threading.

39. Which category do Java threads fall into?

Java threads are created and managed by the Java Virtual Machine. They do not easily fall under the category of either user or kernel threads.

40. What are multithreading models?

Many operating systems provide both kernel threading and user threading, referred to as multithreading models. They are of three types:

1. Many-to-one model (many user-level threads and one kernel thread).
2. One-to-one model
3. Many-to-many model

In the first model, only one user can access the kernel thread, preventing multi-processing (e.g., Green threads of Solaris). The second model allows multiple threads to run on parallel processing systems; creating a user thread requires creating a corresponding kernel thread (disadvantage) (e.g., Windows NT, Windows 2000, OS/2). The third model allows the user to create as many threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor (e.g., Solaris 2, IRIX, HP-UX, and Tru64 Unix).

41. What is a P-thread?

P-thread refers to the POSIX standard (IEEE 1003.1c) defining an API for thread creation and synchronization. This is a specification for thread behavior, not an implementation. The Windows operating system has generally not supported P-threads.

42. What are Java threads?

Java is one of the few languages that supports thread creation and management at the language level. However, because threads are managed by the Java Virtual Machine (JVM), not by a user-level library or kernel, it is difficult to classify Java threads as either user-level or kernel-level.

43. What is process synchronization?

A situation where several processes access and manipulate the same data concurrently, and the outcome of the execution depends on the particular order in which the access takes place, is called a race condition. To guard against race conditions, we need to ensure that only one process at a time can manipulate the same data. The technique we use for this is called process synchronization.

44. What is the critical section problem?

The critical section is the code segment of a process in which the process may be changing common variables, updating tables, writing a file, and so on. Only one process is allowed to enter the critical section at any given time (mutually exclusive). The critical section problem is to design a protocol that processes can use to cooperate. The three basic requirements of a critical section are:

1. Mutual exclusion
2. Progress
3. Bounded waiting

The Bakery algorithm is one of the solutions to the critical section problem.

45. What is a semaphore?

A semaphore is a synchronization tool used to solve complex critical section problems. It is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: Wait and Signal.

46. What is the bounded-buffer problem?

In the bounded-buffer problem, we assume that a pool consists of n buffers, each capable of holding one item. A semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 1. The empty and full semaphores count the number of empty and full buffers, respectively. The empty semaphore is initialized to n, and the full semaphore is initialized to 0.

47. What is the readers-writers problem?

In the readers-writers problem, we divide the processes into two types:

1. Readers (who want to retrieve data only)
2. Writers (who want to retrieve as well as manipulate data)

We can allow multiple readers to read the same data at the same time. However, a writer must be exclusively allowed to access the data. There are two solutions to this problem:

1. No reader will be kept waiting unless a writer has already obtained permission to use the shared object. In other words, no reader should wait for other readers to complete simply because a writer is waiting.
2. Once a writer is ready, that writer performs its write as soon as possible. This means that if a writer is waiting to access the object, no new readers may start reading.

48. What is the dining philosophers' problem?

The dining philosophers' problem is a classic synchronization problem that illustrates the challenges of resource sharing among concurrent processes. It involves five philosophers who spend their lives thinking and eating, sitting at a common circular table with five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and there are five chopsticks (one for each philosopher).

When a philosopher is thinking, she does not interact with her colleagues. When she becomes hungry, she attempts to pick up the two chopsticks closest to her. However, a philosopher can only pick up one chopstick at a time, and she cannot take a chopstick that is already in someone else's hand. Once a philosopher has both chopsticks, she eats without putting them down. After finishing her meal, she puts down both chopsticks and resumes thinking.

49. What is a deadlock?

A deadlock is a situation in which a process requests resources that are not available, leading the process to enter a wait state. A waiting process may never change state again because the resources it has requested are held by other waiting processes. This situation creates a cycle of dependencies that prevents any of the involved processes from making progress.

50. What are the necessary conditions for deadlock?

The necessary conditions for deadlock are:

1. Mutual exclusion: At least one resource must be non-shareable, meaning only one process can use it at a time.
2. Hold and wait: A process holds at least one resource and is waiting to acquire additional resources.
3. No preemption: Resources cannot be forcibly taken from a process; they must be voluntarily released.
4. Circular wait: There is a circular chain of processes, where each process is waiting for a resource held by the next process in the chain. For example, if process p[i]p[i]p[i] is waiting for resource p[j]p[j]p[j], then iii can be 1 to nnn and jjj can be defined as i+1i+1i+1 if i≠ni \neq ni =n or 1 if i=ni = ni=n.

51. What is a resource allocation graph?

A resource allocation graph is a graphical representation of the state of resource allocation in a system, which can help illustrate deadlocks. The graph consists of a set of vertices VVV and edges EEE. The vertices VVV are divided into two types:

- Processes: P={p1,p2,...,pn}P = \{p_1, p_2, \ldots, p_n\}P={p1,p2,...,pn}
- Resources: R={r1,r2,...,rn}R = \{r_1, r_2, \ldots, r_n\}R={r1,r2,...,rn}

A directed edge Pi→RjP_i \rightarrow R_jPi→Rj represents a request for a resource (request edge), while a directed edge Rj→PiR_j \rightarrow P_iRj→Pi indicates that a resource has been allocated to a process (assignment edge). In the graphical representation, processes are depicted as circles and resources as squares. Each instance of a resource is represented as a dot within the square. When a request is fulfilled, the request edge is converted to an assignment edge. When a process releases a resource, the assignment edge is deleted. If a cycle exists in the graph that involves resource types with a single instance, it indicates that a deadlock has occurred, and each process involved in the cycle is considered to be in a deadlock state.

52. What are deadlock prevention techniques?

Deadlock prevention techniques are strategies used to ensure that at least one of the necessary conditions for deadlock cannot hold. The techniques include:

1. Mutual exclusion: Some resources, like read-only files, should be sharable and not mutually exclusive. However, resources that require exclusive access, such as printers, should maintain mutual exclusion.
2. Hold and wait: To prevent this condition, ensure that a process requesting a resource should not hold any other resources. This can be enforced by requiring processes to request all required resources at once.
3. No preemption: If a process holds some resources and requests another resource that cannot be immediately allocated, then all currently held resources should be preempted (released autonomously) to prevent deadlock.
4. Circular wait: To prevent circular wait conditions, impose a total ordering of all resource types and require that each process requests resources in an increasing order of enumeration.

53. What is a safe state and a safe sequence?

A system is in a safe state if there exists a safe sequence of processes that can execute without leading to a deadlock. A sequence of processes is considered safe if, for each process PiP_iPi in the sequence, the resources that PiP_iPi can still request can be satisfied by the currently available resources plus the resources held by all other processes PjP_jPj (where j≠ij \neq ij =i). This ensures that all processes can eventually complete without leading to a deadlock.

Thanks for reading!

Follow for more interview insights and tech tips.