OPERATING SYSTEM

Interview Question & Answers

PART-2

54. What are the deadlock avoidance algorithms?

Deadlock avoidance algorithms dynamically examine the resource allocation state to ensure that a circular wait condition can never exist. The resource allocation state is characterized by the number of available and allocated resources, as well as the maximum demand of each process. The two main algorithms for deadlock avoidance are:

1. Resource allocation graph algorithm: This algorithm uses a graph to represent the allocation of resources and can identify potential deadlocks by analyzing the states of requests and allocations.
2. Banker's algorithm: This algorithm consists of two components:
    - Safety algorithm: Determines if a system is in a safe state by finding a safe sequence of processes.
    - Resource request algorithm: Checks if a resource request can be granted without leading the system into an unsafe state.

55. What are the basic functions of an operating system?

The operating system (OS) controls and coordinates the use of hardware among various application programs. It acts as a resource allocator and manager, deciding how to allocate resources among conflicting requests to operate the computer system efficiently and fairly. Additionally, the OS serves as a control program that manages user programs to prevent errors and improper use of the computer, with a particular focus on the operation and control of I/O devices.

56. Explain briefly about processor, assembler, compiler, loader, linker, and the functions executed by them.

- Processor: The processor, also known as the Central Processing Unit (CPU), is the part of a computer system that executes instructions.
- Assembler: An assembler is a program that translates basic computer instructions into a pattern of bits (machine code) that the processor can use for its operations. The instructions written in assembler language (or assembly language) are converted into machine code by the assembler.
- Compiler: A compiler is a specialized program that processes statements written in a programming language (such as Pascal or C) and translates them into machine language (or code) that the processor can understand. Programmers write source statements using an editor, and then the compiler is run to create an executable file.
- Loader: In an operating system, a loader is a component responsible for locating a given program (whether an application or part of the OS) in offline storage (like a hard disk), loading it into main storage (RAM), and transferring control to that program.
- Linker: A linker combines libraries with object code to produce executable machine code. It resolves references between different pieces of code and ensures that they can work together correctly.

57. What is a Real-Time System?
A real-time system is one that must respond to events within a specified time period. Real-time processes are critical because they require timely and deterministic responses. A real-time operating system is designed to handle such processes successfully, ensuring that they meet their timing constraints and operate reliably in environments where timing is crucial.

58. What is the difference between Hard and Soft real-time systems?
A hard real-time system guarantees that critical tasks complete on time, meaning that all delays in the system must be bounded, from data retrieval to the operating system's response to requests. If a deadline is missed, the consequences can be catastrophic. In contrast, a soft real-time system allows for some flexibility, where critical real-time tasks are prioritized over others, but if they do not complete on time, it may not have severe consequences. However, delays should still be managed, as in hard real-time systems.

59. What is virtual memory?
Virtual memory is a hardware technique that allows a system to appear as if it has more memory than it physically does. It achieves this by using a combination of physical memory (RAM) and disk storage, effectively time-sharing between them. When data in physical memory is not actively being used, it can be temporarily moved to disk storage, freeing up RAM for other tasks.

60. What is cache memory?
Cache memory is a type of random access memory (RAM) that the computer's microprocessor can access more quickly than regular RAM. It stores frequently accessed data and instructions, allowing the microprocessor to check the cache first before retrieving data from slower main memory. If the required data is found in the cache (a cache hit), it significantly speeds up processing.

61. Differentiate between Compiler and Interpreter.
An interpreter reads and executes one instruction at a time, carrying out the actions implied by that instruction without performing any translation of the entire program. In contrast, a compiler translates the entire set of instructions or source code into machine code before execution, producing an executable file that can be run independently of the original source code.

62. What are different tasks of Lexical Analysis?

The primary purpose of the lexical analyzer (or lexer) is to process the input text and deliver a sequence of tokens. This involves partitioning the input into comments (character sequences to be ignored) and basic symbols (character sequences corresponding to terminal symbols of the grammar that defines the structure of the input). The lexer removes unnecessary information, simplifying the input for further analysis.

63. Why is paging used?

Paging is used to address the issue of external fragmentation by allowing the logical address space of a process to be noncontiguous. This means that a process can be allocated physical memory wherever it is available, rather than requiring a contiguous block of memory. Paging divides the process's memory into fixed-size pages, which can be loaded into any available frame in physical memory, making memory management more efficient.

64. What is Context Switch?

A context switch is the process of switching the CPU from one process to another. This involves saving the state (context) of the currently running process and loading the saved state of the next process to be executed. Context switch time is considered overhead because the system does not perform any useful work during the switch. The time it takes for a context switch can vary based on several factors, including memory speed, the number of registers that need to be copied, and whether the system has special instructions for loading or storing all registers efficiently.

65. What are Distributed Systems?

Distributed systems involve distributing computation across multiple physical processors. In a loosely coupled system, each processor has its own local memory, and they communicate with each other through various communication lines, such as high-speed buses or telephone lines.

Advantages of distributed systems include:

- Resource Sharing: Multiple processors can share resources, such as data and applications.
- Computation Speed-Up: Load sharing among processors can increase overall computation speed.
- Reliability: The failure of one processor does not necessarily halt the entire system, enhancing reliability.
- Communications: Improved communication capabilities among different system components.

66. What is the difference between Primary Storage and Secondary Storage?

- Primary Storage (Main Memory): This refers to the only large storage media that the CPU can access directly. It is volatile, meaning that data is lost when the power is turned off.
- Secondary Storage: This serves as an extension of primary storage, providing large non-volatile storage capacity. It retains data even when the power is off and includes devices such as hard drives, SSDs, and optical discs.

67. What is CPU Scheduler?

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
  a. Switches from running to waiting state.
  b. Switches from running to ready state.
  c. Switches from waiting to ready.
  d. Terminates.
- Scheduling under 1 and 4 is nonpreemptive.
- All other scheduling is preemptive.

68. What do you mean by deadlock?

Deadlock is a situation where a group of processes are all blocked and none of them can become unblocked until one of the others becomes unblocked. The simplest deadlock involves two processes, each of which is waiting for a message from the other.

69. What is Dispatcher?

- The dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the user program to restart that program
- Dispatch latency is the time it takes for the dispatcher to stop one process and start another running.

70. What is Throughput, Turnaround time, waiting time, and Response time?

- Throughput: Number of processes that complete their execution per time unit.
- Turnaround time: Amount of time to execute a particular process.
- Waiting time: Amount of time a process has been waiting in the ready queue.
- Response time: Amount of time it takes from when a request is submitted until the first response is produced, not output (for a time-sharing environment).

71. Explain the difference between microkernel and macro kernel?

Micro-Kernel: A micro-kernel is a minimal operating system that performs only the essential functions of an operating system. All other operating system functions are performed by system processes.

Monolithic: A monolithic operating system is one where all operating system code is in a single executable image, and all operating system code runs in system mode.

72. What is multitasking, multiprogramming, multithreading?

Multiprogramming: Multiprogramming is the technique of running several programs at a time using time-sharing. It allows a computer to do several things at the same time and creates logical parallelism. The operating system keeps several jobs in memory simultaneously and selects a job from the job pool to execute. When that job needs to wait for any I/O operations, the CPU switches to another job, ensuring that the CPU is never idle.

Multitasking: Multitasking is the logical extension of multiprogramming. The concept of multitasking is similar to multiprogramming, but the difference is that the switching between jobs occurs so frequently that users can interact with each program while it is running. This is also known as time-sharing systems. A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of the system's time.

Multithreading: An application is typically implemented as a separate process with several threads of control. In some situations, a single application may need to perform several similar tasks. For example, a web server accepts client requests for web pages, images, sounds, and so forth. If the web server ran as a traditional single-threaded process, it could service only one client at a time, resulting in long wait times for clients. Instead, it is efficient to have one process with multiple threads to serve the same purpose. This approach allows the web server to create a separate thread for each client request, enhancing responsiveness, resource sharing, economy, and utilization of multiprocessor architectures.

73. Give a non-computer example of preemptive and non-preemptive scheduling?
Consider any system where people use some kind of resources and compete for them. The non-computer example for preemptive scheduling is traffic on a single-lane road; if there is an emergency or an ambulance, the other vehicles give way to the vehicle that is in need. The example for non-preemptive scheduling is people standing in a queue for tickets.

74. What is starvation and aging?
Starvation: Starvation is a resource management problem where a process does not get the resources it needs for a long time because the resources are being allocated to other processes.
Aging: Aging is a technique to avoid starvation in a scheduling system. It works by adding an aging factor to the priority of each request. The aging factor must increase the request's priority as time passes and ensure that a request will eventually become the highest priority request after it has waited long enough.

75. Different types of Real-Time Scheduling?
Hard real-time systems: Required to complete a critical task within a guaranteed amount of time.
Soft real-time computing: Requires that critical processes receive priority over less fortunate ones.

76. What are the Methods for Handling Deadlocks?

- Ensure that the system will never enter a deadlock state.
- Allow the system to enter a deadlock state and then recover.
- Ignore the problem and pretend that deadlocks never occur in the system; this approach is used by most operating systems, including UNIX.

77. What is a Safe State and its use in deadlock avoidance?
When a process requests an available resource, the system must decide if the immediate allocation leaves the system in a safe state. The system is in a safe state if there exists a safe sequence of all processes. A sequence is safe if, for each process PiP_iPi, the resources that PiP_iPi can still request can be satisfied by currently available resources plus the resources held by all other processes PjP_jPj. If PiP_iPi's resource needs are not immediately available, then PiP_iPi can wait until all PjP_jPj have finished. When PjP_jPj is finished, PiP_iPi can obtain the needed resources, execute, return allocated resources, and terminate. When PiP_iPi terminates, Pi+1P_{i+1}Pi+1 can obtain its needed resources, and so on.
Deadlock Avoidance: Ensures that a system will never enter an unsafe state.

78. Recovery from Deadlock?

Process Termination:

- Abort all deadlocked processes.
- Abort one process at a time until the deadlock cycle is eliminated.
- In which order should we choose to abort?
    - Priority of the process.
    - How long the process has computed, and how much longer to completion.
    - Resources the process has used.
    - Resources the process needs to complete.
    - How many processes will need to be terminated?
    - Is the process interactive or batch?

Resource Preemption:

- Selecting a victim – minimize cost.
- Rollback – return to some safe state, restart the process from that state.
- Starvation – the same process may always be picked as a victim; include the number of rollbacks in the cost factor.

79. Difference between Logical and Physical Address Space?

- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.
- Logical address: Generated by the CPU; also referred to as a virtual address.
- Physical address: Address seen by the memory unit.
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding schemes.

80. Binding of Instructions and Data to Memory?

Address binding of instructions and data to memory addresses can happen at three different stages:

- Compile time: If memory location is known a priori, absolute code can be generated; must recompile code if the starting location changes.
- Load time: Must generate relocatable code if the memory location is not known at compile time.
- Execution time: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Needs hardware support for address maps (e.g., base and limit registers).

81. What is Memory-Management Unit (MMU)?
A hardware device that maps virtual to physical addresses. In the MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.

- The user program deals with logical addresses; it never sees the real physical addresses.

82. What are Dynamic Loading, Dynamic Linking, and Overlays?
Dynamic Loading:

- A routine is not loaded until it is called.
- Better memory-space utilization; unused routines are never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required; implemented through program design.

Dynamic Linking:

- Linking is postponed until execution time.
- A small piece of code, called a stub, is used to locate the appropriate memory-resident library routine.
- The stub replaces itself with the address of the routine and executes the routine.
- The operating system needs to check if the routine is in the process's memory address.
- Dynamic linking is particularly useful for libraries.

Overlays:

- Keep in memory only those instructions and data that are needed at any given time.
- Necessary when a process is larger than the amount of memory allocated to it.
- Implemented by the user; no special support is needed from the operating system. The programming design of the overlay structure is complex.

83. What is fragmentation? Different types of fragmentation?
Fragmentation occurs in a dynamic memory allocation system when many of the free blocks are too small to satisfy any request.

External Fragmentation:

- Happens when a dynamic memory allocation algorithm allocates some memory, and a small piece is left over that cannot be effectively used.
- If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space exists to satisfy a request, but it is not contiguous.

Internal Fragmentation:

- The space wasted inside of allocated memory blocks because of restrictions on the allowed sizes of allocated blocks.
- Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition but not being used.

To reduce external fragmentation:

- Compaction: Shuffle memory contents to place all free memory together in one large block. Compaction is possible only if relocation is dynamic and is done at execution time

.

84. Define Demand Paging, Page fault interrupt, and Trashing?

Demand Paging:

- A paging policy where a page is not read into memory until it is requested, that is, until there is a page fault on the page.

Page Fault Interrupt:

- Occurs when a memory reference is made to a page that is not in memory. The present bit in the page table entry will be found to be off by the virtual memory hardware, signaling an interrupt.

Trashing:

- The problem of many page faults occurring in a short time, referred to as “page thrashing.”

85. Explain Segmentation with Paging.

Segments can be of different lengths, making it harder to find a place for a segment in memory than for a page. With segmented virtual memory, we gain the benefits of virtual memory, but we still need to perform dynamic storage allocation of physical memory. To avoid this complexity, segmentation and paging can be combined into a two-level virtual memory system. In this system, each segment descriptor points to a page table for that segment. This combination provides some advantages of paging (such as easier placement) alongside some benefits of segmentation (like the logical division of the program

86. Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.
A page fault occurs when there is an attempt to access a page that has not yet been brought into main memory. When this happens, the operating system first verifies the memory access, aborting the program if the access is invalid. If the access is valid, the operating system then locates a free frame and requests I/O to read the needed page into that free frame. Upon the completion of the I/O operation, the process table and page table are updated, and the instruction that caused the page fault is restarted.

87. What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?
Thrashing is caused by the under-allocation of the minimum number of pages required by a process, resulting in continuous page faults. The system can detect thrashing by evaluating the level of CPU utilization relative to the level of multiprogramming. If CPU utilization is low while multiprogramming is high, thrashing is likely occurring. To eliminate this issue, the system can reduce the level of multiprogramming by limiting the number of processes that are allowed to run concurrently.

Thanks for reading!

Follow for more interview insights and tech tips.