

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

|  |
| --- |
| **Course Details** |
| **Class:** II B. Tech **Semester:** II **Academic Year:** 2024-25  **Course Title:** OPERATING SYSTEMS  **Course Code:**  **Credits:** 3 **Regulation:** NECR BTECH 23  **Program/Dept.:** UG / CSE  **Faculty:** G. Radhika Deepthi / Dr. B. Sunil Kumar / E. Ramesh Reddy / M. Subhashini |

**2 MARKS QUESTION BANK**

**UNIT – 1: Operating Systems Overview, System Structures**

|  |
| --- |
| **1**.**Define an operating system.** |
| An operating system (OS) is system software that acts as an interface between hardware and users. It manages hardware resources and provides essential services to applications. Examples Windows, Linux, and macOS. |
| **2.List two primary functions of an operating system.** |
| The OS manages resources like the CPU, memory, and I/O devices and ensures efficient utilization. It also provides a user-friendly interface and a platform for application execution. |
| **3.What are the two main types of operating system operations?** |
| OS operations include single-tasking or multi-tasking, where tasks are executed one at a time or simultaneously. Another key operation is mode switching between user-mode and kernel-mode for secure execution. |
| **4. Give examples of two computing environments where operating systems are used.** |
| Operating systems are used in desktop environments (e.g., Windows, macOS) for personal productivity and cloud environments (e.g., Linux on servers) for hosting web applications and services. |
| **5. Name any two services provided by an operating system.** |
| The OS provides program execution services, allowing programs to run efficiently. It also offers file management, enabling the creation, deletion, and access of files. |
| **6. What is the purpose of system calls in an operating system?** |
| System calls provide a controlled interface for programs to request services from the OS, such as accessing files, managing processes, or communicating with devices. |
| **7. Differentiate between user interface and system interface.** |
| The user interface allows interaction between the user and the OS (e.g., GUI or CLI). The system interface connects application programs to OS services, typically via system calls. |
| **8. What is the role of system programs in an operating system?** |
| System programs provide utilities for system management, such as file editors, disk management tools, and debugging aids, ensuring smooth operation and user convenience. |
| **9.Define operating system structure.** |
| Operating system structure refers to the organization of its components, such as monolithic, layered, or microkernel designs, determining how the OS functions and interacts with hardware and software. |
| **10.What is the purpose of booting an operating system?** |
| Booting initializes hardware components and loads the OS into memory, allowing the system to transition from a powered-off state to operational mode. |

**UNIT- 2: Processes, Threads and Concurrency, Scheduling**

|  |
| --- |
| **1.what is a process, and how is it different from a program?** |
| A process is an executing instance of a program, which includes its code, data, and execution context. A program is a static set of instructions, while a process is dynamic and actively running. A program can be considered a set of instructions, while a process is the program in action. |
| **2.List the main states of a process.** |
| The main states of a process are:  **New**: The process is being created.  **Ready**: The process is waiting for the CPU to execute.  **Running**: The process is actively being executed on the CPU.  **Waiting**: The process is waiting for some event (e.g., I/O).  **Terminated**: The process has finished execution. |
| **3.What are the three types of schedulers in an operating system?** |
| **Long-term scheduler**: Decides which processes are admitted to the ready queue.  **Short-term scheduler**: Allocates the CPU to processes in the ready queue.  **Medium-term scheduler**: Manages processes swapped in and out of memory to optimize CPU usage. |
| **4.Differentiate between preemptive and non-preemptive scheduling.** |
| **Preemptive scheduling** allows the CPU to be taken from a running process before it finishes.  **Non-preemptive scheduling** ensures a process runs to completion once it starts. Preemptive scheduling provides better responsiveness but causes more context switching. |
| **5.What happens when a process is terminated?** |
| When a process terminates, the operating system frees the resources allocated to it, including memory and file handles. The process is removed from the process table, and its exit status is recorded. It may enter the "zombie" state briefly for cleanup before complete removal. |
| **6. What is the difference between a parent process and a child process?** |
| A **parent process** creates one or more **child processes**. The parent process typically manages and monitors the child. A child process inherits some resources and properties from the parent process, but they are independent and can execute concurrently. |
| **7. What is the purpose of inter-process communication (IPC)?** |
| IPC allows processes to communicate and share data, which is essential for process synchronization, coordination, and resource sharing. It is especially important for multi-process applications, enabling them to work together efficiently. |
| **8.Differentiate between shared memory and message passing in IPC.** |
| **Shared memory**: Multiple processes access the same memory region to exchange data, providing fast communication.  **Message passing**: Processes communicate by sending messages through a communication channel, often used when processes are on different machines or when shared memory is not feasible. |
| **9. Define a thread and explain its importance.** |
| A thread is the smallest unit of execution within a process, sharing the same memory space as other threads in the same process. Threads are important because they allow for concurrent execution, making programs more efficient by utilizing multi-core processors. |
| **10. Explain the many-to-one threading model.** |
| In the **many-to-one** model, multiple user-level threads are mapped to a single kernel thread. This model is efficient but limits parallelism, as only one thread can execute at a time. It is typically used in systems where concurrency isn't a major requirement. |

**UNIT-3: Synchronization Tools, Deadlocks**

|  |
| --- |
| **1.What are the three requirements of a solution to the critical section problem?** |
| A solution must ensure mutual exclusion (only one process in the critical section), progress (processes can proceed without unnecessary delays), and bounded waiting (a process won't wait indefinitely). |
| **2. How does Peterson’s solution ensure mutual exclusion and progress?** |
| Peterson's solution uses two variables (flag and turn) to ensure only one process enters the critical section at a time while maintaining fairness and progress by alternating access. |
| **3. What is the role of a mutex lock in process synchronization?** |
| A mutex lock enforces mutual exclusion by allowing only one process to acquire the lock at a time, ensuring no two processes access shared resources concurrently. |
| **4. Differentiate between binary semaphores and counting semaphores.** |
| Binary semaphores (0 or 1) control access to a single resource, while counting semaphores manage multiple instances of a resource by using a counter. |
| **5. How do monitors help in process synchronization?** |
| Monitors encapsulate shared data and synchronization methods in one structure, ensuring only one process can execute a monitor procedure at a time, simplifying synchronization. |
| **6. What is the Dining Philosophers Problem, and why is it significant?** |
| The Dining Philosophers Problem models resource sharing and synchronization challenges, highlighting potential deadlocks and starvation when multiple processes compete for limited resources. |
| **7. How is the Producer-Consumer problem solved using semaphores?** |
| Semaphores synchronize producer and consumer processes by using a mutex for mutual exclusion and two semaphores to track buffer space and available items. |
| **8. Define a deadlock and explain its four necessary conditions.** |
| A deadlock is a state where processes cannot proceed due to circular resource dependency. It requires mutual exclusion, hold-and-wait, no preemption, and circular wait. |
| **9. What is the difference between deadlock avoidance and deadlock prevention?** |
| Deadlock avoidance dynamically checks system states to avoid unsafe states, while prevention alters system rules to ensure at least one deadlock condition is never met. |
| **10. Explain the steps involved in detecting a deadlock in a system.** |
| Deadlock detection involves examining resource-allocation tables, simulating resource releases, and identifying if any processes are indefinitely blocked. |

**UNIT-4: Memory Management Strategies, Virtual Memory**

**Management, Storage Management**

|  |
| --- |
| **1.What is contiguous memory allocation, and how does it manage memory?** |
| Contiguous memory allocation assigns a single continuous block of memory to each process, simplifying address calculation but causing fragmentation over time. |
| **2.Explain the difference between logical and physical addresses in memory management.** |
| Logical addresses are generated by the CPU during program execution, while physical addresses refer to actual locations in memory. The MMU translates logical to physical addresses. |
| **3.Describe the structure of a page table and its role in paging.** |
| A page table maps logical page numbers to physical frame addresses. It contains entries with frame numbers, access permissions, and status bits to manage paging effectively. |
| **4.What is swapping, and when is it used in memory management?** |
| Swapping temporarily moves processes between main memory and secondary storage, used when memory is insufficient for all active processes. |
| **5.What is virtual memory, and how does it extend physical memory?** |
| Virtual memory extends physical memory by using disk space to simulate additional memory, allowing processes to execute beyond available RAM size. |
| **6.What is thrashing, and how does it affect system performance?** |
| Thrashing occurs when excessive page faults force frequent swapping, drastically reducing CPU utilization and system performance due to high I/O overhead |
| **7.What is copy-on-write, and how does it optimize memory usage in process creation?** |
| Copy-on-write allows parent and child processes to share the same memory pages initially, copying them only when modifications occur, saving memory. |
| **8.Explain the purpose of page replacement algorithms in virtual memory management.** |
| Page replacement algorithms decide which memory page to replace when a new page needs to be loaded, optimizing memory access and reducing page faults. |
| **9.What are the main components of a mass storage structure?** |
| The main components include hard drives, solid-state drives, controllers, caches, and buses that handle data storage, access, and communication. |
| **10.What is the purpose of the seek time in HDD scheduling?** |
| Seek time measures the delay in moving the read/write head to the desired track on a disk, crucial for determining access speed and efficiency. |

**UNIT-5: File System, Protection**

|  |
| --- |
| **1.What is the file concept, and how are files structured?** |
| A file is a collection of related data stored on disk. Files can be structured as text, binary, or organized into records or data blocks for efficient access. |
| **2.How does a hierarchical directory structure organize files?** |
| A hierarchical directory structure organizes files in a tree format, with directories acting as nodes and files as leaves, enabling efficient navigation and grouping. |
| **3.What is the role of the file-system structure in managing data storage?** |
| The file-system structure organizes storage, manages metadata, and provides interfaces for storing, retrieving, and managing files efficiently. |
| **4.What are the main methods used to implement directories?** |
| Directories can be implemented using linear lists for simplicity or hash tables for faster lookup and better performance. |
| **5.Compare the contiguous and linked allocation methods for file storage.** |
| Contiguous allocation stores files in consecutive blocks for fast access but causes fragmentation, while linked allocation chains blocks together, improving flexibility but slowing access. |
| **6.What is the purpose of file-system mounting, and how is it performed?** |
| File-system mounting integrates a file system into the directory tree. The OS assigns a mount point to the storage device during the mounting process. |
| **7.What is the difference between partitions and file-system mounting?** |
| Partitions divide storage into isolated sections, while mounting connects a partition or device to the file system hierarchy for access. |
| **8.Explain how file sharing is managed in a multi-user system.** |
| File sharing uses access control mechanisms like user permissions, locks, and sharing protocols to enable safe multi-user collaboration. |
| **9.What are the main goals of protection in an operating system?** |
| Protection ensures data integrity, confidentiality, and controlled resource access to prevent unauthorized use or system breaches. |
| **10.Explain the principle of least privilege in the context of protection.** |
| The principle of least privilege grants users/processes only the minimum permissions needed, reducing security risks and potential misuse. |