OS tutorial questions

What are the different types of operating systems?

2. What are the basic functions of an op system?
3. What is user space and kernel space
4. What are the different kinds of kernEl

1. Types of Operating Systems

- **Batch Operating Systems**: Processes jobs in batches without user interaction.
- **Time-Sharing Operating Systems**: Allow multiple users to share system resources simultaneously (e.g., Unix, Linux).
- **Distributed Operating Systems**: Manage a group of distinct computers and present them as a single coherent system (e.g., Google's Android).
- **Real-Time Operating Systems**: Provide immediate processing for critical tasks (e.g., embedded systems).
- **Network Operating Systems**: Provide networking capabilities and manage network resources (e.g., Novell NetWare).
- **Mobile Operating Systems**: Designed for mobile devices (e.g., iOS, Android).

2. Basic Functions of an Operating System

- **Process Management**: Handles creation, scheduling, and termination of processes.
- **Memory Management**: Manages the system's memory allocation and deallocation.
- **File System Management**: Controls how data is stored and retrieved on disk.
- **Device Management**: Manages input/output devices and their communication.
- **User Interface**: Provides interfaces for users to interact with the system (command line, GUI).
- **Security and Access Control**: Protects data and resources against unauthorized access.

3. User Space and Kernel Space

- **User Space**: The memory area where user applications run. It is isolated from the kernel to prevent interference with critical system processes.
- **Kernel Space**: The memory area where the core of the operating system runs, including the kernel itself. It has direct access to hardware and system resources.

4. Different Kinds of Kernel

- **Monolithic Kernel**: A single large kernel that includes all system services (e.g., Linux).
- **Microkernel**: Minimal kernel that handles only essential services, with other services running in user space (e.g., Minix).
- **Hybrid Kernel**: Combines aspects of both monolithic and microkernel designs (e.g., Windows NT).
- **Exokernel**: Provides minimal abstractions to allow application-level management of hardware resources.
- **Nanokernel**: An even more minimalist approach, focusing on very small core functions and pushing most services into user space.

Differentiate between program and a process

6. Define the following: a) Multiprogramming system, b) Parallel systems c) Distributed systems
7. What is context switching?
8. What is a time-sharing system?
9. What are real-time systems?
10. Define the following: a) Loader b) Linker c) Scheduler

5. Difference Between Program and Process

- **Program**: A static set of instructions stored on disk, written in a programming language, and not currently executing. It represents a file containing code.
- **Process**: A dynamic instance of a program that is currently executing. It includes the program code, current activity (state), and allocated resources (memory, CPU).

6. Definitions

- **a) Multiprogramming System**: An operating system that allows multiple programs to reside in memory and execute concurrently by managing the CPU among them. It improves resource utilization.
- **b) Parallel Systems**: Systems that use multiple processors to execute multiple tasks simultaneously, enabling faster computation and processing.
- **c) Distributed Systems**: A system where components located on networked computers communicate and coordinate their actions to achieve a common goal, often providing higher reliability and resource sharing.

7. Context Switching

Context switching is the process of storing the state of a currently running process and loading the state of another process. This allows multiple processes to share the CPU effectively and is essential for multitasking.

8. Time-Sharing System

A time-sharing system allows multiple users to interact with a computer simultaneously by allocating time slices to each user’s processes, making it appear as if they are running concurrently.

9. Real-Time Systems

Real-time systems are designed to process data and provide output within a strict timing constraint. They are critical for applications where timing is crucial, such as embedded systems in medical devices or industrial control systems.

10. Definitions

- **a) Loader**: A program that loads executable files into memory and prepares them for execution by linking necessary libraries and setting up the execution environment.
- **b) Linker**: A tool that combines various object files into a single executable file, resolving references between them and ensuring that all required functions and variables are available.
- **c) Scheduler**: A component of the operating system responsible for determining which process or thread to run at any given time, optimizing CPU utilization and responsiveness.