Home Assignment CO-1

1] In a monolithic OS service like device drivers, file system and network stocks which are internally may be externally subsystem in a microkernel os running in user space.

2] Time sharing supports multiple user multi programming runs multiple program interupts including hard ware, software and timer.

3] Monolithic, microkernel, layered, modular, Hybrid

4] Process creation allocates load, setup, fork()wait, due to resource limit is invalid executable no child.

5] Limited direct execution timer interupt triggers. context switches saving state in process. stable the process table tracks activated process in the time sharing.

6] Scheduling is required for multitasking purpose to allocate cpu timer: its less crucial in single tasking or real time system with resource.

7] Proportional sharing divides cpu by weight multiprocessor scheduling balance task accross CPU.

8] Unix os structure:

Block diagram: Kernel, Shell, User Applications

Kernel: Manages resources, processes and memory.

Shell: Interface between user and kernel

User Application: Run on top of the shell.

9] wait(): Parent process waits for child termination.

exit(): Terminates a process.

Interaction: wait() captures the exit status of the child process.

10] Scheduling Essential vs Not Essential:

Essential: In multitasking environments for resource allocation.

Not Essential: In single task systems where only one process runs.

11] Shell and shell scripting:

Shell: command line interface.

Shell scripting: Automates tasks using commands in a script.

Ex: Automating backups using a bash script.

12] Time sharing vs Distributed Systems:

Time sharing:

Multiple users interact with one system simultaneously.

Distributed systems:

Multiple systems work together to perform tasks.

13] Non-Preemptive vs Distributed Systems:

Time Sharing: Multiple users interact with one system.

Non-Preemptive: A process runs until completion or yields.

Preemptive: The OS can interupt and switch processes to ensure fairness.

14] Process and memory Management:

Process Management: Controls process creation, execution & termination.

Memory Management: Allocates, tracks and manages memory use.

System Performance: Effecient management optimizes CPU & memory utilization

15] Simple Batch System:

Jobs are collected, grouped and executed sequentially without interaction.

16) fork() and exec():

fork(): Duplicates the calling process.

exec(): Replaces the current process memory with a new program.

importance: Fundamental for process creation and execution flow in unix.

17] Role of system calls:

Systemcalls: Interface between user applications and the kernel, allowing safe hardware access.

18] Types of OS:

Batch: Processes jobs in batches without user interaction.

Real-Time: Guarantees response times for crucial tasks.

Time-Sharing: Multiple users interact with the system concurrently.

Distributed: Systems collaborate to perform tasks.

19) TimeSharing vs Distributed Systems:

TimeSharing: Multiple users on one system.

Distributed Systems: Collaboration across multiple Systems.

2a) Microkernel vs Monolithic:

Advantage: Increased modularity and security.

Interaction: Through message passing

Disadvantages: Potentially slower performance due to overhead.

Modular vs layered: Modular allows more flexibility, while layered is more rigid but easier to debug.

2] Process Life Cycle:

Stages: New, Ready, Running, Waiting, Terminated.

Transitions: Eg: Running ⇒ Waiting (I/O request).