Fundamentals

Processes vs Threads

Main : Processes are executed by threads

Memory

Stack

- Stack allocation called by function
- Each function gets its own memory for local variables
- Freed immediately when function ends
- Too many function calls exceed stack capacity -> stack overflow error (JAVA) / **segmentation fault (CPP)**

Heap

- Manual deallocation required
- Accessible by multiple threads
- Heap is persistent until program termination
- **Memory leak :** allocated data is not freed after use

File I/O

Concurrency

Scheduling

System Calls

Processes vs Threads

- Process: own address space
- Thread: shared address space
- Context switching overhead
- When threads help vs hurt

Memory

- Stack vs heap
- Virtual memory
- Paging vs segmentation
- What a segmentation fault actually is
- Memory leaks and how they happen

File I/O

- File descriptors
- stdin/stdout/stderr
- Blocking vs non-blocking I/O

Concurrency

- Race conditions
- Mutex vs semaphore
- Deadlocks (4 conditions)

Scheduling

- Preemptive vs cooperative
- CPU-bound vs I/O-bound

System calls

- What they are
- Why they exist
- User mode vs kernel mode

Practice explaining out loud

If she asks:

> “What happens when a program starts?”

You should mention:

- Executable loaded
- Virtual memory allocated
- Stack/heap setup
- Entry point (`main`)
- Syscalls for I/O

Revision #2
Created 2026-01-29 13:39:48 UTC by Admin
Updated 2026-01-31 11:38:29 UTC by Admin