Os concepts

🧠 Concept	💡 What It Is	📌 Use
Process	A program in execution with allocated resources	Manages independent program execution
Thread	Lightweight unit within a process sharing memory	Enables efficient multitasking
Context Switching	CPU switches from one process/thread to another	Supports multitasking by saving state
Scheduling	Decides which process/thread runs next	Boosts CPU utilization and responsiveness
Deadlock	Two+ processes wait indefinitely for resources	Identifies system freeze due to resource conflict
Starvation	Process waits forever due to resource preference to others	Highlights unfair scheduling or poor priority handling
Race Condition	Incorrect output from unsynchronized access	Diagnoses critical bugs in concurrent systems
Mutex / Semaphore	Tools for synchronization between threads	Ensures data consistency in shared environments
Critical Section	Code segment that must not be run by more than one thread at once	Ensures safe access to shared resources

Paging	Divides memory into fixed-size pages	Efficient memory management
Segmentation	Divides memory into logical segments	Logical organization of memory
Virtual Memory	Uses disk as an extension of RAM	Runs large programs on small RAM
Swapping	Moves processes between RAM and disk	Frees memory during high usage
Memory Management	Handles allocation and deallocation of memory	Prevents memory leaks, improves efficiency
Cache Memory	Fast memory closer to CPU	Speeds up frequent data access
Kernel	Core part of OS managing hardware & system calls	Handles essential system tasks
System Call	Interface between user apps and OS	Allows apps to use OS features (file, memory, etc.)
File System	Structure for storing & accessing data	Manages files and directories
I/O Management	Controls interactions with I/O devices	Handles hardware communication
Interrupts	Signals to CPU for immediate attention	Prioritizes events like keystrokes, errors
Booting	Starting up the OS from storage	Loads kernel and essential services
Daemon	Background process/	Runs maintenance or

	service	server tasks
User Mode / Kernel Mode	Modes for executing code with/without privileges	Protects system from faulty user programs
Signals	Software interrupts to processes	Handles kill, stop, or custom events
Shell	Interface to run system commands	Lets users interact with the OS
Top-Down Parsing	Parsing source code from the highest-level rule	Used in OS compilers and loaders
Thrashing	Excessive swapping of memory pages	Happens due to insufficient RAM
Multitasking	Running multiple tasks simultaneously	Increases CPU usage and responsiveness
Multithreading	Running multiple threads in one process	Efficient use of resources for concurrent tasks
Time Sharing	Dividing CPU time between tasks	Makes systems feel responsive for users
Monolithic vs Microkernel	OS architecture types	Impacts how services and kernel communicate