OPERATING SYSTEM CONCEPTS

INTRODUCTION TO OPERATING SYSTEM

- • An operating system (OS) is system software that controls the overall functioning of a computer.
- • It acts as an interface between the user and computer hardware components.
- • The OS manages hardware resources such as CPU, memory, storage, and I/O devices.
- • It ensures that programs run smoothly without interfering with each other.
- • Without an operating system, a computer cannot perform meaningful tasks.
- • Every modern computing device requires an operating system.

WHAT IS AN OPERATING SYSTEM

- • The operating system is the main control program of a computer system.
- • It starts functioning as soon as the computer is powered on.
- • It controls execution of all user and system programs.
- • It allocates resources to programs when required.
- • It ensures security and proper usage of the system.
- • It makes complex hardware easy for users to operate.

NEED FOR OPERATING SYSTEM

- • Manages multiple programs running at the same time.
- • Prevents misuse of hardware resources.
- • Provides user-friendly interaction with system.
- • Ensures efficient use of CPU and memory.
- • Protects data and programs from damage.
- • Simplifies computer operation.

FUNCTIONS OF OPERATING SYSTEM

- • Process management to control program execution.
- • Memory management to allocate RAM efficiently.
- • File system management to store data systematically.
- • Device management to control hardware devices.
- • Security management to protect system.
- • User interface management for interaction.

PROCESS MANAGEMENT

- • A process is a program currently being executed by CPU.
- • The OS creates a process when a program starts.
- • It assigns CPU time to processes using scheduling.
- • It manages synchronization between processes.
- • It handles deadlock situations.
- • Enables multitasking and multiprocessing.

PROCESS STATES

- • New state represents newly created process.
- • Ready state means process is waiting for CPU.
- • Running state means process is executing.
- • Waiting state occurs during I/O operations.
- • Terminated state means process has finished execution.
- • OS controls transition between states.

PROCESS CONTROL BLOCK (PCB)

- • PCB stores all information related to a process.
- • It contains process ID and process state.
- • It stores CPU register contents.
- • It includes memory management information.
- • OS uses PCB for scheduling decisions.
- • Essential for multitasking.

CPU SCHEDULING

- • CPU scheduling decides which process gets CPU time.
- • Improves utilization of CPU.
- • Reduces waiting and turnaround time.
- • Ensures fairness among processes.
- • Increases system throughput.
- • Implemented using scheduling algorithms.

SCHEDULING ALGORITHMS

- • First Come First Serve schedules processes in arrival order.
- • Shortest Job First selects process with least burst time.
- • Priority Scheduling executes based on priority.
- • Round Robin gives fixed time slice to each process.
- • Each algorithm has advantages and disadvantages.
- • Selection depends on system requirements.

MEMORY MANAGEMENT

- • Memory stores programs and data temporarily.
- • OS allocates memory space to each process.
- • Prevents one process from accessing another’s memory.
- • Frees memory when process terminates.
- • Supports virtual memory concept.
- • Improves system efficiency.

MEMORY MANAGEMENT TECHNIQUES

- • Contiguous memory allocation assigns continuous memory.
- • Paging divides memory into fixed-size pages.
- • Segmentation divides memory into logical segments.
- • Virtual memory uses disk as additional memory.
- • Reduces memory wastage.
- • Managed by OS.

VIRTUAL MEMORY

- • Allows execution of large programs using limited RAM.
- • Uses hard disk as extension of main memory.
- • Pages are swapped between disk and memory.
- • Improves multitasking capability.
- • Reduces system crashes.
- • Increases system performance.

DEADLOCK

- • Deadlock occurs when processes wait indefinitely.
- • Each process holds a resource and waits for another.
- • No process can proceed.
- • Causes system halt.
- • Reduces performance.
- • OS must handle deadlocks.

DEADLOCK CONDITIONS

- • Mutual exclusion means resources cannot be shared.
- • Hold and wait means process holds one resource and waits.
- • No preemption means resources cannot be taken forcibly.
- • Circular wait means circular chain of processes.
- • All conditions must exist.
- • Necessary for deadlock.

DEADLOCK HANDLING

- • Deadlock prevention avoids one of the conditions.
- • Deadlock avoidance predicts and avoids deadlock.
- • Deadlock detection finds deadlock after occurrence.
- • Deadlock recovery breaks deadlock.
- • OS chooses strategy based on system.
- • Improves reliability.

FILE SYSTEM MANAGEMENT

- • File system organizes data on storage devices.
- • Controls file creation, deletion, and access.
- • Manages directories.
- • Ensures data security and integrity.
- • Supports backup and recovery.
- • Managed by OS.

FILE OPERATIONS

- • Creating new files.
- • Reading data from files.
- • Writing data to files.
- • Deleting unnecessary files.
- • Closing files after use.
- • Controlled by OS.

DISK MANAGEMENT

- • Disk stores data permanently.
- • OS manages disk space allocation.
- • Schedules disk access requests.
- • Reduces access time.
- • Improves overall performance.
- • Ensures data safety.

DEVICE MANAGEMENT

- • OS controls all hardware devices.
- • Uses device drivers for communication.
- • Allocates devices to processes.
- • Handles interrupts from devices.
- • Coordinates I/O operations.
- • Improves efficiency.

I/O SYSTEM

- • Manages input and output operations.
- • Uses buffering techniques.
- • Uses spooling methods.
- • Handles interrupts.
- • Improves data transfer speed.
- • Managed by OS.

INTERRUPTS

- • Interrupt is a signal to CPU.
- • Temporarily stops current execution.
- • Used to handle urgent tasks.
- • Improves system responsiveness.
- • Generated by hardware or software.
- • Managed by OS.

SECURITY AND PROTECTION

- • Protects system from unauthorized users.
- • Controls access permissions.
- • Protects data integrity.
- • Prevents system misuse.
- • Ensures secure operation.
- • Managed by OS.

SYSTEM CALLS

- • Interface between application programs and OS.
- • Used to request OS services.
- • Provides controlled access to resources.
- • Examples include read, write, open.
- • Essential for program execution.
- • Implemented by OS.

TYPES OF OPERATING SYSTEMS

- • Batch operating system.
- • Time sharing operating system.
- • Distributed operating system.
- • Network operating system.
- • Real-time operating system.
- • Mobile operating system.

BATCH OPERATING SYSTEM

- • Executes jobs in batches.
- • No user interaction during execution.
- • Used in early computer systems.
- • High turnaround time.
- • Suitable for large data processing.
- • Less flexible.

TIME SHARING OPERATING SYSTEM

- • Multiple users share CPU time.
- • Each process gets time slice.
- • Supports multitasking.
- • Fast response time.
- • Used in multi-user environments.
- • Improves efficiency.

DISTRIBUTED OPERATING SYSTEM

- • Multiple computers connected together.
- • Share resources transparently.
- • Improves reliability and performance.
- • Reduces workload on single system.
- • Appears as single system.
- • Used in large networks.

NETWORK OPERATING SYSTEM

- • Manages network resources.
- • Provides file and printer sharing.
- • Controls user access.
- • Provides centralized security.
- • Used in servers.
- • Example: Windows Server.

REAL-TIME OPERATING SYSTEM

- • Responds within strict time limits.
- • Used in critical applications.
- • High reliability.
- • Used in medical and industrial systems.
- • Delays not acceptable.
- • Ensures real-time response.

MOBILE OPERATING SYSTEM

- • Designed for smartphones and tablets.
- • Supports touch interface.
- • Manages battery efficiently.
- • Supports mobile apps.
- • Examples include Android and iOS.
- • Optimized for portability.

OPERATING SYSTEM STRUCTURE

- • Defines internal organization of OS.
- • Monolithic structure.
- • Layered structure.
- • Microkernel structure.
- • Modular structure.
- • Affects performance and security.

MONOLITHIC OPERATING SYSTEM

- • All services in one kernel.
- • Fast execution speed.
- • Difficult to maintain.
- • Large kernel size.
- • Less secure.
- • Example: UNIX.

LAYERED OPERATING SYSTEM

- • OS divided into layers.
- • Each layer performs specific task.
- • Easy to debug.
- • Easy to maintain.
- • Organized structure.
- • Slightly slower.

MICROKERNEL OPERATING SYSTEM

- • Minimal services in kernel.
- • Other services in user space.
- • High security.
- • High reliability.
- • Modular design.
- • Used in modern OS.

SUMMARY AND CONCLUSION

- • Operating system is core of computer system.
- • Manages resources efficiently.
- • Ensures security and stability.
- • Supports multitasking.

THANK YOU

Shamna Subaida Khalid

shamnaplpy@gmail.com