UNIT - 1

Operating System

An operating system (OS) is system software that manages computer hardware and software resources, and provides common services for computer programs.

Characteristics / Functions of Operating Systems

1. **Resource Management:** Handles allocation and deallocation of resources.
2. **Process Management:** Controls the execution of programs (processes).
3. **Memory Management:** Allocates memory space to running programs and manages memory
4. **File Management:** Provides mechanisms for creating, deleting, accessing, and modifying files.
5. **Device Management:** Acts as an intermediary between software and hardware devices.
6. **Security:** Controls access to system resources and protects against unauthorized access.
7. **User Interface (UI):** Provides a way for users to interact with the operating system. (CLI / GUI)

Types of Operating Systems

1. **Batch Operating System:** Programs are submitted in batches and processed one after another.
2. **Multiprogramming OS:** Allows multiple programs to be loaded into memory.
3. **Multiprocessing OS:** Distributes tasks among multiple CPUs.
4. **Multitasking OS:** Allows users to run multiple programs concurrently.
5. **Time-Sharing OS:** Multiple users on a single computer system.
6. **Real-Time OS(RTOS):** Guarantees predictable response times for critical tasks.
7. **Distributed OS:** Distributes tasks and data across the network.
8. **Network Operating System (NOS):** Manages resources on a network.

Concept of System Calls

System calls are the programmatic way a program requests services from the operating system's kernel, the core of the OS.

Types of System Calls:

- **Process Control:** Manage the creation, termination, and scheduling of processes.
- **File Management:** Create, read, write, and delete files and directories.
- **Device Management:** Interact with hardware devices like printers, disks, and networks.
- **Information Maintenance:** Get information about the system, like memory usage.
- **Communication:** Allow processes to exchange data and synchronize their actions.

Process Management

Fundamental function of an OS that deals with the creation, scheduling, and termination of processes. A process is essentially a program that's actively being executed.

Process: A process is an instance of a program that's currently being executed by an OS.

Process State:

- **New:** Just created, waiting for resources.
- **Ready:** Waiting for the CPU to be assigned.
- **Running:** Actively executing instructions.
- **Waiting/Blocked:** Needs a resource (like I/O) before continuing.
- **Terminated:** Finished execution.

Process Control Block (PCB)

A data structure used by the Operating System (OS) to manage processes.
Holds all relevant information about a process for the OS to track its state and manage resources.

Information stored in a PCB:

- **Process State:** Running, waiting, ready, terminated.
- **Process ID (PID):** Unique identifier for the process.
- **Program Counter (PC):** Memory address of the next instruction to be executed.
- **CPU Registers:** Values stored in CPU registers specific to the process.
- **Memory Management Information:** Memory allocated to the process.
- **I/O Status Information:** Open files and devices associated with the process.

Context Switching

The process of saving the state of a running process and then switching to a different process.

- **Multitasking:** Enables a single CPU to appear to run multiple programs concurrently.
- **Process management:** Allows the OS to manage multiple processes and prioritize tasks.

CPU Scheduling

The process of deciding the order in which processes will be allocated to the CPU for execution.
This ensures efficient utilization of CPU resources in a multiprogramming environment.

1. **Long-Term Scheduler (Job Scheduler):** Controls the number of processes in ready state.
2. **Short-Term Scheduler (CPU Scheduler):** Selects a process from the queue for execution.
3. **Medium-Term Scheduler:** Swaps processes between main memory and secondary storage.

Scheduling Criteria

1. **CPU Utilization:** Measures how busy the CPU is.
2. **Throughput:** Represents the number of processes completed per unit time.
3. **Turnaround Time:** The total time taken for a process to finish execution.
4. **Waiting Time:** The time a process spends waiting in the ready queue for its turn on the CPU.
5. **Response Time:** The time it takes for a process to start responding after submitting a request.

Scheduling Algorithms

1. **First Come First Served (FCFS):** Processes are served in the order they arrive.
2. **Shortest Job First (SJF):** Prioritizes processes with the shortest execution time.
3. **Longest Job First (LJF):** Prioritizes processes with the longest execution time.
4. **Priority Scheduling:** Assigns priorities to processes. Higher priority processes are served first.
5. **Round Robin (RR):** Processes are allocated a short time slice (quantum).

Critical Section Problem

A section of code in a multi-threaded program where shared resources are accessed.
Concurrent execution in this section can lead to inconsistencies and errors (like race conditions).

Key Properties for Solutions:

- **Mutual Exclusion:** Only one thread can be in the critical section at a time.
- **Progress:** If no thread is in the critical section, a waiting thread should eventually enter.
- **Bounded Waiting:** There should be a limit on how long a thread waits to enter.

Inter-Process Communication (IPC)

Mechanisms that allow processes to communicate and exchange data with each other.
Types of IPC:

- **Shared Memory:** Processes share a designated memory region to exchange data directly.
- **Message Passing:** Processes send and receive messages through queues or mailboxes.
- **Pipes:** Processes communicate through unidirectional data streams.
- **Semaphores:** Ensures that only one process can access a shared resource at a time.

Semaphores

Synchronization tools used to control access to shared resources between processes.
Semaphores don't transfer data, they coordinate access to prevent conflicts.
Operations:

- **P(wait):** Decrements the semaphore value. Process blocks until another process signals.
- **V(signal):** Increments value. If any processes are blocked waiting, one is woken up.

Types:

- **Binary Semaphores:** Only two values (0 or 1) - ensures only one process accesses the resource.
- **Counting Semaphores:** Value range beyond 1 - controls resources with multiple instances.

UNIT - 2

Memory Management

Memory management efficiently allocates RAM and prevents programs from interfering with each other's memory space. It ensures smooth program execution and effective use of this resource.

Logical and Physical Address Space

Logical / Virtual Address Space: Set of all logical addresses generated by the CPU for a program.

Physical Address Space: Set of all physical addresses in the main memory (RAM).

Key Differences:

- **Visibility:** Users work with logical addresses, while physical addresses are hidden MMU.
- **Mapping:** MMU translates logical addresses to physical addresses before memory access.
- **Portability:** Logical addresses are portable across different systems.

Swapping in Memory Management

Temporarily moves inactive processes to disk to free RAM for active ones, enabling more programs to run concurrently than physical RAM allows.

Contiguous Memory Allocation

The operating system allocates a single, continuous block of memory to a process.

- **Fixed Partitioning:** Memory is divided into equal-sized blocks beforehand.
- **Variable Partitioning:** Blocks can be of varying sizes, allowing for a better fit.

Advantages / Disadvantages:

- + Faster memory access due to contiguous data storage.
- – Memory wastage due to fragmentation.
- – Not suitable for systems with dynamic memory requirements.

Multiple Partitions

Divides main memory into fixed-size chunks called partitions. Each partition holds a single process at a time.

Advantages / Disadvantages:

- + Easy to protect memory space for the operating system.
- – **Fragmentation:** Wasted memory due to fragmentation.

Fragmentation

1. External Fragmentation: Occurs when there's enough total free memory to fit a new process, but it's divided into unusable small blocks.

2. Internal Fragmentation: Occurs when allocated memory has unused space within it due to memory allocation strategies.

Compaction

It is a technique used to address external fragmentation. It involves shuffling the allocated memory blocks in the main memory to create one larger contiguous block of free space.

Paging

Divides both physical memory and processes into fixed-size blocks called pages and frames, respectively. Pages can be stored anywhere in physical memory as long as a frame is available.

Logical vs. Physical Address:

- The CPU generates logical addresses for memory access within a process.
- A Memory Management Unit (MMU) translates the logical address into a physical address using a page table.

Page Table: The page table is a data structure maintained by the OS. It maps logical page numbers to physical frame numbers.

Benefits of Paging:

- **Non-contiguous Allocation:** Processes don't require contiguous memory blocks.
- **Simplified Memory Management:** can load only required pages of a process to memory.
- **Protection:** Page tables can define access permissions for each page.

Virtual Memory Management

It utilizes secondary storage (hard disk) to extend the capacity of primary memory (RAM).

Benefits:

1. Run Larger Programs: Allows programs exceeding RAM size to run by loading parts as needed.
2. Increased Multiprogramming: Enables running multiple processes simultaneously.
3. Memory Protection: Isolates processes, preventing them from interfering with each other.

Implementation:

- **Division:** Divides virtual memory and physical memory into fixed-size blocks.
- **Translation:** A MMU in the CPU translates virtual addresses used by programs to physical addresses in RAM.
- **Demand Paging:** Loads only the required program pages into RAM when accessed, reducing initial load time. Triggers a page fault to swap data between RAM and disk.

Demand Paging

Technique where a program's sections (pages) are loaded into RAM only when needed, triggered by a page fault. This optimizes RAM usage by deferring loading of unused pages.

Benefits / Drawbacks:
+ Runs larger programs on limited RAM.
+ Enables efficient multitasking by allowing more processes in memory.
– **Page Faults:** Can slow down program execution if they occur frequently.
– **Overhead:** Managing page tables and swapping pages adds some processing overhead.

Page Replacement Algorithms

Algorithms to decide which memory pages to swap out to secondary storage (disk) when a page fault occurs. Page fault happens when a requested page isn't present in main memory.

1. **First In, First Out (FIFO):** The oldest page (at the queue's front) gets replaced.
2. **Least Recently Used (LRU):** Replaces the page that hasn't been used for the longest time.
3. **Optimal Page Replacement:** Replaces the page that won't be used for the longest time.

UNIT - 3

Deadlock

A state where a set of processes are **blocked**, waiting for resources held by each other.

Conditions/Characterization of Deadlock:

- **Mutual Exclusion:** Resources are exclusive (one process at a time).
- **Hold and Wait:** Processes hold resources while waiting for others.
- **No Preemption:** Resources cannot be forcibly taken away.
- **Circular Wait:** A circular chain of processes waiting for resources.

Preventing Deadlock

- **Mutual exclusion:** Not realistic for most resources.
- **Hold and wait:** Complex to implement, can lead to under-utilization.
- **No preemption:** Risky, can cause data inconsistency.
- **Resource ordering:** Assign priorities to resources, ensuring processes only request resources in a specific order.

Deadlock Avoidance

Deadlock avoidance proactively prevents deadlocks. The OS anticipates resource needs and approves requests only if the future state is guaranteed to be safe.

Deadlock Detection and Recovery

- **Detection:** The system employs techniques like resource-allocation graphs or the banker's algorithm to detect deadlocks.

- **Recovery:** Once a deadlock is identified, the system takes steps like process termination or resource preemption to break the deadlock and restore normal operation.

File System

A file system is a method used by an operating system to organize, store, and manage files and folders on a storage device like a hard drive, solid state drive (SSD), or USB flash drive.

Common File Systems:

- **NTFS (New Technology File System):** Offers features like file permissions, encryption etc.
- **FAT (File Allocation Table):** Used in older MS-DOS systems and some digital cameras.
- **HFS+ (Hierarchical File System):** Used by Apple Macintosh computers.
- **ext4 (Fourth Extended Filesystem):** Used in Linux. Supports large files and partitions.

File Concept

A file is a digital container that stores information on a computer.
Files reside on secondary storage devices for persistent storage, unlike RAM which is volatile.

Identification: Each file has a unique name to distinguish it from others.

Organization: Files are organized within directories (also called folders) to create a hierarchy.

File Access Methods

1. Sequential Access: Processes data one record at a time, starting from the beginning.
2. Direct Access: Allows jumping directly to any specific location in the file.
3. Indexed Sequential Access: Uses an index to locate specific records quickly.

Directory Structure

- **Organization System:** Organizes files using a hierarchy of folders.
- **Tree Structure:** Directories branch out like a tree, with a main directory at the top.

Benefits:

- **Easier to find files:** Files grouped by category make them easier to locate.
- **Improved Organization:** Reduces clutter and keeps things tidy.
- **Efficient Management:** Simplifies file management tasks like backup and access control.

Types of Directory Structures

1. Single-Level Directory Structure: All files reside directly in the root directory.
2. Two-Level Directory Structure: Each user gets a dedicated directory for their files.
3. Tree-Structured Directory Structure: Files and subdirectories organized hierarchically, resembling an upside-down tree.
4. Acyclic-Graph Directory Structure: Similar to tree structure but allows for some looping.

5. General-Graph Directory Structure: Most complex structure with unrestricted loops and connections.

File System Protection

1. Permissions: Defines who can access a file and what they can do (read, write, execute).
2. Access Control Lists (ACLs): Granular control specifying access permissions for individual users or groups.
3. Encryption: Scrambles file content using a key, making it unreadable without authorization.

File System Structure

Organizes files and folders on a storage device for efficient storage, retrieval and access.

- **Directories (Folders):** Contain files and other subdirectories, creating a hierarchy
- **Files:** Store actual data like text, images, videos etc.
- **File Metadata:** Additional information about a file like size, creation date, permissions etc.

Logical vs Physical Structure:

- **Logical Structure:** The tree-like directory structure users see.
- **Physical Structure:** The way data is actually stored on the storage device.

Directory Implementation

1. Linear List: Stores filenames with pointers to their data blocks in a sequential order.
2. Hash Table: Uses a hash function to quickly locate files based on their filenames.
3. B-Trees: Offer efficient searching, insertion, and deletion operations for very large directories.
4. Self-Balancing Trees: automatically maintain a balanced structure for search performance.

File Allocation Methods

1. Contiguous Allocation: Stores files in a continuous block of disk space.
2. Linked Allocation: Stores files in non-contiguous blocks, linked together by pointers.
3. Indexed Allocation: Stores a table (index) containing block addresses for the file.

Free Space Management

Free space management is a crucial part of file systems in operating systems. It tracks and allocates unused storage space on a disk drive for efficient storage of files.

1. Bit Vector: Uses a bit for each block on the disk.
2. Linked List: Links all free blocks together in a chain. Requires searching the list to find a block.
3. Grouping: Allocates fixed-size blocks and keeps track of free groups.
4. Counting: Maintains a counter for the number of free blocks.

UNIT - 4

Introduction to Linux Operating System

Free and open-source operating system (OS) based on the Linux kernel.

Key Features of Linux:

- **Free & Open-Source:** Free to use and modify, with a large community.
- **Stable and secure:** Known for reliability and strong security features.
- **Cost-effective:** Free to use and modify, reducing software licensing costs.
- **Flexible:** Highly customizable to fit specific needs.
- **Command Line:** Powerful command line interface for efficiency.
- **Multitasking & Multi-user:** Multiple users and programs can run simultaneously.
- **Hardware Compatibility / Versatile:** Runs on a wide range of devices.
- **Package Management:** Easy software installation and removal with package managers.

Basic Utilities in Linux

File Manipulation:

- **ls:** Lists files and directories in the current directory.
- **cd:** Changes directory.
- **mkdir:** Creates a new directory.
- **cp:** Copies files or directories.
- **mv:** Moves or renames files or directories.
- **rm:** Removes files or directories.

File Viewing and Editing:

- **cat:** Displays the contents of a file.
- **more/less:** View files one screen at a time (useful for long files).
- **nano/vim:** Text editors for creating and modifying files.

System Information:

- **pwd:** Shows your current working directory (where you are in the file system).
- **uname:** Displays information about the system kernel.
- **whoami:** Tells you the name of the user you are currently logged in as.
- **hostname:** Shows the hostname of the machine.
- **df:** Displays information about disk usage.

Network Utilities:

- **ping:** Checks connectivity to another host by sending packets.
- **wget:** Downloads files from the internet.

Other Useful Utilities:

- **sudo:** Allows you to run commands as another user, typically the root user.
- **man:** Provides detailed information (manual pages) about other commands.
- **clear:** Clears the terminal screen.
- **help:** Provides basic help for some built-in shell commands.

Working with Files in Linux

File Types:

- **Regular Files:** These are the most common files containing data.
- **Directories:** Folders that organize other files.
- **Special Files:** Provide access to devices like hard drives or printers.
- **Symbolic Links (Symlinks):** Shortcuts pointing to other files.
- **Named Pipes (FIFOs):** Allow data exchange between processes.
- **Socket Files:** Facilitate network communication.

File Management Commands:

- `touch`: Creates an empty file
- `mkdir`: Creates a new directory
- `cp`: Copies a file
- `mv`: Moves or renames a file
- `rm`: Deletes files
- `rmdir`: Deletes empty directories
- `find`: Locates files based on criteria
- `cat`: Displays file contents
- `less` / `more`: View contents a page at a time
- `chmod` / `chown`: Control access to files

Shells in Linux

A shell is a program that acts as an interface between the user and the Linux operating system.

Key Functions of Shell:

- Interprets user-typed commands.
- Executes programs and utilities.
- Provides a way to manage files and directories.
- Allows for automation through shell scripting.
- More control over the system compared to a graphical interface.

Types of Shells in Linux

1. Bourne Shell (sh): The original Unix shell, known for its simplicity.
2. C Shell (csh): Similar to C. Offers features like filename completion and command history.
3. Bourne Again Shell (bash): Offers a good balance of features and ease of use.

4. **Z Shell (zsh)**: Powerful shell with a lot of customization options and plugins.
5. **Friendly Interactive Shell (fish)**: Designed to be user-friendly and easy to learn.

Shell Programming

Shell programming involves writing scripts that automate tasks in Unix-based systems using CLI.

Shell Scripts:

- Programs written in a shell's scripting language.
- Contain a series of shell commands.
- Executed line by line by the shell.
- Saved as plain text files with a `.sh` extension (e.g., `myscript.sh`).

Text Editors in Linux

1. **Vim/Vi:** Powerful and highly customizable, Popular among programmers for its efficiency.
2. **Nano:** Simpler and user-friendly alternative to Vim.
3. **gedit (GNOME):** Default editor for GNOME desktops. Offers basic functionalities.
4. **Kwrite (KDE):** Default editor for KDE desktops. Similar to gedit with a focus on KDE integration.
5. **Sublime Text:** Feature-rich editor with a clean interface and powerful plugins.
6. **Emacs:** Another powerful editor with extensive customization options.
7. **Atom/VSCode:** Modern editors with a focus on extensibility and collaboration.

Key Features of Vim

1. **Modal Editing:** Vim operates in different modes for specific tasks:

    - **Normal Mode (default):** Use keyboard shortcuts for navigation, deletion, copying, etc.
    - **Insert Mode:** Enter text like a regular editor.
    - **Visual Mode:** Select text visually for editing.
    - **Command-Line Mode:** Enter commands for saving, quitting, searching, etc.

2. **Efficient Text Manipulation:** Powerful motions to select text quickly.
3. **Highly Customizable:** Extensive configuration options through a simple text file.
4. **Lightweight and Versatile:** Runs efficiently on any system with minimal resources.