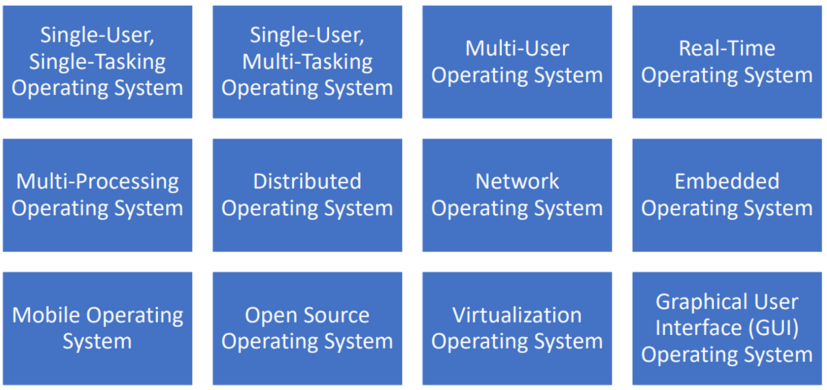
Operating system

Topic 1: OPERATING SYSTEM OVERVIEW

An **operating system (OS) is** a software that serves as an intermediary between computer hardware and the computer user.

It provides a user interface and manages computer hardware and software resources.



The primary functions of an operating system include:
- process management
- memory management
- file system management
- device management
- security

OS Managers
- Monitor resources continuously
- Enforce policies - Who gets what, when, how much
- Allocate resources
- Deallocate resources

Processor Manager: in charge of CPU
Responsibilities:
- Tracks process status (An instance of program execution)
- Two levels of responsibilities:
- Handle jobs entering the system -Job Scheduler
- Manage each process within jobs - Process Scheduler

Memory Manager: in charge of the main memory (RAM)
Responsibilities:
- Preserving OS area
- Checking validity of memory space request
- Setting up memory tracking table
- Tracks usage of memory by sections

• Needed in multiuser environment
• Deallocating memory to reclaim it

Network Manager: In charge of controlling, planning, allocating and monitoring network resources
Responsibilities:
• A software utility that aims to simplify the use of computer networks.

File Manager
Responsibilities:
• Choosing most efficient resource allocation method
• Printers, ports, disk drives, etc. based on scheduling policy
• Allocating the device
• Starting device operation
• Deallocating the device

Device Manager
Responsibilities:
• Choosing most efficient resource allocation method
• Printers, ports, disk drives, etc. based on scheduling policy
• Allocating the device
• Starting device operation
• Deallocating the device

A user interface (UI): is a means through which a user interacts with a computer system, software application, or electronic device. It serves as a bridge between the user and the underlying technology.

Types of user interfaces:

- **Graphical user interface:** A GUI is the visual representation of the interface, typically seen on computer screens. it uses graphical elements such as icons, buttons, windows, and menus to facilitate user interactions.
- Command line interface: Users interact with a system by entering text-based commands. This is often used by more technical users.
- **Voice user interface:** VUIs enable users to interact with technology using spoken language.
- **touchscreen**: Touchscreen interfaces are prevalent in smartphones, tablets, and some computers. Users interact with the system by touching the screen directly.
- **Web-user interface:** Web UIs are specifically designed for web applications and websites. They use a combination of text, images, links, and interactive elements to present information and enable user interactions through web browsers.

Operating systems example:

MS-DOS: Stands for Microsoft Desktop operating system. Doesn't support multi-user multi-tasking

Windows: Produced my Microsoft, use Graphical user interface.

Mac-OS: The official name is Macintosh, produces by apple

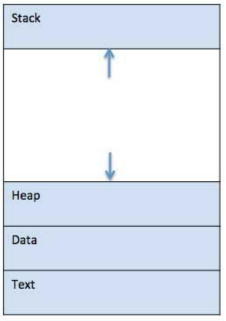
Linux: Freely distributed open source operating system that runs on a number of hardware planforms.

Topic 2: PROCESS DESCRIPTION & CONTROL

A process refers to a fundamental concept that represents an independent program or task being executed.

- It consists of the program's code, its data, and the various system resources it requires to function.
- Each process has its own memory space and runs independently of other processes, allowing for multitasking and concurrent execution on a computer.
- Processes are managed by the operating system, which allocates CPU time and system resources to them.
- They can communicate with each other through inter- process communication mechanisms and are essential for the proper functioning of modern operating systems.
- Processes can be created, terminated, paused, and resumed, making them a fundamental building block for managing and organizing tasks in a computer system.

When a program is loaded into memory and becomes a process, it is typically divided into four primary sections:
• Text
• Data
• Heap
• Stack



Text
• The text section contains the executable code of the program.
• It includes the instructions that the CPU will execute.
• This section is typically read-only to prevent accidental modification of the program's code.
• Example: In a word processing program, the text section contains the code responsible for functions like opening, saving, and formatting documents.

Data
• The data section includes global and static variables.
• These variables have a predefined size and are used for storing data that persists throughout the program's execution.

• Example: In a game, the data section may contain variables that store the player's score, game settings, and the state of the game world.

Heap
• The heap is a dynamic memory area used for allocating memory during the program's execution.
• Data in the heap is allocated and deallocated as needed and does not have a predetermined size.
• Example: In a web browser, the heap is used to allocate memory for rendering web pages, images, and other content, which can vary in size and be loaded as needed.

Stack
• The stack is used for managing function calls and local variables.
• It operates as a Last-In-First-Out (LIFO) data structure.
• When a function is called, a new stack frame is created to store local variables and function parameters.
• When the function exits, the stack frame is removed.
• Example: In a text editor, when you open a new document, a stack frame is created to manage the document's editing session, including the cursor position and undo/redo history

Process State
A process can be in various states, including:
•
New - The process is being created.
•
Ready - The process is waiting to be assigned to a processor.

• **Running** - Instructions are being executed.
•
Waiting - The process is waiting for some event to occur(such as an I/O completion or reception of a signal).
•
Terminated - The process has finished execution

Process Control Block (PCB): is a data structure that is used to store the information of a process that might be needed to manage the scheduling of a particular process.

Each process is associated with a data structure known as a PCB, which contains information about the process, such as:
• program counter,
• registers, and
• other essential details
• This allows the operating system to manage and switch between processes efficiently.

Components of PCB:

Process ID: is a unique identifier assigned to each process in the system. It distinguishes one process from another. The PID helps the operating system keep track of and manage processes efficiently.

Program Counter (PC): The Program Counter is a register that stores the address of the next instruction to be executed by the process. It allows the operating system to save and restore the state of the process during context switches, ensuring that the process can continue from where it left off.

CPU Registers: CPU Registers store the current values of various CPU registers, including the Accumulator, Index Register, and Stack Pointer. Saving and restoring these values is crucial during context switches to maintain the process's state.

Scheduling Information: This component contains details related to process scheduling, such as the process's priority, scheduling state, and time quantum if it's a pre-emptive scheduling system. It helps the operating system determine when and how long a process can execute.

Memory Management Information: This section includes information about the memory allocated to the process, such as the base and limit registers, which define the memory boundaries for the process.• It helps prevent processes from accessing each other's memory.

Accounting Information: Accounting information keeps track of resource usage, like CPU time, clock time, and the number of I/O operations performed by the process.

I/O Status Information: I/O Status Information maintains the status of open files, devices, and any pending I/O requests associated with the process.

inter-Process Communication (IPC) Information: If the process communicates with other processes, this part of the PCB contains details about shared memory segments, message queues, or other IPC mechanisms being used by the process.

State of Process: This field indicates the current state of the process, such as "Running," "Ready," or "Blocked." It helps the operating system manage and prioritize processes for execution.

Pointer to Parent Process: If a process creates another process (e.g., a child process), this field contains a reference to the parent process.• It's useful for process hierarchy and management.

Pointer to Next PCB: This is essential for efficient process scheduling and management.

Process Scheduling:

refers to the management of multiple processes competing for CPU time.
• It's a critical component of multitasking and multiprogramming environments, ensuring that the CPU is efficiently utilized while providing fairness, responsiveness, and the ability to meet various process requirements.

The Goal of Process Scheduling

- The primary goal of process scheduling is to maximize CPU utilization while minimizing response time and ensuring fairness among processes.
- The scheduler aims to make the best use of available CPU resources.

The Scheduling Policies:

Different scheduling policies determine the order in which processes are selected to run.
• Common scheduling policies include:
• First-Come, First-Served (FCFS)
• Shortest Job Next (SJN) - also known as Shortest Job First (SJF)
• Round Robin
• Priority-based
• Multilevel Feedback Queues (MLFQ)

Topic 3: Threads

Definition

thread: is the smallest unit of execution within a process

It represents an independent flow of control and shares the same resources with other threads belonging to the same process

Threads enable concurrent execution, allowing multiple tasks to be performed simultaneously

Components of a Thread

1. **Thread ID (TID)**: Unique identifier for each thread.
2. **Program Counter (PC)**: Tracks the address of the next instruction.
3. **Register Set**: Contains general-purpose registers and the stack pointer.
4. **Stack**: Stores local variables, function parameters, and return addresses.
5. **Thread State**: Indicates the current status (running, ready, blocked).
6. **Thread Priority**: Determines execution order when multiple threads are ready.
7. **Thread-specific Data**: Unique data for each thread.
8. **Thread Control Block (TCB)**: Data structure containing thread information.
9. **Thread Local Storage (TLS)**: Unique data storage for each thread.
10. **Execution Context**: Information needed to restore the thread's state.

Multithreading is commonly used to achieve:

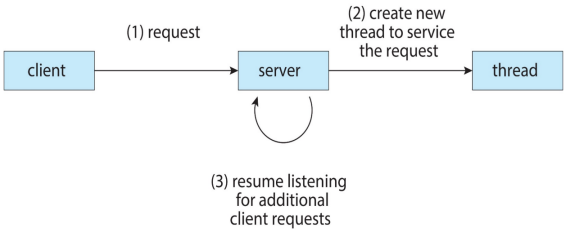
• **Parallelism** - Multithreading enables parallelism by allowing different threads to execute different parts of a program simultaneously. This is particularly beneficial in scenarios where tasks can be performed concurrently.
•
Improve system responsiveness - Multithreading allows multiple threads to execute independently within the same program.
•
Efficiently utilize resources - Threads within a process share the same resources, including memory space. This allows for more efficient communication and data sharing between threads.

Single thread vs multithread

- **Single-threaded**: Executes one task at a time.
- **Multithreaded**: Executes multiple tasks concurrently, improving efficiency and responsiveness.

Multithread Server Architecture:

- A multithread server architecture is a design approach in which a server application uses multiple threads to handle multiple client requests concurrently.
- Each client request is assigned to a separate thread, allowing the server to process multiple requests simultaneously.
- This architecture is commonly employed in networked applications, such as web servers, where responsiveness and the ability to handle multiple clients concurrently are crucial.



Objectives of Multithread Server Architecture

• **Concurrency:** Handle multiple client requests at the same time.

• **Responsiveness:** Improve responsiveness by avoiding delays caused by processing one request before moving on to the next.

• **Resource Utilization:** Efficiently use system resources by distributing tasks among multiple threads.

• **Scalability:** Scale the server to accommodate an increasing number of clients without a proportional decrease in performance.

How it works:

1. **Client Requests :** Multiple clients send requests to the web server simultaneously.
2. **Thread Allocation:** The server, using a thread pool, assigns a separate thread to handle each incoming client request.
3. **Concurrent Execution:** Threads operate concurrently, processing their assigned client requests independently.
4. **Responsiveness:** Since each client request has its dedicated thread, the server can respond promptly to multiple clients, improving overall system responsiveness.

5. **Thread Recycling:** Once a thread has completed processing a client request, it is returned to the thread pool, making it available to handle another incoming request.

Types of thread:

- **User thread:** Managed by the application and not the kernel. These are lightweight and offer more control to the programmer.
- **Kernel thread:** Managed by the operating system, providing better parallelism but with higher overhead.

Multithreading Models

1. **Many-to-One**: Many user threads mapped to a single kernel thread.
2. **One-to-One**: Each user thread corresponds to one kernel thread.
3. **Many-to-Many**: Many user threads mapped to a smaller or equal number of kernel threads.

Benefits of Threads

1. **Responsiveness**: Continues execution even if part of the process is blocked.
2. **Resource Sharing**: Easier sharing of memory and data within the same process.
3. **Cost**: Cheaper than process creation, lower overhead for thread switching.
4. **Scalability**: Utilizes multi-core architectures effectively.

Topic 4: CONCURRENCY

Introduction to Concurrency

- **Concurrency**: ability of the system to execute multiple independent tasks or processes simultaneously.
- **Importance**: Optimizes system performance and resource utilization.
- **Key Concepts related to concurrency**: Processes, threads, and synchronization.

Processes and Threads

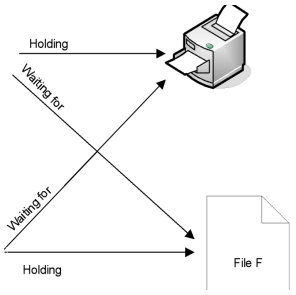
- **Process**: An independent program or task being executed.
- **Thread**: The smallest unit of execution within a process.
- **Synchronization**: Manages access to shared resources to prevent conflicts and ensure data consistency.

Mutual Exclusion

- **Definition**: Ensures only one process/thread accesses a critical section or shared resource at a time.
- **Importance**: avoid conflicting the other process assigned to the same resource at the same time.
- **Example**: Without mutual exclusion, if the user with Process 1 and Process 2 accessing the resource (printer) concurrently, the user will probably get a half-printed page of Microsoft words, and half printed page of Microsoft Excel.
- Mutual exclusion gives rise to another problem which OS must handle – deadlock.

Deadlock

- **Definition**: A situation where two or more processes are unable to proceed because each is waiting for the other to release a resource.
- **Example**:
  - Process P1 has a printer allocated and is attempting to open a file F.
  - Process P2 has already obtained exclusive access to file F and now requests printer.
  - Each process is waiting for a resource held by the other, while holding a resource required by the other.



- It's a state of circular waiting.
- Several Conditions that Cause Deadlocks

Condition	Explanation
Mutual Exclusion	A resource is assigned to only one process at a time
Hold and Wait	A situation whereby process own a resource and claiming the other resource.
No pre-emption	A situation whereby a resource cannot been taken from the process unless the process release the resource.
Circular Wait	A situation whereby each waiting process is waiting for the next process in the chain and none is able to get the resource.

Deadlock Handling

1. **Prevention**: Ensure that at least one of the necessary conditions for deadlock cannot hold.
2. **Avoidance**: Allocate resources only if it leads to a safe state.
3. **Detection**: Identify deadlock after it occurs and recover.

Starvation

- **Definition**: process does not get the resources it needs for a long time because the resources are being allocated to other processes.
- **Cause**: Can be due to deadlock or high-priority processes monopolizing resources.
- **Solution**: Aging – technique of gradually increasing the priority of processes that wait in the system for a long time

Issues with concurrency

Race conditions: Occur when multiple processes or threads try to access and modify shared data simultaneously, leading to unpredictable results.

Deadlocks: Happen when two or more processes are unable to proceed because each is waiting for the other to release a resource.

Overcoming Concurrency Issues

1. **Locks**: Provide mutual exclusion by allowing only one thread to access a critical section at a time.
2. **Atomic Operations**: Ensure operations are executed as a single, uninterruptible unit.
3. **Semaphores**: Control access by limit the number of threads accessing a resource simultaneously.
4. **Transactional Memory**: Execute a series of operations atomically, with rollback on failure

Topic 5: Memory management

Introduction to Computer Memory

Computer memory is a fundamental component that allows
a computer to store and retrieve data quickly. it comes in two types

- **RAM (Random Access Memory)**: Volatile memory used for temporary storage during processing.
- **ROM (Read-Only Memory)**: Non-volatile memory that retains data even when power is off.

Von Neumann Architecture

- This architecture implies that for any operation, including data manipulation, the data must be brought into the main memory first.

- Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

So how does it works?

- The storage drive permanently stores programs and files.
- When data is requested, for example you open a file, the processor accesses the storage drive and transfers long-term data (the stored file) to the memory for short-term access.
- As you work on a file or switch between programs, the processor may access data from memory if it has already been transferred from the storage drive.
- The goal is to enhance the system's performance by keeping frequently used data in the faster, volatile memory (RAM) for quicker access.

Memory Management

- **Definition**: The process of controlling and coordinating computer memory.
- **Goals**: optimize the use of memory resources to ensure efficient program execution.

Importance of Memory Management

1. **Optimal Resource Utilization**: Allocates and frees memory space as needed.
2. **Prevention of Memory Conflicts**: Allocates separate memory regions to prevent interference between processes.
3. **Dynamic Memory Allocation**: Allows programs to request and release memory dynamically.
4. **Avoidance of Fragmentation**: Uses methods like paging and segmentation to mitigate fragmentation issues.
5. **Prevention of System Crashes**: Ensures processes receive necessary memory resources, preventing system instability.
6. **Virtual Memory Support**: Combines physical RAM and disk space to simulate larger memory sizes.
7. **Enhanced System Performance**: Optimizes memory usage to run multiple processes concurrently.
8. **Security and Isolation**: Segregates memory spaces to prevent unauthorized access.

Memory Management Methods

1. **Contiguous Memory Allocation**: Allocates a single block of memory to a process. It’s simple, but can lead to fragmentation.
2. **Non-contiguous Memory Allocation**: Allocates scattered memory blocks to a process.
3. **Swapping**: Temporarily moves processes between main memory and disk.
4. **Paging**: Divides memory into fixed-size pages, reducing external fragmentation.
5. **Segmentation**: Divides logical address space into variable-sized segments.

Fragmentation

Fragmentation occurs as processes are stored and removed from memory; it will create free memory space which are too small to use by other processes.

- **Internal Fragmentation**: Occurs when allocated memory is larger than needed.
- **External Fragmentation**: Occurs when free memory is not contiguous.
- **Compaction**: Compaction is a technique used to address external fragmentation in dynamic partitioning.

Partition Allocation

1. **Fixed Partitioning**: Divides memory into fixed-size partitions; simple but can lead to inefficient use.
2. **Dynamic Partitioning**: Allocates and deallocates memory as needed, reducing fragmentation.

Partition Allocation Algorithms

1. **First Fit**: Allocates the first available partition that fits the process.
2. **Best Fit**: Allocates the smallest partition that fits the process.
3. **Worst Fit**: Allocates the largest available partition, leaving the largest free space.

Topic 6: Virtual Memory

Introduction to Virtual Memory

- **Definition**: Virtual memory is a memory management technique that uses secondary storage (like a hard drive) as an extension of the main memory (RAM).
- **Purpose**: It helps computers handle a shortage of physical memory by temporarily moving data from RAM to disk storage.

Importance of Virtual Memory

- **Overcoming Physical Memory Limitations**: Allows running applications even if they can't all fit into physical RAM simultaneously.
- **Running Multiple Applications**: Enables efficient multitasking.
- **Enabling Larger Programs**: Supports memory-intensive processes.
- **Enhancing System Stability**: Improves overall system performance and stability.
- **Efficient Resource Management**: Optimizes the use of memory resources.

How Virtual Memory Works

To enable the seamless transfer of virtual memory content to physical memory, the operating system employs a process known as **paging**.

- **Paging**: Virtual memory is partitioned into fixed-size segments called pages. Pages are moved from disk storage to main memory as needed.

When a particular page is needed, the operating system orchestrates its migration from disk to the main memory (RAM). the operating system performs a crucial task known as **address translation**.

- **Address Translation**: The operating system translates virtual addresses to physical addresses during page transfer.

Memory Management Unit (MMU)

- **Role**: The MMU is a **hardware component** that work in conjunction with the operating system to translates virtual addresses into physical addresses.
- **Function**: Works with the operating system to manage memory effectively and ensure seamless program operation.
- **How it works:** Use a section of memory to translate virtual addresses into physical addresses via a **series of table lookups.**

Segmentation

- **Definition**: is a memory management method where a program's address space is divided into logical segments, each with a specific purpose.
- **Advantages**: Organizes and secures memory space.
- **Issues (fragmentation)**:
  - Since segmentation relies on memory that is located in single large blocks, it is highly possible that enough free space is available to load a new module but can not be utilized – external fragmentation.
  - Segmentation may also suffer from internal fragmentation if segments are not variable-sized, where memory above the segment is not used by the program but is still “reserved” for it.

Paging

- **Definition**: is a memory management technique that involves dividing physical memory and virtual memory into fixed size blocks called pages.
- **Advantages**: Reduces external fragmentation and provides an easier interface for programs.
- **How it Works: Each unit of transfer**, referred to as a **page**, is of a fixed size and swapped by the virtual memory manager outside of the program’s control. Instead of utilizing a segment addressing approach, as seen in segmentation, paging uses a **linear sequence** of virtual addresses which are mapped to physical memory as necessary.
- **Internal Fragmentation**: May occur due to fixed page sizes.

Page Fault

- **Definition**: Occurs when a program accesses a page not currently in main memory, triggering the OS to load the required page from secondary storage.

- **Mechanism**: Signals the OS to bring the needed page into RAM, allowing the program to continue execution.

Page Replacement Algorithms

- **FIFO (First In, First Out)**: Replaces the oldest page first.
- **LRU (Least Recently Used)**: Replaces the least recently used page.
- **LFU (Least Frequently Used)**: Replaces the page used least often in the past.
- **OPT (Optimal)**: Replaces the page that will not be used for the longest time in the future.

Additional Methods for Managing Virtual Memory

- **Demand Paging**: Loads pages into memory only when needed, reducing initial loading time.
- **Memory-Mapped Files**: Maps virtual memory directly to files on disk, facilitating efficient data sharing between processes.

Topic 7: Uniprocessor Scheduling

Introduction

- **Purpose**: Allocate computer resources among competing processes using scheduling techniques.
- **Objectives**: Ensure fairness, prevent starvation, maximize processor efficiency, and maintain low overhead.

Aim of Scheduling

- Assign processes to be executed by the processor over time, focusing on:
  - Response time
  - Throughput
  - Processor efficiency

Scheduling Criteria

1. **CPU Utilization**: Maximize CPU usage for better system performance.
2. **Throughput**: Increase the number of processes completed per unit time.
3. **Turnaround Time**: Minimize the total time taken to execute a process.
4. **Waiting Time**: Reduce the time a process spends in the ready queue.
5. **Response Time**: Shorten the time from request submission to the first response.

Types of Scheduling

1. **Long-term Scheduling**: Determines if a new process should be created and added to the process pool.

2. **Medium-term Scheduling**: Part of the swapping function, decides if a process should be loaded into main memory.
3. **Short-term Scheduling**: Decides which ready process to execute next, also known as dispatching.

Scheduling Algorithms

1. **First-Come-First-Served (FCFS)**
   - Simple and non-preemptive.
   - Processes are executed in the order they arrive.
   - **Advantages**: Simple, fair for batch processing.
   - **Disadvantages**: High waiting and turnaround times, especially for short processes.
2. **Shortest Job First (SJF)**
   - Executes processes with the shortest burst time first.
   - **Advantages**: Minimizes average waiting and turnaround times.
   - **Disadvantages**: Can lead to starvation, burst time not always known in advance.
3. **Longest Job First (LJF)**
   - Opposite of SJF, prioritizes longest burst time.
   - **Advantages**: Ensures long jobs get done.
   - **Disadvantages**: High waiting and turnaround times, short jobs may starve.
4. **Priority Scheduling**
   - Processes are executed based on priority levels.
   - **Advantages**: High-priority processes are executed quickly.
   - **Disadvantages**: Can lead to starvation of low-priority processes.
5. **Round Robin (RR)**
   - Processes are executed in a cyclic order for a fixed time slice.
   - **Advantages**: Fair, prevents starvation, good for time-sharing systems.
   - **Disadvantages**: Context switching overhead, not suitable for processes with varying execution times.

Key Concepts

- **Waiting Time (WT)**: Total time a process waits in the ready queue.
- **Service Time/Burst Time (ST/BT)**: Time required to complete a process.
- **Turnaround Time (TAT)**: Total time from submission to completion of a process.

Topic 8: Multiprocessor and Real-Time Scheduling

Introduction

- **Multiprocessing OS**: Utilizes more than one CPU to improve performance by parallelizing tasks. Common examples include Unix, Linux, and Solaris.
- **Multiprocessor System**: Consists of multiple CPUs that operate concurrently, dividing tasks for parallel execution.
- **Uniprocessor System**: Has a single CPU that executes instructions sequentially, leading to potential bottlenecks.

Multiprocessor vs. Uniprocessor

- **Performance**: Multiprocessors offer enhanced performance by parallelizing tasks, while uniprocessors are limited by a single CPU's capabilities.
- **Resource Utilization**: Multiprocessors allow better resource distribution and concurrent task execution, reducing idle time. Uniprocessors have limited resource utilization.
- **Fault Tolerance**: Multiprocessors have improved fault tolerance by rerouting tasks to functioning processors in case of failure. Uniprocessors have a single point of failure.
- **Examples**: High-performance servers and supercomputers (multiprocessors) vs. personal computers and laptops (uniprocessors).

Multiprocessor Scheduling

- **How It Works**: Divides tasks among multiple processors, allowing simultaneous processing to reduce overall completion time.
- **Classifications**:
  1. **Symmetric Multiprocessing (SMP)**:
     - All processors share memory and I/O bus.
     - Each processor executes the same OS copy and makes independent decisions.
     - **Advantages**: Failure of one processor does not affect others; efficient resource utilization.
     - **Disadvantages**: Complex, costly, difficult synchronization.
  2. **Asymmetric Multiprocessing (AMP)**:
     - One master processor controls, others are slaves.
     - **Advantages**: Cost-effective, easy to design and manage, scalable.
     - **Disadvantages**: Uneven workload distribution, single point of failure.
  3. **Processor Affinity**: Assigns specific processors to tasks for optimized performance.
     - **Example**: Ensuring a program runs on a specific core for efficiency.

Scheduling Techniques

- **Load Balancing**: Distributes tasks evenly across processors to prevent overload.
- **Gang Scheduling**: Schedules a group of related processes to run simultaneously on multiple processors, ensuring synchronized progress.

Real-Time Systems

- **Definition**: Processes information and responds to events within a specified time frame.
- **Types**:
  1. **Hard Real-Time Systems**: Must meet deadlines, failure leads to catastrophic consequences (e.g., medical devices, aerospace).
  2. **Soft Real-Time Systems**: Timely response is important but flexible (e.g., multimedia systems).
- **Characteristics**: Strict time constraints, predictability, concurrency, reliability, efficient task scheduling, fault tolerance.

Real-Time Scheduling

- **Importance**: Manages tasks to meet specific deadlines.
- **Types**:
  1. **Static Scheduling**: Fixed priorities and deadlines assigned during system design.
  2. **Dynamic Scheduling**: Priorities and deadlines assigned dynamically based on current system state.
- **Algorithms**:
  1. **Static Table-Driven Approaches**: Optimal order of task execution determined in advance.
  2. **Static Priority-Driven Preemptive Approaches**: Assigns fixed priorities to tasks.
  3. **Dynamic Planning-Based Approaches**: Identifies feasible schedules dynamically.
  4. **Dynamic Best Effort Approaches**: Focuses on meeting deadlines, aborting tasks if deadlines are missed.

Topic 9: Input/Output Management & Disk Scheduling

Introduction

- **I/O (Input/Output)**: Refers to communication between the computer and external devices.
- **OS Role**: Manages I/O to enhance reliability and performance using device drivers and controllers.

I/O Request Handling

- **Components**:
  - **I/O Traffic Controller**: Manages paths for I/O operations.
  - **I/O Scheduler (Disk Scheduler)**: Decides the order of I/O request service.
  - **I/O Device Handler**: Manages communication between CPU and peripherals.

I/O Traffic Controller

- **Functions**:
  - Checks for available paths.
  - Selects optimal paths based on algorithms.
  - Manages path availability and efficiency.

I/O Scheduler (Disk Scheduler)

- **Functions**:
  - Allocates devices and control units.
  - Decides the service order under heavy I/O loads.
  - Works with the traffic controller to track and serve I/O requests.

I/O Device Handler

- **Role**: Facilitates data transfer between CPU and external devices, handling interrupts and scheduling algorithms.

I/O Handling Algorithms

1. **First-Come-First-Serve (FCFS)**: Services requests in arrival order.
2. **Shortest Seek Time First (SSTF)**: Prioritizes requests with minimal disk arm movement.
3. **SCAN (Elevator) Algorithm**: Moves the disk arm in one direction, servicing requests until the end, then reverses.
4. **Circular LOOK**: Similar to SCAN but only goes as far as the last request before reversing.
5. **Deadline Monotonic Scheduling**: Prioritizes requests based on deadlines.
6. **Round Robin**: Services requests in a circular order, each for a fixed time slice.

Redundant Array of Independent Disks (RAID)

- **Purpose**: Improves data reliability and performance through redundancy and parallelism.
- **Standard RAID Levels**:
  - **RAID 0 (Striping)**: Distributes data across multiple disks without redundancy.
  - **RAID 1 (Mirroring)**: Duplicates data across two or more disks for redundancy.
  - **RAID 2**: Bit-level striping with dedicated Hamming-code parity (rarely used today).
  - **RAID 5**: Combines striping with parity for fault tolerance and performance.
- **Non-Standard RAID Levels**:
  - **RAID-DP**: Developed by NetApp, includes double parity.
  - **RAID 10**: Combines mirroring (RAID 1) and striping (RAID 0).
  - **RAID-Z**: Designed for ZFS, similar to RAID 5 with enhancements.
- **Nested/Hybrid RAID Levels**: Combine two or more RAID levels for better performance and redundancy.
  - **RAID 50**: Combines RAID 5 and RAID 0.

- **RAID 60**: Combines RAID 6 and RAID 0.

Topic 10: File Management

Introduction to File Management

- **Definition**: Critical component of OS that governs how data is stored, organized, and accessed.
- **Functions**: Ensure data integrity, security, and ease of use, manage file attributes, control access, and provide interaction methods for users and applications.

Objectives of File Management

1. **File Organization**: Logical and efficient organization of files and folders for easy access.
2. **Data Security**: Protect files from unauthorized access and provide data recovery mechanisms.
3. **Data Sharing**: Enable multiple users to access and edit the same file simultaneously.
4. **File Backup**: Create copies of important files to prevent data loss.
5. **File Compression**: Reduce file size to save disk space and ease transfer.
6. **Space Management**: Efficiently allocate and deallocate storage space.

Types of File Structures

1. **Text File**: Non-executable file containing sequences of numbers, symbols, and letters.
2. **Source File**: Executable file containing a series of functions and processes.
3. **Object File**: File containing object codes in assembly or machine language.

Types of File Systems

1. **FAT (File Allocation Table)**: An older file system used by older versions of Windows and other operating systems.
2. **NTFS (New Technology File System)**: A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.
3. **ext (Extended File System)**: A file system commonly used on Linux and Unix- based operating systems.
4. **HFS (Hierarchical File System)**: A file system used by macOS.
5. **APFS (Apple File System)**: A new file system introduced by Apple for their Macs and iOS devices.

Properties of File Management

- Files are organized in a hierarchical structure similar to a tree.
- Hierarchical file systems have a special directory at the root.

File Access Permissions

- **Linux-Based OS Permissions**:

- Three permission groups: owner, group, and others.
- File types indicated by the first character: '-' for file, 'd' for directory.
- Permissions: 'r' for read, 'w' for write, 'x' for execute, '-' for no permission.

File Directory

- **Definition**: Collection of files containing information about the files, including attributes, location, and ownership.
- **Directory Operations**:
  - Search for a file
  - Create, delete, and rename a file
  - List directory contents
  - Traverse the file system

File Access Methods

1. **Sequential Access**: Access data one record after another.
2. **Direct Access**: Read and write records rapidly in no particular order.
3. **Index Access**: Uses an index to map logical keys to physical addresses, enabling quick data access.

File Allocation Methods

1. **Continuous Allocation**: Allocates a single set of contiguous blocks at file creation.
   - **Advantages**: Supports sequential and direct access, minimal seeks.
   - **Disadvantages**: Internal and external fragmentation, difficult to increase file size.
2. **Linked Allocation (Non-contiguous allocation)**: Each block contains a pointer to the next block.
   - **Advantages**: Flexible file size, no external fragmentation.
   - **Disadvantages**: Slower access, no direct access, overhead of pointers.
3. **Indexed Allocation**: Uses a special index block containing pointers to all file blocks.
   - **Advantages**: Supports direct access, no external fragmentation.
   - **Disadvantages**: Greater pointer overhead, inefficient for very small files.

Topic 11: OS security and protection

Introduction

- OS security is crucial for safeguarding integrity, confidentiality, and availability.
- The OS manages hardware resources and supports all software applications, making it a target for cyber threats.

Cyber Threats & OS

Operating Systems are targets for several reasons:

- **Centralized Control:** OS manages critical system functions.
- **Boot Process Manipulation:** OS boot process can be exploited.
- **Access to Resources:** OS controls access to system resources.
- **User Authentication:** OS handles user authentication processes.
- **Privilege Escalation:** Exploiting vulnerabilities to gain higher privileges.
- **Network Communication:** OS manages network interactions.
- **Economic Incentives:** Attacking OS can lead to financial gain.
- **Software Compatibility:** Vulnerabilities in software compatibility.

Intruders & OS

Common infiltration methods:

- **Drive-By Downloads:** Automatic download of malicious software.
- **Exploiting Software Vulnerabilities:** Attacking software flaws.
- **Zero-Day Exploits:** Exploiting unknown vulnerabilities.
- **Social Engineering:** Manipulating people into divulging information.
- **Remote Access Tools (RATs):** Gaining remote control.
- **Unsecured Network Connections:** Exploiting unsecured networks.

Malicious Software

- Malicious software, commonly known as malware, refers to any software intentionally designed to cause harm, exploit vulnerabilities, or perform malicious actions on a computer system, network, or device.
- Malware can take various forms, each serving a specific purpose for the attacker.
- Forms of malware include:
  - **Adware and Browser Hijackers**
  - **Rootkits and Kernel-level Attacks**
  - **Trojans and Backdoors**
  - **Ransomware**
  - **Worms and Self-Replicating Malware**
  - **Spyware and Keyloggers**

Access Control

- Access control is the process of protecting a resources so that it is used only by those allowed to use it
- **Discretionary Access Control (DAC)**: the owner of the resource decides who gets in and changes permission as needed. The owner can give that job to others.

- **Mandatory Access Control (MAC)**: permission to access a system or any resource is determined by the sensitivity of the resource and the security level of the subject. It cannot be given to someone else. This makes MAC stronger than DAC.
- **Non-Discretionary Access Control**: are closely monitored by the security administrator and not the system administrator
- **Rule-based Access Control**: is maintained by the data owner based on a list of rules which determines access of a user towards objects.
- **Content-dependent Access Control**: Access based on data content.

Hardening Operating Systems

- Hardening an operating system refers to the **process of securing and configuring** it to

  reduce vulnerabilities and **enhance overall security.**
- The goal of OS hardening is to minimize the potential attack surface and make it more resistant to unauthorized access, exploitation, and other security threats.
- **Techniques**:
  1. **Security Updates and Patch Management**: Regularly applying updates.
  2. **User Account Management**: Limiting privileges and implementing strong passwords.
  3. **Network Configuration**: Configuring firewalls, disabling unnecessary services, restricting network access.
  4. **File System Security**: Applying permissions and encryption.
  5. **Application Whitelisting**: Restricting execution to predefined applications.
  6. **Audit Logging and Monitoring**: Enabling logs and monitoring activities.
  7. **Disabling Unnecessary Services and Features**: Reducing the attack surface.
  8. **Security Policies and Group Policies**: Standardizing security settings.

Data Backup & Archive

- **Data Backup**: Data backup involves **creating copies of essential files, databases,** or **entire systems** to protect **against data loss** due to various reasons, such as hardware failure, accidental deletion, or cyberattacks.
  - The primary purpose of data backup is to provide a **redundant copy** of data that can be used to restore the original data in case of a loss or corruption.
  - Backups are typically **performed regularly**, with the frequency depending on the organization's data retention policies and the criticality of the data.
- **Data Archive**: Data archiving is the process of **moving data that is no longer actively used** to a **separate storage system** for long-term retention.
  - Archived data is often **retained for historical**, regulatory, or compliance purposes.
  - The main purpose of data archiving is to **free up space** on primary storage systems, **improve performance**, and ensure that important historical data is preserved for legal, regulatory, or business reasons.

    - Archiving is generally done **less frequently than backups** and typically involves data that is not frequently accessed but still needs to be retained.

- **Relation to OS**:
    - Disaster recovery.
    - Storage management.
    - Efficient resource usage.
    - Compliance with regulations.
    - Security through backups.
    - System state and full system backups.

Definitions Collection

Operating System (OS)

Operating System (OS): A software that serves as an intermediary between computer hardware and the computer user, providing a user interface and managing computer hardware and software resources.

Process

Process: A fundamental concept representing an independent program or task being executed, consisting of the program's code, data, and various system resources it requires to function.

Thread

Thread: The smallest unit of execution within a process, representing an independent flow of control and sharing the same resources with other threads belonging to the same process.

Concurrency

Concurrency: The ability of a system to execute multiple independent tasks or processes simultaneously, optimizing system performance and resource utilization.

Mutual Exclusion

Mutual Exclusion: Ensures that only one process/thread accesses a critical section or shared resource at a time to avoid conflicts and ensure data consistency.

Deadlock

Deadlock: A situation where two or more processes are unable to proceed because each is waiting for the other to release a resource.

Starvation

Starvation: A condition where a process does not get the resources it needs for a long time because the resources are being allocated to other processes.

Memory Management

Memory Management: The process of controlling and coordinating computer memory to optimize resource utilization, prevent conflicts, and enhance system performance.

Fragmentation

Fragmentation: The condition where memory is divided into small, non-contiguous blocks, making it difficult to allocate large contiguous blocks of memory.

Paging

Paging: A memory management technique that involves dividing physical memory and virtual memory into fixed-size blocks called pages, reducing external fragmentation.

Virtual Memory

Virtual Memory: A memory management technique that uses secondary storage (like a hard drive) as an extension of the main memory (RAM), enabling efficient multitasking and handling of large programs.

Process Control Block (PCB)

Process Control Block (PCB): A data structure used to store information about a process, including program counter, registers, memory management information, and scheduling information.

File Management

File Management: A critical component of the OS that governs how data is stored, organized, and accessed, ensuring data integrity, security, and ease of use.

Data Backup

Data Backup: Creating copies of essential files, databases, or entire systems to protect against data loss due to hardware failure, accidental deletion, or cyberattacks.

Data Archive

Data Archive: The process of moving data that is no longer actively used to a separate storage system for long-term retention, often for historical, regulatory, or compliance purposes.

Access Control

Access Control: The process of protecting resources so that they are used only by those allowed to use them, involving methods like Discretionary Access Control (DAC), Mandatory Access Control (MAC), Non-Discretionary Access Control, and Rule-based Access Control.

Hardening Operating System

Hardening Operating System: Securing and configuring an operating system to reduce vulnerabilities and enhance overall security by implementing various security measures, configurations, and best practices.

RAID (Redundant Array of Independent Disks)

RAID: A technology that improves data reliability and performance through redundancy and parallelism, with levels such as RAID 0 (striping), RAID 1 (mirroring), RAID 5 (striping with parity),

and others.

User Interface (UI)

User Interface (UI): A means through which a user interacts with a computer system, software application, or electronic device, acting as a bridge between the user and the underlying technology. Types include Graphical User Interface (GUI), Command Line Interface (CLI), Voice User Interface (VUI), Touchscreen, and Web User Interface.

Process Scheduling

Process Scheduling: The management of multiple processes competing for CPU time, aiming to maximize CPU utilization, minimize response time, and ensure fairness among processes.

Memory Management Unit (MMU)

Memory Management Unit (MMU): A hardware component that translates virtual addresses into physical addresses, working with the OS to manage memory effectively and ensure seamless program operation.

Segmentation

Segmentation: A memory management method where a program's address space is divided into logical segments, each with a specific purpose.

Page Fault

Page Fault: Occurs when a program accesses a page not currently in main memory, triggering the OS to load the required page from secondary storage.

Scheduling Algorithms

Scheduling Algorithms: Techniques used to determine the order of process execution. Examples include First-Come-First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling, Round Robin (RR), and others.