Introduction to Operating Systems

DSC 315: Computer Organization & Operating Systems

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

March 06, 2026

1. Introduction to Operating Systems
   - Computer-System Organization
   - Interrupts
   - I/O Structure
   - Storage Structure
   - Computer-System Architecture
   - Clustered Systems
   - Operating-System Operations
   - Resource Management
   - Process Management
   - Memory Management
   - File-System Management
   - Security and Protection
   - Distributed Systems
   - Kernel Data Structures

IISER THIRUVANANTHAPURAM

Introduction to Operating Systems

What is operating system?

- An operating system $\rightarrow$ a software that **manages** a computer's hardware
- acts as an **intermediary** $\rightarrow$ between the computer **user** and the computer **hardware**.
- fundamental responsibility $\rightarrow$ allocate shared resources to programs.

What does it do?

- performs no useful function by itself. but provides an environment within which other programs can do useful work.

- Users' Point of view :$\rightarrow$ Ease of Use with Performance and Security
- Systems Point of view :$\rightarrow$ Resource Controller and Allocator

Computer System Structure

Computer system can be divided into four components:

1. **Hardware** – provides basic computing resources
   - CPU, memory, I/O devices
2. **Operating system**
   - Controls and coordinates use of hardware among various applications and users
3. **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
   - Word processors, compilers, web browsers, database systems, video games
4. **Users**
   - People, machines, other computers

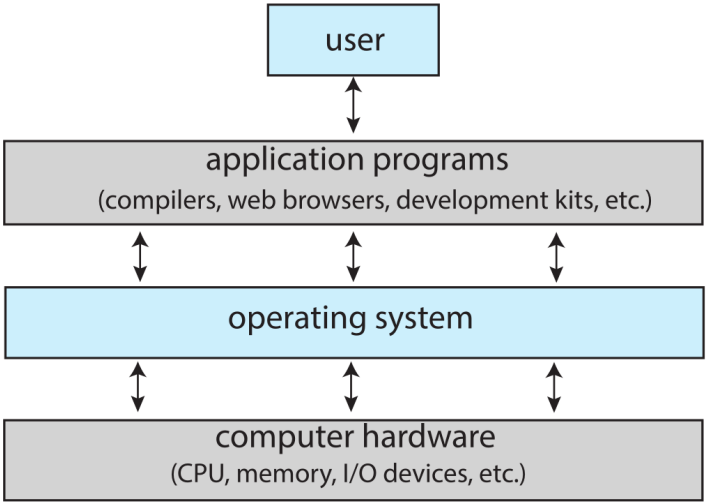

Figure 1.1 Abstract view of the components of a computer system.

Operating System

- No universally accepted definition
- **Practical view**
  - "Everything a vendor ships when you order an operating system" –>In practice this varies widely

Definition.

The common functions of controlling and allocating resources are then brought together into one piece of software:

- **Kernel**
  - The program that runs at all times on the computer
  - Core part of the operating system
- **Other software**
  - **System programs**: shipped with the OS but not part of the kernel
  - **Application programs**: programs not associated with the OS
- **Middleware**
  - Software frameworks providing services to applications
  - Examples: databases, multimedia, graphics

Why Learn?

- to proper, efficient, effective, and secure programming(as almost all code runs on top of an OS)

What Operating Systems Do

- **Depends on the point of view**
- **Users**
  - Want convenience, ease of use, and good performance
  - Usually do not care about resource utilization
- **Shared systems (mainframe / minicomputer)**
  - Must keep all users satisfied
  - OS acts as a **resource allocator and control program**
  - Ensures efficient hardware use and manages execution of user programs
- **Dedicated systems (workstations)**
  - Have dedicated resources
  - Often access shared resources from servers
- **Mobile devices (smartphones, tablets)**
  - Limited resources
  - Optimized for usability and battery life
  - Interfaces: touch screens, voice recognition
- **Embedded computers**
  - Little or no user interface
  - Operate mostly without user intervention

IISER THIRUVANANTHAPURAM

Computer-System Organization

Computer-System Organization

1. computer system → multiple CPUs & device controllers → connected through a bus
2. bus provides access between components and shared memory
3. operating systems have a **device driver** for each **device controller**.
4. The CPU and the device controllers can execute in parallel,

How operates? Key Structures

- Interrupts
- Storage structure
- I/O structure

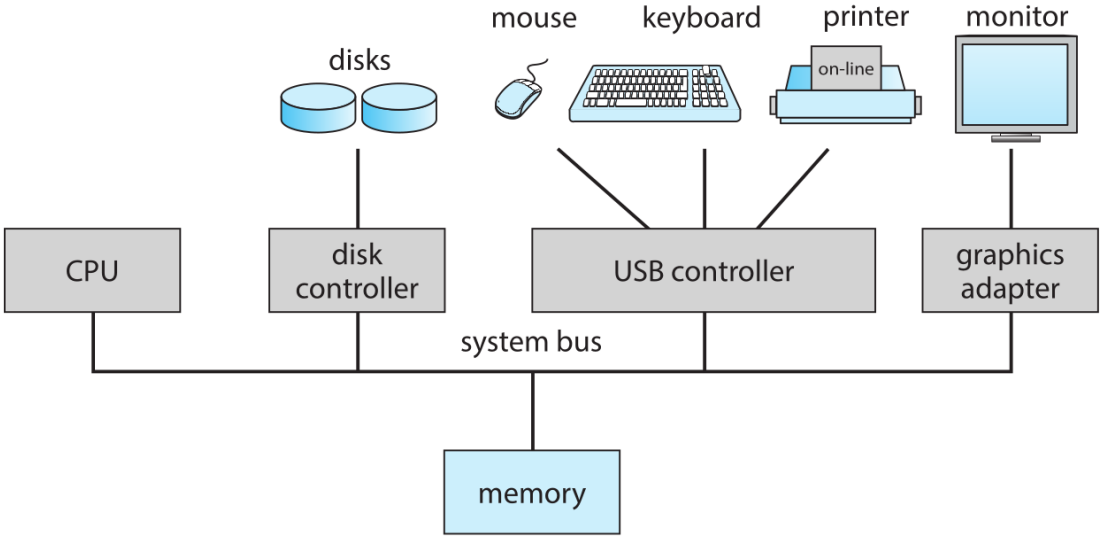

Figure 1.2 A typical PC computer system.

Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller manages a particular device type
- Each device controller has a local buffer
- Each controller type has an **OS device driver** to manage it
- **Data movement**
  - CPU moves data between main memory and controller buffers
  - I/O transfers data between the device and the controller's local buffer
- **Interrupt**
  - Device controller signals completion by generating an interrupt to the CPU

Interrupts

Definition

An **interrupt** in an **Operating System (OS)** is a `signal` that temporarily stops the CPU's current task so it can immediately handle a higher priority event or request.

Common Functions of Interrupts

1. Interrupt transfers control to the **interrupt service routine (ISR)**
   - Usually through an **interrupt vector** containing addresses of service routines
2. Interrupt architecture must save the address of the interrupted instruction
3. **Trap / Exception**
   - Software generated interrupt
   - Caused by an error or a user request
4. Operating systems are **interrupt driven**

IISER THIRUVANANTHAPURAM

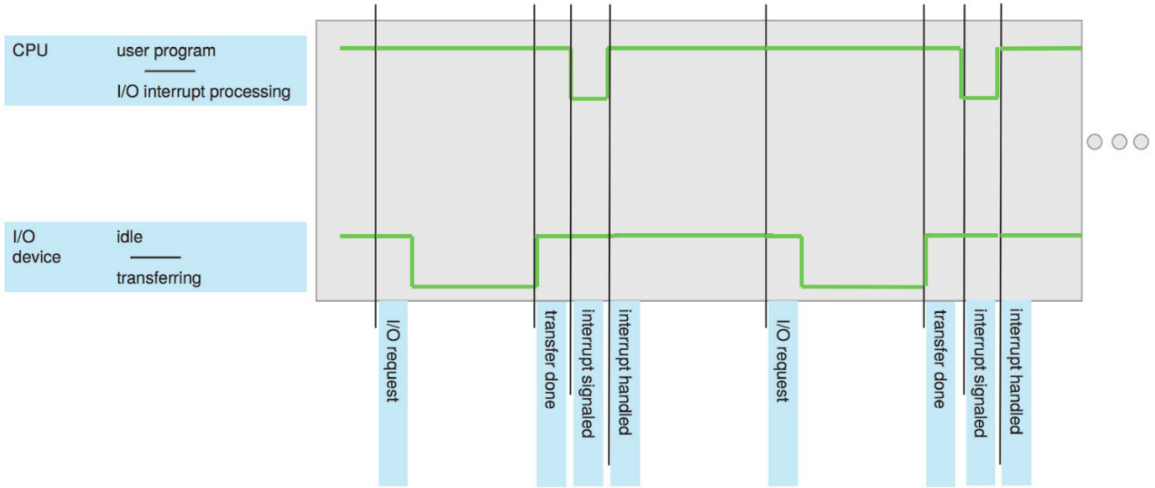

Figure 1.3 Interrupt timeline for a single program doing output.

Interrupt Handling

- Operating system preserves the **CPU state**
  - Stores register values and the program counter
- Operating system determines **which type of interrupt occurred**
- Separate code segments handle different interrupts
  - Each interrupt type has its own handler
  - Appropriate action is executed for that interrupt

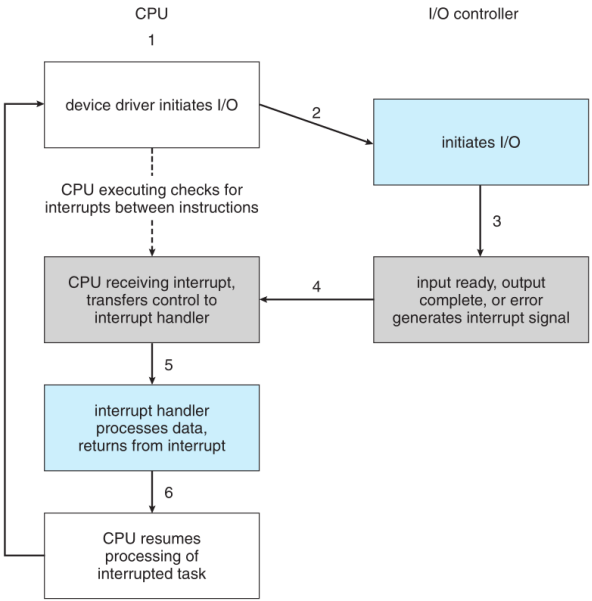

Figure 1.4 Interrupt-driven I/O cycle.

I/O Structure

I/O Structure

- **Two methods for handling I/O**
  - Control returns to the user program **after I/O completion**
  - Control returns to the user program **without waiting for I/O completion**
- **Return after I/O completion**
  - `wait` instruction keeps CPU idle until next interrupt
  - Possible wait loop (memory contention)
  - Only one I/O request at a time, no simultaneous I/O processing
- **Return without waiting (asynchronous I/O)**
  - **System call** lets user program request to wait for I/O completion
  - **Device status table** stores device type, address, and state
  - OS checks and updates the table when interrupts occur

Storage Structure

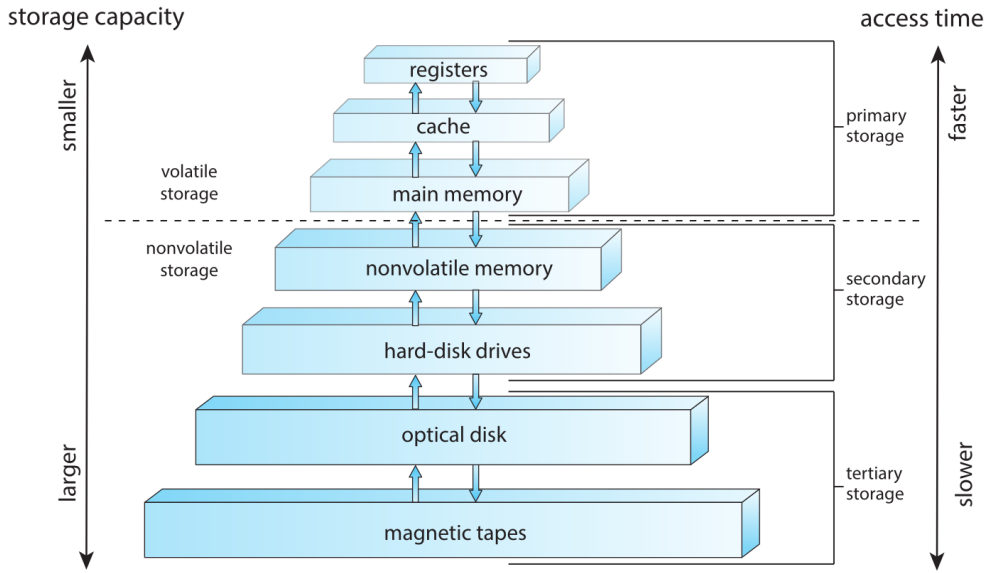

Figure 1.6 Storage-device hierarchy.

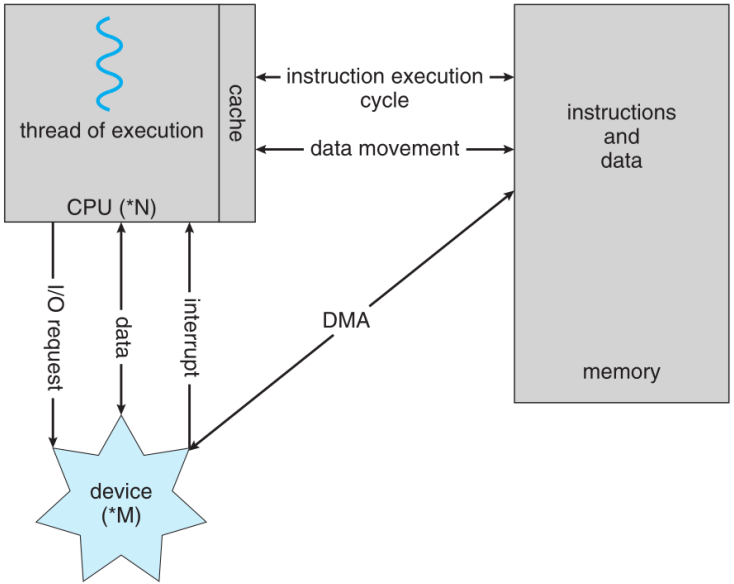

Figure: How a modern computer system works.

Computer-System Architecture

Single-Processor vs Multiprocessor Systems

- **Single-Processor Systems**
  - One general-purpose CPU
  - Executes all instructions and manages system activities
  - May include special-purpose processors (for example I/O controllers)
  - Most traditional computers were single-processor systems
- **Multiprocessor Systems**
  - Two or more CPUs sharing memory and I/O devices
  - Also called **parallel systems** or **tightly coupled systems**
  - Advantages:
    - Increased throughput
    - Economy of scale
    - Increased reliability (graceful degradation)

IISER THIRUVANANTHAPURAM

Definitions of Computer System Components

- **CPU**
  - Hardware that executes instructions
- **Processor**
  - A physical chip containing one or more CPUs
- **Core**
  - Basic computation unit inside a CPU
- **Multicore**
  - Multiple computing cores on the same CPU
- **Multiprocessor**
  - A system containing multiple processors
- In general usage:
  - **CPU** refers to a single computational unit
  - **Core** / **Multicore** refers to one or more cores within a CPU

IISER THIRUVANANTHAPURAM

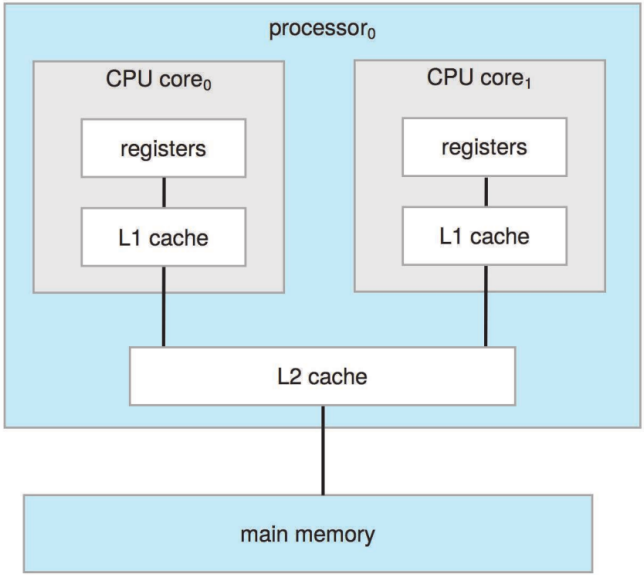

Figure: A dual-core design with two cores on the same chip.

Clustered Systems

Clustered Systems

- Multiple computer systems working together
- Usually share storage through a **Storage Area Network (SAN)**
- Provide **high availability** and survive hardware failures
- **Types of clustering**
  - **Asymmetric clustering**
    - One machine runs the application
    - Another machine stays in hot standby
  - **Symmetric clustering**
    - Multiple nodes run applications
    - Nodes monitor each other
- Some clusters support **High Performance Computing (HPC)**
  - Applications must be written to run in parallel
- Some systems use a **Distributed Lock Manager (DLM)**
  - Prevents conflicting operations between nodes

IISER THIRUVANANTHAPURAM

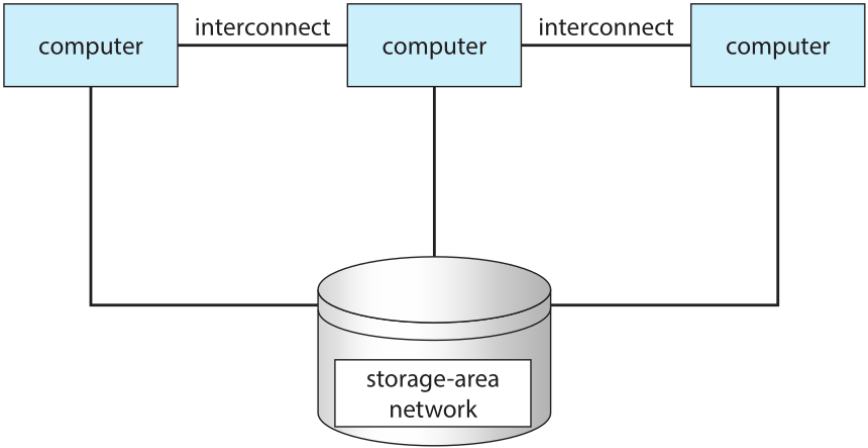

Figure: General structure of a clustered system.

Operating-System Operations

Operating-System Operations

- **Bootstrap program**
  - Small program that initializes the system
  - Loads the operating system kernel
- **Kernel initialization**
  - Kernel is loaded into memory
  - Starts **system daemons** (services running outside the kernel)
- **Interrupt driven system**
  - **Hardware interrupts**: generated by hardware devices
  - **Software interrupts** (exceptions or traps)
    - Software errors (for example division by zero)
    - Request for OS service via **system call**
    - Process problems such as infinite loops or illegal memory modification

IISER THIRUVANANTHAPURAM

Multiprogramming (Batch System)

- A single user cannot always keep the CPU and I/O devices busy
- **Multiprogramming** organizes multiple jobs (code and data) so that the CPU always has a job to execute
- Only a **subset of jobs** is kept in memory at a time
- The operating system selects one job and runs it using **job scheduling**
- When a job waits (for example for I/O), the OS **switches to another job**

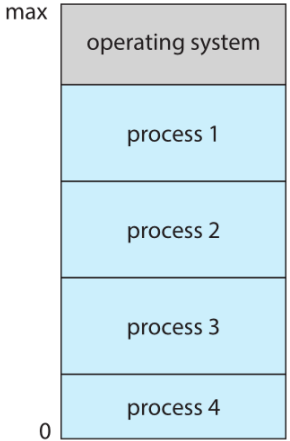

Figure: Memory layout for a multiprogramming system.

Multitasking (Timesharing)

- Logical extension of batch systems
- CPU switches between jobs very frequently, allowing **interactive computing**
- Desired **response time** is typically less than 1 second
- Each user runs at least one program in memory, called a **process**
- If several jobs are ready at the same time, the OS uses **CPU scheduling**
- If processes do not fit in memory, **swapping** moves them in and out of memory
- **Virtual memory** allows processes to run even if they are not completely in memory

IISER THIRUVANANTHAPURAM

Dual-Mode Operation

- Allows the operating system to **protect itself and system components**
- Two execution modes:
  - **User mode**
  - **Kernel mode**
- Hardware provides a **mode bit**
  - Distinguishes whether the system runs user code or kernel code
  - User programs execute in **user mode**
  - OS kernel executes in **kernel mode**
- **Mode switching**
  - System call switches mode to kernel
  - Return from system call switches back to user mode
- Some instructions are **privileged**
  - Can be executed only in kernel mode

IISER THIRUVANANTHAPURAM

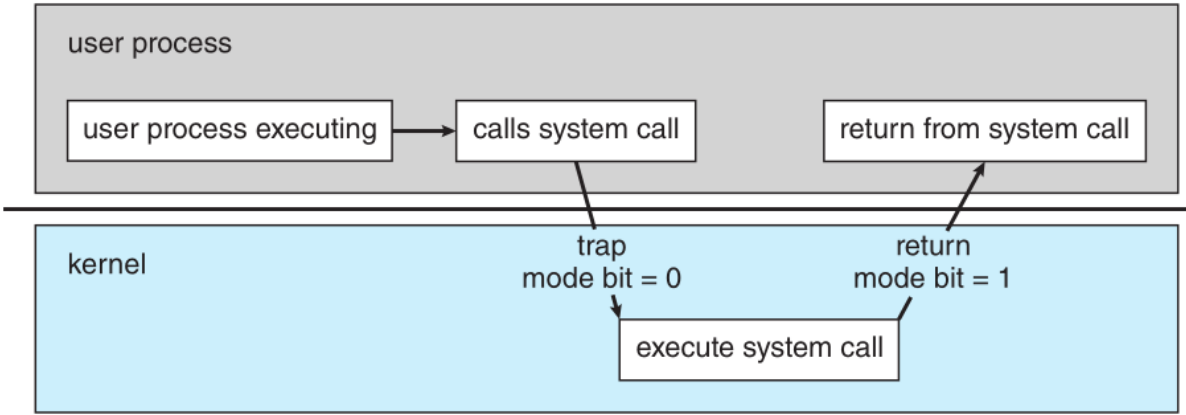

Figure: Transition from user to kernel mode.

Timer

- **Purpose**
  - Prevents infinite loops or processes hogging system resources
- **Operation**
  - Timer generates an interrupt after a specified time period
  - Uses a counter that is decremented by the system clock
- **Control**
  - Operating system sets the timer using a privileged instruction
  - When the counter reaches zero, an interrupt is generated
- **Usage**
  - Timer is set before scheduling a process
  - Allows the OS to regain control or terminate programs exceeding their time limit

Resource Management

1. Process Management
2. Memory Management
3. File-System Management
4. Mass-Storage Management
5. Cache Management
6. I/O System Management

Process Management

Process Management

- A **process** is a program in execution
  - Program: passive entity
  - Process: active entity and unit of work in the system
- A process requires resources
  - CPU, memory, I/O devices, files
  - Initialization data
- When a process terminates, the OS reclaims reusable resources
- **Single-threaded process**
  - One program counter indicating the next instruction
  - Instructions execute sequentially
- **Multi-threaded process**
  - Multiple threads, each with its own program counter
- Systems typically run many processes concurrently
  - Both user and operating system processes
  - CPU multiplexing provides concurrency

IISER THIRUVANANTHAPURAM

Process Management Activities

- The operating system performs several activities related to **process management**
  - Creating and deleting both **user and system processes**
  - **Suspending and resuming** processes
  - Providing mechanisms for **process synchronization**
  - Providing mechanisms for **process communication**
  - Providing mechanisms for **deadlock handling**

Memory Management

Memory Management

- To execute a program, all or part of its **instructions and data** must be in memory
- **Memory management** determines what is in memory and when
  - Helps optimize CPU utilization and system response time
- **Memory management activities**
  - Keeping track of which memory locations are in use and by whom
  - Deciding which processes (or parts) and data to move into or out of memory
  - Allocating and deallocating memory space as needed

File-System Management

File-System Management

- OS provides a **uniform logical view** of information storage
  - Physical storage is abstracted into logical units called **files**
  - Storage media controlled by devices such as disk drives and tape drives
  - Devices differ in access speed, capacity, transfer rate, and access method
- **File-system management**
  - Files are typically organized into **directories**
  - **Access control** determines who can access files
- **Operating system activities**
  - Creating and deleting files and directories
  - Providing primitives to manipulate files and directories
  - Mapping files onto secondary storage
  - Backing up files onto stable (non volatile) storage

IISER THIRUVANANTHAPURAM

Mass-Storage Management

- Disks store data that does not fit in main memory or must be kept for a long time
- Proper management of mass storage is crucial
- Overall system performance often depends on the disk subsystem and its algorithms
- **Operating system activities**
  - Mounting and unmounting storage devices
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

IISER THIRUVANANTHAPURAM

Caching

- Important principle used at many levels of a computer
  - Hardware, operating system, and software
- Frequently used information is copied from **slower storage** to **faster storage** temporarily
- The **cache** is checked first
  - If data is present, it is used directly from cache (fast)
  - Otherwise data is fetched from slower storage and placed in cache
- Cache is smaller than the storage being cached
- **Cache management issues**
  - Cache size
  - Replacement policy

IISER THIRUVANANTHAPURAM

Migration of Data "A" from Disk to Register

- In multitasking systems, the most **recent value** of data must always be used, regardless of where it is stored in the storage hierarchy
- In **multiprocessor systems**, hardware must ensure **cache coherency**
  - All CPUs should see the most recent value of the data in their caches
- In **distributed systems**, the problem becomes more complex
  - Multiple copies of the same data may exist
  - Special mechanisms are needed to maintain consistency

IISER THIRUVANANTHAPURAM

Security and Protection

Protection and Security

- **Protection**
  - Mechanisms that control access of processes or users to system resources
- **Security**
  - Defense of the system against internal and external attacks
  - Examples: denial of service, worms, viruses, identity theft, theft of service
- Systems distinguish among **users** to determine permissions
- **User identity**
  - Each user has a **User ID (UID)**
  - UID associated with the user's files and processes for access control
- **Group identity**
  - **Group ID (GID)** defines sets of users
  - Used to manage permissions for files and processes
- **Privilege escalation**
  - Allows a user to temporarily obtain higher privileges

IISER THIRUVANANTHAPURAM

Distributed Systems

Distributed Systems

- A **distributed system** is a collection of separate, possibly heterogeneous, computers connected through a network
- The network provides a communication path, commonly using **TCP/IP**
- Types of networks
  - Local Area Network (LAN)
  - Wide Area Network (WAN)
  - Metropolitan Area Network (MAN)
  - Personal Area Network (PAN)
- **Network Operating System**
  - Provides services for communication across systems
  - Enables systems to exchange messages
  - Creates the illusion of a **single unified system**

IISER THIRUVANANTHAPURAM

Kernel Data Structures

For different purposes, Operating system uses different data structures, such as

- Lists, Stacks, and Queues
- Trees
- Hash Functions and Maps

References

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com
Course webpage: https://laltu-sardar.github.io/courses/corgos_2026.html.