

Chapter 1: Introduction

adapted from Silberschatz, Galvin, Gagne

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization

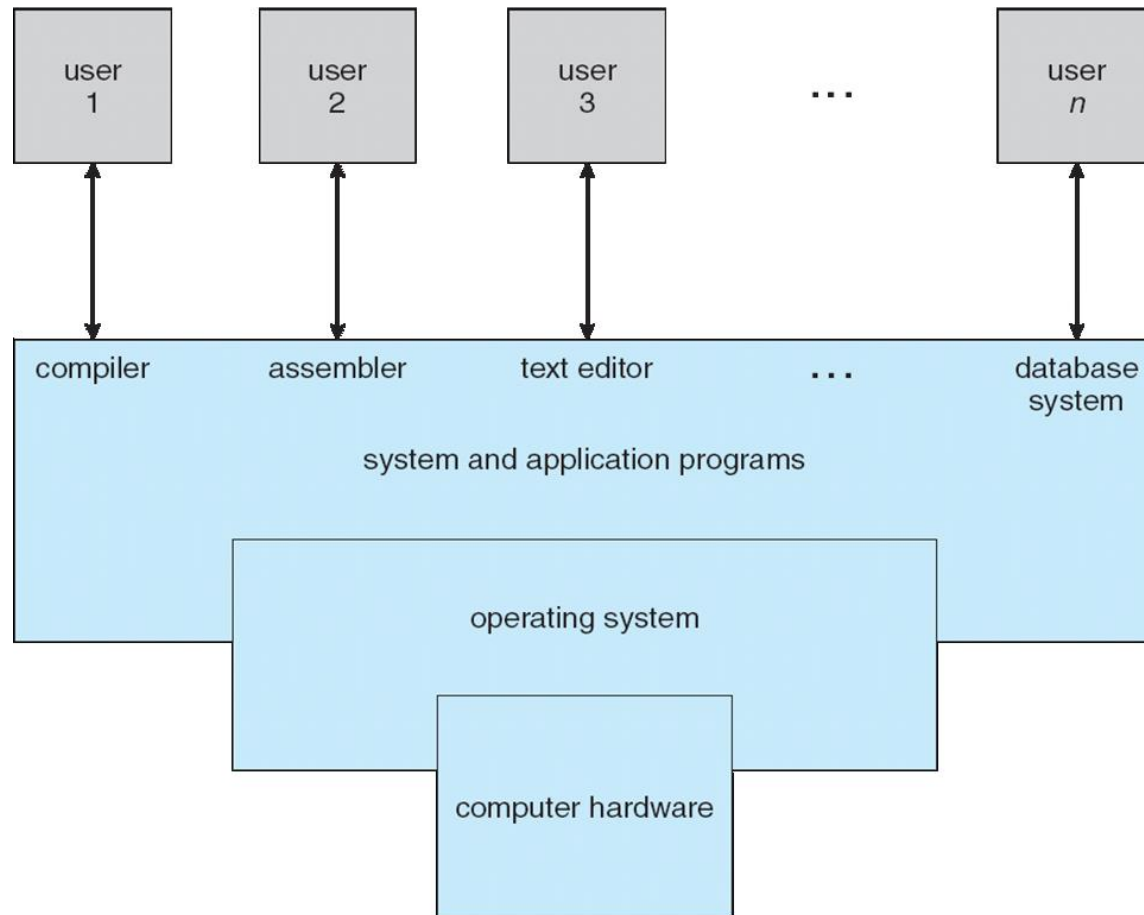
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
 - Execute user programs and make solving user problems easier
 - Make the computer system convenient to use
 - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components
 - Hardware – provides basic computing resources
 - ▶ CPU, memory, I/O devices
 - Operating system
 - ▶ Controls and coordinates use of hardware among various applications and users
 - 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
 - Users
 - ▶ People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
 - Manages all resources
 - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
 - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont)

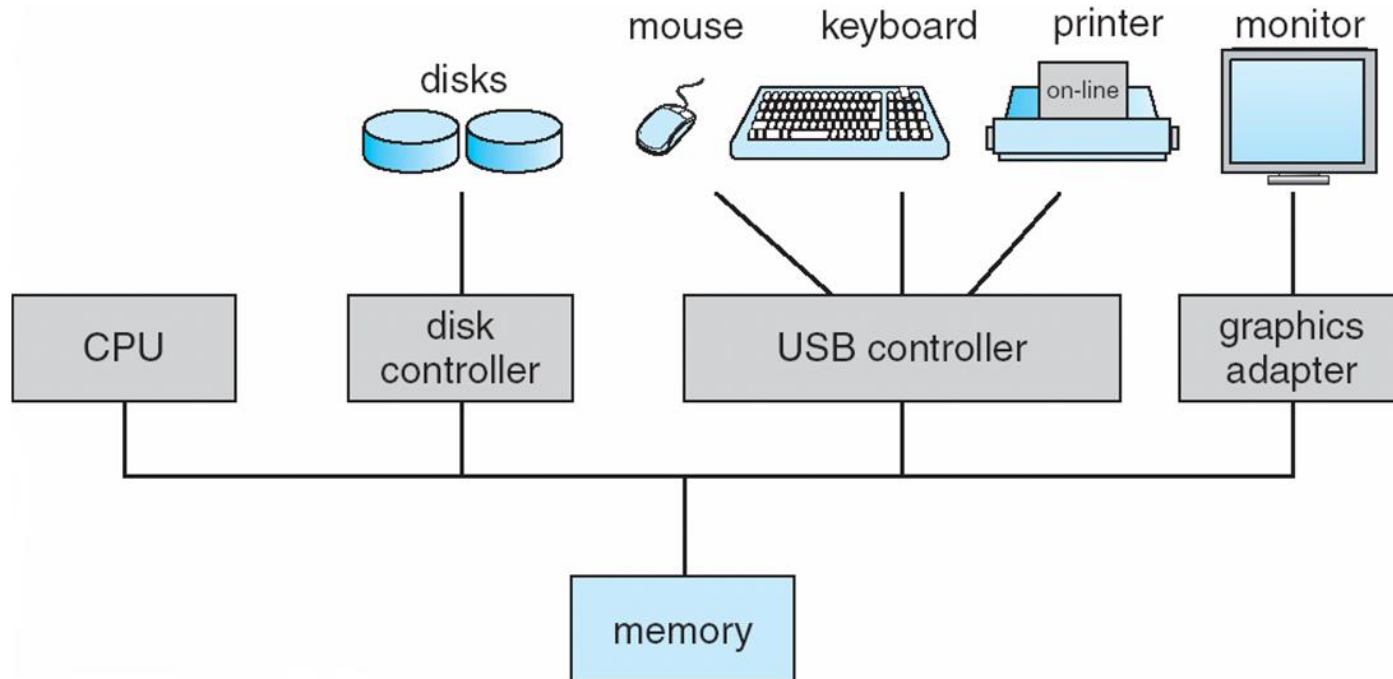
- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation
 - But varies wildly
- “The one program running at all times on the computer” is the **kernel**.
Everything else is either a system program (ships with the operating system) or an application program

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
 - Typically stored in ROM or EPROM, generally known as **firmware**
 - Initializes all aspects of system
 - Loads operating system kernel and starts execution

Computer System Organization

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an *interrupt*

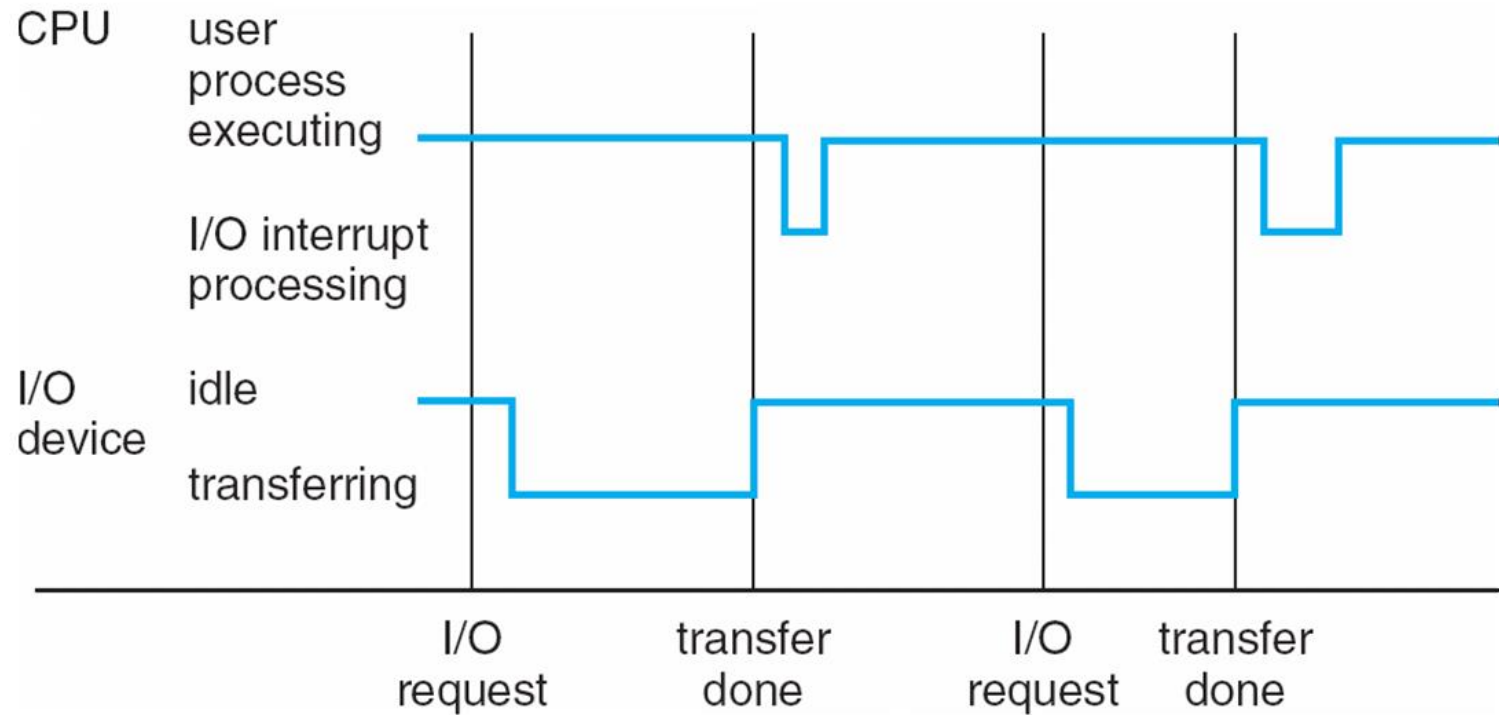
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
 - **polling** based interrupt system: the interrupt handler poll or send a signal to each device in turn in order to find out which one sent the interrupt request.
 - **vectored** interrupt system: hardware informs the part of the computer that handles I/O interrupts that a request for attention from an I/O device has been received and also identifies the device that sent the request
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline



Direct Memory Access (DMA)

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

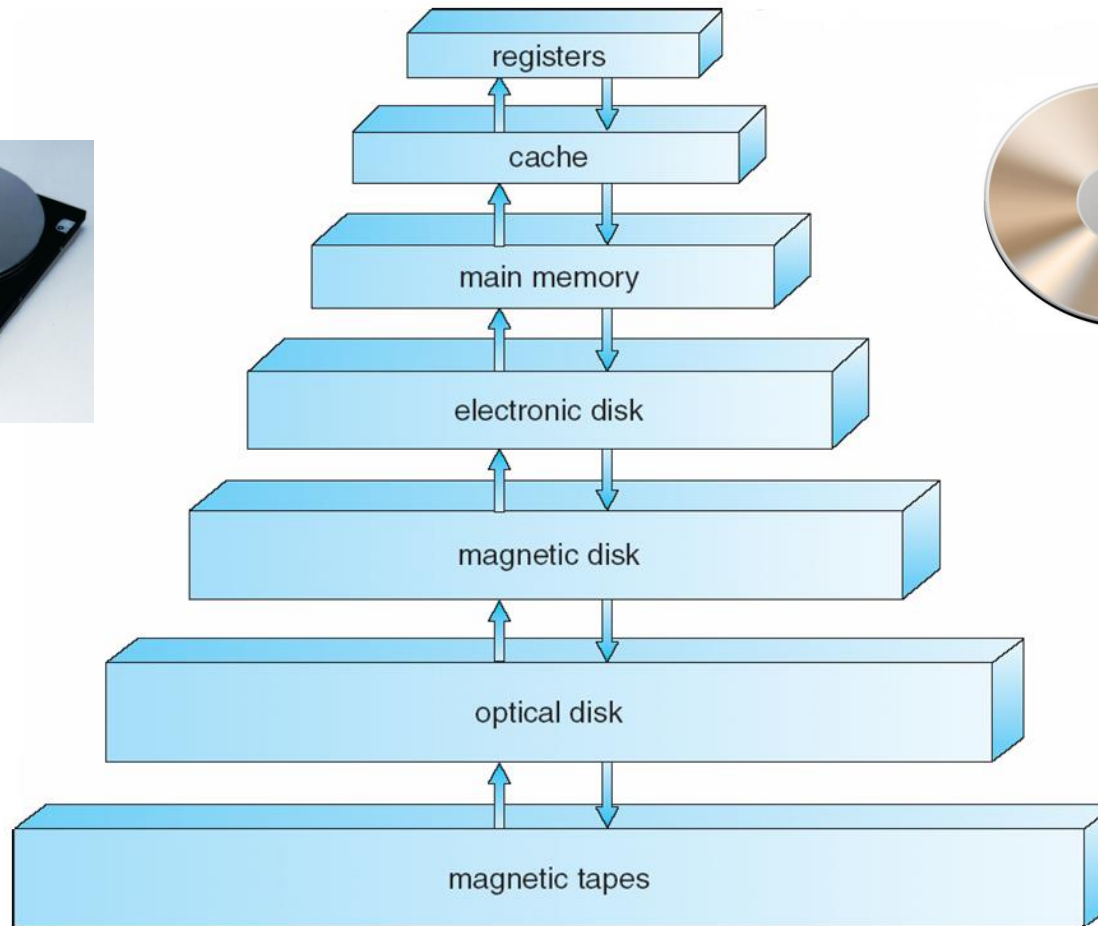
Storage Structure

- Main memory – only large storage media that the CPU can access directly
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
 - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
 - The **disk controller** determines the logical interaction between the device and the computer

Storage Hierarchy

- Storage systems organized in hierarchy
 - Speed
 - Cost
 - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage

Storage-Device Hierarchy



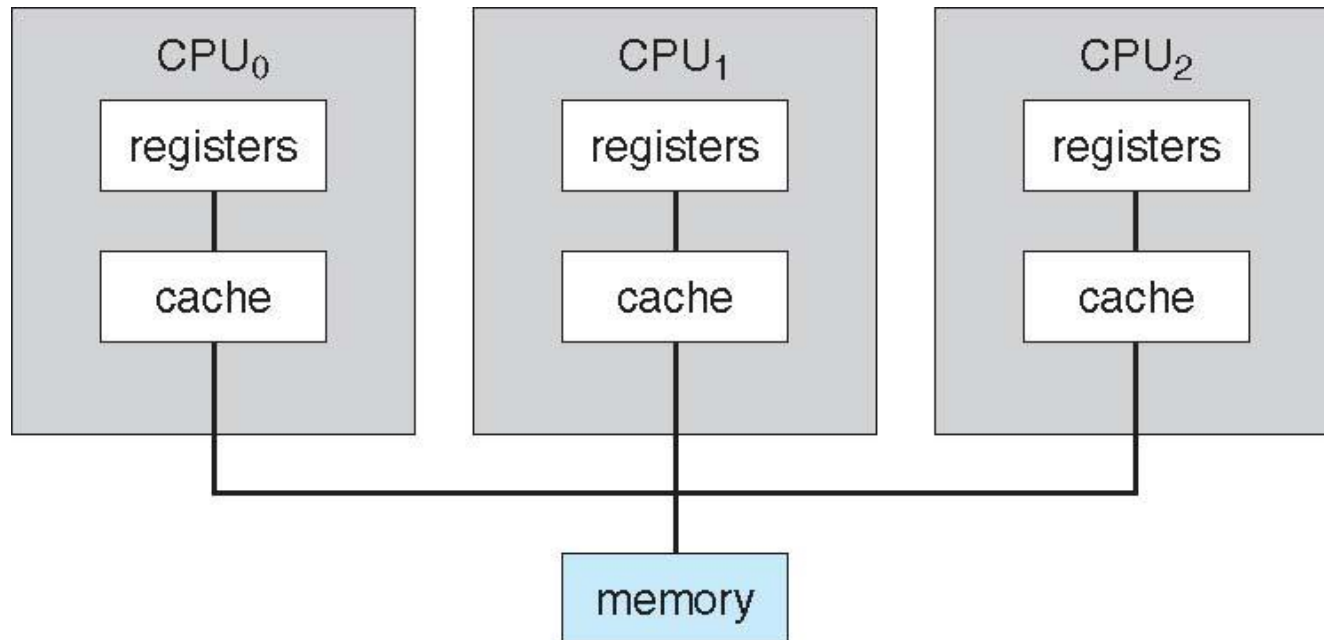
Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (fast)
 - If not, data copied to cache and used there
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy

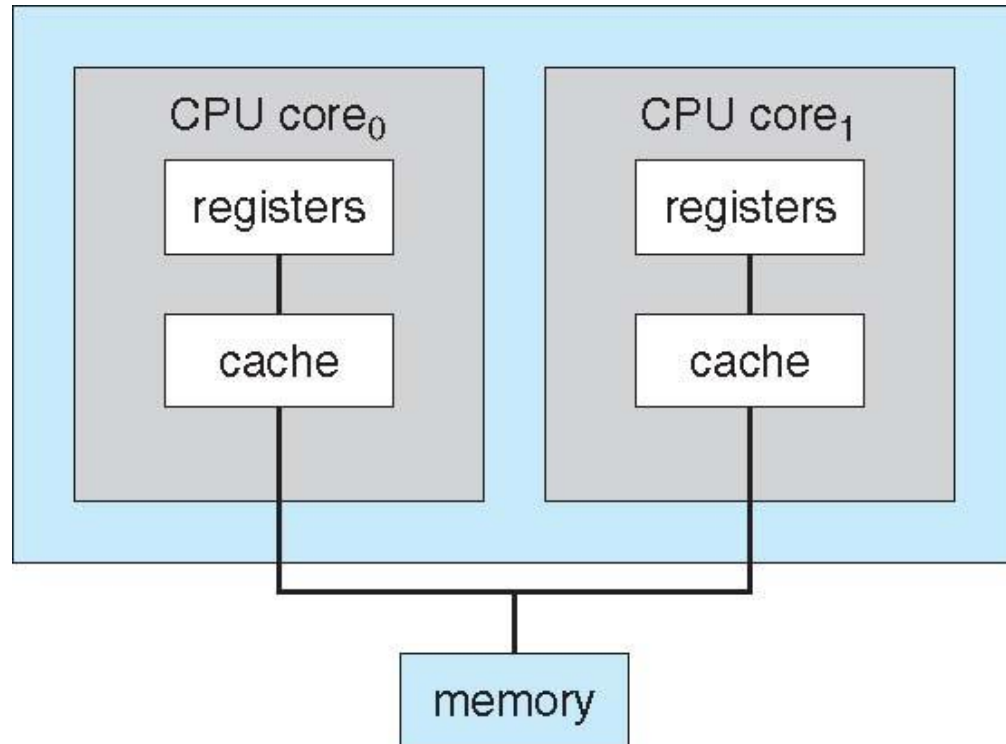
Computer-System Architecture

- Many old systems use a single general-purpose processor
 - e.g., Intel Pentium 4
- **Multiprocessors** systems growing in use and importance
 - Also known as **parallel systems**, **multi-core systems**
 - Advantages include
 1. **Increased throughput**
 2. **Economy of scale**
 3. **Increased reliability**
 - Two types
 1. **Asymmetric Multiprocessing**
 2. **Symmetric Multiprocessing (SMP)**

Symmetric Multiprocessing Architecture



A Dual-Core Design



Clustered Systems

- Like multiprocessor systems, but multiple systems working together
 - Usually sharing storage via a [storage-area network \(SAN\)](#)
 - Provides a [high-availability](#) service which survives failures
 - [Asymmetric clustering](#) has one machine in hot-standby mode
 - [Symmetric clustering](#) has multiple nodes running applications, monitoring each other
 - Some clusters are for [high-performance computing \(HPC\)](#)
 - Applications must be written to use [parallelization](#)

Operating System Structure

- **Multiprogramming** needed for efficiency
 - Single user cannot keep CPU and I/O devices busy at all times
 - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
 - A subset of total jobs in system is kept in memory
 - One job selected and run via **job scheduling**
 - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
 - **Response time** should be < 1 second
 - Each user has at least one program executing in memory \Rightarrow **process**
 - If several jobs ready to run at the same time \Rightarrow **CPU scheduling**
 - If processes don't fit in memory, **swapping** moves them in and out to run
 - **Virtual memory** allows execution of processes not completely in memory

Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- Timer to prevent infinite loop / process hogging resources
 - Set interrupt after specific period
 - Operating system decrements counter
 - When counter zero generate an interrupt
 - Set up before scheduling process to regain control or terminate program that exceeds allotted time

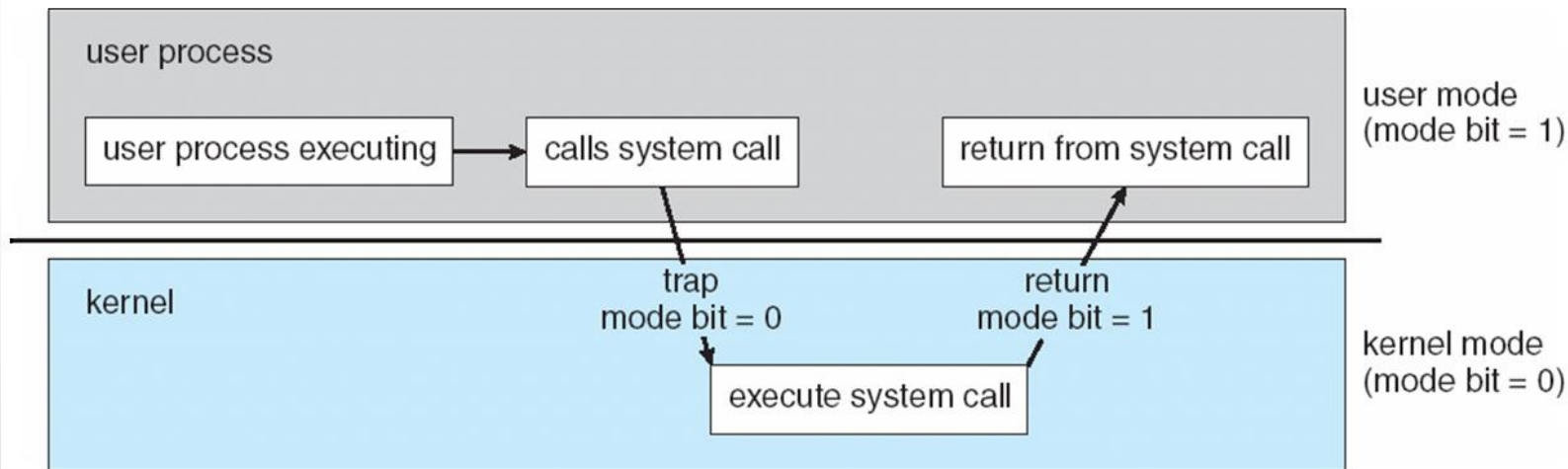
Transition from User to Kernel Mode

■ **Dual-mode** operation allows OS to protect itself and other system components

- **User mode** and **kernel mode**

- **Mode bit** provided by hardware

- ▶ Provides ability to distinguish when system is running user code or kernel code
- ▶ Some instructions designated as **privileged**, only executable in kernel mode
- ▶ System call changes mode to kernel, return from call resets it to user



Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
 - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with 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

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit - **file**
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - ▶ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
 - Files usually organized into directories
 - Access control on most systems to determine who can access what
 - OS activities include
 - ▶ Creating and deleting files and directories
 - ▶ Primitives to manipulate files and dirs
 - ▶ Mapping files onto secondary storage
 - ▶ Backup files onto stable (non-volatile) storage media

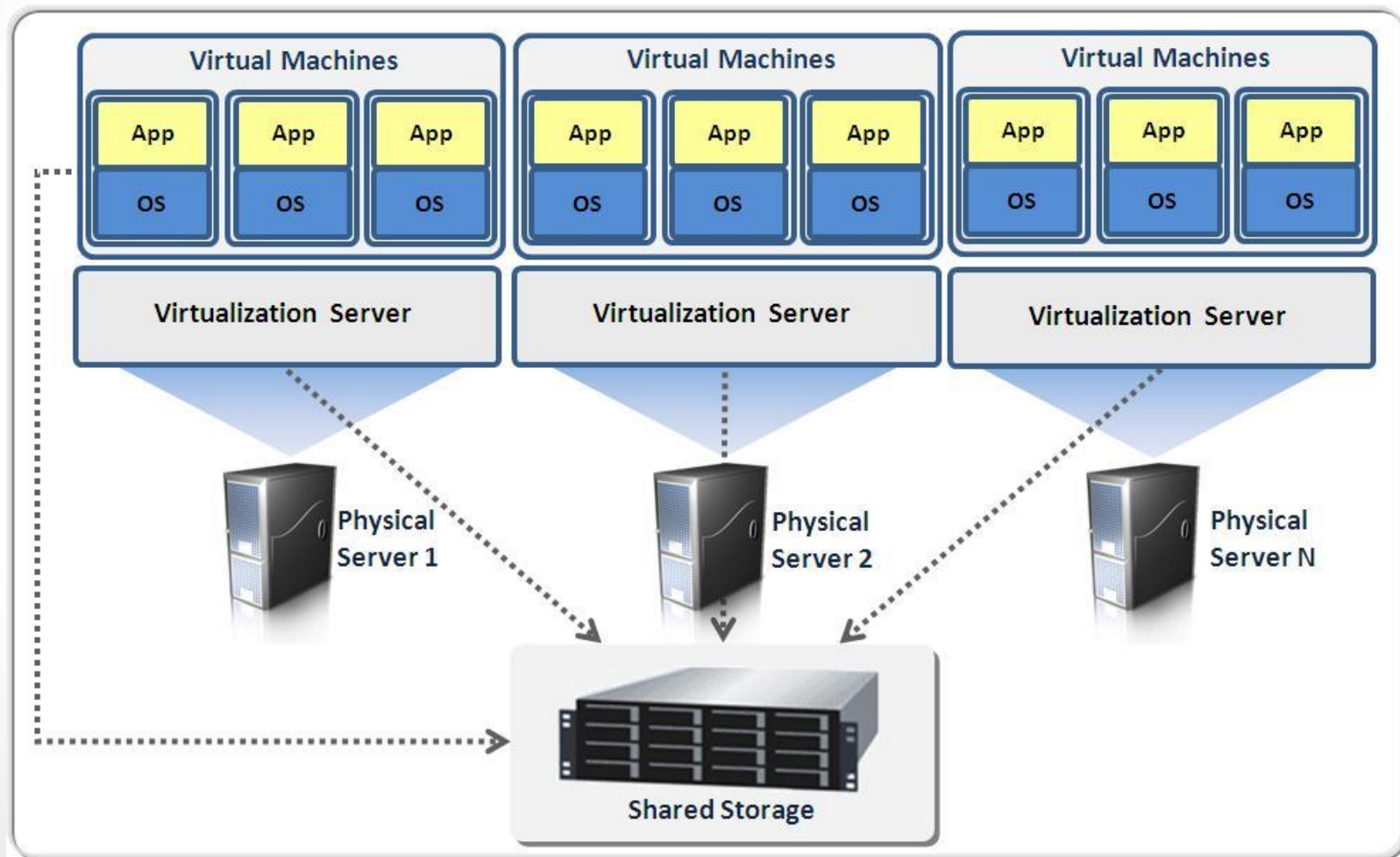
I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
 - General device-driver interface
 - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to effective ID with more rights

Virtualization



Cloud Computing

Computing
Resources

Servers



Storage



Network



Applications



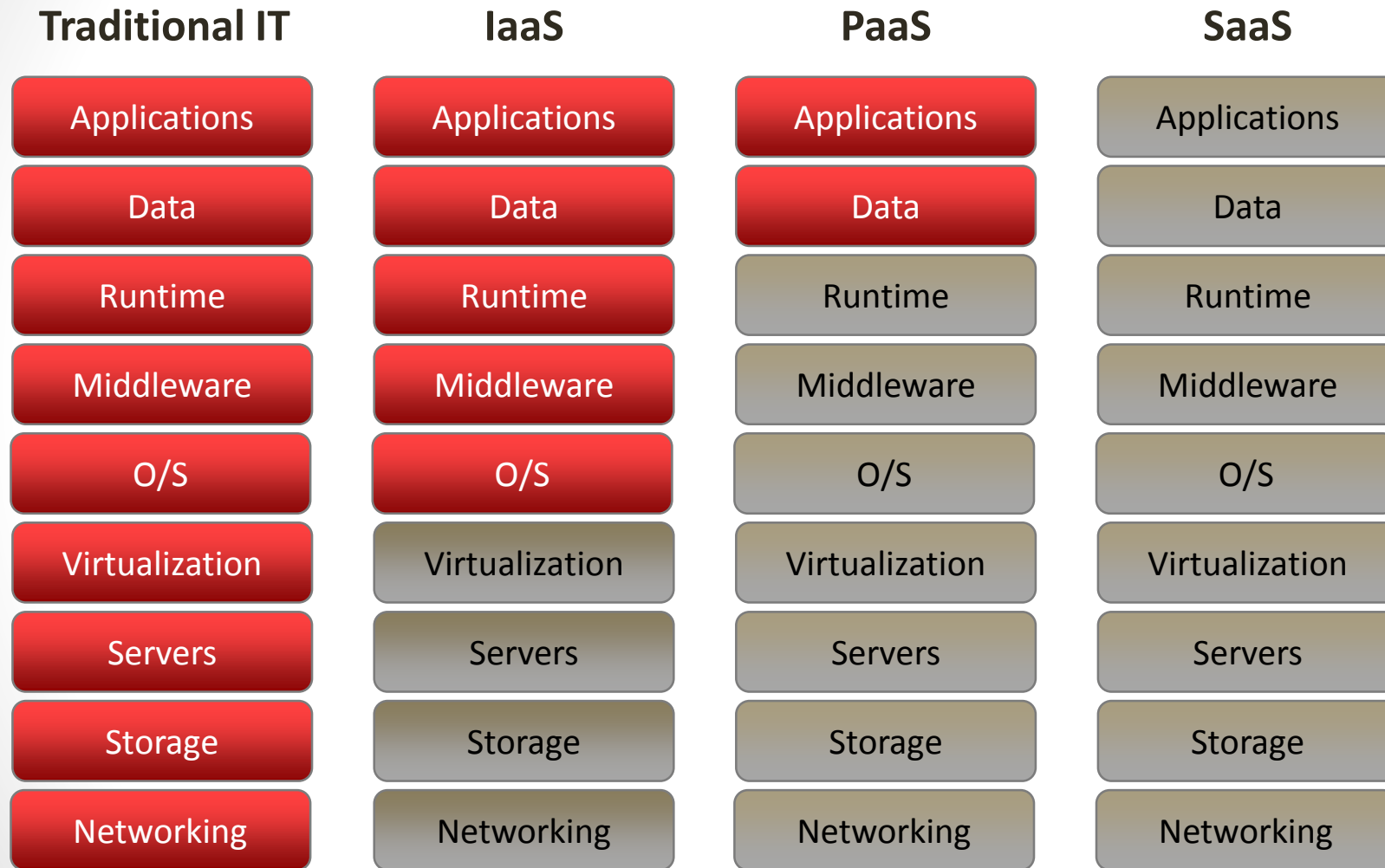
Service
Model

Infrastructure
as a Service
(IaaS)

Platform as a
Service
(PaaS)

Software as a
Service
(SaaS)

Cloud Service Models



Managed by users

Managed by providers

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary [closed-source](#)
- Counter to the [copy protection](#) and [Digital Rights Management \(DRM\)](#) movement
- Started by [Free Software Foundation \(FSF\)](#), which has “copyleft” [GNU Public License \(GPL\)](#)
- Examples include [GNU/Linux](#), [BSD UNIX](#) (including core of [Mac OS X](#)), and [Sun Solaris](#)

End of Chapter 1