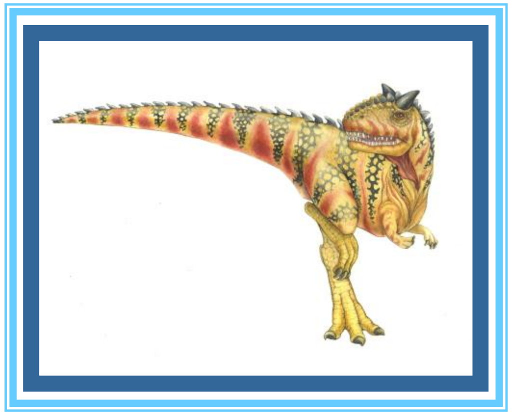
Chapter 1: Introduction

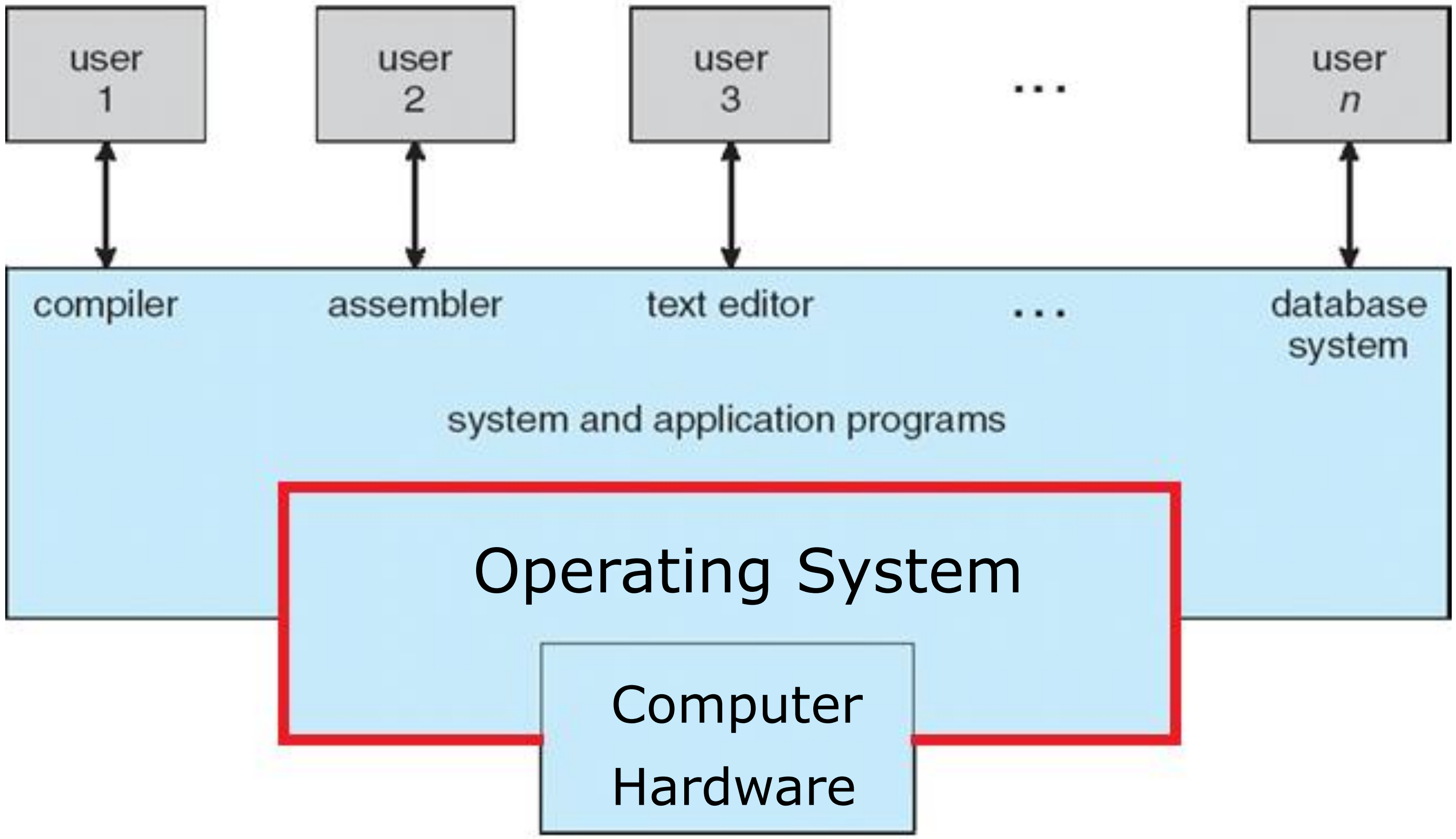


Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Operating-System Operations
  - Process Management
  - Memory Management
  - Storage Management
  - Protection and Security
- Kernel Data Structures

What Operating Systems Do

Four Components of a Computer System



What Operating Systems Do

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
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

What Operating Systems Do

Different Goals for Different Systems

- Depends on the point of view
- For **personal computer**, users want convenience → Care about ease of use or performance, but not resource utilization
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy → optimize resource utilization
- **Handheld** computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as **embedded computers** in devices and automobiles

What Operating Systems Do

Operating System Definition

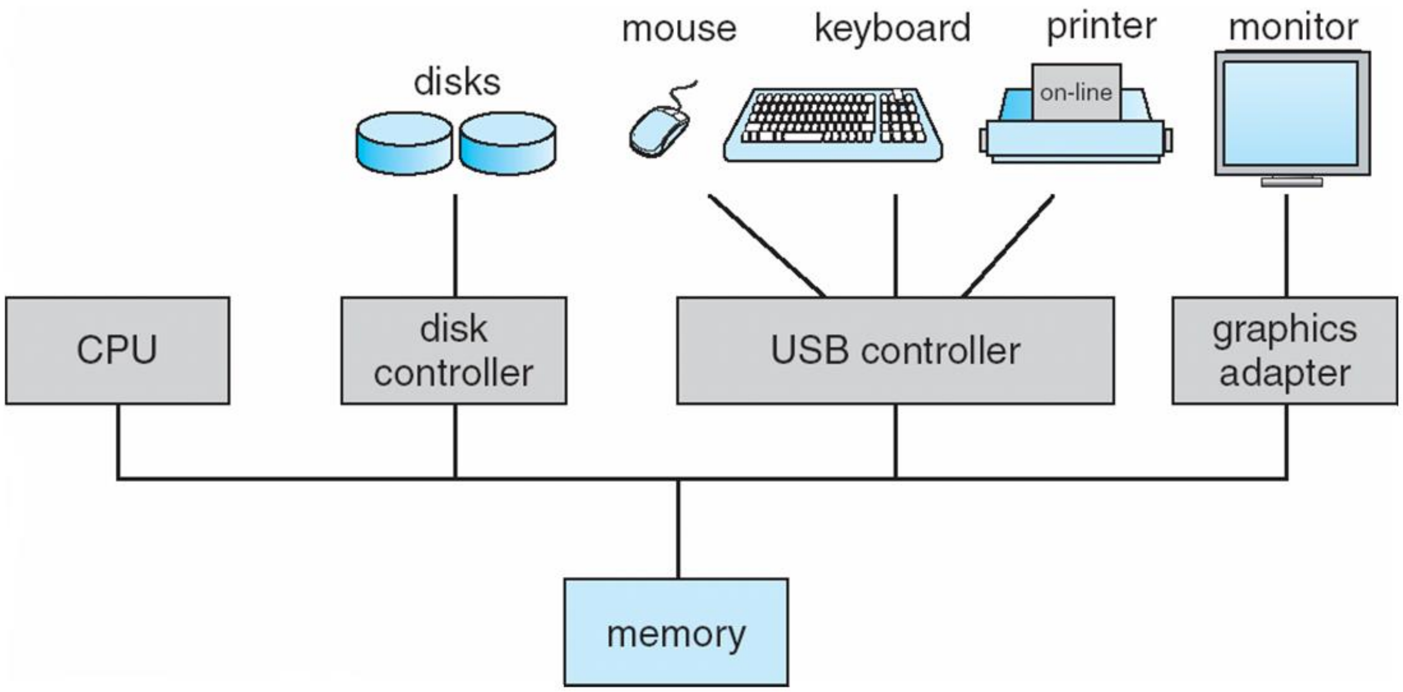
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

- **No universally accepted definition**

Computer System Organization

Computer-System Operation

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



Computer System Organization

Computer-System Operation: Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, known as **firmware**
  - Initializes all aspects of system
    - CPU registers, memory content, device controller
  - Loads operating system kernel and starts execution
- Kernel starts providing services to the system.
  - In UNIX, first system process is “init” which starts many other daemons.
- The system waits for some event to occur
  - OS is event driven (Interrupt driven)

Computer System Organization

Computer-System Operation: Interrupts

- The occurrence of an event is signaled by an **interrupt** from either hardware or software
  - Hardware can trigger an interrupt at any time by sending a signal to CPU (by way of system bus)
  - Software-generated interrupt is called a trap or exception, caused either by an error or a user request (executing a special operation called system call)
- Interrupt Service Routine (**ISR**)
  - When CPU is interrupted, it stops what it is doing and immediately transfer execution to ISR
  - On ISR completion, CPU resumes to the interrupted instruction.

Computer System Organization

Computer-System Operation: Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- The operating system preserves the state of the CPU by storing **registers** and the **program counter** (the address of the interrupted instruction)
- An operating system is **interrupt driven**

Computer System Organization

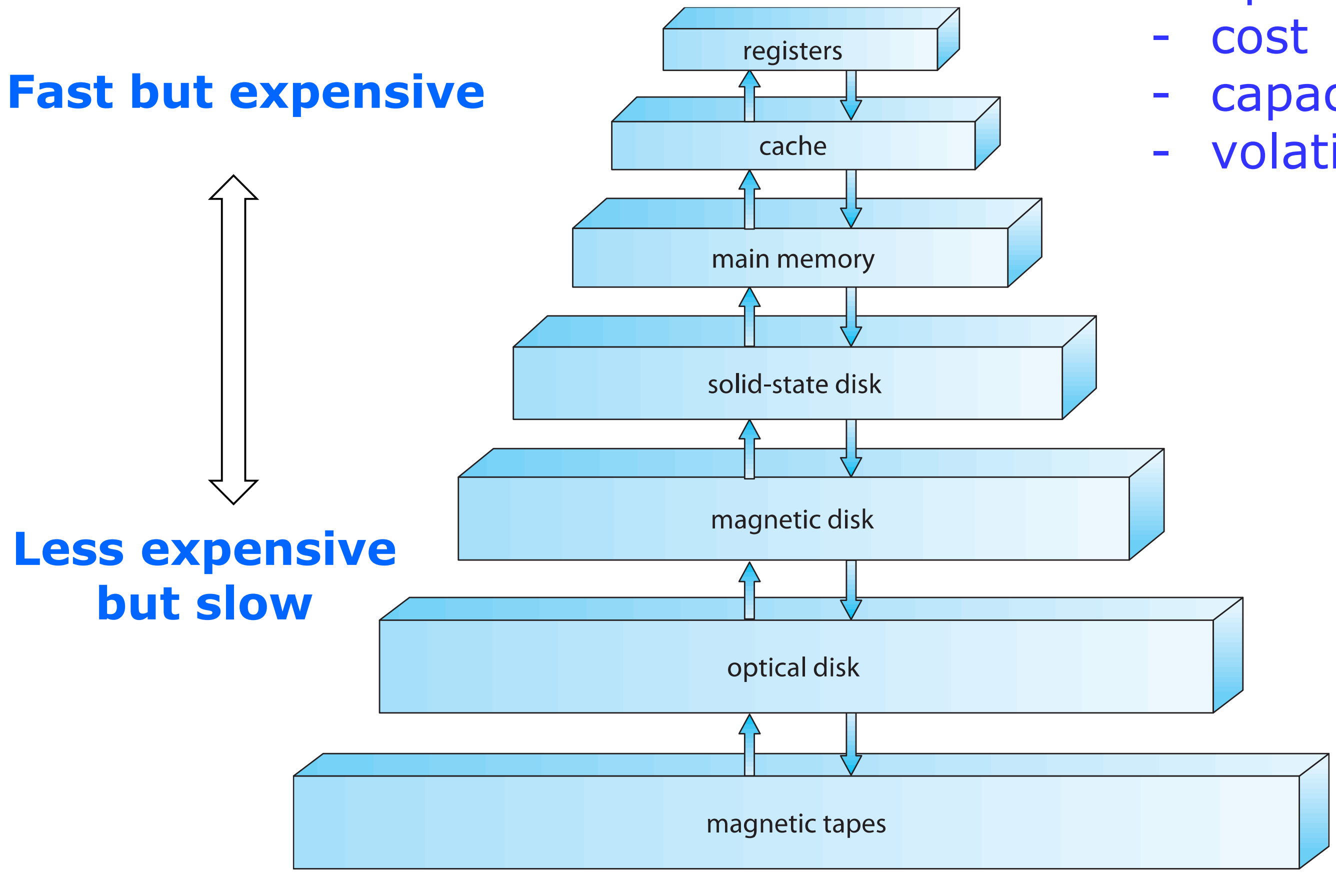
Storage Structure

- Main memory – CPU can only load instructions only from memory, so any programs to run must be stored there.
  - **Random access,** typically **volatile** ➔ e.g., SRAM, DRAM
  - non-volatile: EEPROM, ROM (bootstrap)
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Magnetic disks – metal platters covered with magnetic material
    - Disk surface is logically divided into **tracks**, subdivided into **sectors**
    - The **disk controller** determines the logical interaction between the device and the computer
  - **Solid-state disks (SSD)** – faster than magnetic disks, nonvolatile
    - Various technologies, Becoming more popular

Computer System Organization

Storage Hierarchy

Difference among various storage
- Speed
- cost
- capacity
- volatility



Computer System Organization

Storage Structure - Caching

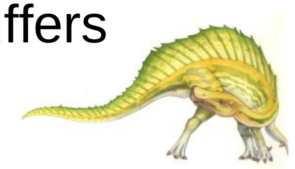
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
  - e.g., main memory can be viewed as a cache for secondary storage
- Faster storage (cache) checked first to determine if information 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 is an important design problem
  - Cache size and replacement policy

Computer System Organization

I/O Structure

- 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 and a set of registers
- OS has a device driver for each device controller
- I/O operation
  - Device driver starts an I/O operation by sending commands to the proper register of the device controller
  - Device controller examines the command register and moves data to/from the device from/to its local buffer
  - Device controller informs CPU that it has finished its operation by causing an ***interrupt***
  - CPU moves data from/to main memory to/from local buffers

  ➔ Interrupt-driven I/O



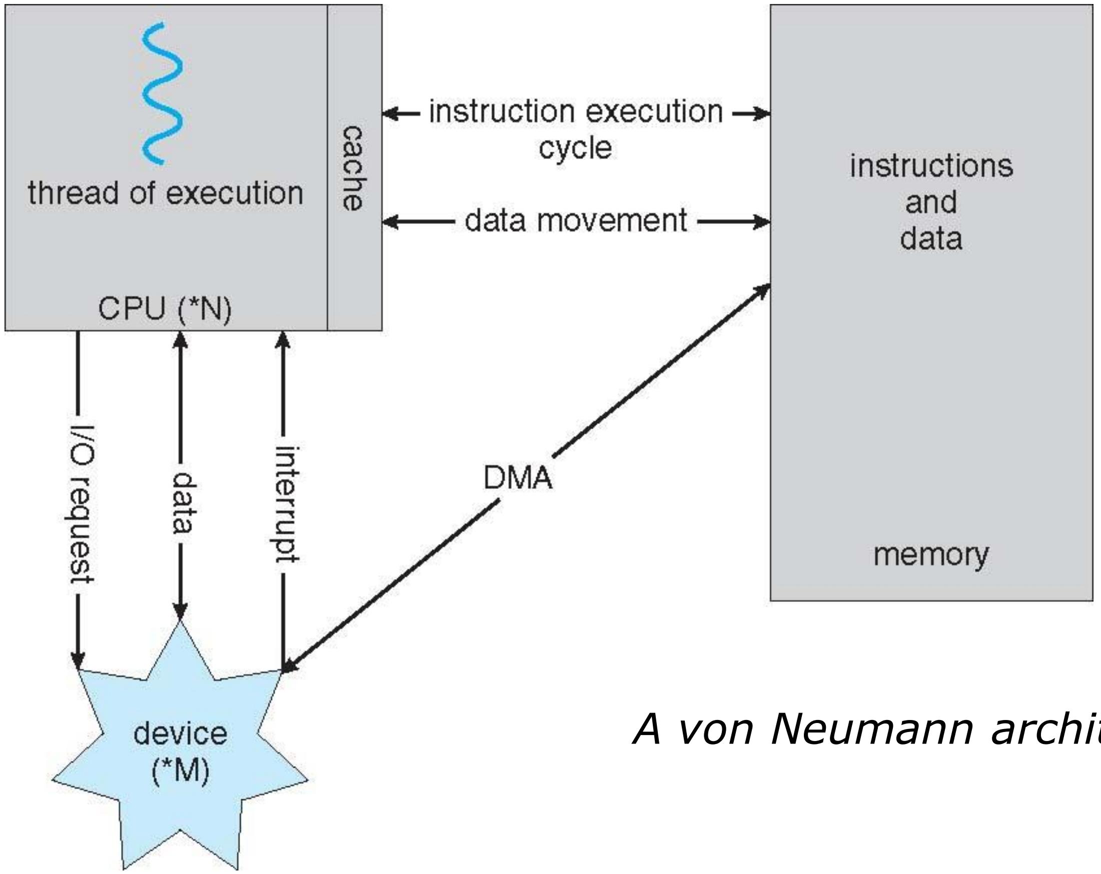
Computer System Organization

I/O Structure

- **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 local buffer directly to main memory without CPU intervention
  - Only one interrupt is generated per block, rather than the one interrupt per byte

Computer-System Architecture

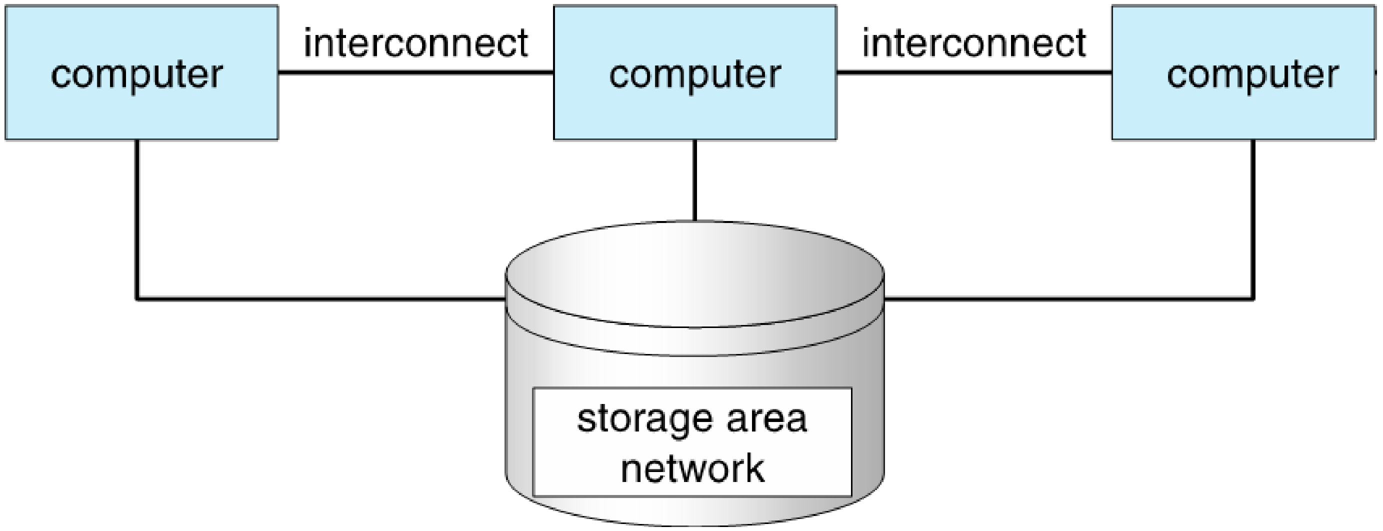
- Single Processor System
  - Most systems use **a single general-purpose** processor
    - Some systems have special-purpose processors as well



A von Neumann architecture

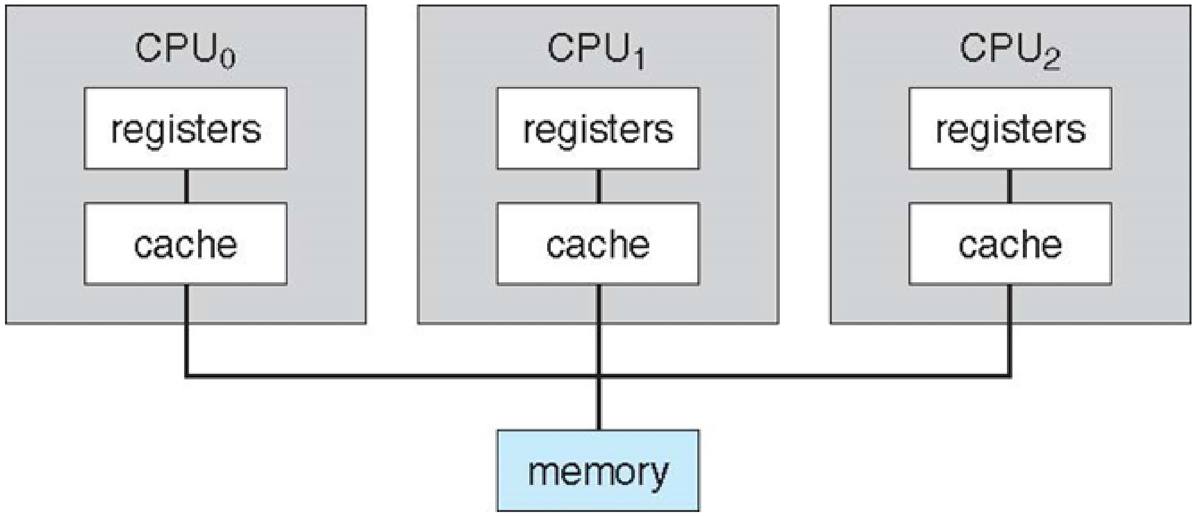
Computer-System Architecture

- **Clustered** Systems (loosely coupled)
  - Multiple systems working together by network
  - 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



Computer-System Architecture

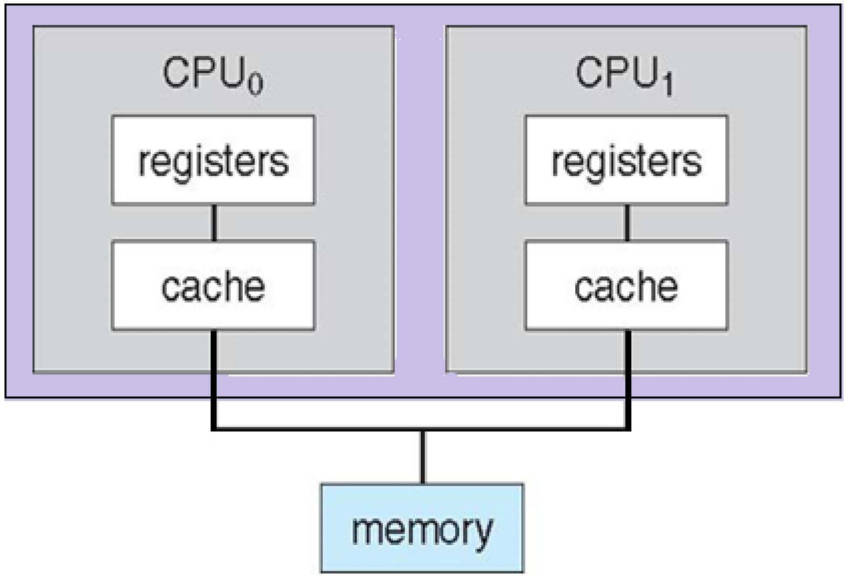
- **Multiprocessors** systems (parallel systems, tightly-coupled systems)
  - Increased throughput
  - Economy of scale
  - Increased reliability: graceful degradation (or fault tolerant)
- Two types:
  1. **Asymmetric Multiprocessing** (master-slave relationship)
  2. **Symmetric Multiprocessing**



Computer-System Architecture

- Multiprocessors systems : **Multicore**
  - More efficient than multiple chips with single cores because on-chip communication is faster than between-chip communication.
  - Less power than multiple single-core chips

Dual-core chip



Operating System Structure

- **Multiprogramming** (increase CPU utilization)
  - 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**) (creating interactive computing)
  - Response time should be < 1 second
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out
  - **Virtual memory** allows execution of processes not all in memory

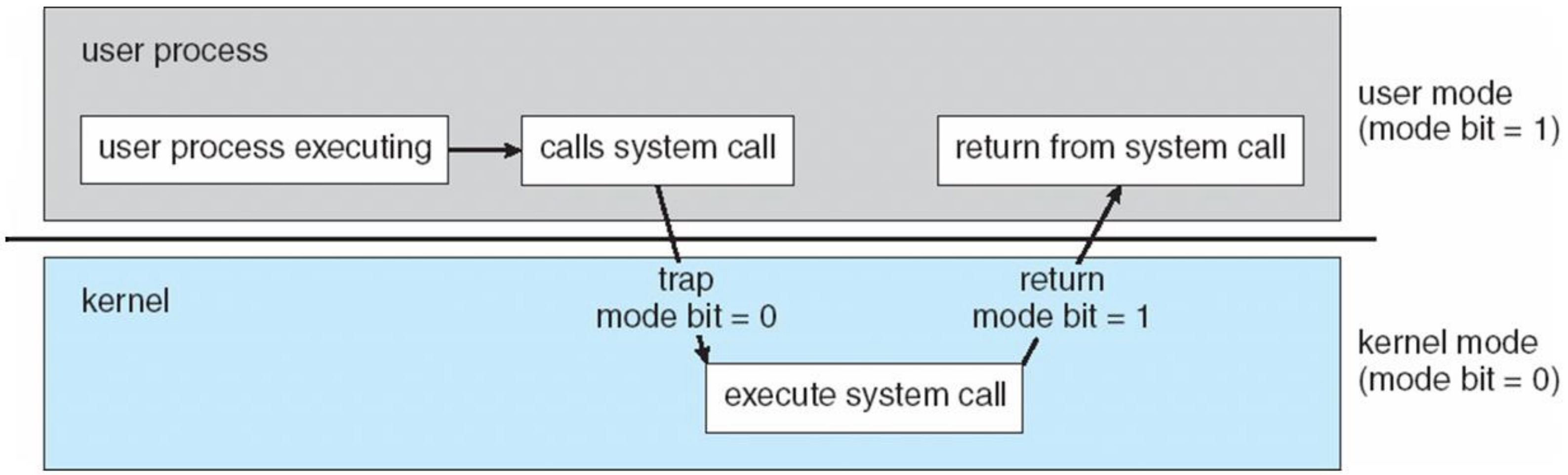
Operating-System Operations

- A properly designed system must ensure that an incorrect program cannot cause other program to execute incorrectly.
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - To distinguish when system is running user code or kernel code
    - **privileged** instructions: only used in kernel mode
    - System call changes mode to kernel, return from call resets it to user mode
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Operating-System Operations

Transition from User to Kernel Mode

- 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



Operating-System Operations

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 running concurrently on one or more CPUs
  - Concurrency by multiplexing CPUs among them

Operating-System Operations

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

Operating-System Operations

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

Operating-System Operations

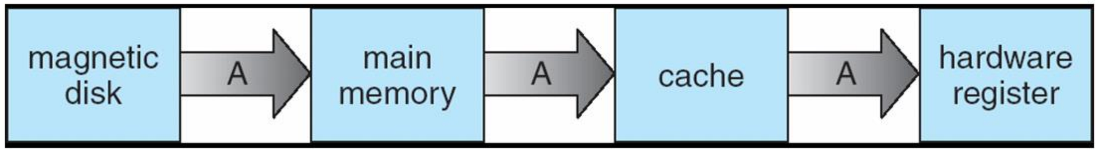
Storage Management

- OS provides uniform, logical view of information storage
  - Various storage devices (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
  - OS abstracts physical properties to logical storage unit - **file**
- File-System management
  - Files usually organized into directories
  - Access control on 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

Operating-System Operations

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory. Entire speed of computer operation hinges on disk subsystem and its algorithms



Migration of Integer A from Disk to Register

- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have most recent value in their cache

Operating-System Operations

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

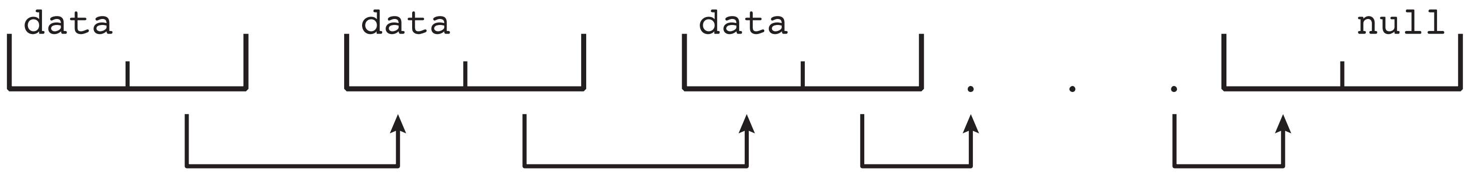
Operating-System Operations

Protection and Security

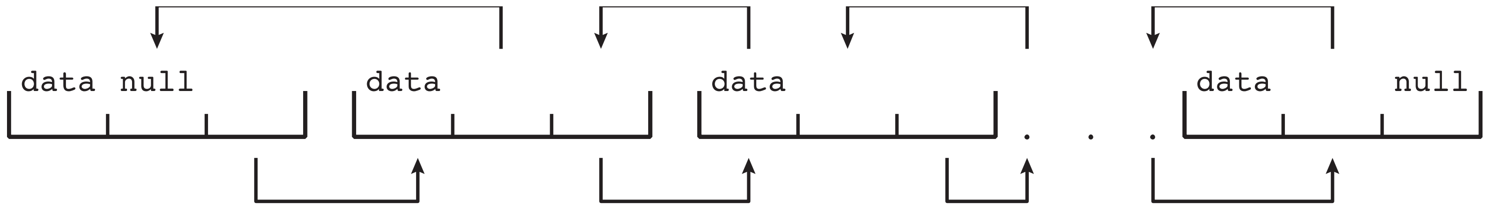
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first 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 have more rights

Kernel Data Structures

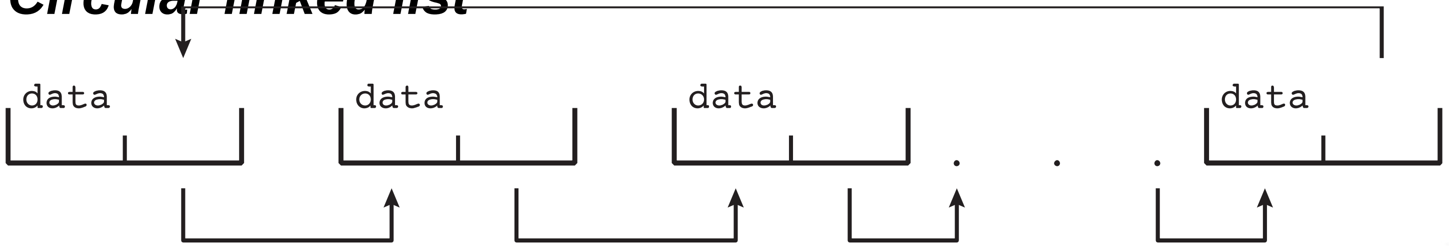
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

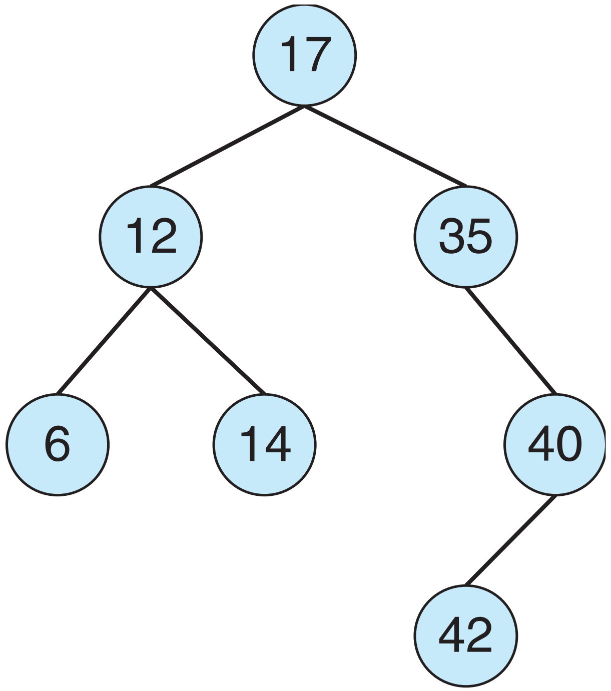


- ***Circular linked list***



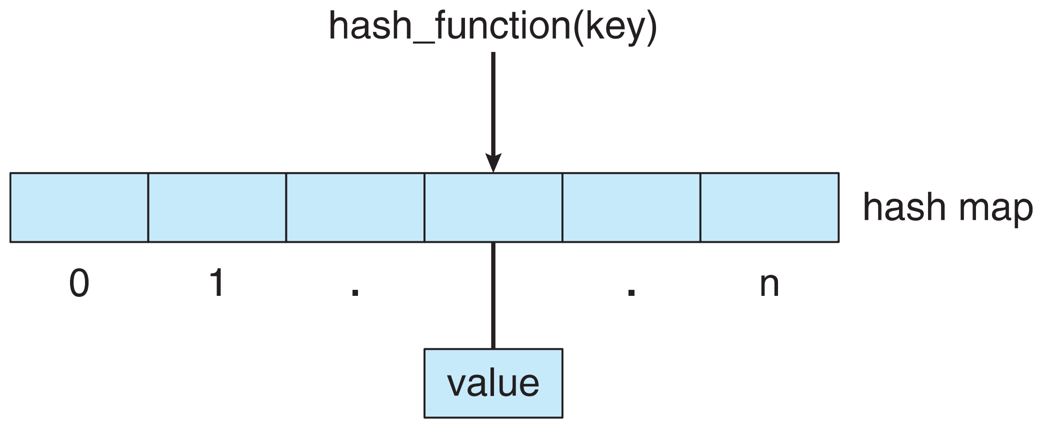
Kernel Data Structures

- **Stack**
  - **Last in first out (LIFO)**
- **Queue**
  - **First in first out (FIFO)**
- **Binary search tree**
  left_child <= right_child
  - **Balanced binary search tree** is *O(lg n)*
    - *Because n items has at most lg n levels.*



Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

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** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration