

# Chapter 1: Introduction

---





# 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





# Objectives

---

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments





# Operating System Goals

---

- Execute user programs and solve user problems easier
- Make the computer system be convenient to use
- Use the computer hardware **efficiently**
  - Limited resources
    - > applications/users need to wait for the resources
  - Most of OS concepts are designed for the efficiency





# What Operating Systems Do

---

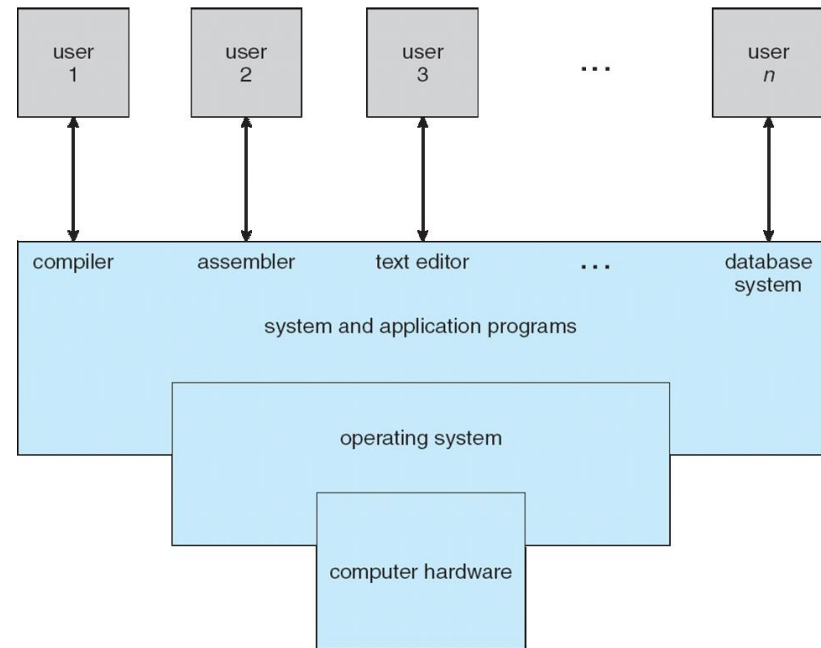
- It differs by the view points
  - User
    - ▶ **Convenience**, **ease of use** and **good performance**
    - ▶ Does not care about resource utilization
  - Shared computer (mainframe or minicomputer)
    - ▶ Keep **all users** be satisfied with their tasks
  - Users of dedicate systems (workstations)
    - ▶ Frequently use shared resources from servers
  - Handheld computers
    - ▶ **Poor resource**
    - ▶ Optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles





# Computer System Structure

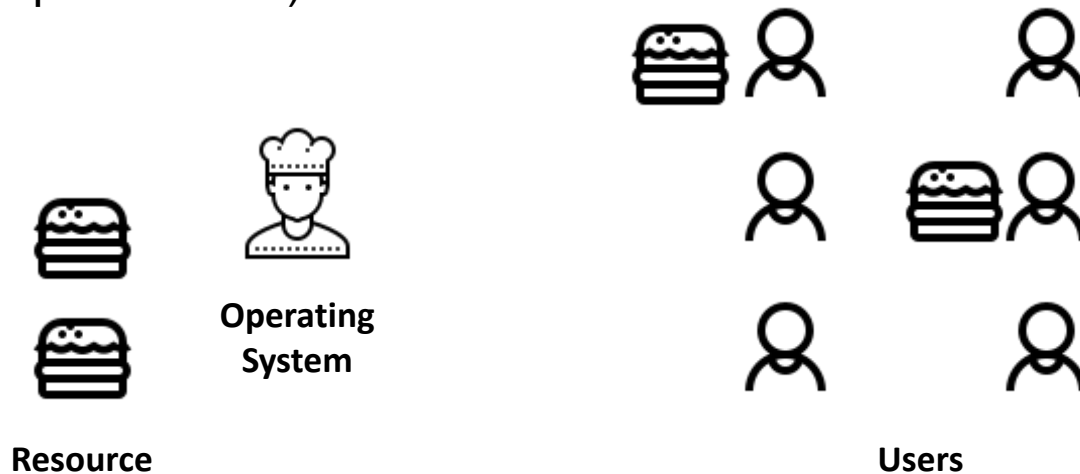
- Computer system can be divided into four components
  - Hardware
    - CPU, memory, I/O devices, ...
  - Operating system
    - Controls and coordinates the use of hardware among various applications and users
  - Application programs
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers



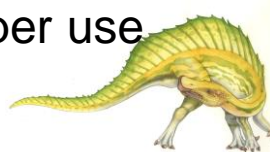


# Operating System Definition

- OS is a **resource allocator**
  - Manages all resources (CPU, memory, network, disk...)
  - For efficient and fair resource use (ex> Which user/process uses the printer first?)



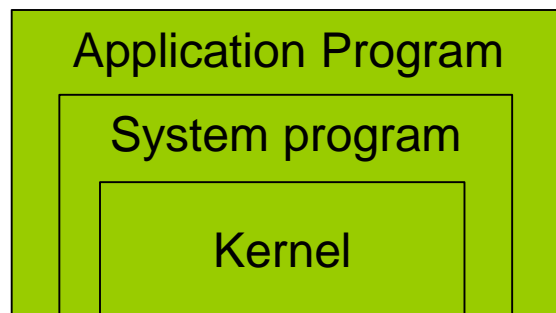
- OS is a **control program**
  - For robust and safety
  - Controls execution of programs to prevent errors and improper use of the computer





# Operating System Definition (Cont.)

- No universally accepted definition exists
- “Everything a vendor ships when you order an operating system” is good approximation, but it varies widely
- “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.

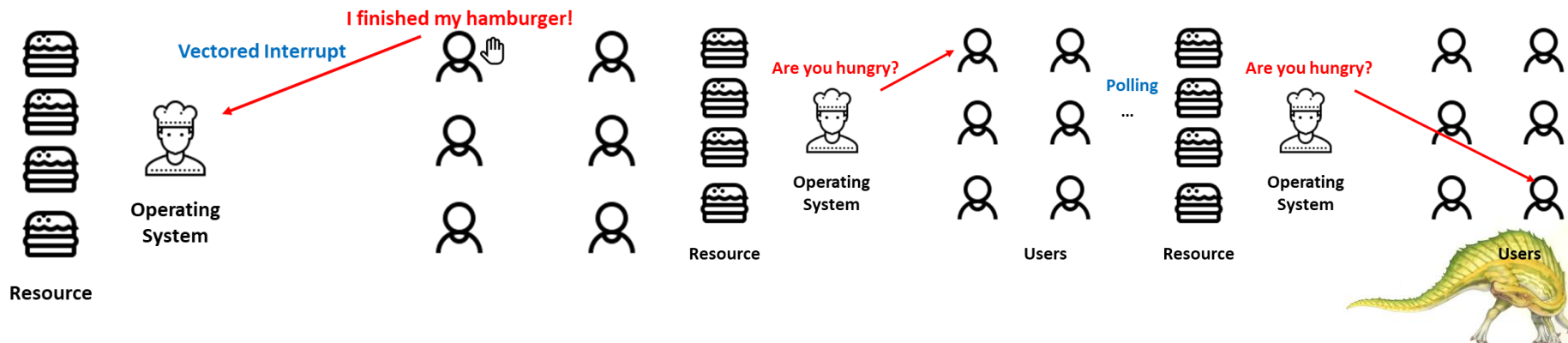






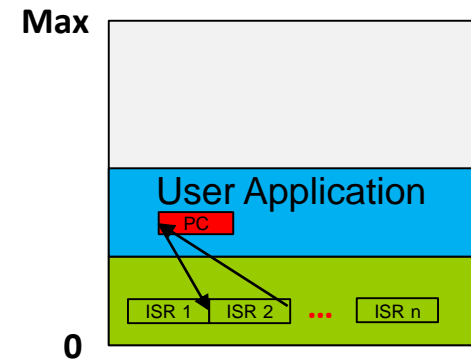
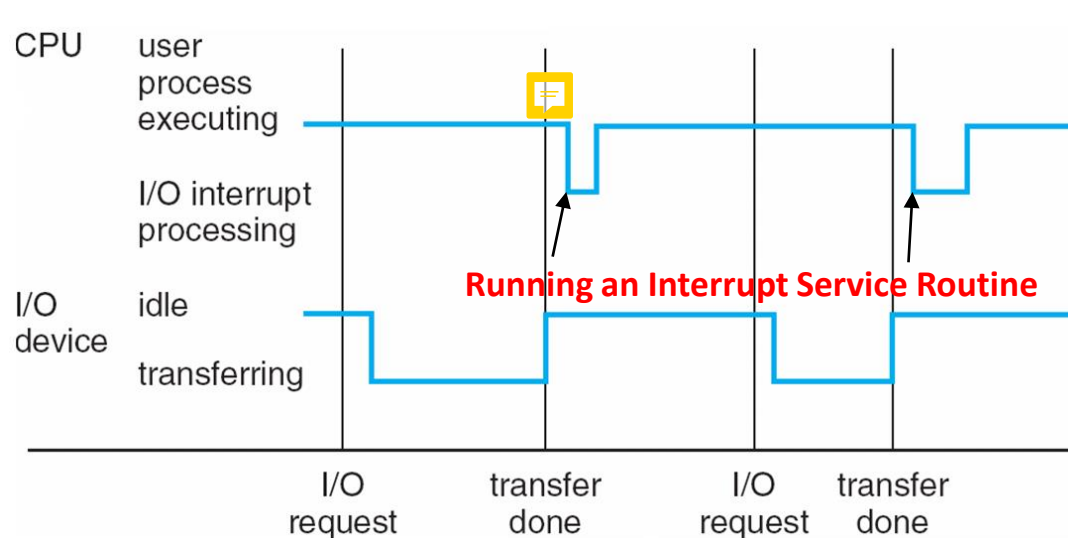
# Interrupt Handling


- I/O request from a user application, a kernel application, or a hardware controller (Ex> file operations, device operations, ...)
- OS stores a state of the current running process by storing registers and the program counter, run the interrupt code, and restore from it
- OS determines which type of interrupt has occurred:
  - **Polling**
  - **Vectored interrupt system**
- OS separates segments of code and determines what action should be taken for each type of interrupt





# Interrupt Timeline



- When an I/O request occurs, I/O device driver transfers data between the device buffer and the device
- Interrupt procedure 
  - 1) Stops a user process from running
  - 2) Moves a program counter (PC) to an interrupt service routine (ISR)
  - 3) Runs an ISR code at PC
  - 4) Returns to the stopped process





# I/O Structure

---

- Synchronous I/O
  - After I/O starts, a process **stops until I/O completion**
    - Ex> `ch = getch(); printf("Key: %c\n", ch);`
    - **No simultaneous** I/O processing
- Asynchronous I/O:
  - After I/O starts, a process runs regardless of I/O completion
    - **Device-status table** contains entry for each I/O device indicating its type, address, and state
    - OS manages I/O device table to determine device status and to modify table entry to include interrupt status
    - **System call**: an interrupt generated by a software
- Device Driver manages I/O to handle controllers of each device
  - Provides uniform interface between controller and kernel





# Storage Definitions and Notation Review

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.

A **kilobyte**, or **KB**, is 1,024 bytes

a **megabyte**, or **MB**, is  $1,024^2$  bytes

a **gigabyte**, or **GB**, is  $1,024^3$  bytes

a **terabyte**, or **TB**, is  $1,024^4$  bytes

a **petabyte**, or **PB**, is  $1,024^5$  bytes



Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).





# Storage Structure

---

- Main memory – the largest storage media that the CPU can access directly
  - **Random access memory** (RAM) 
  - Typically **volatile** 
- 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
  - Solid-state disks – faster than magnetic disks, nonvolatile
    - Various technologies
    - Becoming more popular





# 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

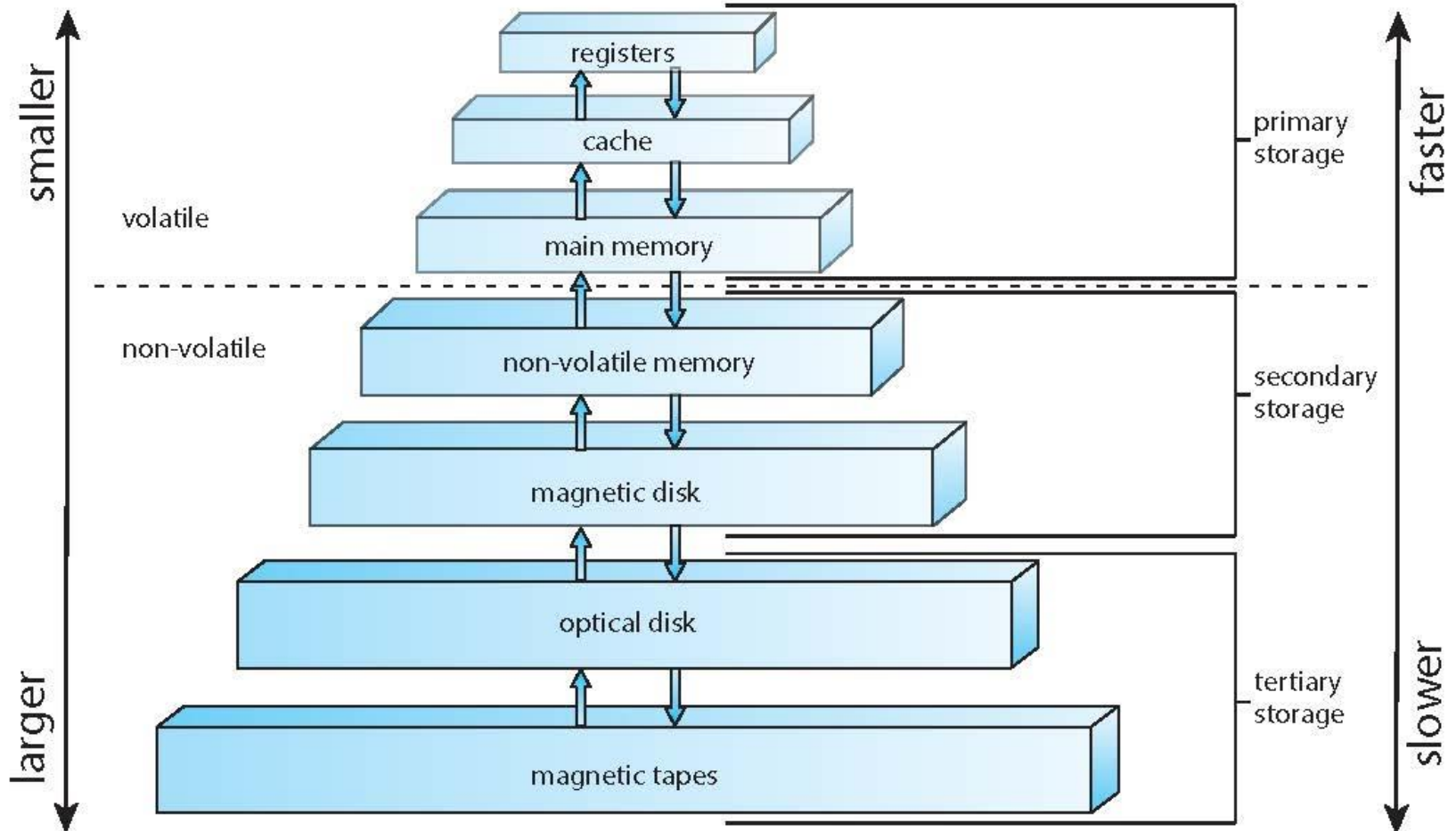




# Storage-Device Hierarchy

storage capacity

access time



Which storage device is the most expensive?





# Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Temporarily copies information in use from slower to faster storage
- When a process asks for data, faster storage (cache) is checked first
  - If there it is (**Hit**), cache data is used directly
    - ▶ Cache -> Register
  - If there is no data in cache (**Miss**), data is copied from a slower storage to the cache and the copied data is used
    - ▶ Cache -> Memory -> Cache -> Register
- Cache size is usually smaller than the storage being cached
  - Cache management: important design problem
  - Cache size and replacement policy

What if cache size is larger than the storage?









# Direct Memory Access (DMA) Structure

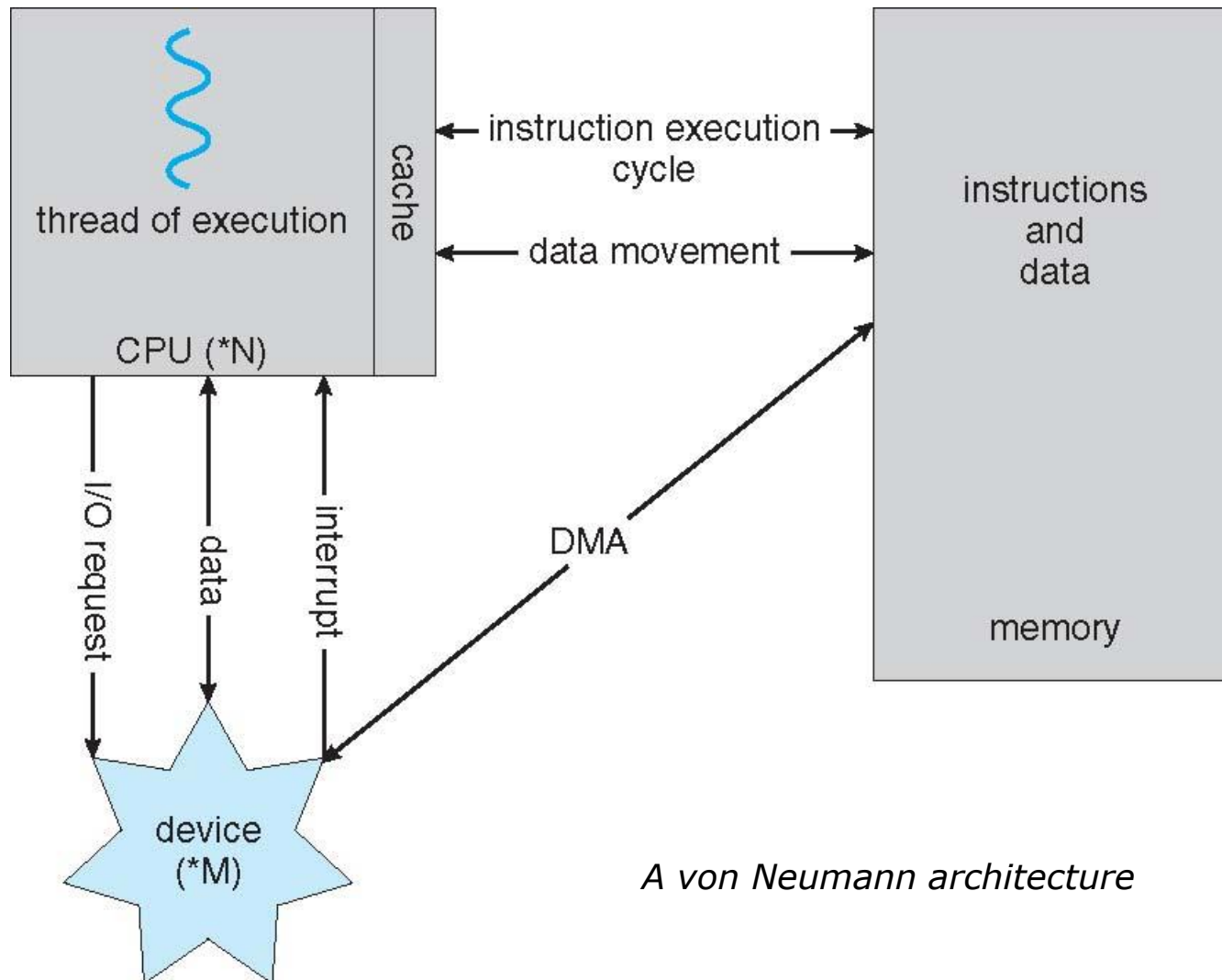
---

- To transmit information at close to memory speeds for high-speed I/O devices
- Device controller transfer blocks of data between buffer storage and main memory directly without CPU intervention 
  - > CPU runs another calculations already by seeing its cache
- Only one interrupt occurs per block, rather than one interrupt per byte 





# How a Modern Computer Works



*A von Neumann architecture*





# Computer-System Architecture

---

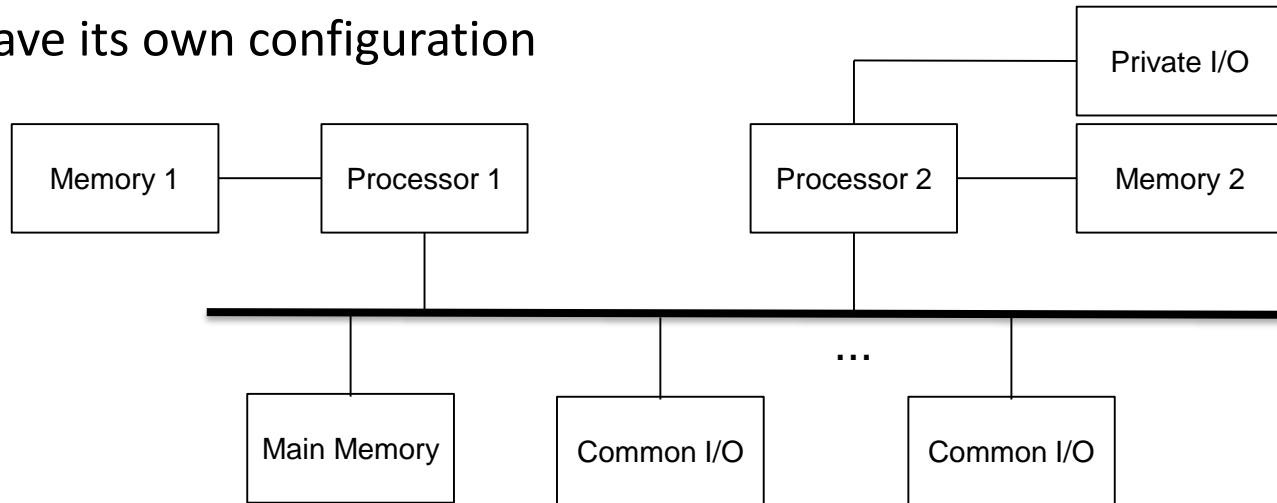
- Most systems use a single general-purpose processor
- Multiprocessor systems are growing
  - Also known as **parallel systems, tightly-coupled systems**
  - Advantages include:
    - Increased throughput
    - Economy of scale
    - Increased reliability – graceful degradation or fault tolerance
  - Two types:
    - **Asymmetric Multiprocessing**
      - > Each processor is assigned to a specific task
    - **Symmetric Multiprocessing (SMP)**
      - > All processors are equal and each processor performs all tasks





# Asymmetric Multiprocessing (AMP)

- In Asymmetric Multiprocessing architecture, a single processor may have its own configuration

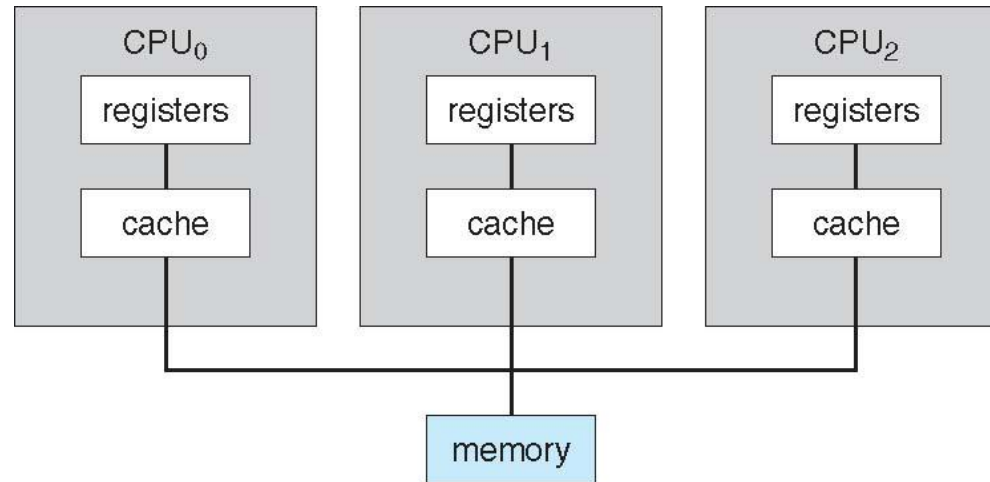


- SMP is a special form of AMP



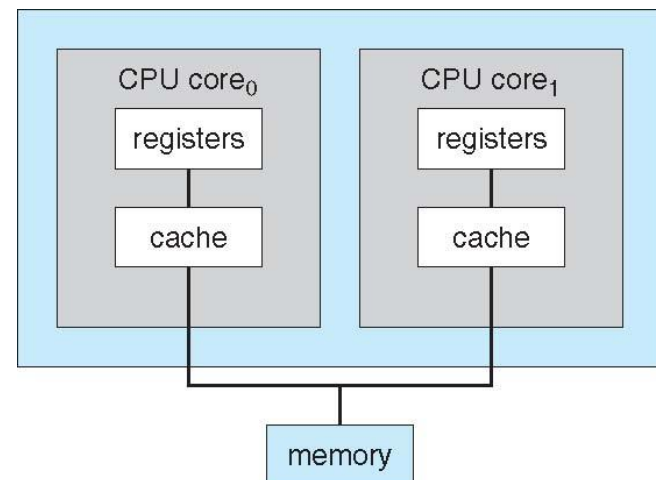


# Symmetric Multiprocessing (SMP) Architecture



## A dual-core design

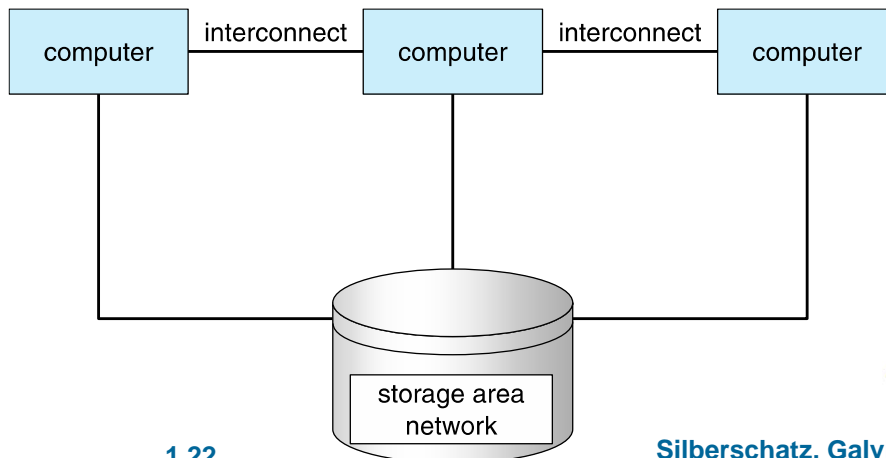
- Multi-chip and multi-core





# Clustered Systems

- Multiple systems work together, **loosely coupled systems**
  - Usually share storages via a **storage-area network** (SAN)
  - Provides a high-availability service which survives from failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are designed for **high-performance computing** (HPC)
    - Applications must be written to use parallelization
  - Some clusters have **distributed lock manager** (DLM) to avoid operation conflicts

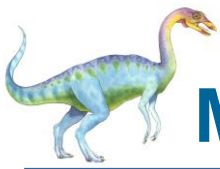




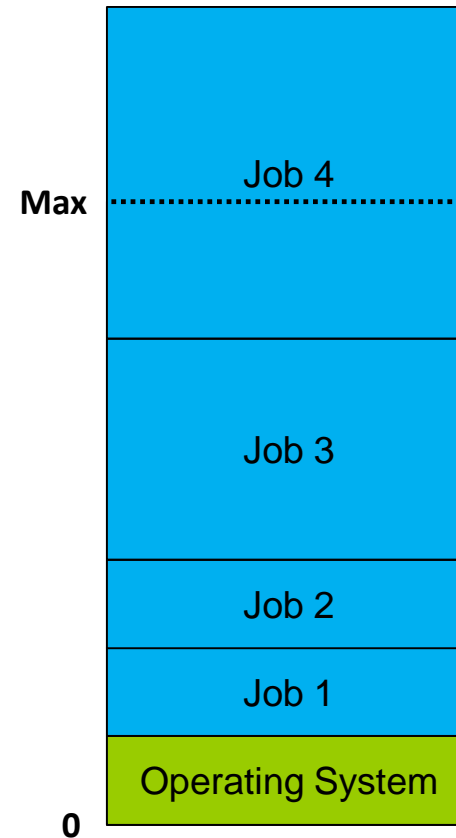
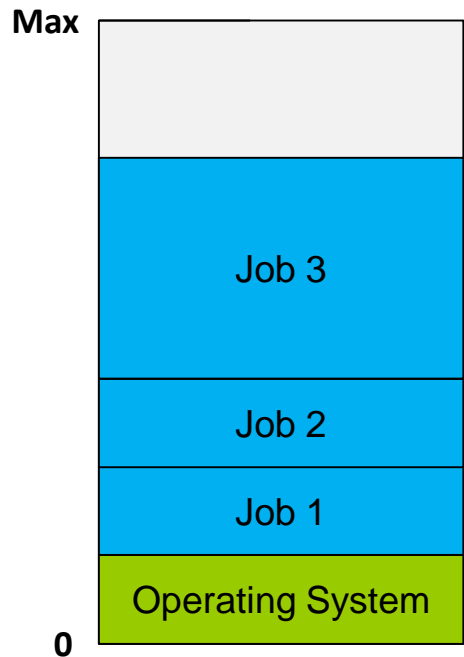
# Operating System Structure

- Multiprogramming for efficiency
  - A single user cannot keep CPU and I/O devices busy at all times
  - **Multiprogramming** organizes jobs so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job is selected and runs by job scheduling
  - When a job has to wait (for synchronous I/O for example), OS switches to another job
- Timesharing (**multitasking**) is a logical extension of multiprogramming
  - CPU switches jobs so frequently that users can interact with each running job
  - Interactive computing
    - Very short response time is essential
    - Each user has at least one program executing in memory ⇒ **process**
    - If several jobs are ready to run at the same time ⇒ **CPU scheduling**
  - Virtual memory allows execution of processes that are not completely loaded in a memory





# Memory Layout for Multiprogrammed System




What if the memory does not fit for all jobs?







# Operating-System Operations

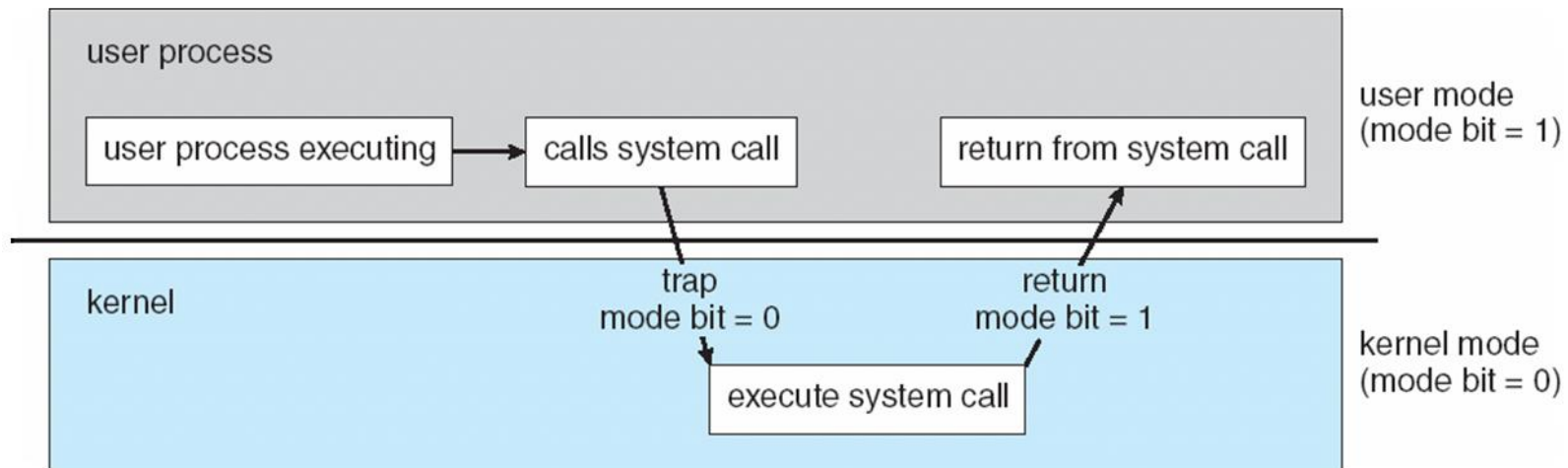
- Interrupt is driven by hardware and usually occurs **asynchronously**
- Software error or request creates exception or trap
  - Division by zero, request for operating system service
- **Dual-mode operation** allows OS to protect itself and other system components 
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware in CPU
    - Provides ability to distinguish when a system is running user code or kernel code
    - Some instructions are designated as privileged and only executable in kernel mode
    - System call changes mode to kernel, and return from call resets the mode to user
- Modern CPUs support multi-mode operations
  - i.e. virtual machine manager (VMM) mode for guest VMs





# Transition from User to Kernel Mode

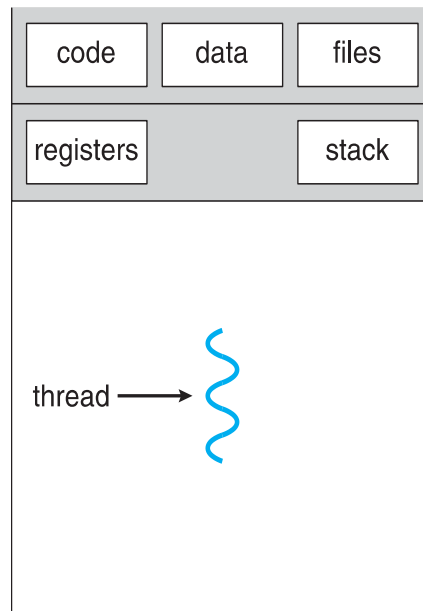
- Example: timer to prevent infinite loop that wasting resources
  - A timer is set to occur an interrupt after a certain duration
  - Operating system sets a counter (privileged instruction)
  - Keep the counter be decremented by the physical clock
  - When the counter becomes zero, internal clock generate an interrupt and OS regains control
  - OS now determines
    - Terminate the program because its time slot is exceeded, or
    - Extends the program's counter



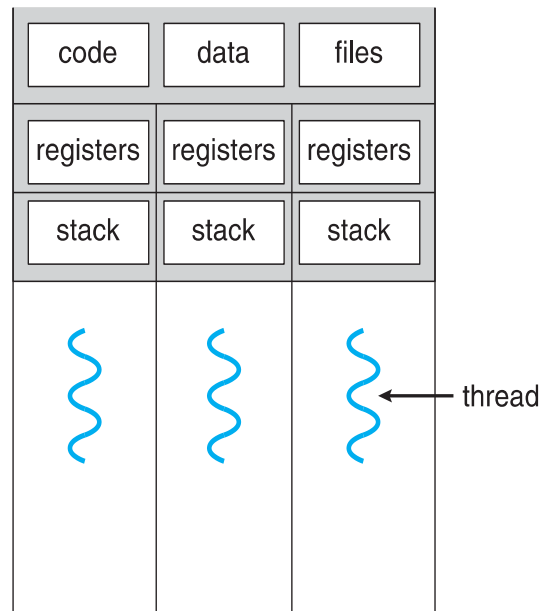


# Process Management

- 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
  - Initialized data



single-threaded process



multithreaded process





# Process Management (cont.)

---

- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - CPU loads codes or data from main memory at where PC points
- A multi-threaded process has **one program counter per thread**
  - In other words, we call PC as a context
- Typically system has many processes, some users, some operating systems 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
  - Process communication
  - Deadlock handling





# Memory Management

---

- Running a program means that all or part of instructions should be in memory
  - Processes can access data if and only if they are in memory
- Memory management determines what to store in main memory
  - To optimize CPU utilization and faster computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory (**swapping**)
  - Allocating and deallocating memory space as needed





# Storage Management

---

- OS provides uniform, logical view of information storage
  - Physically, storage handles binary data stream
  - Abstracts physical properties to logical storage unit - **file**
  - Each device include varying properties of access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories for easier interpretation
  - **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 directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media





# Mass-Storage Management

---

- Usually disks are used to store 1) data that does not fit in main memory or 2) data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation **hinges** on disk subsystem and its algorithms
  - This is why SSD drastically boosts up the performance
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling (the order of disk I/Os)
- Some storage does not require fast speed
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)







# Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

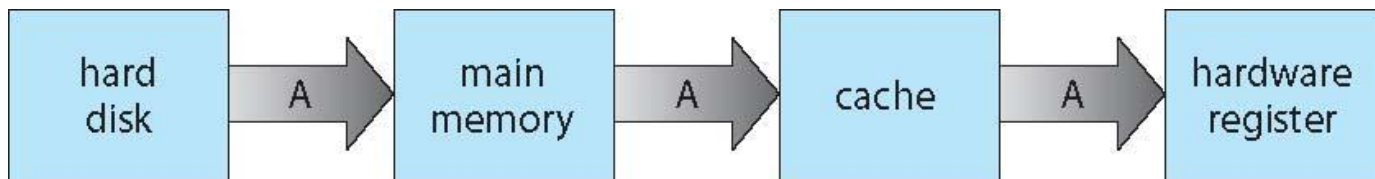
Movement between levels of storage hierarchy can be explicit or implicit





# Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use the most recent value, no matter where it is stored in the storage hierarchy



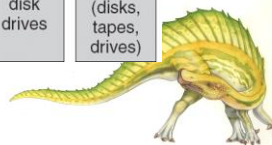
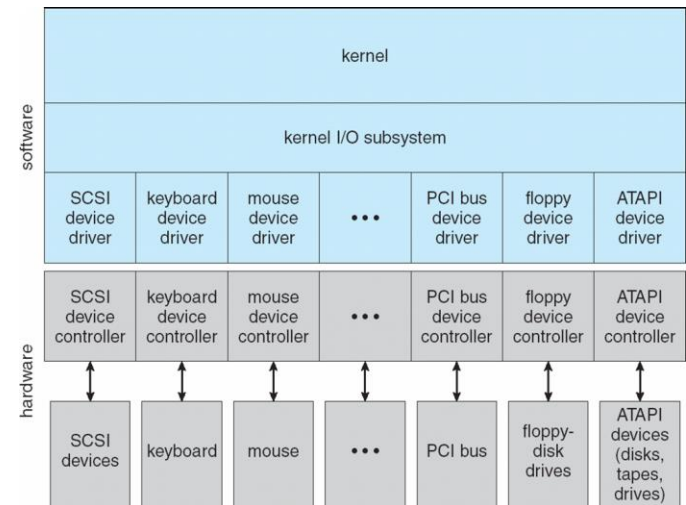
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache memories
- Distributed environment situation is even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17





# I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem is 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



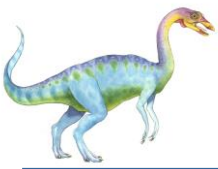


# Protection and Security

---

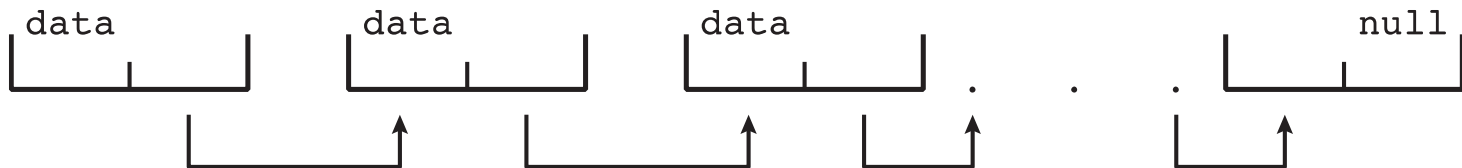
- **Protection** – any mechanism for controlling access of processes or users to resources defined by OS
- **Security** – defense of the system against internal/external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish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



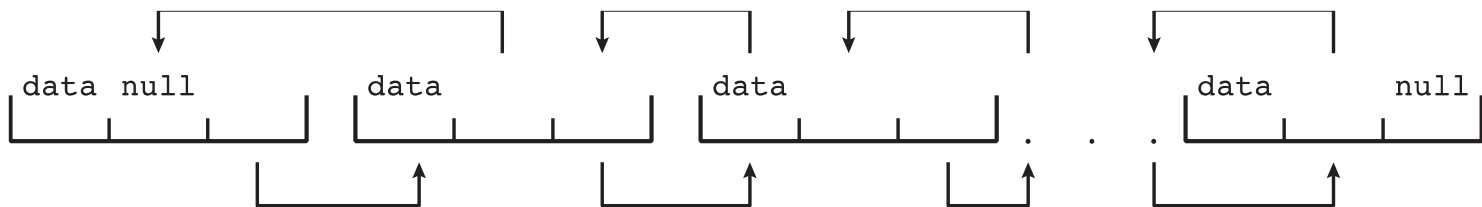


# Kernel Data Structures

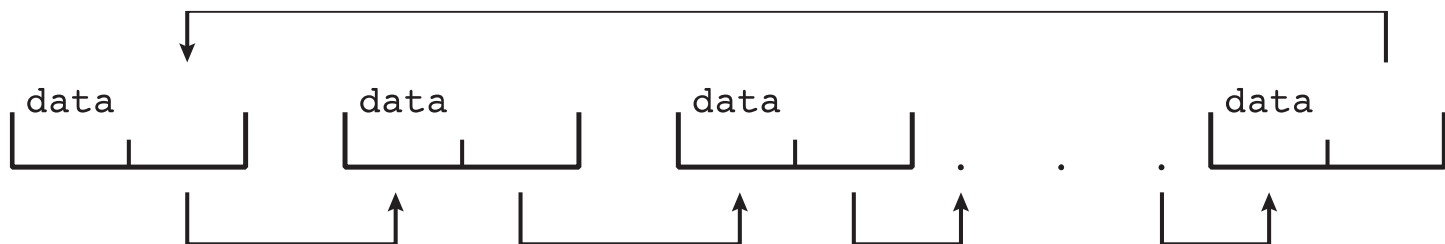
- Many similar to standard programming data structures
- Singly linked list



- Doubly linked list



- Circular linked list



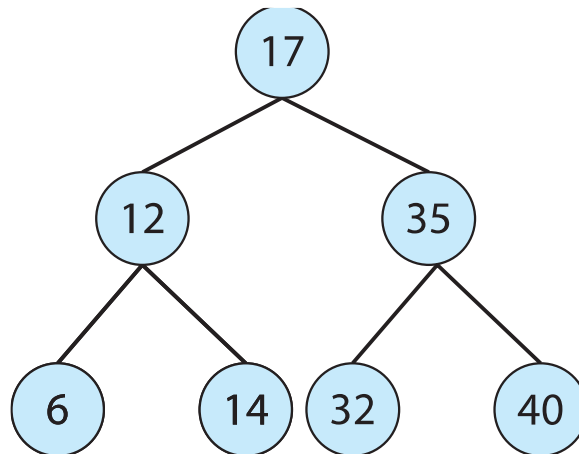


# Kernel Data Structures

- **Binary search tree**

left  $\leq$  right

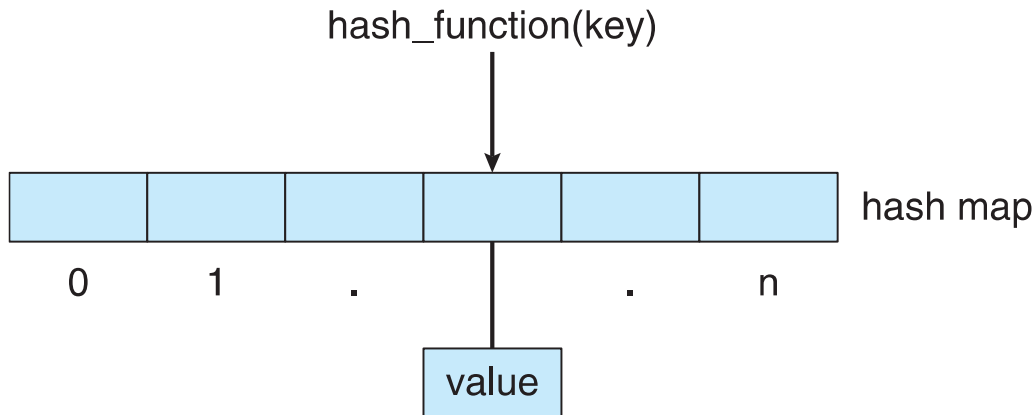
- Search performance is  $O(n)$
- Balanced binary search tree is  $O(\lg n)$





# Kernel Data Structures

- **Hash function** can create a hash map (Virtual Memory)



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





# Computing Environments - Traditional

---

- Stand-alone general purpose machines
- But has been blurred as most systems are becoming interconnected with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (thin clients) are like web terminals
- Mobile computers interconnect via wireless networks
- Networking is becoming ubiquitous – even home systems use firewalls to protect home computers from network attacks via internet







# Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?



?

=



- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like augmented reality
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**





# Computing Environments – Distributed

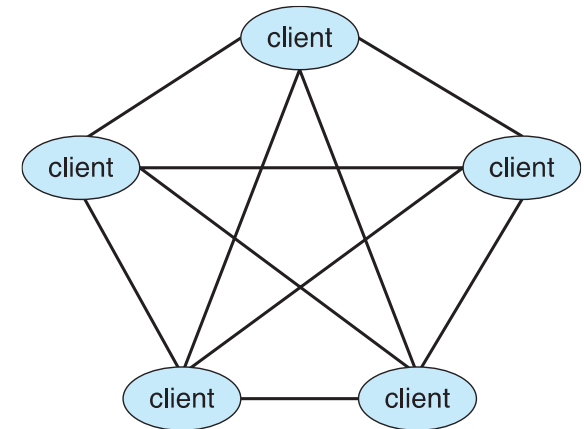
- Distributed computing
  - Collection of **separate**, possibly **heterogeneous**, systems networked together
    - ▶ Network is a communications path, TCP/IP most common
      - Local Area Network (LAN)
      - Wide Area Network (WAN)
      - Metropolitan Area Network (MAN)
      - Personal Area Network (PAN)
  - **Network Operating System** provides features between systems across network
    - ▶ Communication scheme allows systems to exchange messages
    - ▶ Illusion of a single system





# Computing Environments - Peer-to-Peer

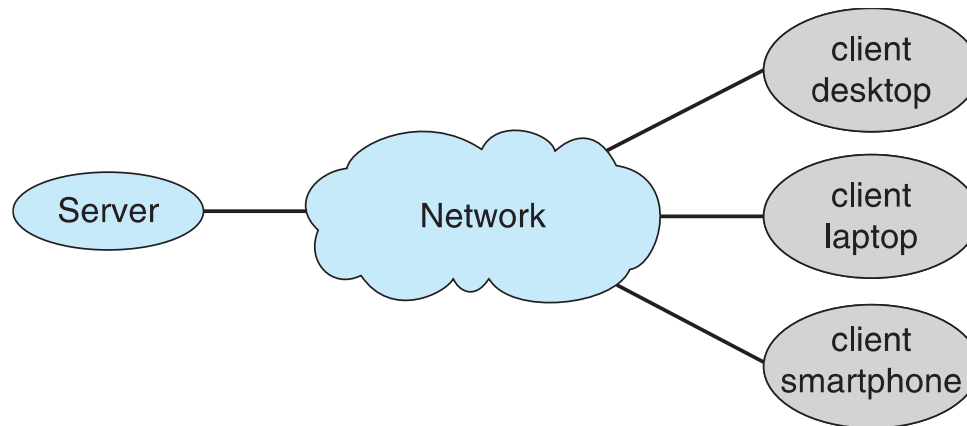
- One of distributed system models
- P2P does not distinguish clients and servers
  - All nodes are considered peers
  - Each may act as client, server or both
  - Each node must join in P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via **discovery protocol**
  - Examples include Napster and Gnutella, **Voice over IP** (VoIP) such as Skype





# Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now servers, responding to requests generated by clients
    - Compute-server system provides an interface to client to request services (i.e., database, high-power computing)
    - File-server system provides interface for clients to store and retrieve files





# Computing Environments - Virtualization

---

- Allows operating systems to run applications within other OSES
  - Vast and growing industry
- **Emulation**: when the source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally **slowest** method
  - When computer language is not compiled to native code – Interpretation
- **Virtualization**: OS is natively compiled for CPU, and running guest OSES that are also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP host OS
  - **VMM** (virtual machine manager) provides virtualization services





# Computing Environments - Virtualization

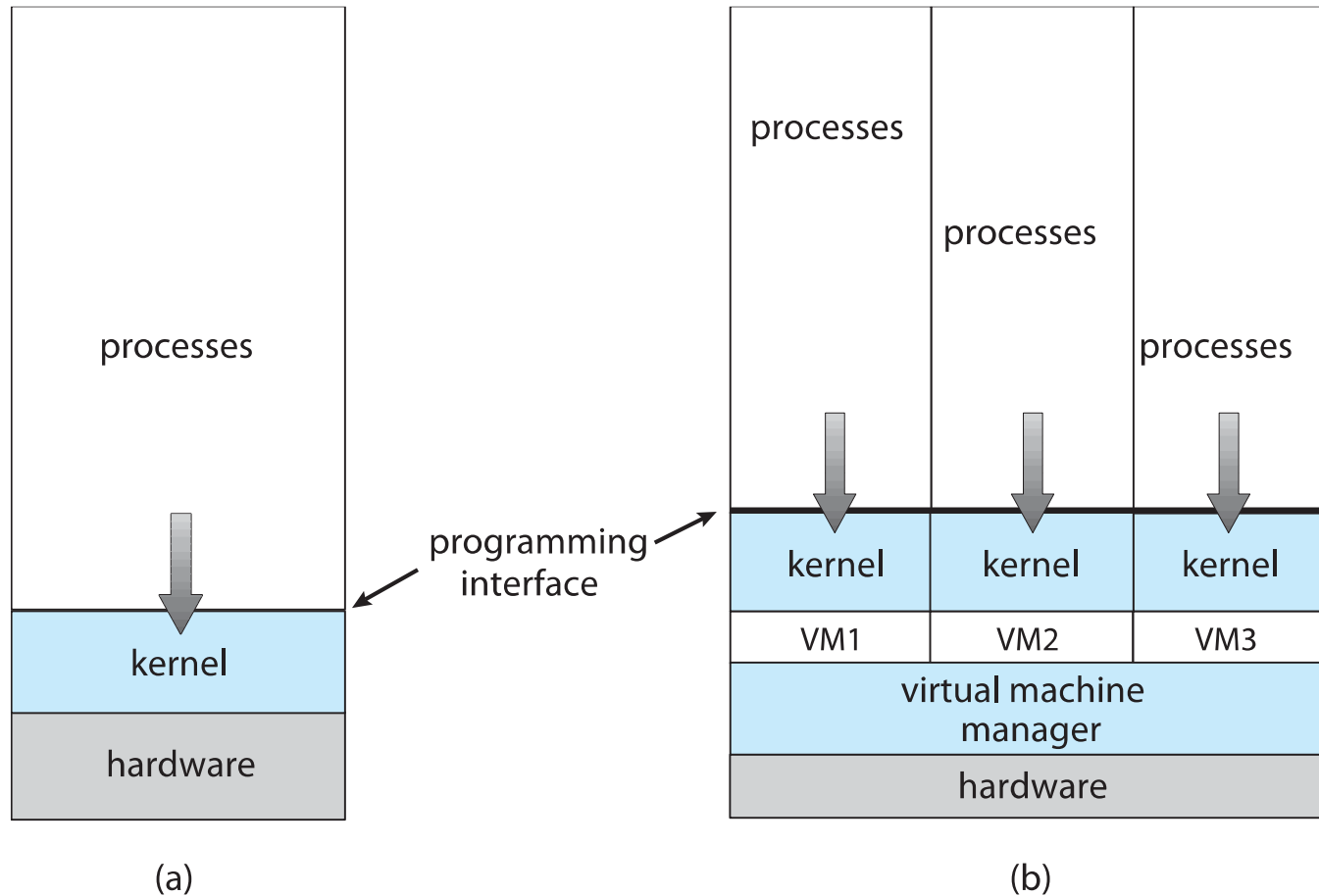
---

- Use cases include laptops and desktops running multiple OSES for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSES without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer no longer run on host operating systems but rather are the hosts)





# Computing Environments - Virtualization





# Computing Environments – Cloud Computing

---

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for its functionality
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application development via the Internet (i.e., a compiler, a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage resources available over Internet (i.e., storage available for backup use)

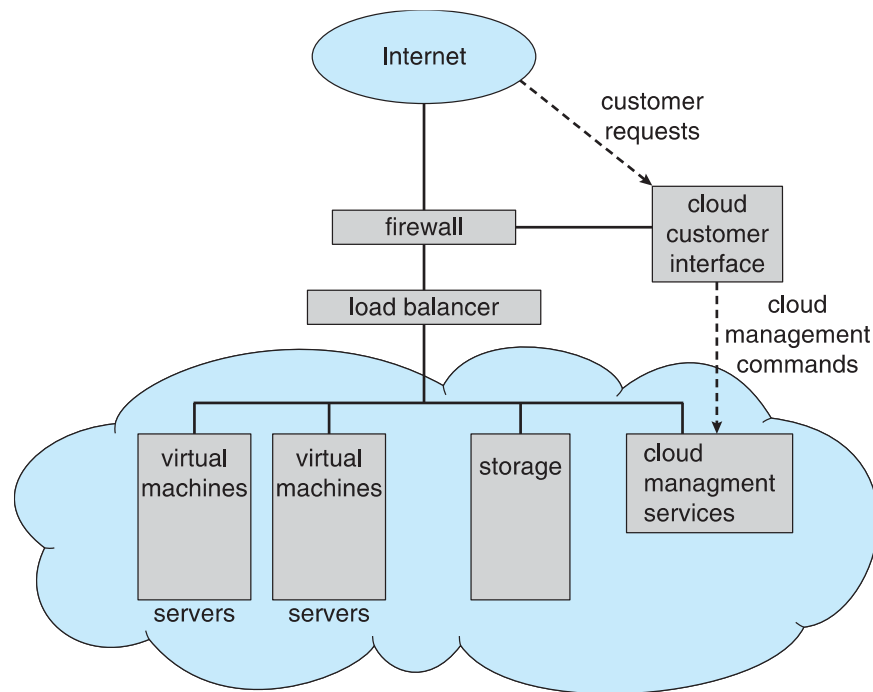






# Computing Environments – Cloud Computing

- Cloud computing environments composed of traditional OSES, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications





# Summary

---

- This course will mainly handle how OS generally operates a local computer
  - System Calls
  - Process
  - Thread
  - Synchronization
  - Main Memory
  - Virtual Memory
  - File System

