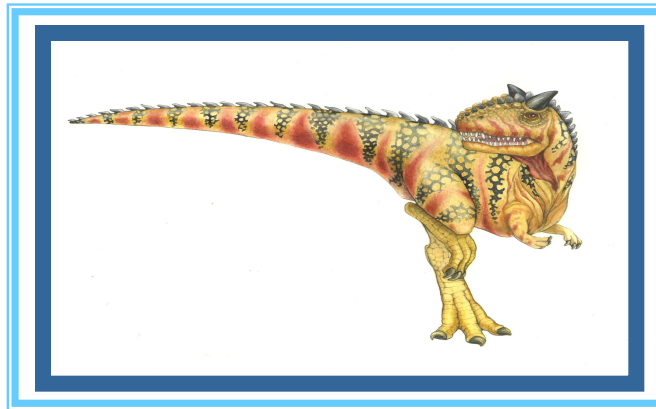


Chapter 1: Introduction





Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems





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
- To explore several open-source operating systems





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





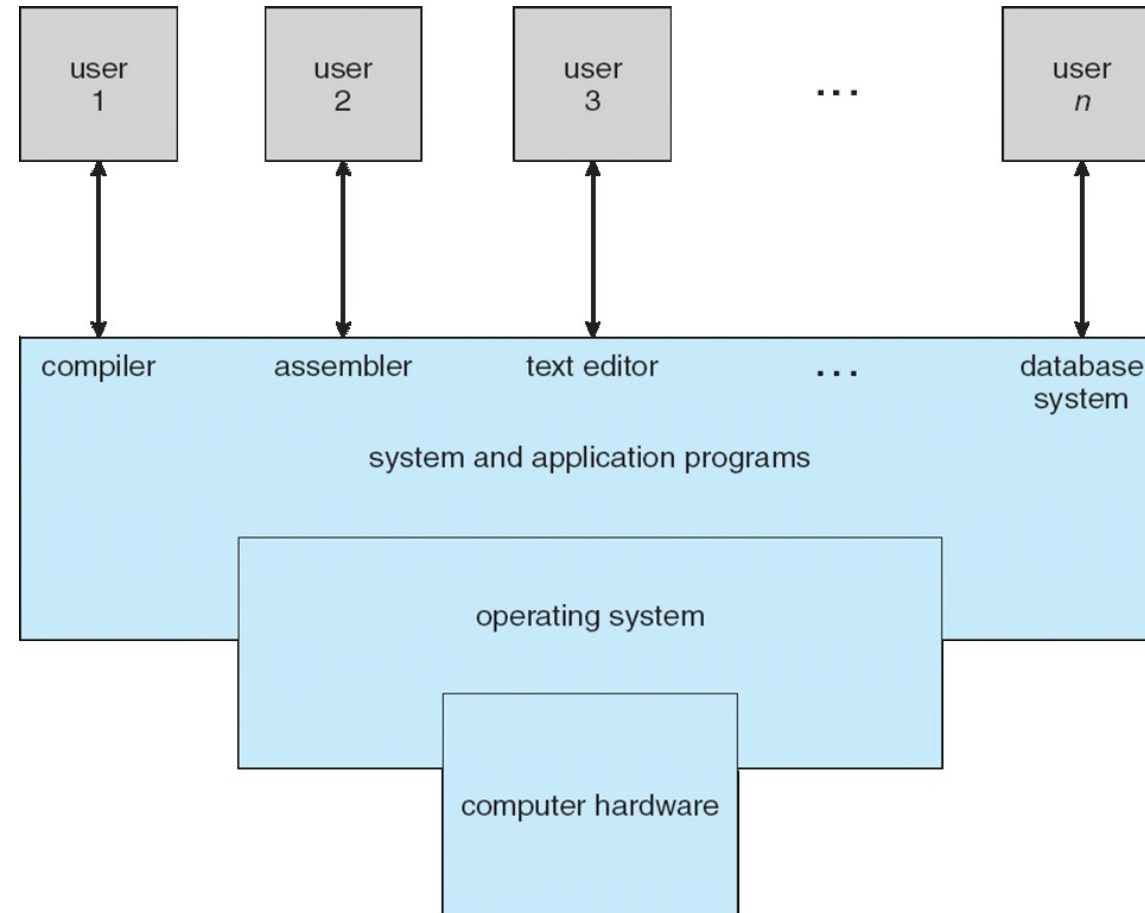
Computer System Structure

- 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





What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
 - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- 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





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

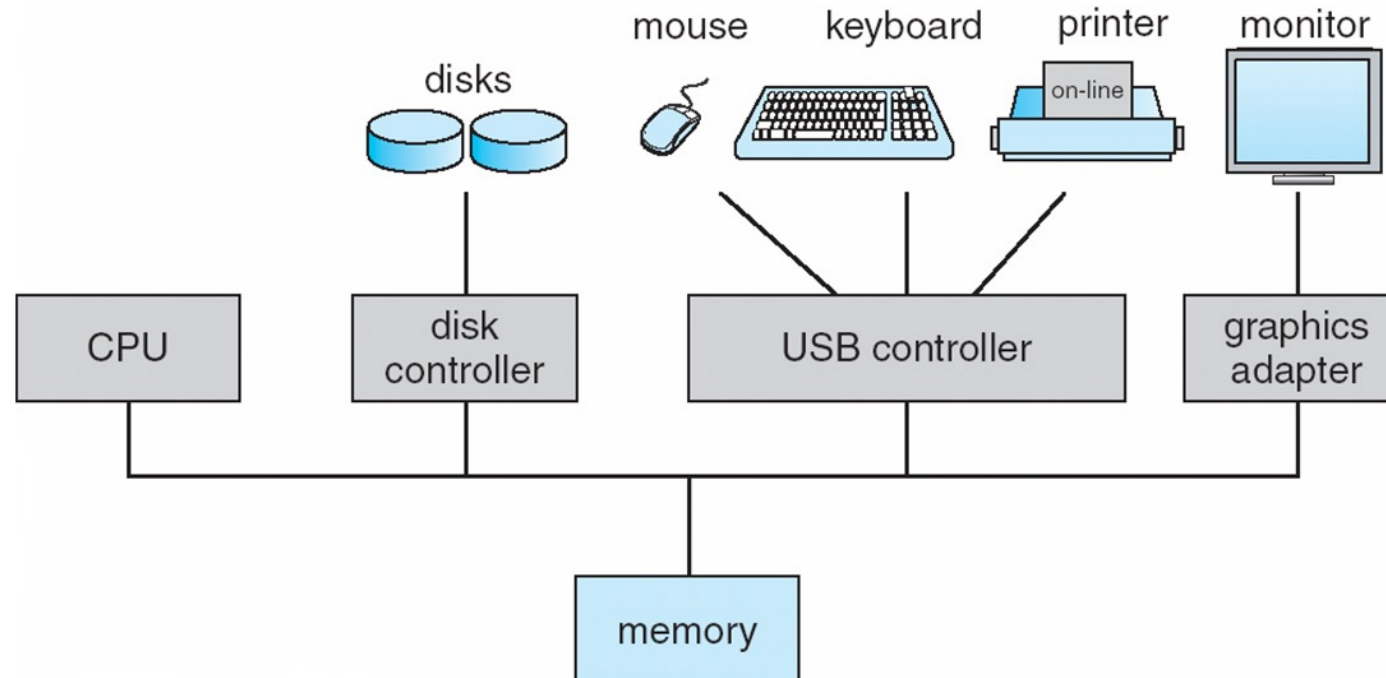
- **bootstrap program** is loaded at power-up or reboot
 - Typically stored in ROM or EPROM, generally known as **firmware**
 - Initializes all aspects of system
 - Loads operating system kernel and starts execution





Computer System Organization

- Computer-system operation
 - 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**





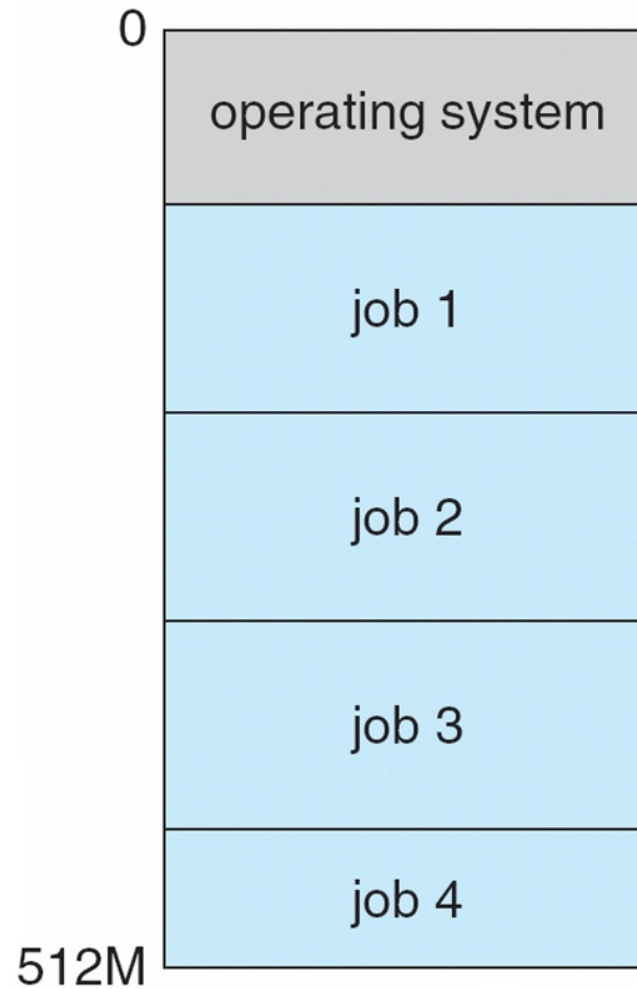
Operating System Structure

- **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





Memory Layout for Multiprogrammed System





Operating-System Operations

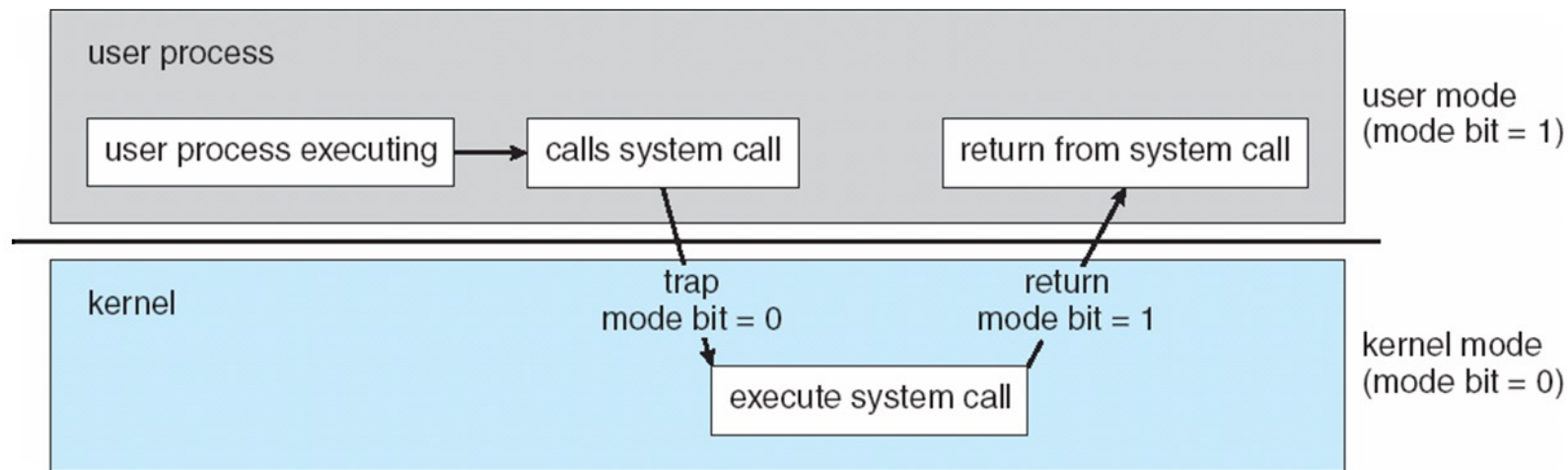
- **Interrupt driven** by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect 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
- Increasingly CPUs support multi-mode operations
 - i.e. **virtual machine manager (VMM)** mode for guest **VMs**





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 or terminate program that exceeds allotted time

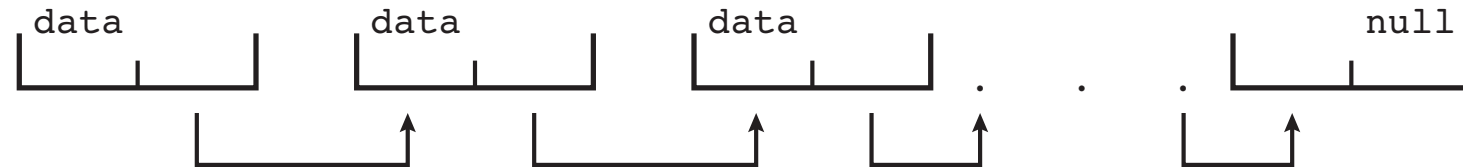




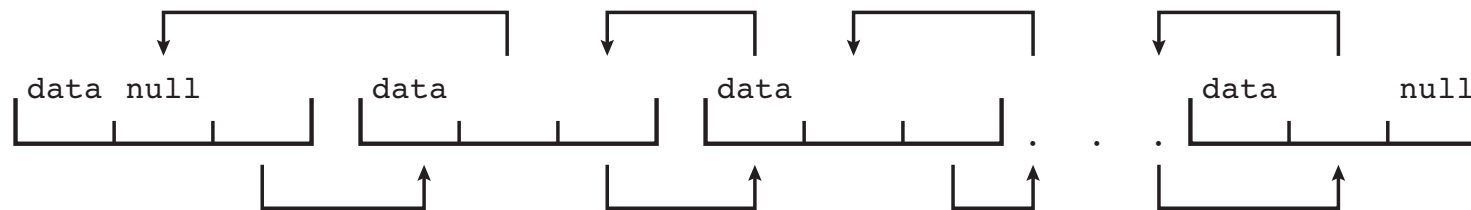
Kernel Data Structures

- Many similar to standard programming data structures

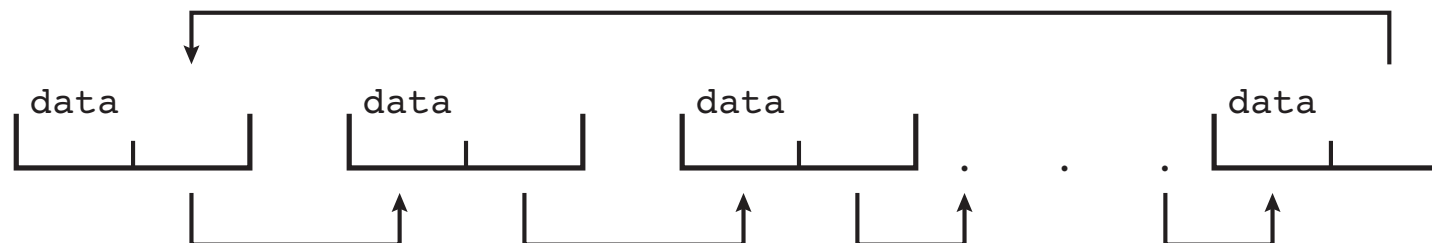
- ***Singly linked list***



- ***Doubly linked list***



- ***Circular linked list***



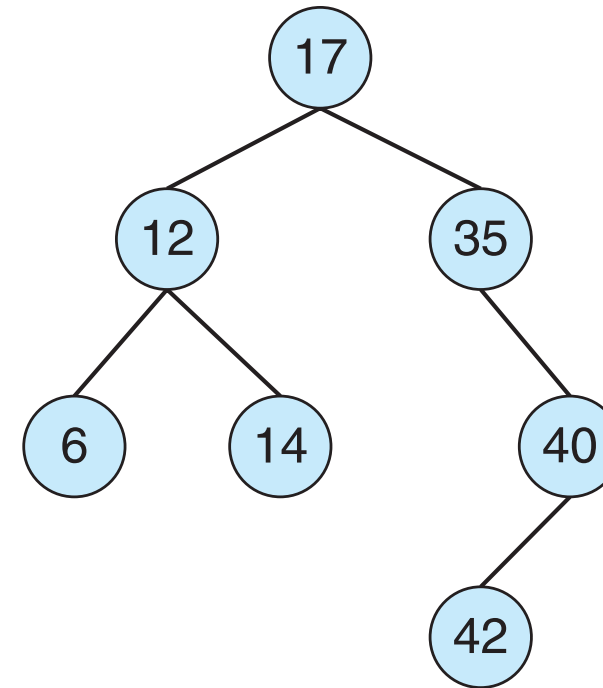


Kernel Data Structures

■ Binary search tree

left \leq right

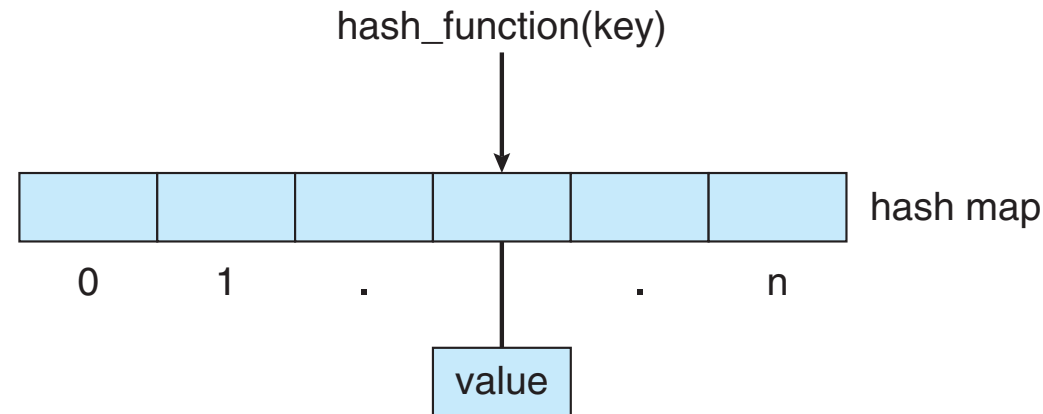
- Search performance is $O(n)$
- **Balanced binary search tree** is $O(\lg n)$





Kernel Data Structures

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



- **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>`





Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect 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 becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks





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

- 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

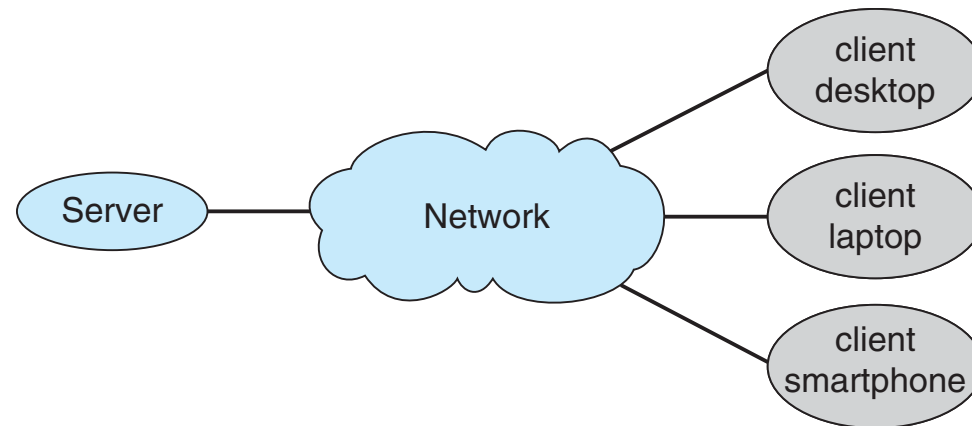




Computing Environments – Client-Server

■ Client-Server Computing

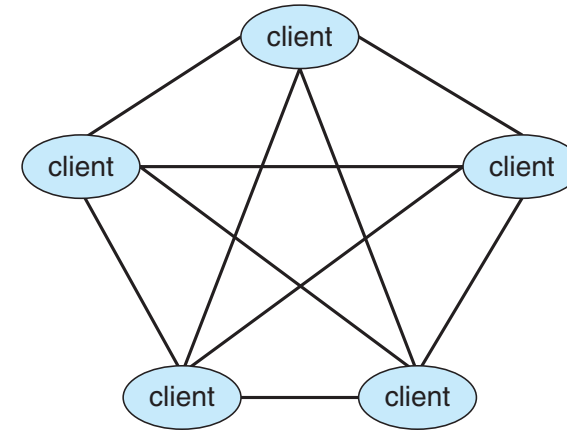
- Dumb terminals supplanted by smart PCs
- Many systems now **servers**, responding to requests generated by **clients**
 - ▶ **Compute-server system** provides an interface to client to request services (i.e., database)
 - ▶ **File-server system** provides interface for clients to store and retrieve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





Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
 - Instead all nodes are considered peers
 - May each act as client, server or both
 - Node must join 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 - Virtualization

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





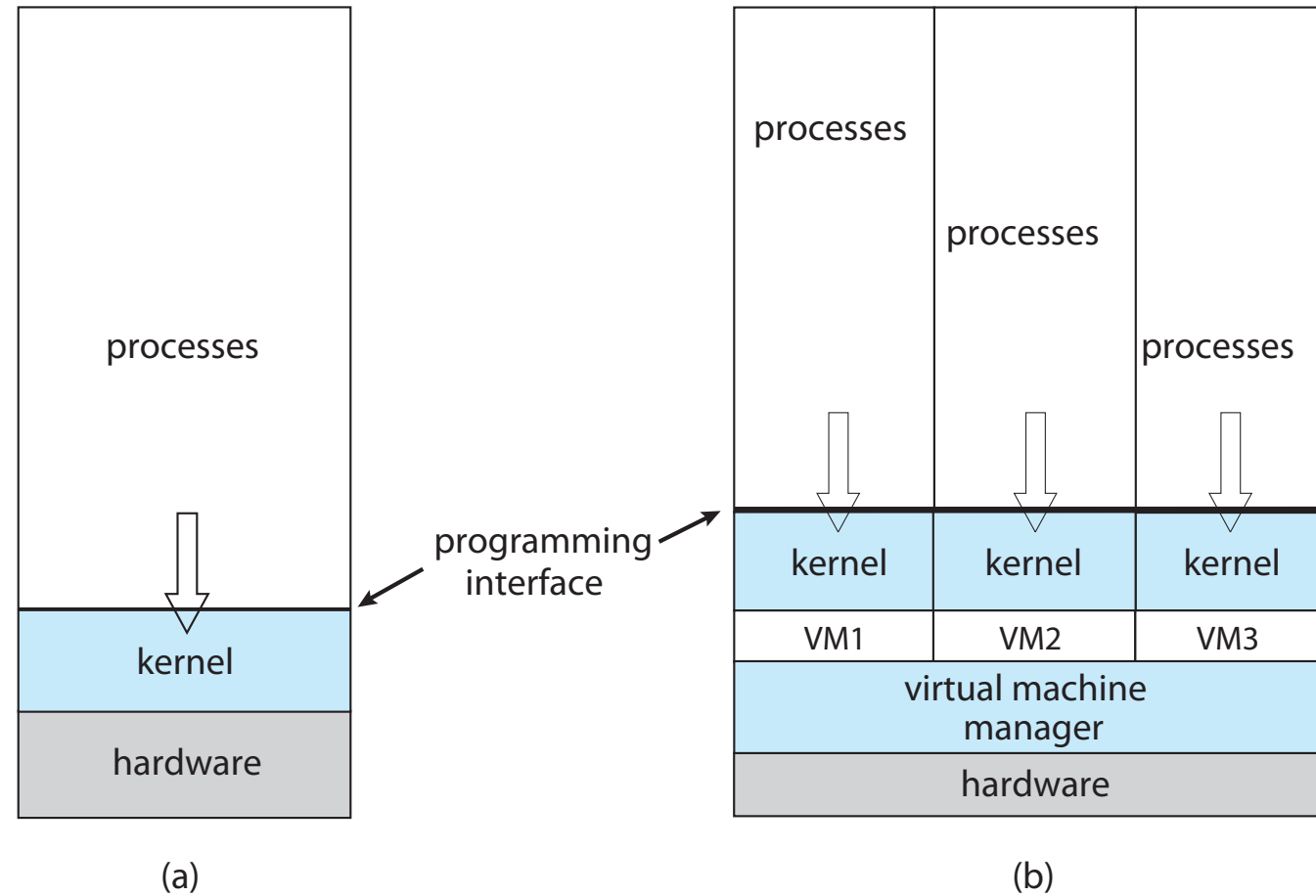
Computing Environments - Virtualization

- Use cases involve 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)





Computing Environments - Virtualization





Computing Environments – Cloud Computing

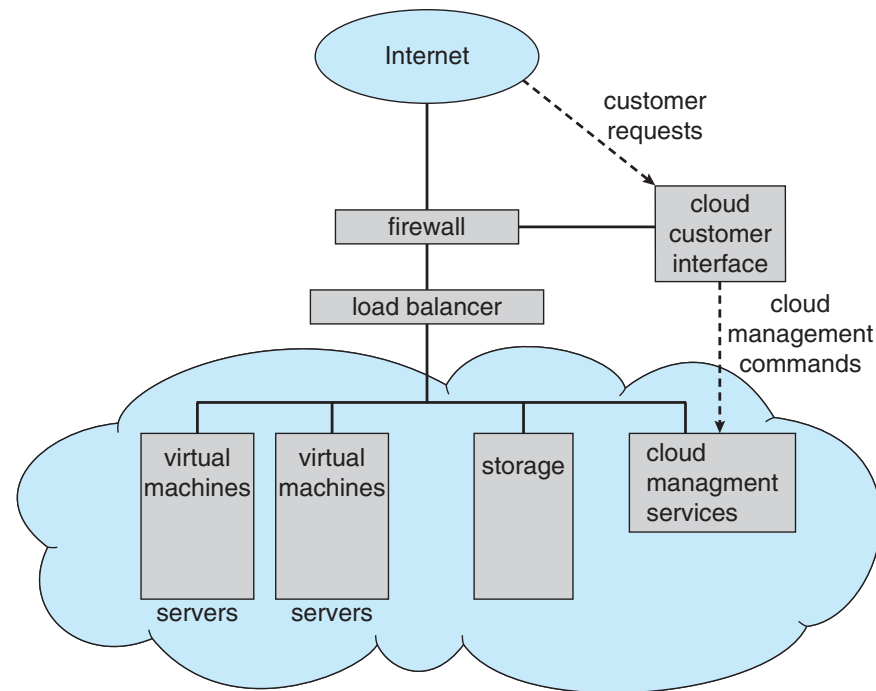
- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
 - Amazon **EC2** has thousands of servers, millions of VMs, PBs of storage available across the Internet, pay based on usage
- Many types
 - **Public cloud** – available 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 use via the Internet (i.e. a database server)
 - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e. storage available for backup use)





Computing Environments – Cloud Computing

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





Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
 - Vary considerable, special purpose, limited purpose OS, **real-time OS**
 - Use expanding
- Many other special computing environments as well
 - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
 - Processing ***must*** be done within constraint
 - Correct operation only if constraints met





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

