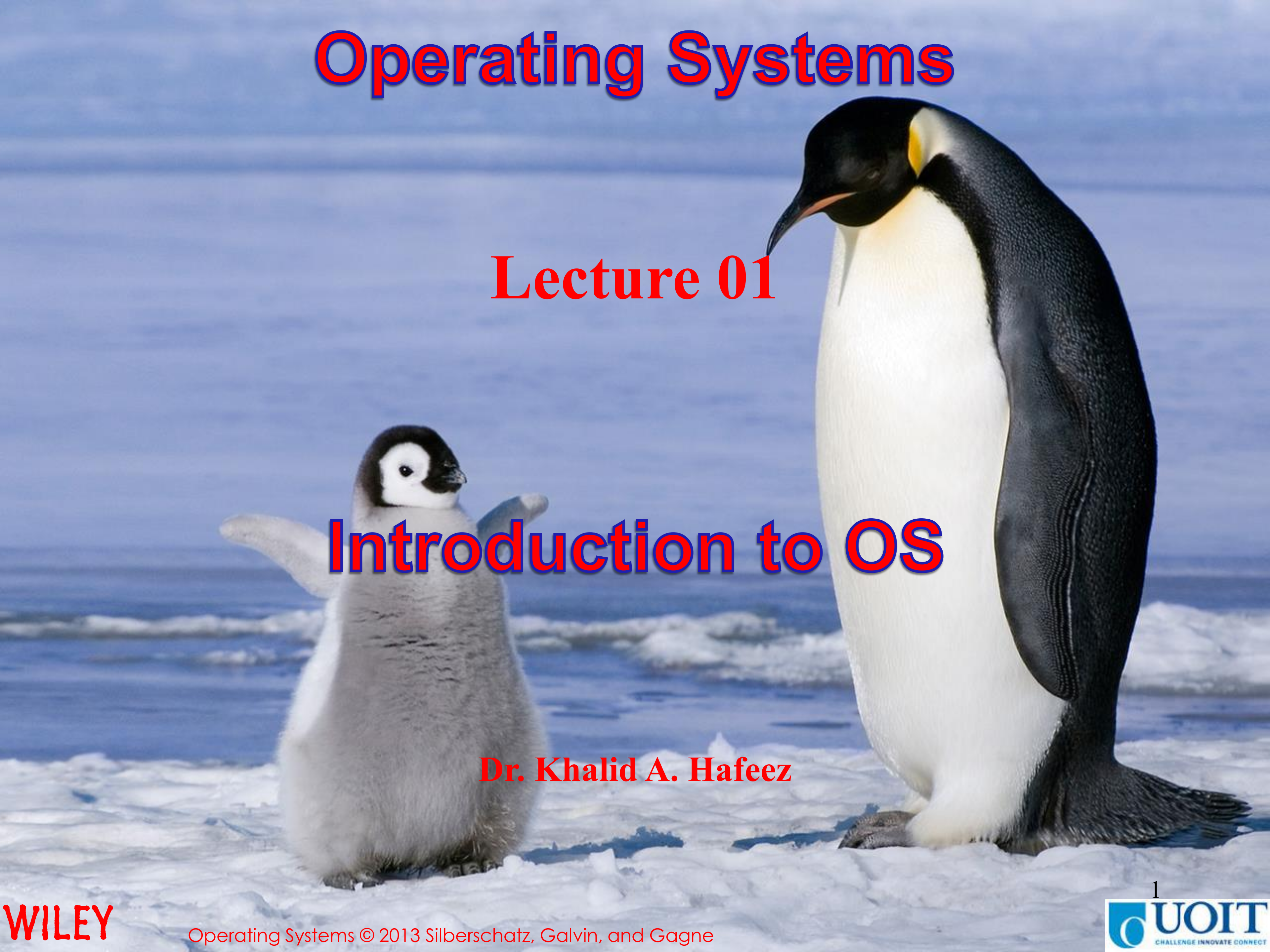


Operating Systems

Lecture 01

Introduction to OS

Dr. Khalid A. Hafeez

Objectives

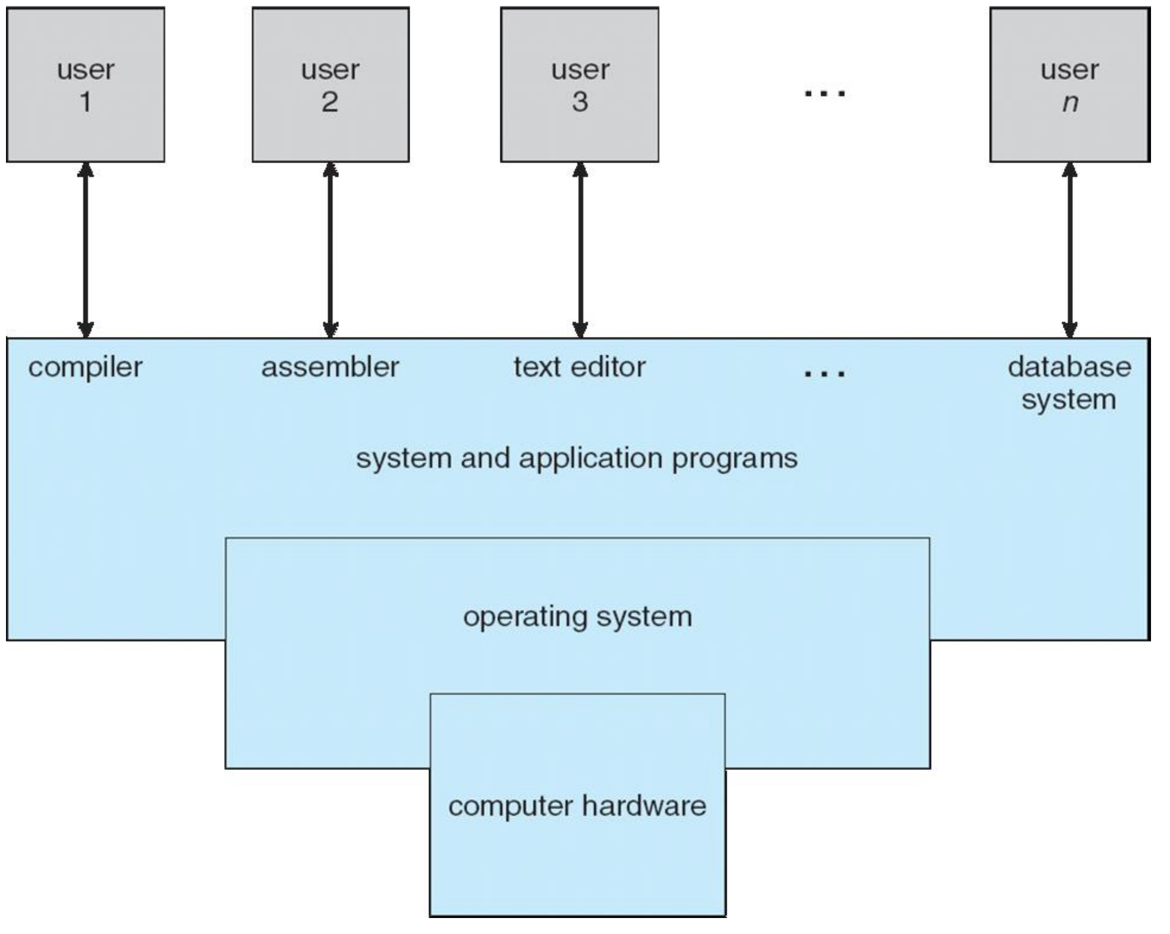
- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

Introduction

- 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
  - *Some operating systems are designed to be convenient, others to be efficient, and others to be some combination of the two.*

Computer System Structure

- Computer system can be divided into four components:



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

What Operating Systems Do

- Exploring the OSes from two viewpoints:
  - **User view:**
    - Users want convenience, **ease of use** and **good performance**
      - Don't care about **resource utilization**
    - But shared computer such as **mainframe** or **minicomputer** must keep all users happy
    - Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
    - Handheld computers (smartphones) 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

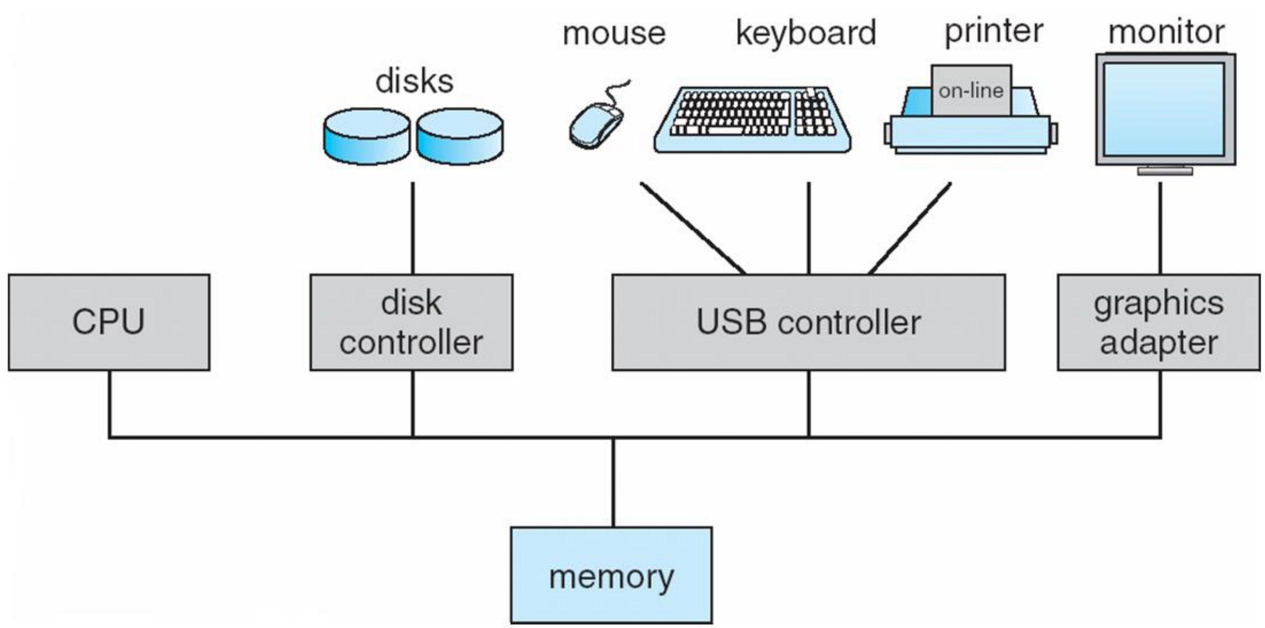
- Exploring the OSes from two viewpoints:
  - **System view:**
    - The OS is the program that most intimately involved with the hardware
    - We can view the OS as a:
      - Resource allocator
        - Manages all resources
        - Decides between conflicting requests for efficient and fair resource use
      - Control program
        - Controls the execution of user programs to prevent errors and improper use of the computer

What is Operating System?

- Operating System Definition:
  - No universally accepted definition
  - "Everything a vendor ships when you order an operating system" is a good approximation, but varies wildly
  - Computer hardware is constructed toward executing user programs.
  - Since bare hardware alone is not easy to use, application programs are developed.
  - These programs require certain common operations, such as those controlling the I/O devices.
  - The common functions of controlling and allocating resources are then brought together into one piece of software: the operating system.
  - Common definition:
    - "The one program running at all times on the computer" is the **kernel**.
    - And two other types of programs:
      - a system program (ships with the operating system) ,
      - an application program.

Computer System Organization

- Computer System Operation:
  - One or more CPUs, device controllers connected through common bus that provide access to shared memory
  - The CPU and the device controllers can execute in parallel, competing for memory cycles



Computer System Organization

- 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

Computer System Organization

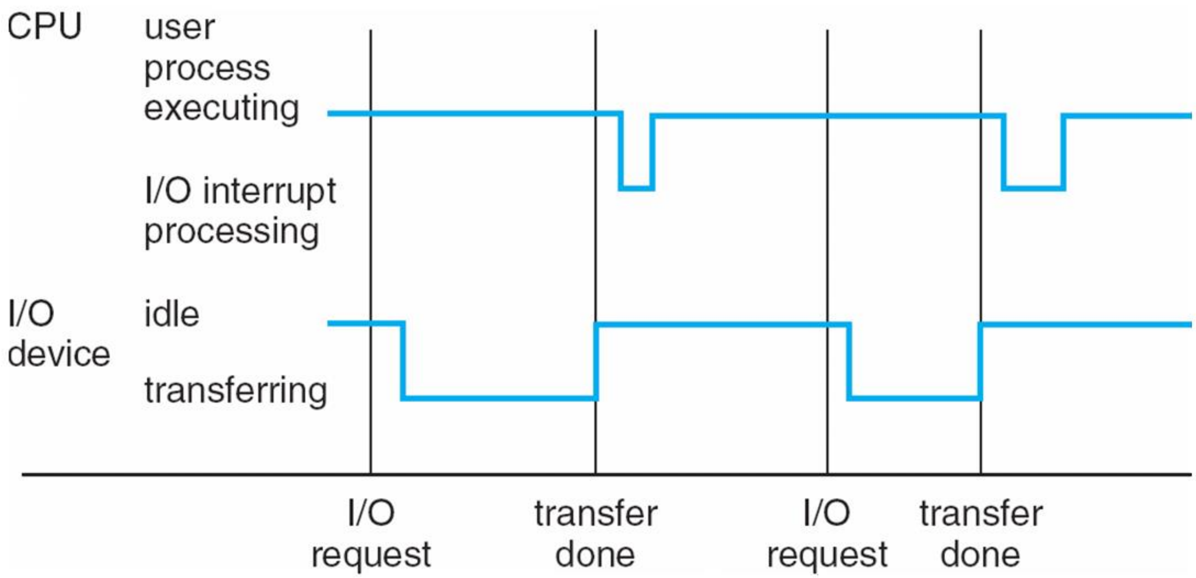
- Computer System Operation:
  - For a computer to start running, it needs an initial program to run
    - bootstrap program is loaded at power-up or reboot
      - Typically stored in **ROM** or **EEPROM**, generally known as **firmware**
      - Initializes all aspects of system
      - Loads operating system kernel and starts execution
      - The kernel then starts providing services to the system and its users.
      - Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become system processes, or system daemons that run the entire time the kernel is running.
      - The system is now running and waiting for an event to occur

Computer System Organization

- Common Functions of Interrupts:
  - The occurrence of an event is usually signaled by an interrupt from either the hardware or the software:
    - Hardware trigger an interrupt by sending a **signal** to the CPU
    - Software trigger an interrupt by executing a special operation called a **system call**
  - 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
  - A trap or exception is a software-generated interrupt caused either by an error or a user request
  - An operating system is interrupt driven

Computer System Organization

- Common Functions of Interrupts:
  - Interrupt timeline



Interrupt timeline for a single process doing output.

Computer System Organization

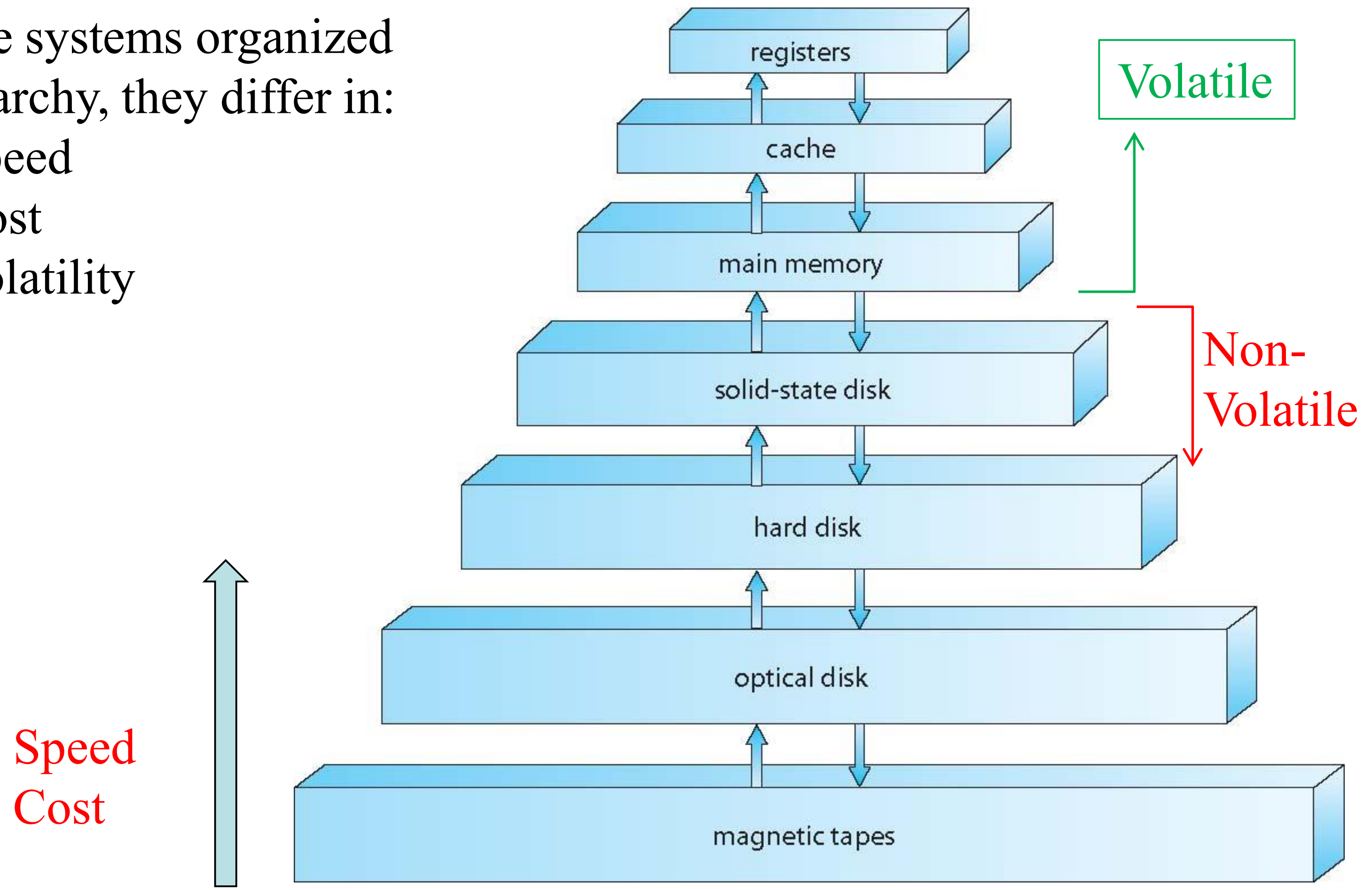
- Interrupt Handling:
  - The operating system preserves the state of the CPU by storing registers and the program counter
  - Determines which type of interrupt has occurred:
    - **polling**
    - **vectored** interrupt system
  - Separate segments of code determine what action should be taken for each type of interrupt

Computer System Organization

- Storage Structure:
  - Main memory: only large storage media that the CPU can access directly
    - **Random access**
    - Typically **volatile**
  - Secondary storage: extension of main memory that provides large **nonvolatile** storage capacity
  - Hard 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 hard disks, nonvolatile
    - Various technologies
    - Becoming more popular

Computer System Organization

- Storage Hierarchy:
  - Storage systems can be organized in a hierarchy according to speed and cost.
  - Storage systems organized in hierarchy, they differ in:
    - Speed
    - Cost
    - Volatility



Computer System Organization

- Caching:
  - **Caching**: copying information into faster storage system; main memory can be viewed as a cache for secondary storage
  - 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 Organization

- Storage Definitions and Notation Review:
  - The basic unit of computer storage is the bit.
  - A byte is 8 bits, and on most computers it is the smallest convenient chunk of storage.
  - A word, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.
  - Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
    - kilobyte, or KB, is 1,024 bytes
    - megabyte, or MB, is 1,0242 bytes
    - gigabyte, or GB, is 1,0243 bytes
    - terabyte, or TB, is 1,0244 bytes
    - petabyte, or PB, is 1,0245 bytes

Computer System Organization

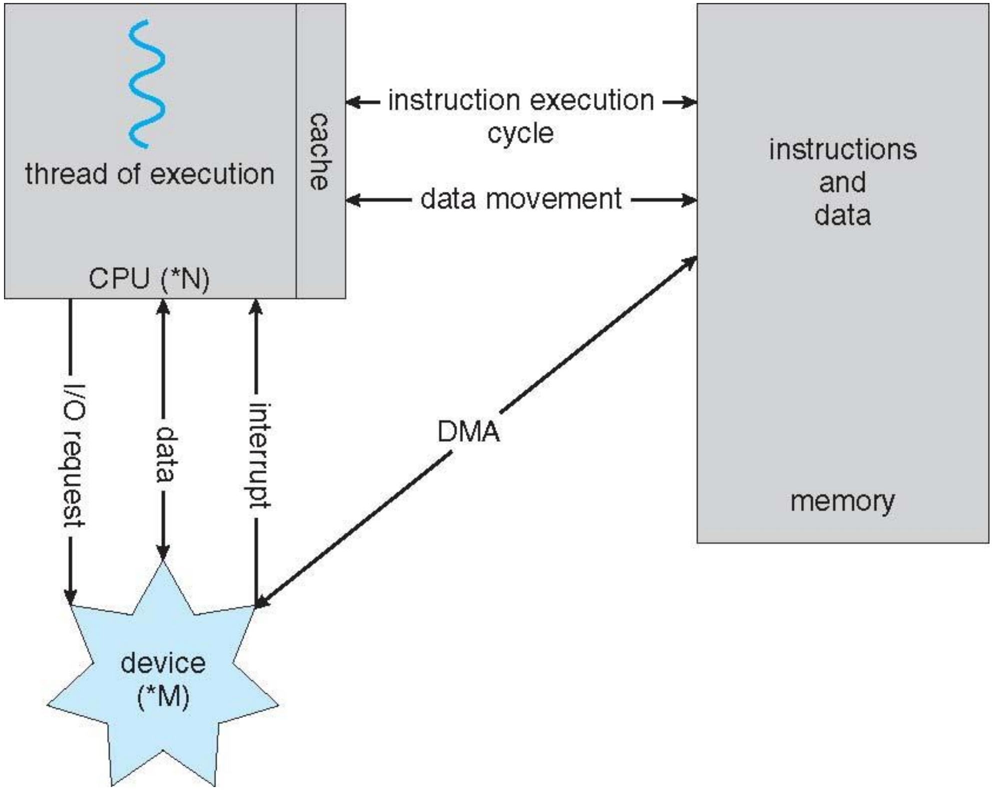
- I/O Structure:
  - A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
  - Each device controller is in charge of a specific type of device.
  - Operating systems have a **device driver** for each device controller to manage I/O

Computer System Organization

- I/O Structure:
  - Interrupt-driven I/O is good for moving small amounts of data
    - The device driver loads the appropriate registers within the device controller.
    - The device controller determines what action to take (such as “read a character from the keyboard”).
    - The controller transfers the data from the device to its local buffer.
    - Then the device controller informs the device driver via an interrupt that it has finished its operation.
    - The device driver then returns control to the operating system
  - This can produce high overhead when used for bulk data movement
    - To solve this problem, direct memory access (DMA) is used

Computer System Organization

- Direct Memory Access Structure:
  - 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

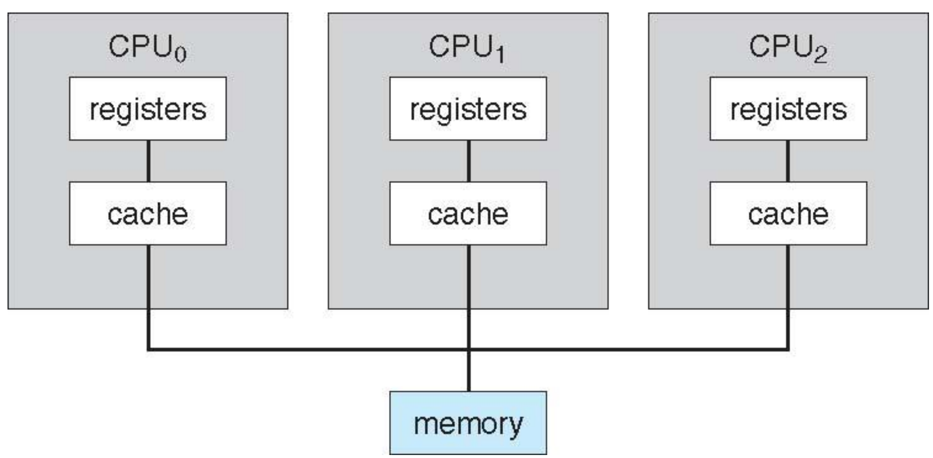


Computer-System Architecture

- Single-Processor Systems:
  - Most systems use a single general-purpose processor
    - Most systems have special-purpose processors as well
      - For keyboard, disks, …
- Multiprocessors systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - They have two or more processors in close communication, sharing the computer resources
  - Advantages include:
    1. **Increased throughput:** more work done in less time
    2. **Economy of scale:** cost less than equivalent multiple single-processor systems
    3. **Increased reliability:** graceful degradation or fault tolerance

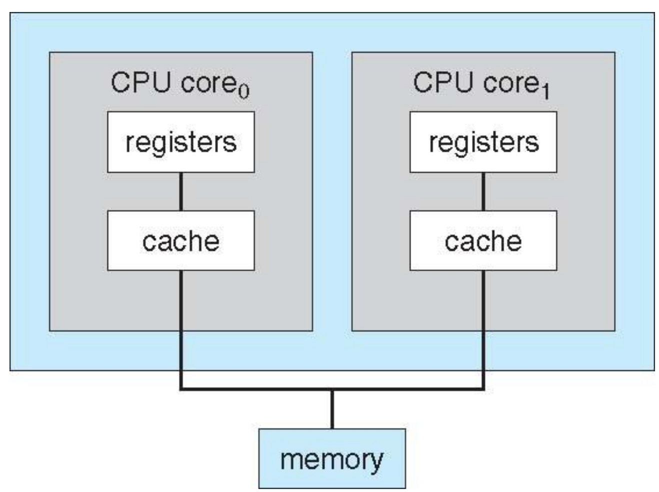
Computer-System Architecture

- The multiple-processor systems in use today are of two types:
  - Asymmetric Multiprocessing: each processor is assigned a specie task.
    - This scheme defines a **boss–worker** relationship.
      - The boss processor schedules and allocates work to the worker processors.
  - Symmetric Multiprocessing (SMP):
    - It is the most common
    - All processors are peers: each processor performs all tasks
    - Symmetric Multiprocessing Architecture:



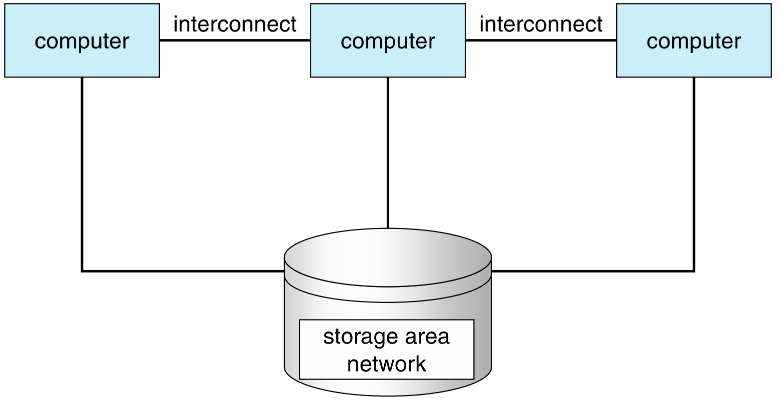
Computer-System Architecture

- Multi-Core CPUs:
  - They are more efficient than multiple chips with single cores
    - Because on-chip communication is faster than between-chip communication.
  - Uses significantly less power than multiple single-core chips
  - These multicore CPUs appear to the operating system as N standard processors.
  - dual-core design with two cores on the same chip.



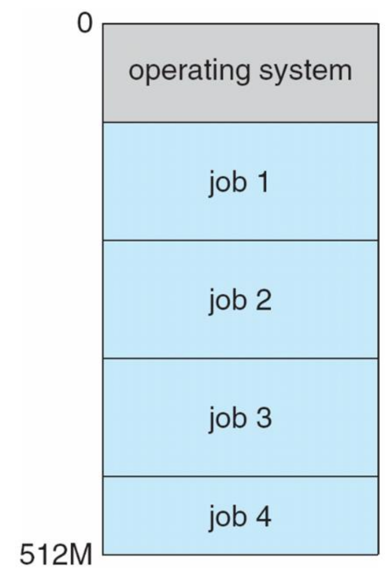
Computer-System Architecture

- Clustered Systems:
  - They are 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**
    - Some have **distributed lock manager** (**DLM**) to avoid conflicting operations



Operating System Structure

- Important Aspects of Operating Systems:
  - **Multiprogramming** (**Batch system**) is 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



Operating System Structure

- Important Aspects of Operating Systems:
  - **Timesharing** (**multitasking**) is logical extension to multiprogramming 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 ⇨ **process**
    - 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 to run
    - **Virtual memory** allows execution of processes that are larger than actual physical memory

Operating-System Operations

- Modern operating systems are interrupt driven:
  - If there are no processes, no I/O devices, and no users to whom to respond, an operating system will sit quietly.

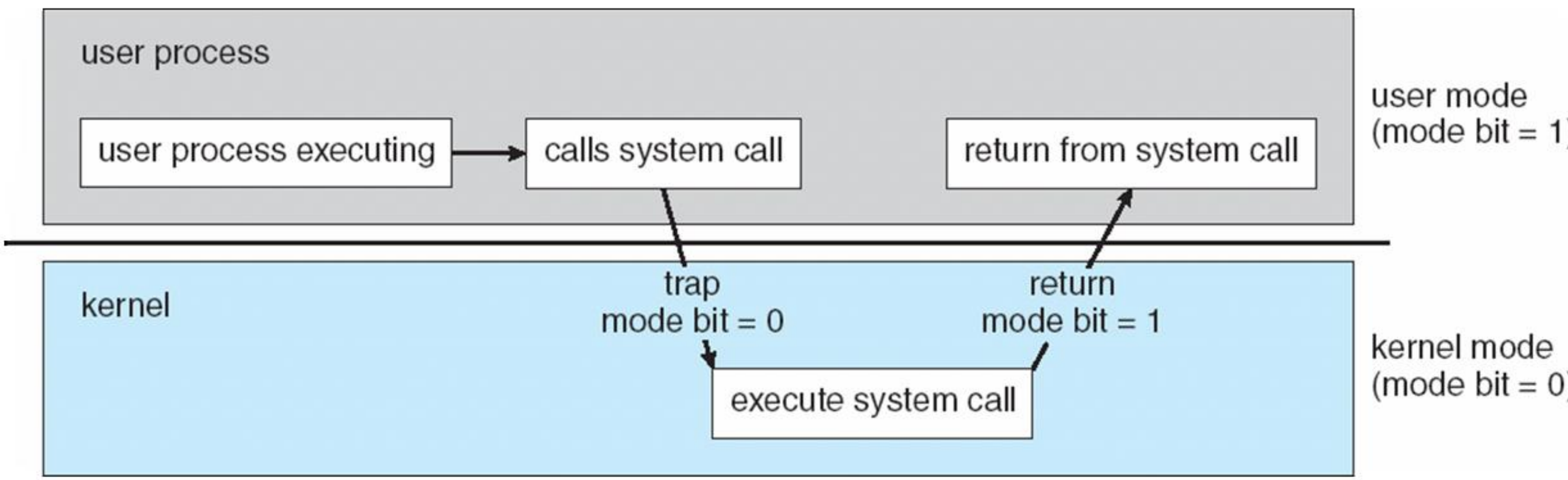
  - **Interrupt driven** by hardware or software
    - Hardware interrupt by one of the devices
    - Software interrupt (called **exception** or **trap):**
      - Software error (e.g., division by zero)
      - Request for operating system service
      - Other process problems include infinite loop, processes modifying each other or even the operating system itself

Operating-System Operations

- Dual-Mode and Multi-Mode Operation:
  - To ensure the proper execution of the OS, we must be able to distinguish between the execution of operating-system code and user-defined code.
  - Therefore, computer systems provide <u>hardware bit</u> to differentiate among various modes of execution.
  - **Dual-mode** operation allows OS to protect itself and other system components
    - **User mode** and **kernel mode** (called supervisor, system, or privileged mode)
    - **Mode bit** provided by hardware (kernel (0), user (1))
      - Provides ability to distinguish when system is running user code or kernel code
      - Some instructions designated as **privileged**, only executable in kernel mode (such as: switch mode, I/O control, timer management, and interrupt management instructions)

Operating-System Operations

- Dual-Mode and Multi-Mode Operation:
  - Transition from user to kernel mode:
    - System call changes mode to kernel, return from call resets it to user



Operating-System Operations

- Dual-Mode and Multi-Mode Operation:
  - The concept of modes can be extended beyond two modes
    - Increasingly CPUs support multi-mode operations
    - CPU uses more than one bit to set and test the mode.
    - Example:
      - CPUs that support virtualization frequently, have a separate mode to indicate when the **virtual machine manager** (**VMM**) and the virtualization management software is in control of the system.
      - In this mode, the VMM has more privileges than user processes but fewer than the kernel

Operating-System Operations

- Transition from User to Kernel Mode:
  - To ensure that the OS maintains control over the CPU, we use timers.
  - Timer is used to prevent a user program to get stuck in infinite loop / process hogging resources
    - Timer is set to interrupt the computer after some time period
    - Keep a counter that is decremented by the physical clock.
    - Operating system set the counter (it is a privileged instruction)
    - When counter reaches zero, it generates an interrupt
    - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Process Management

- 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, a 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

- Process Management Activities:
  - The operating system is responsible for the following activities in connection with process management:
    - Scheduling processes and threads on the CPUs
    - 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 the instructions must be in memory
  - All (or part) of the data that is needed by the program must be in memory.
  - Memory management determines what is in memory and 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

- Storage Management:
  - OS provides uniform, logical view of information storage
    - It 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
    - It is one of the most visible components of an operating system.
    - 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
      - Support primitives to manipulate files and directories
      - Mapping files onto secondary storage
      - Backup files onto stable (non-volatile) storage media

Storage Management

- Mass-Storage Management:
  - Usually disks used to store data that does not fit in main memory or 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
  - OS is responsible for the following activities:
    - Free-space management
    - Storage allocation
    - Disk scheduling
  - Some storage need not be fast
    - Tertiary storage includes optical storage (CD, DVD), magnetic tape
      - Still must be managed – by OS or applications
      - Varies between WORM (write-once, read-many-times) and RW (read-write) formats

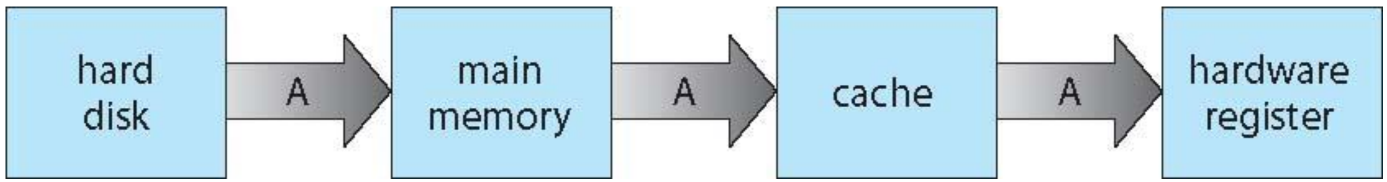
Storage Management

- Caching:
  - Information is normally kept in main memory, when it is used, it is copied into a faster storage system (the cache) on a temporary basis.
  - Cache management is an important design problem.
    - Cache size and the replacement policy can result in greatly increased performance
  - Main memory can be viewed as a fast cache for secondary storage
  - 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

Storage Management

- Cache Coherency:
  - In a hierarchical storage structure, the same data may appear in different levels of the storage system.
  - For example
    - Migration of integer “A” from Disk to Register



    - 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 the most recent value in their cache
    - Distributed environment situation even more complex
      - Several copies of a datum can be kept in different computers
      - Distributed systems must ensure that, when a replica is updated in one place, all other replicas are brought up to date on time

Storage Management

- I/O Systems:
  - One purpose of OS is to hide peculiarities of hardware devices from the user
  - The I/O subsystem consists of several components:
    - 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
  - Only the device driver knows the peculiarities of the specific device to which it is assigned.

Protection and Security

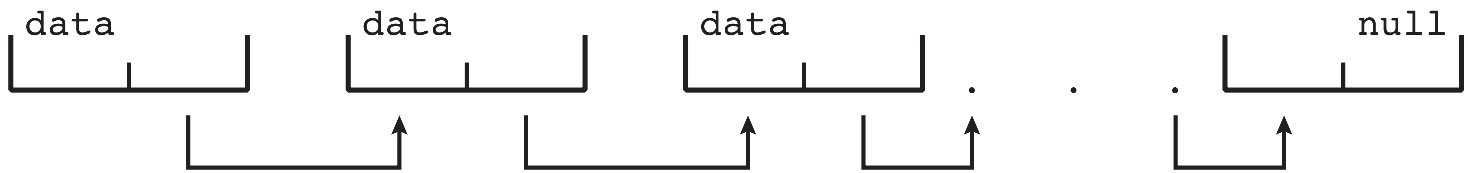
- Protection and Security:
  - **Protection:** any mechanism for controlling access of processes or users to resources defined by the OS
  - **Security:** defense of the system against internal and external attacks
    - Huge range, including denial-of-service, worms, viruses, identity theft, theft of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

Kernel Data Structures

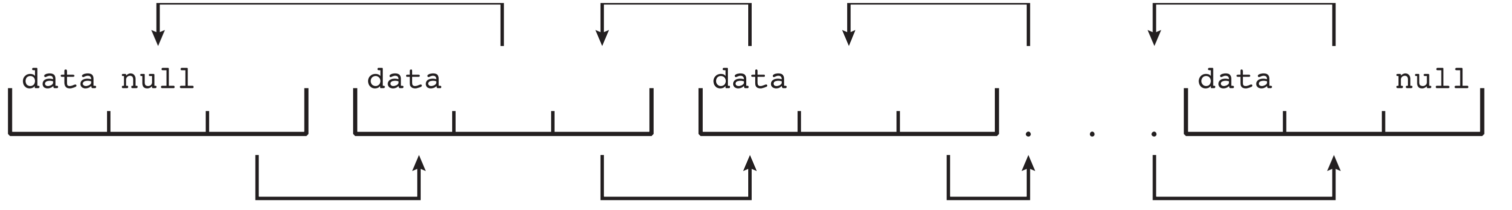
- Lists, Stacks, and Queues:
  - An array is a simple data structure in which each element can be accessed directly.
    - For example, main memory is constructed as an array.
  - A lists is the most fundamental data structures in computer science.
    - each item in an array can be accessed directly,
    - the items in a list must be accessed in a particular order.
    - linked list is the most common method for implementing the list,
    - Lists are used for constructing more powerful data structures, such as stacks and queues

Kernel Data Structures

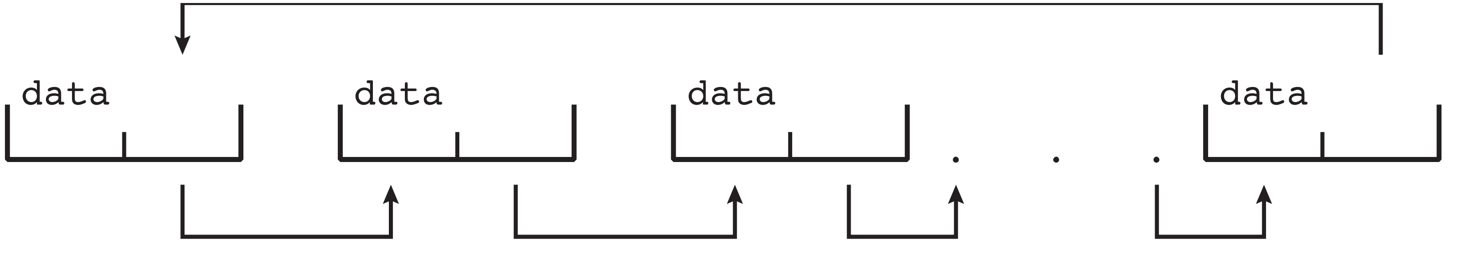
- Lists, Stacks, and Queues:
  - **Linked lists are of several types:**
    - Singly linked list, each item points to its successor,



    - Doubly linked list, a given item can refer either to its predecessor or to its successor

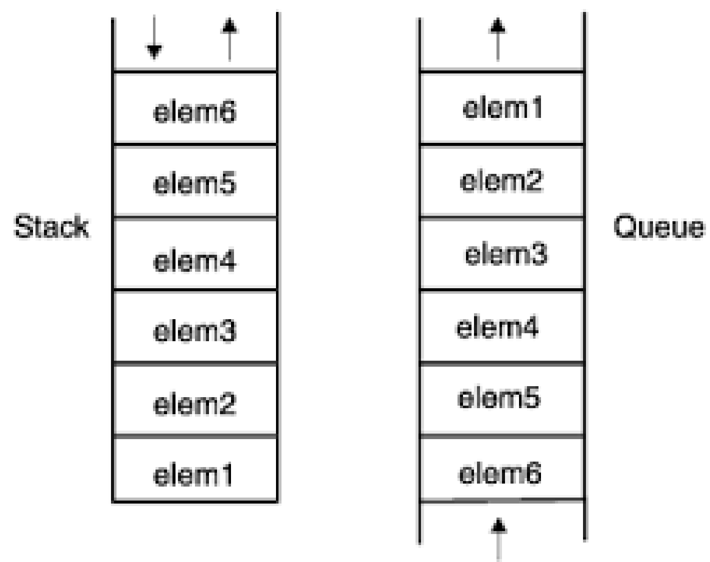


    - Circularly linked list, the last element in the list refers to the first element, rather than to null,



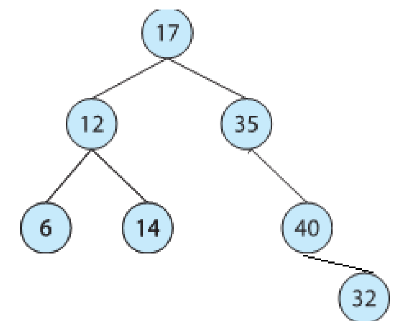
Kernel Data Structures

- Lists, Stacks, and Queues:
  - Stack:
    - It is a sequentially ordered data structure that uses the last in, first out (LIFO) principle for adding and removing items
  - Queue:
    - It is a sequentially ordered data structure that uses the first in, first out (FIFO) principle

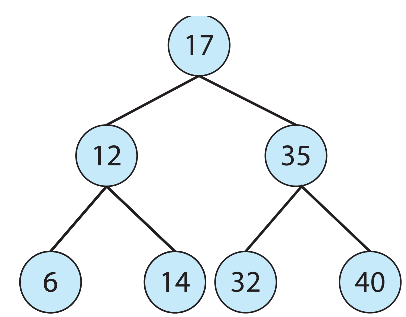


Kernel Data Structures

- Trees:
  - A tree is a data structure that can be used to represent data hierarchically.
  - Data values in a tree structure are linked through parent–child relationships
  - In a binary tree, a parent may have at most two children (left child and the right child).
  - A binary search tree additionally requires an ordering between the parent's two children in which *left_child <= right_child.*
    - Search performance is *O(n)*

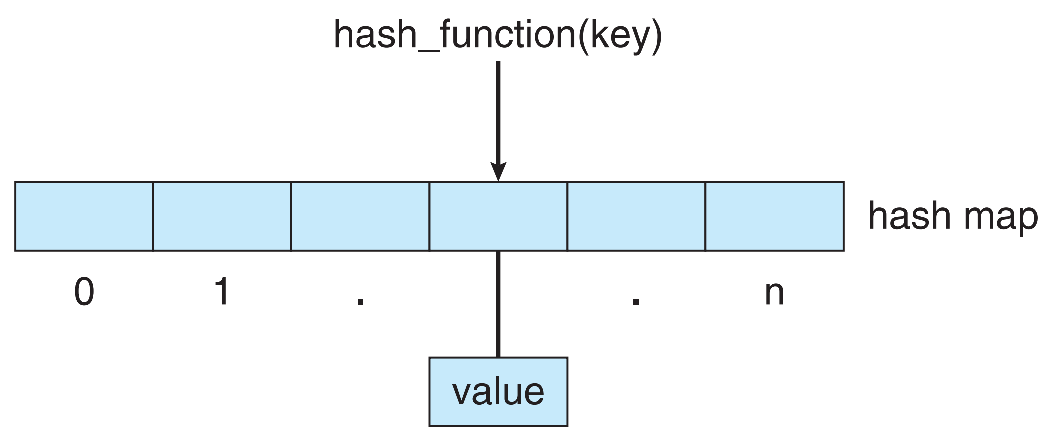


  - An algorithm can be used to create a balanced binary search tree.
    - A tree containing **n** items has at most ***log(n)*** levels,
    - Search performance is ***O(log(n))***



Kernel Data Structures

- Hash Functions and Maps:
  - A hash function takes data as its input, performs a numeric operation on this data, and returns a numeric value.
    - This numeric value can then be used as an index into a table (typically an array) to quickly retrieve the data.
    - Performance can be as good as $O(1)$
    - Hash functions are used extensively in operating systems.
    - Hash collision can be accommodated by having a linked list at that table location that contains all of the items with the same hash value
    - A hash map, which associates (or maps) [key:value] pairs using a hash function



Kernel Data Structures

- Bitmaps:
  - A bitmap is a string of n binary digits that can be used to represent the status of n items.
    - Example,
      - Consider the bitmap 001011101
        - Resources 2, 4, 5, 6, and 8 are unavailable; resources 0, 1, 3, and 7 are available.

Computing Environments

- Traditional Computing:
  - 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

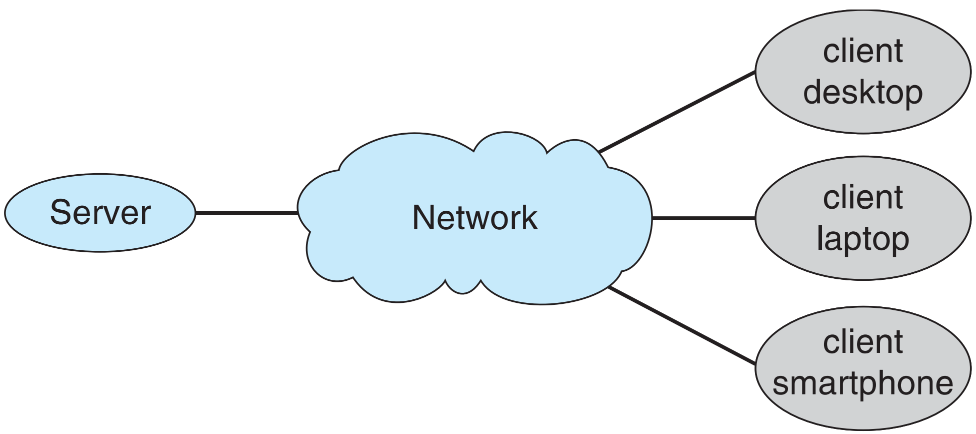
- Mobile Computing:
  - Handheld smartphones, tablets, etc
  - What is the functional difference between them and a “traditional” laptop?
  - Extra feature – more OS features (GPS, gyroscope)
  - Allows new types of apps like ***augmented reality***
  - Use IEEE 802.11 wireless, or cellular data networks for connectivity
  - Leaders are **Apple iOS** and **Google Android**

Computing Environments

- Distributed Computing:
  - 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

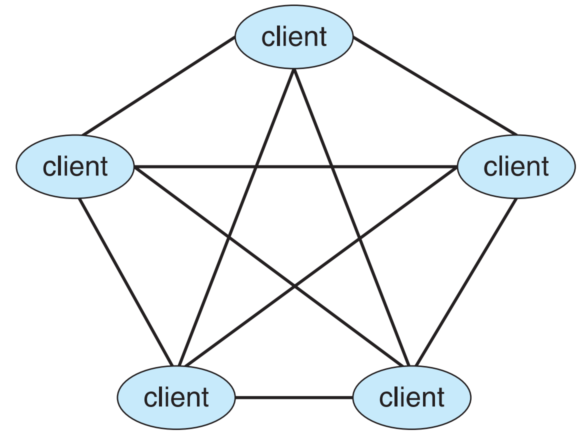
Computing Environments

- Client-Server:
  - 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)
      - Example: A server running a database that responds to client requests for data
    - **File-server system** provides interface for clients to store and retrieve files
      - Example: a web server that delivers files to clients running web browsers.



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

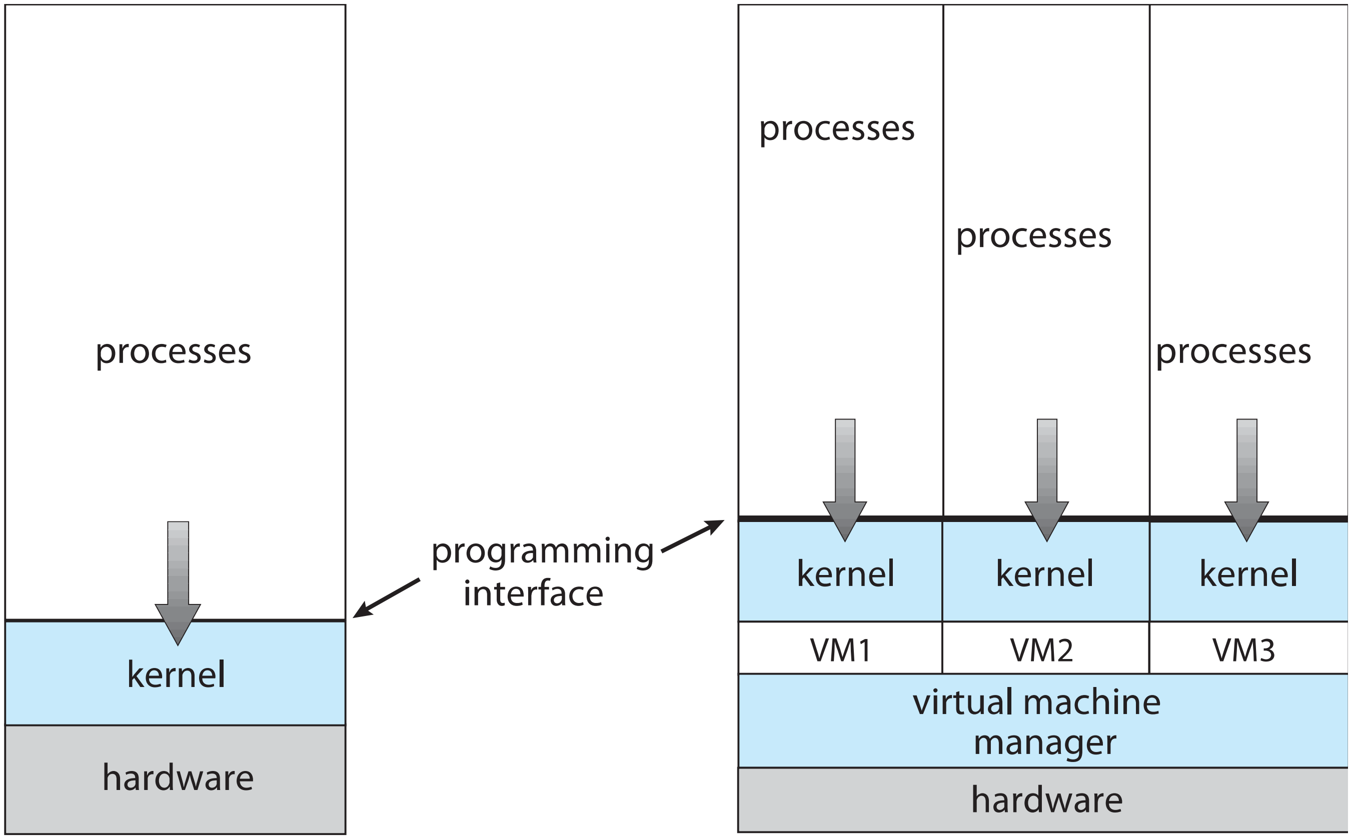
- Virtualization:
  - Allows operating systems to run as applications within other OSes
    - Vast and growing industry
  - Emulation used when source CPU type different from target type (i.e. PowerPC to Intel x86)
    - Generally slowest method
    - When computer language not compiled to native code – 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 (virtual machine Manager) 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:



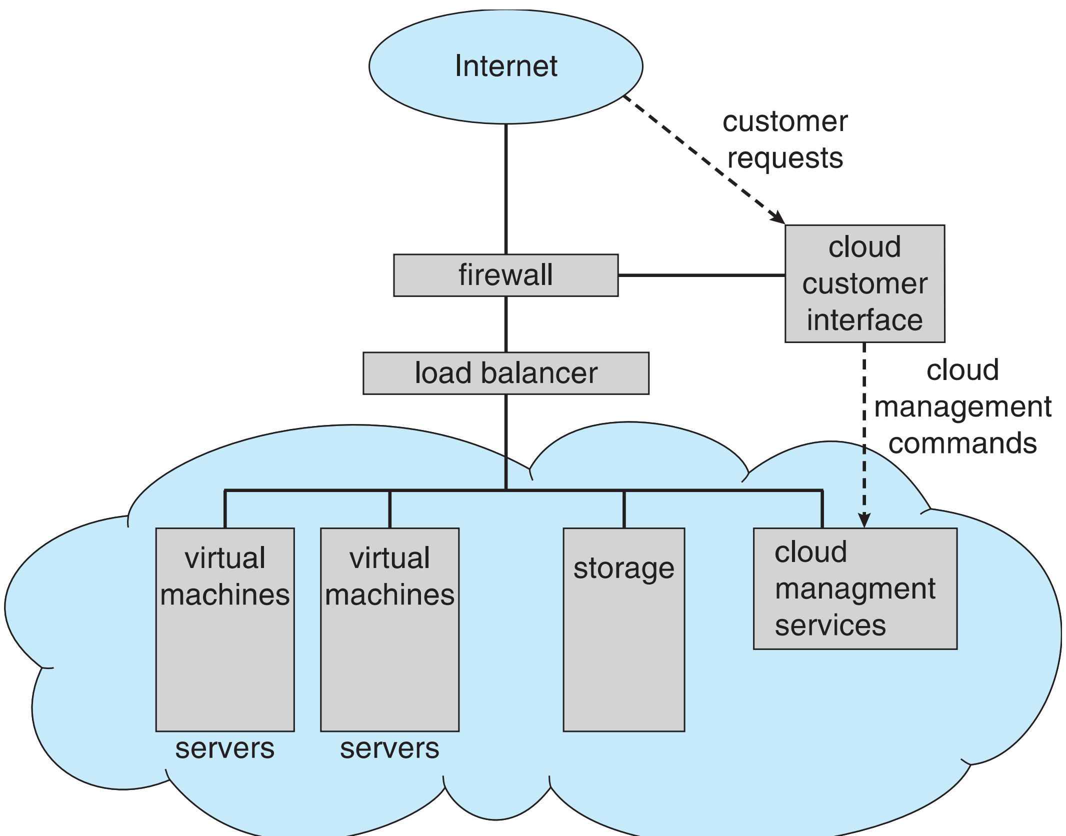
(a) (b)

Computing Environments

- Cloud Computing:
  - Delivers computing, storage, even apps as a service across a network
  - It is logical extension of virtualization because it uses virtualization as the base for its functionality.
    - Amazon Elastic Compute Cloud (EC2) has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
  - Many types of cloud computing:
    - 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 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



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