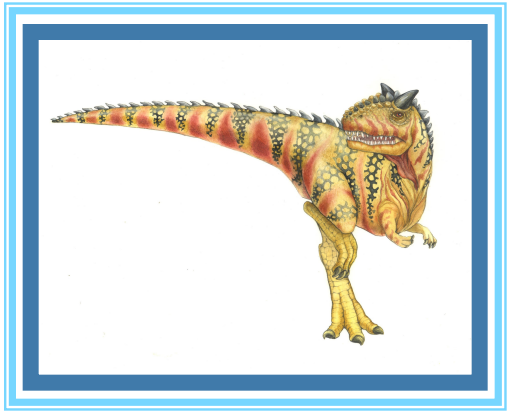
# COMP 3511 Operating System

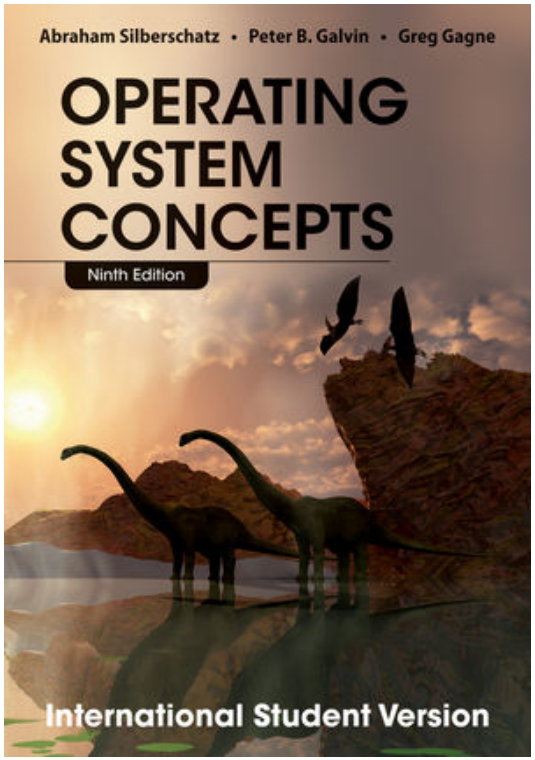


# Lectures and Tutorials

- L-1
  - Instructor: Bo Li
  - Lectures: Tuesday and Thursday 10:30 am – 11:50 am
  - Venue: **Room 2502**
- Lab sessions:
  - Lab 1A Monday 3:00 pm - 4:50 pm Room 4214
  - Lab 1B Wednesday 10:30 am - 12:20 pm Room 4214
- Web site: http://course.cs.ust.hk/comp3511/

# Textbook

- Operating System Concepts, A. Silberschatz, P. B. Galvin and G. Gagne, 9th Edition




# Lecture, Tutorials and Grading Scheme

- Lecture notes and tutorial materials
  - Download course materials before class
  - Homework and programming assignments will be put online
- Grading is based on
  - Four Written Assignments – 20% (5% each)
  - Two Programming Assignments – 25% (10% and 15%)
  - Midterm Exam - 20%
  - Final Exam - 35%

# Plagiarism Policy

- There are differences between collaborations, discussions and copy!

- 1st time: all involved get ZERO marks, and will be reported to ARR
- 2nd time: need to terminate (Fail grade)
- Cheating in Midterm or Final exam results in automatic Fail grade

# Course Prerequisite

- COMP 2611 or equivalent
  - Basic computer organization knowledge, computer system, CPU, memory hierarchy, interrupt, DMA, storage, I/O devices
- Programming
  - UNIX environment
  - C/C++ programming

# Lecture Format

- Lectures:
  - Slides are available before class
  - It is important to attend the lectures (not all materials and concepts can be covered in slides)
  - If you miss any lectures, learn from the lecture notes and textbook
- Tutorials
  - Supplement the lectures with more examples
  - Some exercises
  - Unix environment, editor, how to compile and run programs, Makefile
  - **Nachos**
- Reading the corresponding materials in the textbook
  - Slides do not cover everything
- Chapter summary
  - Comprehensive summary at the end of each chapter

# Assignments

- Written assignments
  - Due by time specified
  - Contact TAs directly for any disputes on the grading
  - Re-grading requests will only be granted within **one week** after the homework grades are released
  - Late policy: 15% reduction, only one day delay is allowed.
- Programming assignments
  - Individual projects
  - Due by time specified
  - Run on Unix
  - Submit it using CASS – You need to register for an account
  - Re-grade policy will be announced
  - Late policy : 15% reduction, only one day delay is allowed.

# Midterm and Final Examinations

- Midterm Exam
  - March 31, 2015 (Week #9, Tuesday) 7:00 pm - 8:30 pm
  - Venues: 3006 and 3007
- Final Exam
  - TBD
- All exams are closed-book, closed-notes
- No make-up exams will be given unless
  - under very unusual circumstances, e.g., sickness, with letters of proof
  - Instructor must informed beforehand

# Tips for Learning

- Attend lectures
  - Download lecture notes prior to lectures
  - Important concepts are explained
- Complete homework alone
  - This is an exercise to test your knowledge
- Spend 30 minutes each week to review the content
  - Weekly summary can help
  - This will save you lots of time later when you prepare for exams
  - You can not expect to learn everything two days before exams no matter how smart you are
  - Knowledge is accumulated incrementally
- Start your project earlier
  - Have a plan for the project
- Raise questions!
  - Do not delay your questions until exams

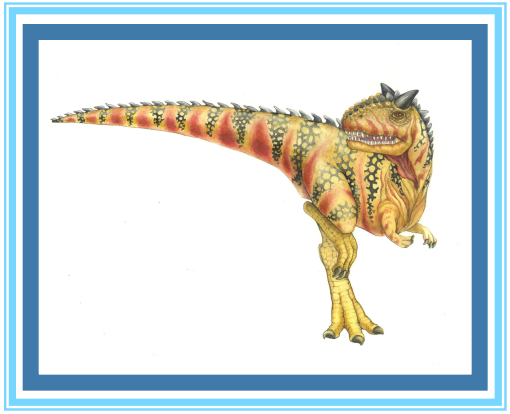
# What you are suppose to learn

- Define the fundamental principles, strategies and algorithms used in the design and implementation of operating systems
- Analyze and evaluate operating system functions
- Analyze the structure of an operating system kernel, and identify the relationship between the various subsystems
- Identify the typical events, alerts, and symptoms indicating potential operating system problems
- Recognize and evaluate the source code of the NACHOS operating system
- Design and implement programs for basic operating system functions and algorithms

# Course Outline

- Overview (1.5 week)
  - Basic OS concept
  - System architecture
- Process and Thread (6 weeks)
  - Process and thread (2 weeks)
  - CPU scheduling (1 week)
  - Synchronization (2 weeks)
  - Deadlock (1 week)
- Memory and storage (5 weeks)
  - Memory management (1.5 weeks)
  - Virtual memory (1.5 weeks)
  - File system (1 week)
  - Secondary storage and I/O (1 weeks)

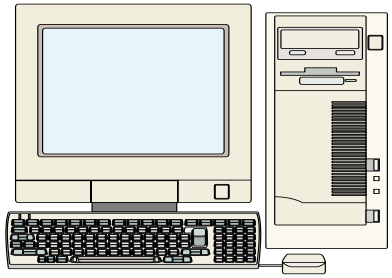
# Chapter 1: Introduction

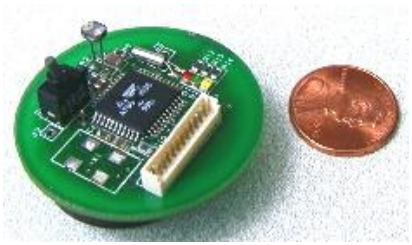


# Chapter 1: Introduction

- Computer-System Architecture
- What Operating Systems Do
- Operating-System Structure
- Operating-System Operations
- Computing Environment

# Computing Devices Everywhere

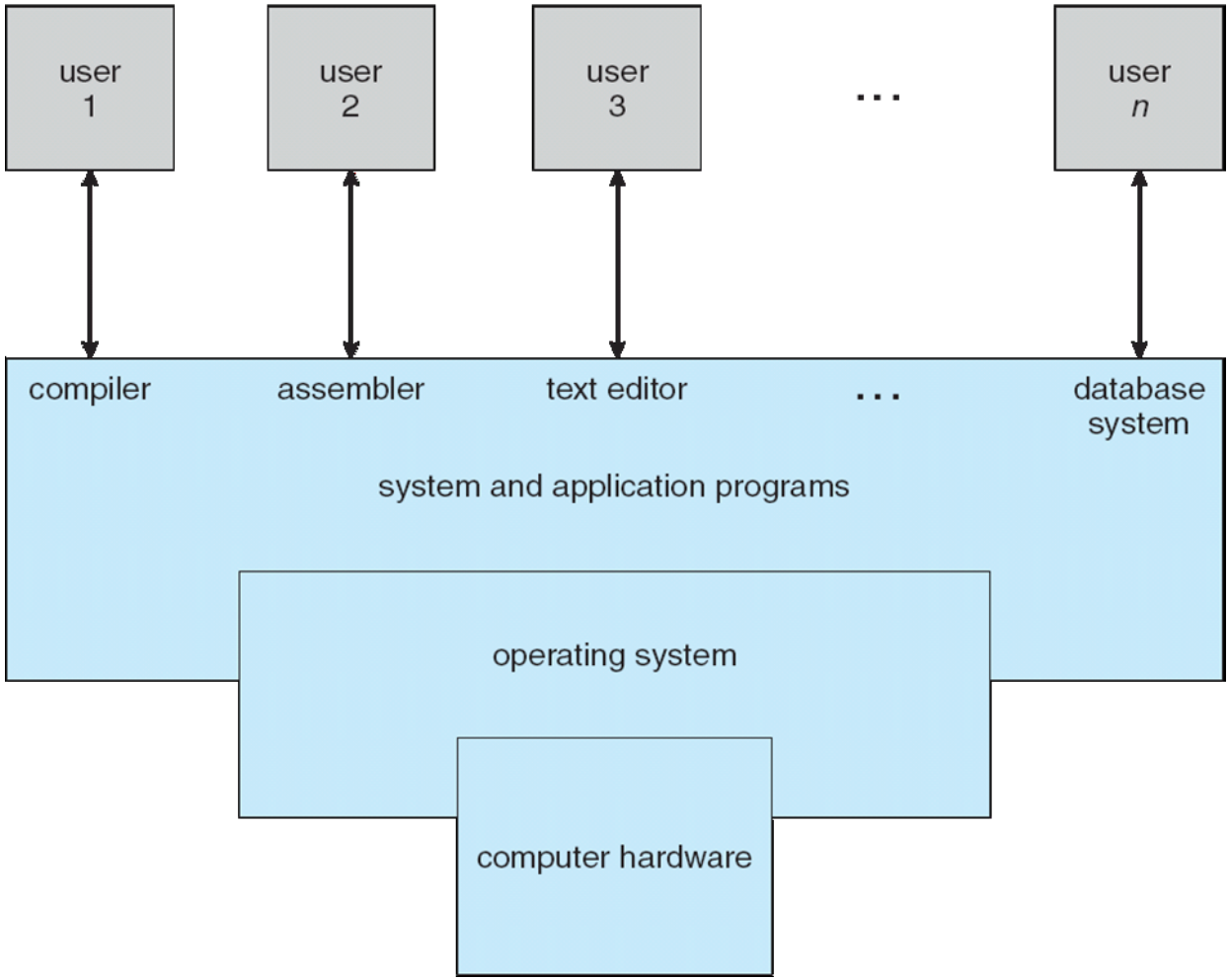





# Computer System Structure

- 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
  - **System and 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

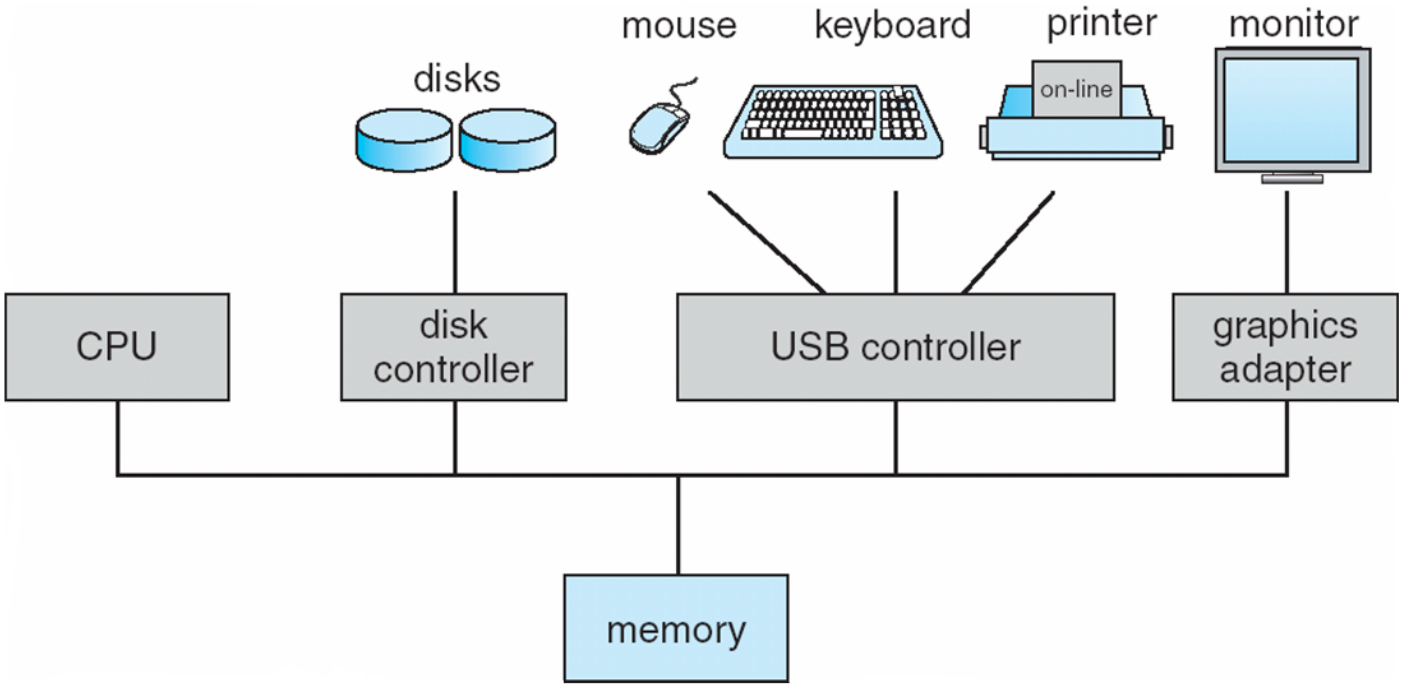



# Computer Startup

- **Bootstrap program** is loaded at power-up or reboot time
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of the 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

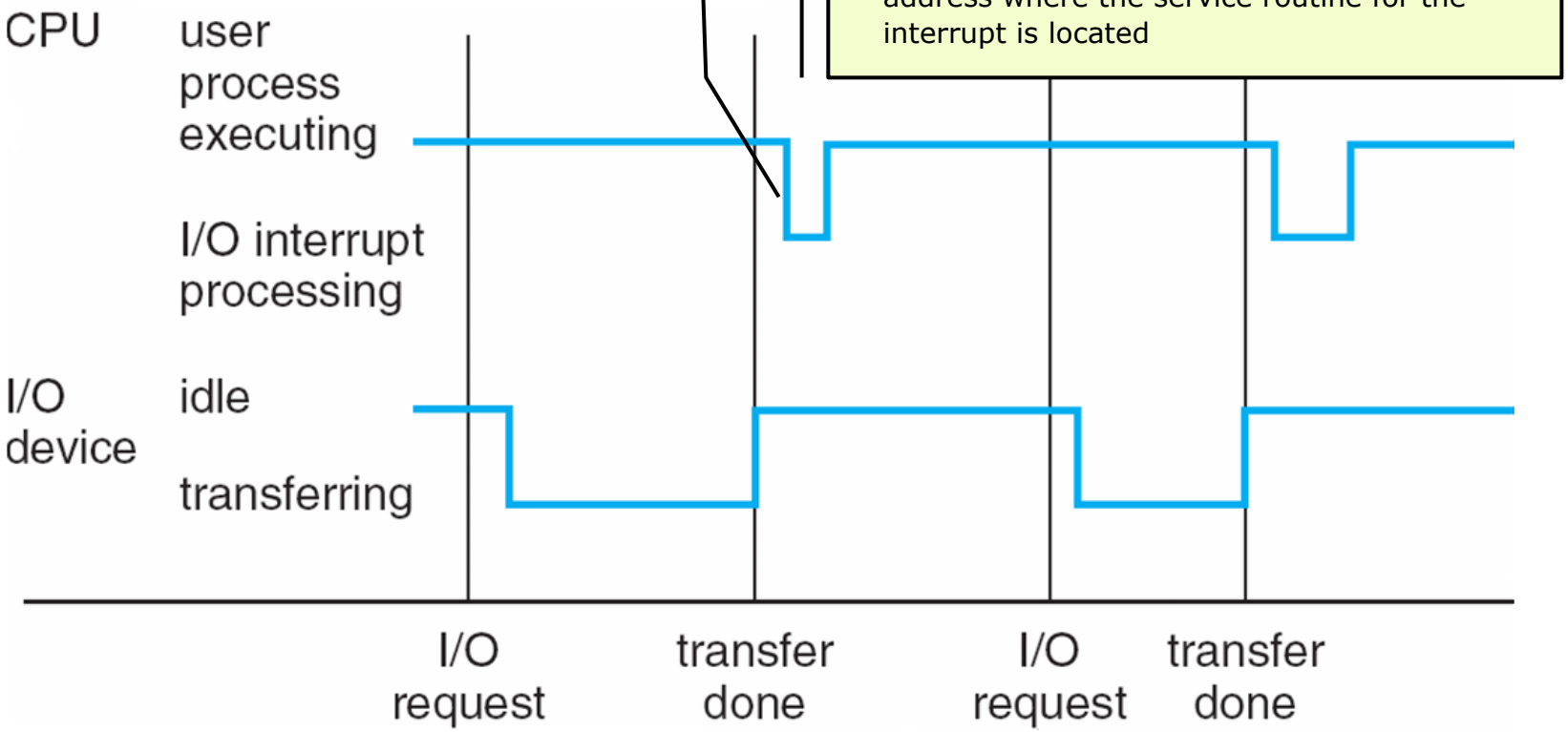
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU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

# Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines or called interrupt handlers
- Interrupt architecture must save the address of the interrupted instruction
- Determines which type of interrupt has occurred:
  - *polling*
  - *vectored* interrupt system
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- **A trap or an exception** is a software-generated interrupt caused either by an error (e.g., division by zero) or a user request for operation system service
- Operating systems are **interrupt driven**

# Interrupt Timeline

- When the CPU is interrupted, it stops what it is doing and immediately transfer execution to a fixed location
- The fixed location usually contains the starting address where the service routine for the interrupt is located




# 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 KB; a **gigabyte**, or **GB**, is 1,024 MB; a **terabyte**, or **TB**, is 1,024GB; and a **petabyte**, or **PB**, is 1,024 TB. 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).

# Direct Memory Access

- The problem with interrupt-based I/O operations
  - The speed mismatch between CPU, memory and I/O device
- 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, so CPU can be released to execute instructions for other process or program

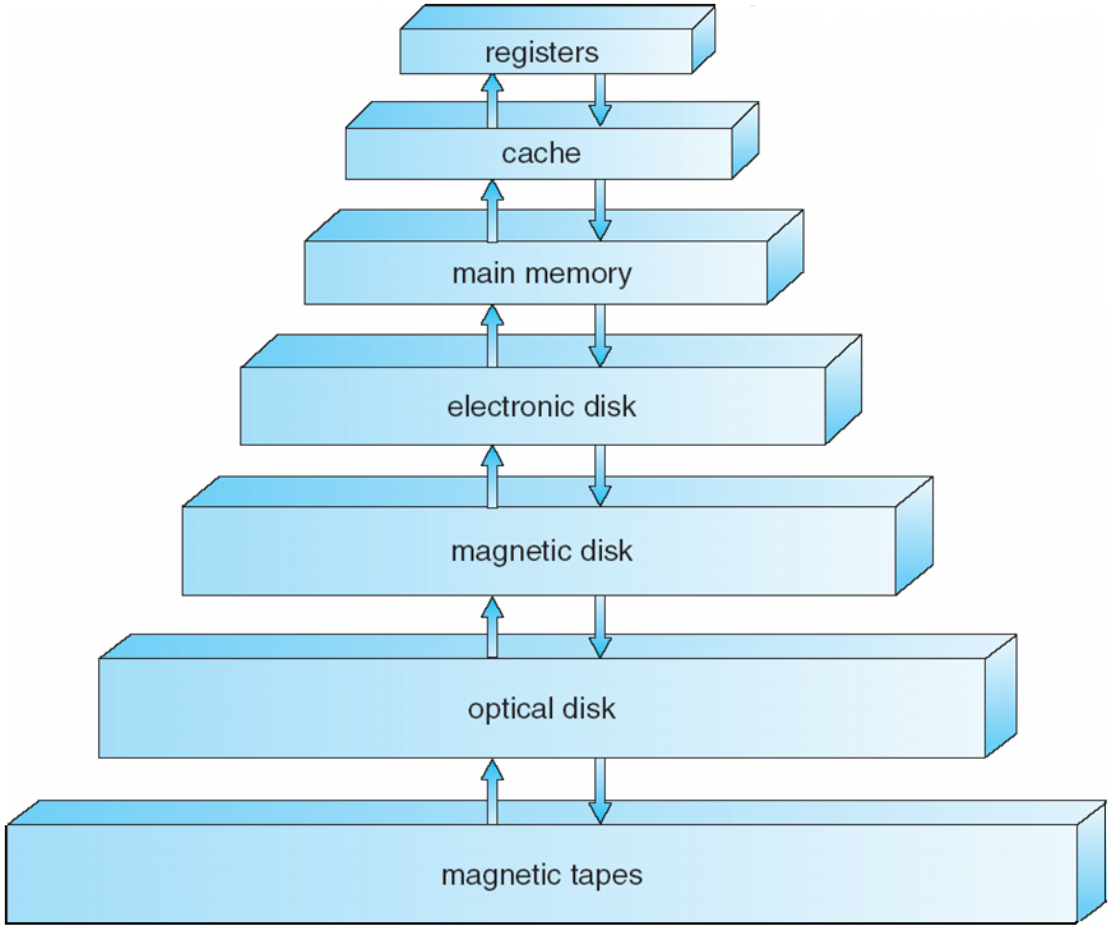
# Storage Hierarchy Structure

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

Main memory is a volatile storage device that loses its contents when power is turned off

- Cache – fast memory close to CPU
- Main memory – only large storage media that the CPU can access directly
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material

# Storage-Device Hierarchy




# Caching

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

# Performance of Various Levels of Storage

- The performance data is updated constantly

| Level | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Name | registers | cache | main memory | disk storage |
| Typical size | < 1 KB | > 16 MB | > 16 GB | > 100 GB |
| Implementation technology | custom memory with multiple ports, CMOS | on-chip or off-chip CMOS SRAM | CMOS DRAM | magnetic disk |
| Access time (ns) | 0.25 – 0.5 | 0.5 – 25 | 80 – 250 | 5,000.000 |
| Bandwidth (MB/sec) | 20,000 – 100,000 | 5000 – 10,000 | 1000 – 5000 | 20 – 150 |
| Managed by | compiler | hardware | operating system | operating system |
| Backed by | cache | main memory | disk | CD or tape |

# How do We Tame Complexity?

- Every piece of computer hardware is different
  - Different CPU
    - Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    - Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - Cable, DSL, Wireless, Firewalls,…
- Questions:
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?
  - ……

# What is an Operating System?

- A program that acts as an intermediary between a user of a computer and computer hardware
- Operating system goals:
  - Control and coordinate the use of system resources (hardware and software)
  - Make the computer system convenient to use for users
  - Use the computer hardware in an efficient and protected manner

# What does Operating System Do?

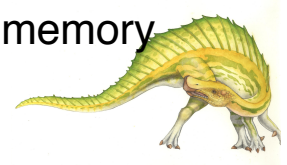
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
  - Prevent errors and improper use of the computer
- OS is a **facilitator**
  - Provides facilities that everyone needs
  - Standard Libraries, Windowing systems
  - Make application programming easier, faster, less error-prone

# Operating System Definition

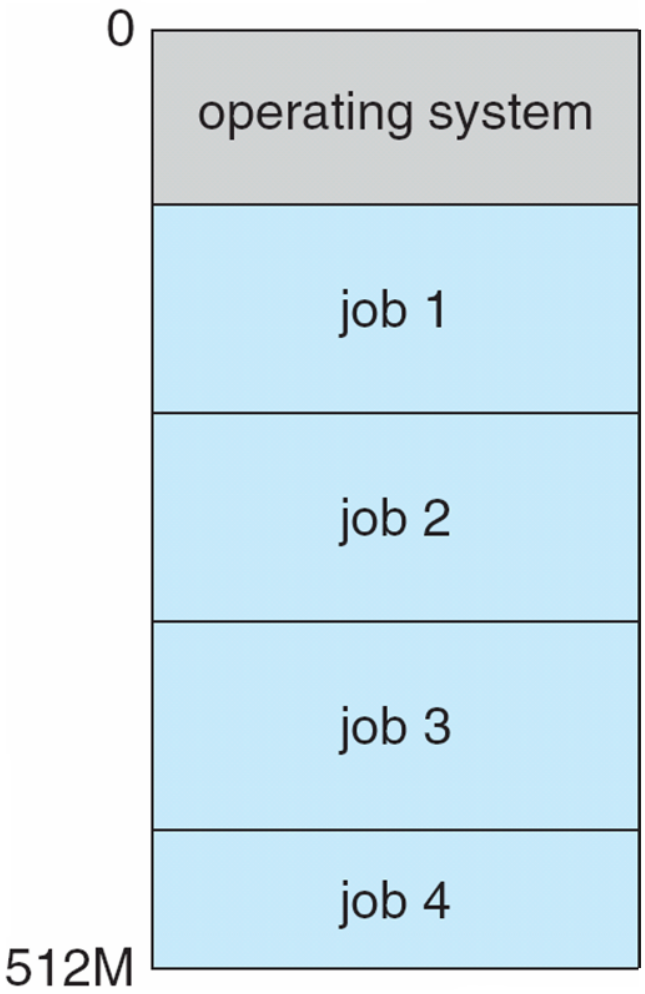
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the kernel
  - Everything else is either a system program (ships with the operating system) or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

# 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 ⇨ **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 not completely in memory



# Memory Layout for Multiprogrammed System

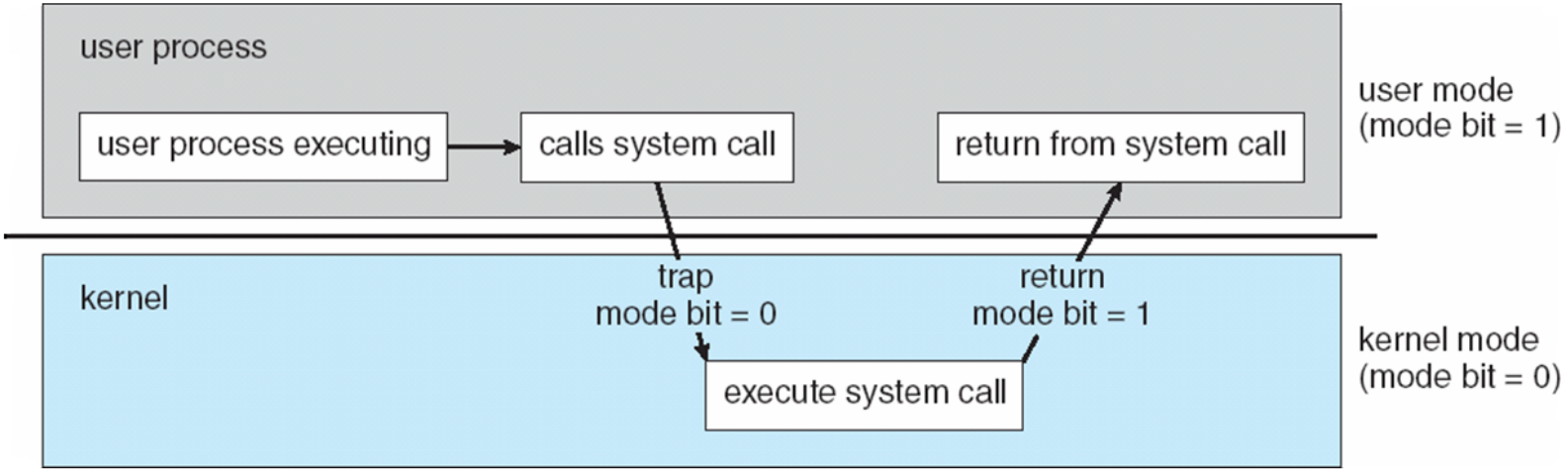



# Operating-System Operations

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode, or supervisor mode, system model**
  - **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

# Dual-mode Operation

- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions




- The dual mode of operation provides us with a rudimentary means for protecting the operating system from errant users and errant users from one another

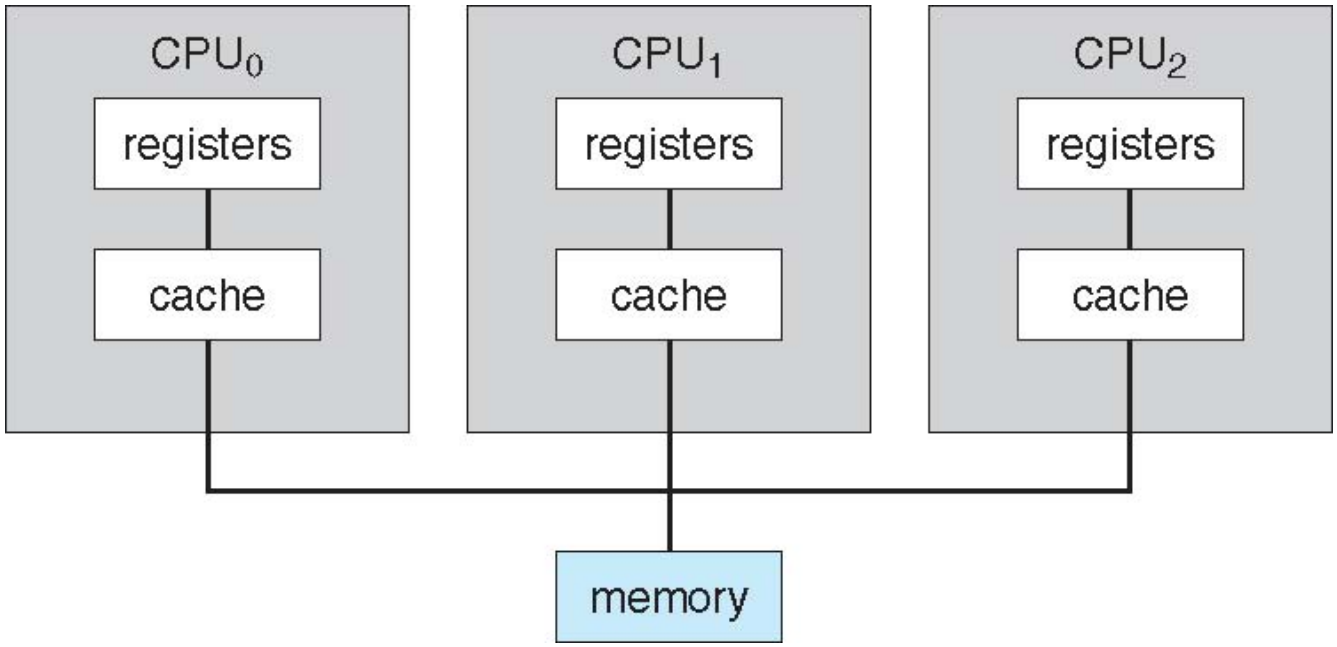
# Timer

- Must ensure the OS maintains control over the CPU
- Must prevent a user program from getting stuck in an infinite loop or never returning control to the OS
- **Timer** is used 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

# Computer-System Architecture

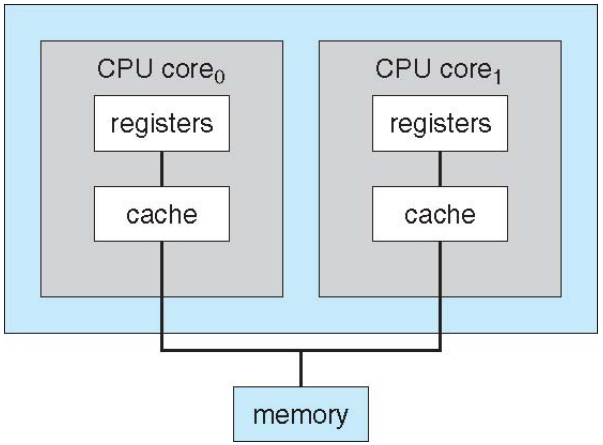
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – one master CPU distributes tasks among multiple slave CPUs
    2. **Symmetric Multiprocessing**

# Symmetric Multiprocessing Architecture




# A Dual-Core Design

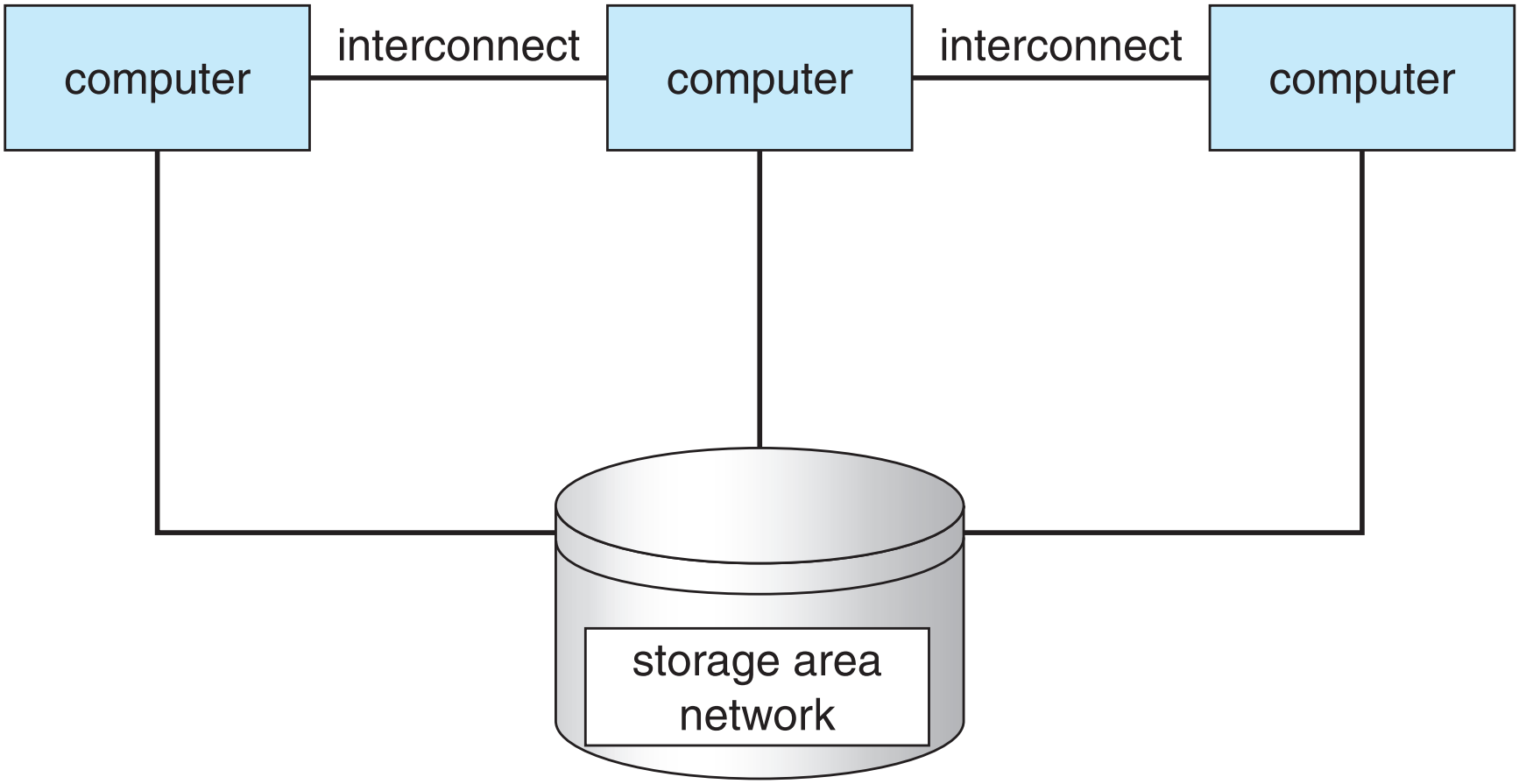
- **UMA** and **NUMA** architecture variations
  - Uniform Memory Access and Non Uniform Memory Access Architectures
- Multi-chip and **multicore**




# Clustered Systems

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

# Clustered Systems




# 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

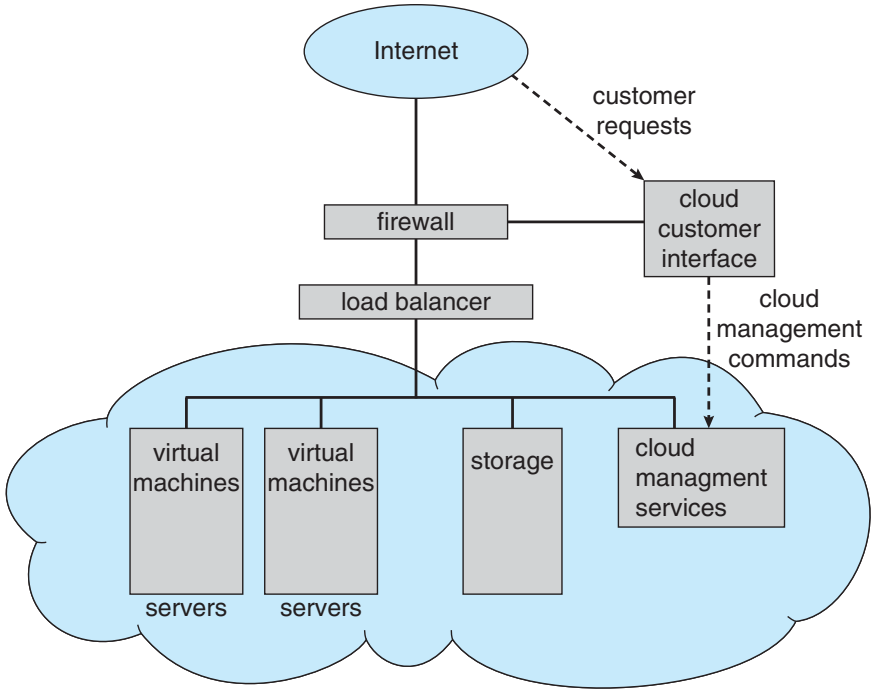
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders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

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




# 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)

# Open-Source Operating Systems

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

# End of Chapter 1