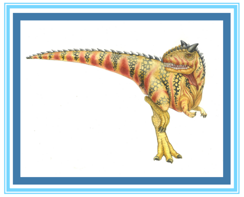
COMP 3511 Operating System



Lectures

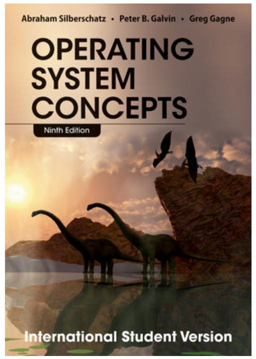
- **L-1**
  - Lectures: Monday and Wednesday 9:00 am – 10:20 am
  - Venue: **Room 2465** (Lift 25/26)
- **L-2**
  - Lectures: Tuesday and Thursday 4:30 pm – 5:50 pm
  - Venue: **Room 2465** (Lift 25/26)
- Web site: http://course.cse.ust.hk/comp3511/
- Instructor: Bo Li

Labs and Tutorials

- Lab sessions:
  - Lab 1 Thursday 6:00 pm - 7:50 pm Room 4214
  - Lab 2 Tuesday 1:00 pm - 2:50 pm Room 4214
  - Lab 3 Thursday 1:30 pm - 3:20 pm Room 4214
  - Lab 4 Tuesday 6:00 pm - 7:50 pm Room 4214
  - Lab 5 Monday 5:30 pm - 7:20 pm Room 4214
  - Lab 6 Monday 1:30 pm - 3:20 pm Room 4214
- Web site: http://course.cse.ust.hk/comp3511/

Textbook

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



Grading Scheme

- Grading is based on
  - 4 homework (written assignments) – 20% (5% each)
    - HW #1 (week 2-4), HW #2 (week 5-7), HW #3 (week 8-10), HW #4 (week11-13)
  - 2 projects (programming assignments) – 25% (10% and 15%)
    - Project #1 (week 4-7), Project #2 (week 9-13)
  - Midterm Exam - 20% (week 8)
  - 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 ELEC 2300 and COMP 1002 or COMP 1004 (prior to 2013-14) or COMP 2011 or COMP 2012H
  - Basic computer organization knowledge, computer system, CPU, memory hierarchy, interrupt, DMA, storage hierarchy, I/O devices

- Programming
  - UNIX environment – CASS account
  - C/C++ programming

Lecture Format

- Lectures:
  - Lecture notes are made available before the lecture

- Tutorials and Labs
  - Unix environment, editor, how to compile and run programs, Makefile
  - Labs on **Nachos** - instructional software on UNIX
  - Supplement the lectures with more examples and exercises
  - Programming or project instructions

- Reading the corresponding materials in the textbook
  - Lecture notes do not and can not cover everything

- Chapter Summaries
  - Comprehensive summary at the end of each chapter

Assignments

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

Midterm and Final Examinations

- Midterm Exam
  - October 23, 2015 (Friday) 7:00 pm – 9:00 pm
  - Venues: LTC and LTD
- Final Exam
  - TBD
- All exams are closed-book and closed-notes
- No make-up exams will be given unless
  - under very unusual circumstances, e.g., sickness, with letters of proof
  - The instructor must be informed before the exam

Tips for Learning

- Attend lectures
  - Download lecture notes prior to lectures
  - Important concepts are explained
- Complete homework independently
  - This is an exercise to test your knowledge and how much you learn
- Spend 30 minutes each week to review the content
  - Weekly or chapter summary can help
  - This can save you lots of time later when you prepare for exams
  - You can not expect to learn everything 2-3 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

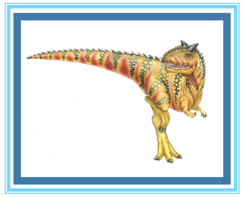
What you are suppose to learn

- 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 (3 lectures)
  - Basic OS concept
  - System architecture
- Process and Thread (11 lectures)
  - Process and thread (3 lectures)
  - CPU scheduling (3 lectures)
  - Synchronization (3 lectures)
  - Deadlock (2 lectures)
- Memory and storage (10 lectures)
  - Memory management (3 lectures)
  - Virtual memory (3 lectures)
  - File system (2 lectures)
  - Secondary storage and I/O (2 lectures)

Chapter 1: Introduction



Chapter 1: Introduction

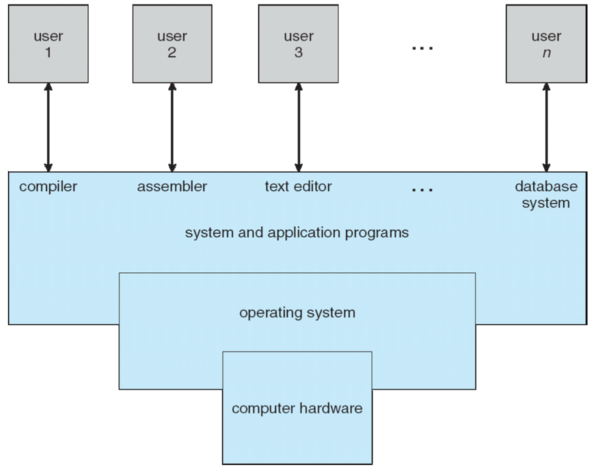
- Computer-System Architecture
- What Operating Systems Do
- Operating-System Structure
- Computing Environment

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

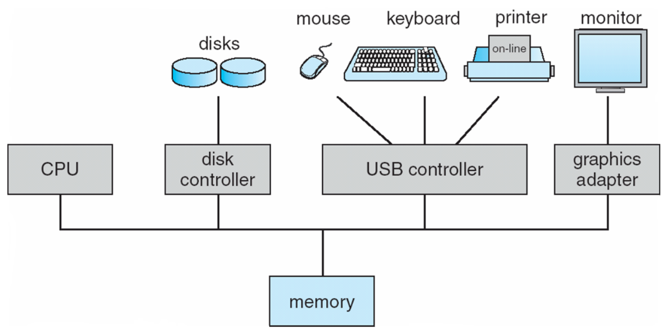


Computer Startup

- **Bootstrap program** is loaded at power-up or reboot time
  - Typically stored in ROM or EPROM (erasable programmable ROM), known by a general term **firmware**
  - It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
  - It loads operating system kernel and starts execution, which can start to provide services to the system and users.
  - Some services are provided outside the kernel, by system programs that are also loaded into memory at boot time to become system processes, or system daemons that run the entire time when the kernel is running.
  - On UNIX, the first system process is "*init*", and it starts other daemons.

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



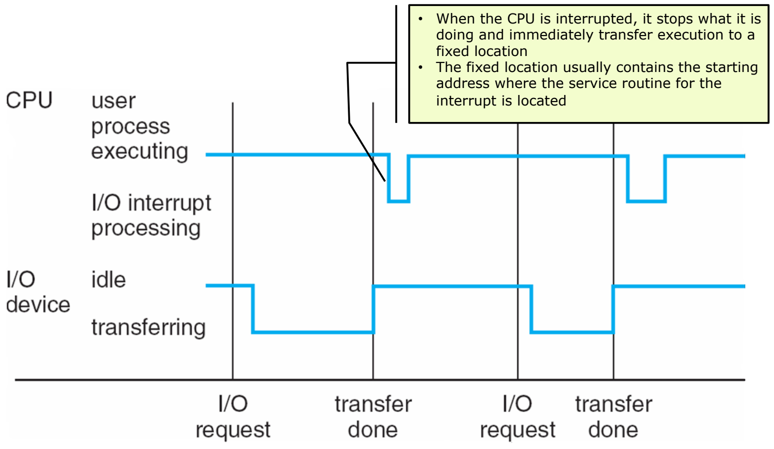
Computer-System Operation

- 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 an interrupt service routine generally, through an interrupt vector, which contains the addresses of all interrupt service routines (also called interrupt handlers), which is a program that handles a particular interrupt
- 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



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, in bit per second or **bps**).

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 that are 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

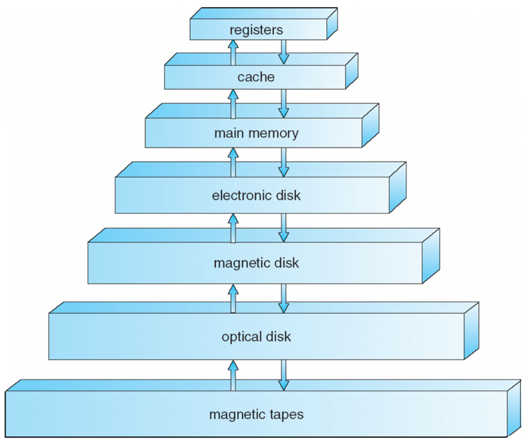
Storage Hierarchy Structure

- 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
  - HIT: if it is, information used directly from the cache (fast)
  - MISS: 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 (services)
  - 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

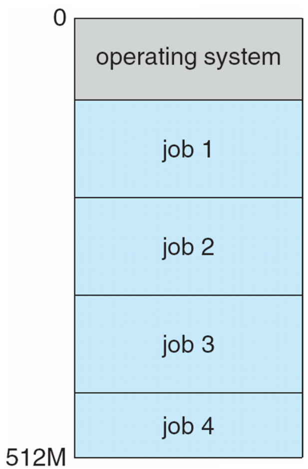
Operating System Definition

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But this varies greatly across systems
- "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
- Mobile OS often includes a core kernel and middleware that supports databases, multimedia, and graphics (to name a few)

Operating System Structure

- **Multiprogramming** needed for efficiency
  - A single program 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, while all jobs are initially kept on the disk in the **job pool**
  - One job selected and run via job scheduling. When it has to wait (for I/O for example), CPU switches to another job
- **Timesharing** (**multitasking**) is a logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing platform – user input vs. computer response
  - Response time typically should be < 1 second
  - Each user has at least one program executing in memory ⇨process
  - If several jobs are ready to run at the same time ⇨ CPU scheduling
  - A time-shared OS allows many users to share the computer simultaneously, giving each user the impression that the entire computer is dedicated to it.

Memory Layout for Multiprogrammed System

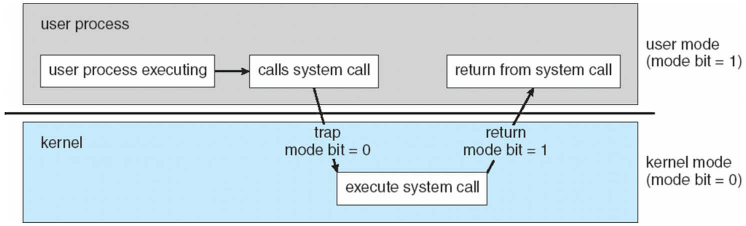


Operating-System Operations

- **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**, which can only be only executable in kernel mode (for example, I/O control, timer and interrupt management)
    - System call changes the mode to kernel mode, return from the system call and resets the mode to user mode

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 that 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

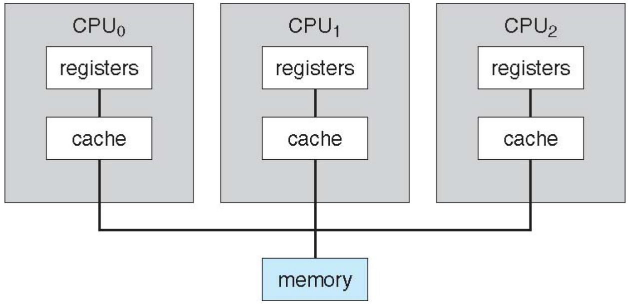
Single-Processor System

- Until recently, most systems used a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well, which may come in the form of device-specific processors, such as disk and graphics controllers, or on mainframes like I/O processors
  - All of these special-purposes processors run a limited instruction set and do not run user processes.

Multiprocessor Systems

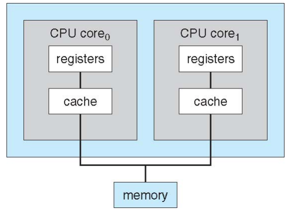
- Multiprocessors systems have begun to dominate the landscape of computing, also known as parallel systems, or multicore systems
- Two or more processors in close communication, sharing computer bus and sometime clock, memory, and peripheral devices.
  - Advantages include:
    1. Increased throughput – more work can be done
    2. Economy of scale – cost less than equivalent multiple single -processor system
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types:
    1. Asymmetric Multiprocessing – one master CPU distributes tasks among multiple slave CPUs, or boss-worker relationship
    2. Symmetric Multiprocessing – most common

Symmetric Multiprocessing - SMP



- Each processor has its own set of registers and local cache
- All processors share physical memory

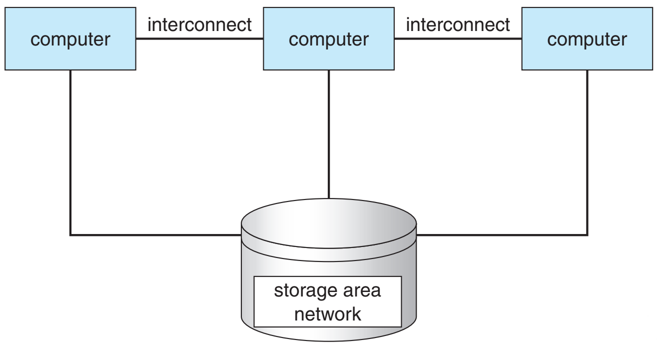
A Dual-Core Design



- Multiprocessing can change memory access mode from Uniform Memory Access (**UMA**) to Non Uniform Memory Access (**NUMA**)
- In UMA, access to any RAM from any CPU takes the same amount of time, while in NUMA, some parts of memory may take longer to access than other parts, creating a performance penalty, which OS needs to minimize
- **Multicore:** multiple computing cores on a single chip, fasters (on-chip communications) and less power than multiple single-core chips.

Clustered Systems

- Unlike multiprocessor systems, a cluster system is composed of two or more individual systems, and considered loosely coupled
- Each system can be a singe-processor system or a multicore



Clustered Systems

- A cluster system usually shares storage via a storage-area network (SAN) and provides a high-availability service which survives failures
  - Asymmetric clustering has one machine in *hot-standby mode* (does nothing but monitoring) while other is running applications
  - Symmetric clustering has multiple nodes running applications, and are monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization to run on all computers in the cluster concurrently
  - Some have distributed lock manager (DLM) to avoid conflicting operations when accessing shared data

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

Computing Environments - Traditional

- Stand-alone general purpose machines, yet 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 breaches

Computing Environments - Mobile

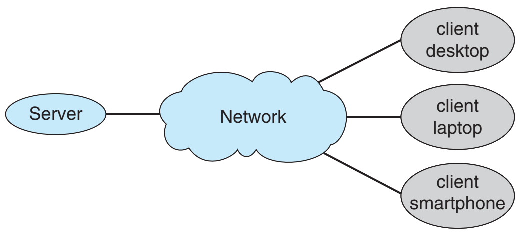
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a "traditional" laptop?
- Extra features – more OS features (GPS, accelerometers, and gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for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

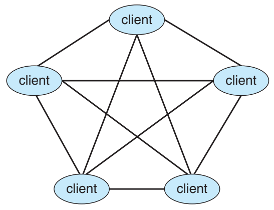
Computing Environments – Client-Server

- 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 a 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

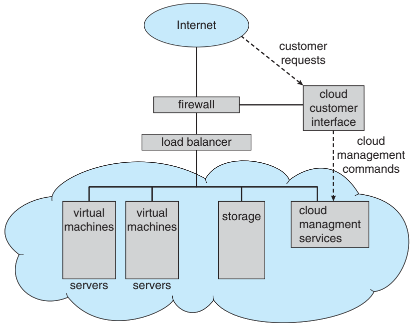
- Allows operating systems to run as applications within other OSes
  - Vast and growing industry
- **Virtualization** – OS natively compiled for CPU, running within another OS, also native to that CPU
  - This allows the user to install multiple operating systems to run applications written for operating systems other than the native host
  - An Apple laptop running Mac OS X on x86 CPU can run a Windows guest to allow execution of Windows 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)

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
  - Car engines, robots, DVDs, microwave ovens, everywhere.
  - Vary considerable, special purpose, limited purpose OS, real-time OS
  - Usage expanding rapdily
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Embedded systems almost always run real-time operating system,
- 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, Linux is the most common one, while Microsoft Window is a well-known close-source approach
- Apple's Mac OS X and iOS, hybrid approach containing an open -source kernel named Darwin yet with other close-source components
- **Benefits**: programmers can contribute to the code, arguably more secure, bugs may be easily located or faster
- Counter to the copy protection and Digital Rights Management (DRM) movement, otherwise would not be effective if code are open-source
- Free Software Foundation (FSF) – Richard Stallman started GNU project in 1983 to create a free and open-source UNIX compatible OS
- Examples include GNU/Linux, BSD UNIX (including core of Mac OS X), and Sun Solaris

End of Chapter 1