Lecture 1: Introduction

Instructor: Mitch Neilsen

Office: N219D

Administrivia

- Course web page: K-State OnLine – CIS 520

  https://online.ksu.edu/Axio/UMS/CourseLogin?course=CIS520

- Textbook: Operating System Concepts, 7th-9th ed., by Silberschatz, Galvin, and Gagne

- Office: N219D, M,W 10:00-11:30, open door policy

- Lectures and Labs:
  - Lectures: M,W 11:30 (B), N236, or 1:30 (A), N122
  - Labs: F 11:30, N128/N126, or 1:30, N128/N122

Course Goals

- Introduce you to operating system concepts
  - Hard to use computer effectively without interacting with OS
  - Understanding the OS makes you a more effective programmer
- Cover important systems concepts in general
  - Caching, concurrency, memory management, I/O, protection ..
- Teach you to deal with larger software systems
  - Version control, automated testing, teamwork
  - Warning: Some people will consider this course to be hard
- Prepare you to take advanced systems courses

Programming Assignments

- Start by implementing parts of the Pintos Operating System
  - Built for x86 hardware, use emulators – bochs and qemu
- Implementation projects: threads, multiprogramming, virtual memory, file systems, etc.
- Project 0 – set up Pintos, and add a new test case, this Friday
  - Complete individually
- Implement remaining projects in team of 2-3
  - Select your partners soon

Grading

- Homework = 10%
- Quizzes = 25%
- Final = 15%
- Projects = 50%
  - For each project, 50% based on passing test cases, remaining points based on design and style.
  - Most team's projects will pass most test cases
    - Please turn in working code, or only partial credit for test cases
  - Means design and style matter a lot
    - Large software systems are not just about producing working code
    - Need to produce code that other people can understand
    - That's why we have group projects

Style

- Must turn in design document with code
  - We supply a template for each project’s design document
- GTA will manually inspect code for correctness
  - Must implement the design
  - Must handle corner cases (e.g., handle malloc failure, etc.)
- Will deduct points for error-prone code w/o errors
  - Don’t use global variables if automatic ones suffice
  - Don’t use deceptive names for variables
- Code must be easy to read
  - Indent code, keep lines and (when possible) functions short
  - Use uniform coding style (try to match existing code)
  - Include comments
  - Don’t leave in reams of commented-out garbage code

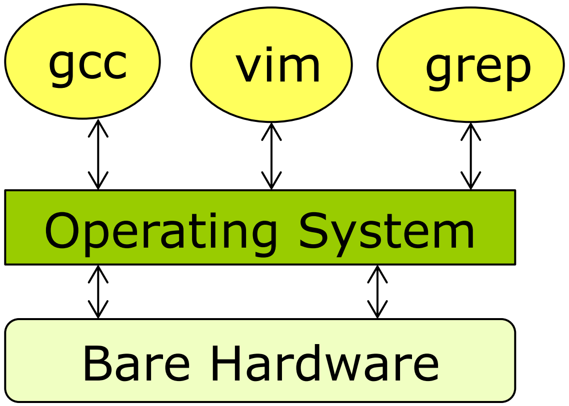
Assignment Requirements

- Don’t looks at other teams solutions to the projects
- Can read, but don’t copy verbatim other OS code; e.g., Linux, Open/FreeBSD, etc.
- Cite any code that inspired your code
  - As long as you cite what you use, it is not cheating
- Projects due generally in two weeks, and homework is generally due in one week from date assigned.
  - Five points are deducted for each “working = M-F” day late

Course Topics

- Introduction – What is an Operating System?
  - Layer of Abstraction – common system calls
  - Resource manager providing access to system resources
  - Control program providing protection and security
- Process and Thread Management
  - Concurrency and synchronization
  - Scheduling
- Memory Management – Virtual Memory, Caching
- File Systems – I/O, secondary storage, networked file systems
- Protection and Security
- Contemporary Op. Sys. – Linux, Android, Windows 7/8
  - Virtualization, cloud computing

What is an Operating System?

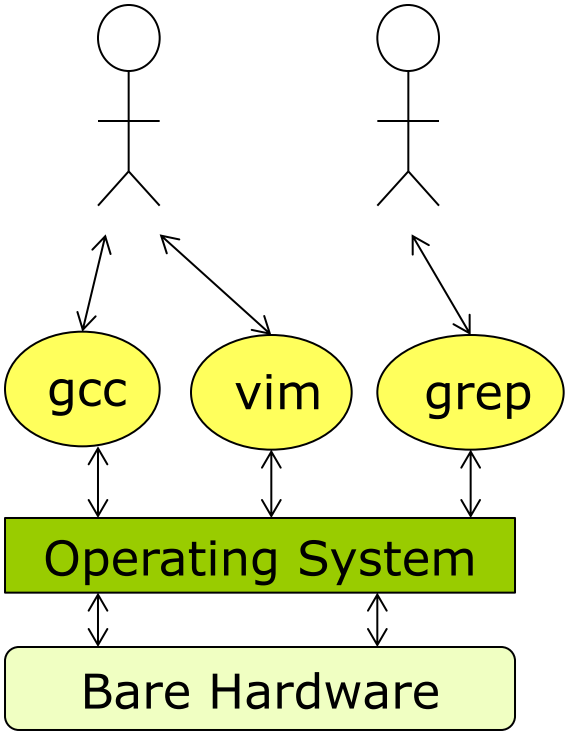


- **A layer between applications and hardware**

- Operating system goals:
  - Provide a layer between applications and hardware
  - Make hardware useful to the programmer
  - [Usually] Provide abstractions for applications (user programs)
    - Manage and hide details of hardware
    - Provide access to hardware through system call interface
  - [Often] Provide protection – prevent one process/user from clobbering another one, and security – non-disclosure, data integrity
  - 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:
  - **Users**
    - People, machines, other computers
  - **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
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices

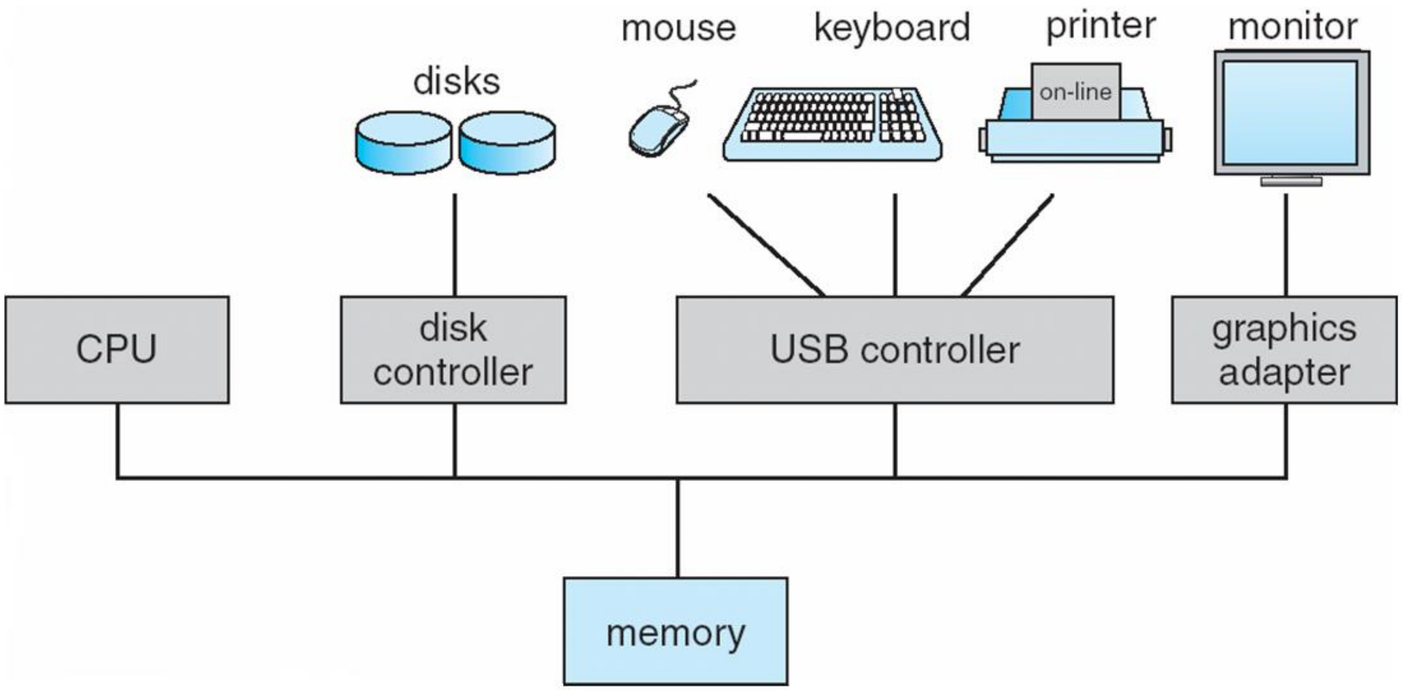


What Operating Systems Do

- 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 dedicate 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 or a specialized user interface, such as embedded computers in devices and automobiles

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

Common Functions of Interrupts

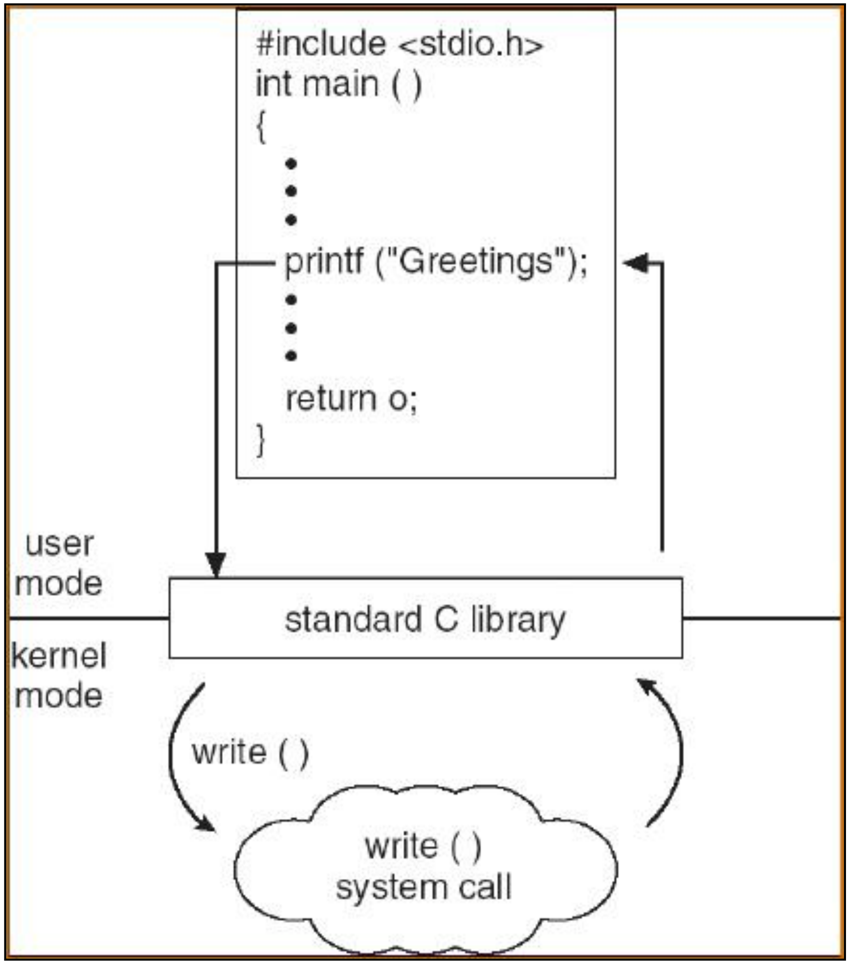
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is typically **interrupt driven**

I/O Structure

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS to allow user program to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

System Call Example

- Standard C library implemented in terms of syscalls
  - printf – in libc has same privileges as application
  - Library function calls write( ) – in kernel, can send bits out on serial port with elevated privileges



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 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 Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory 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

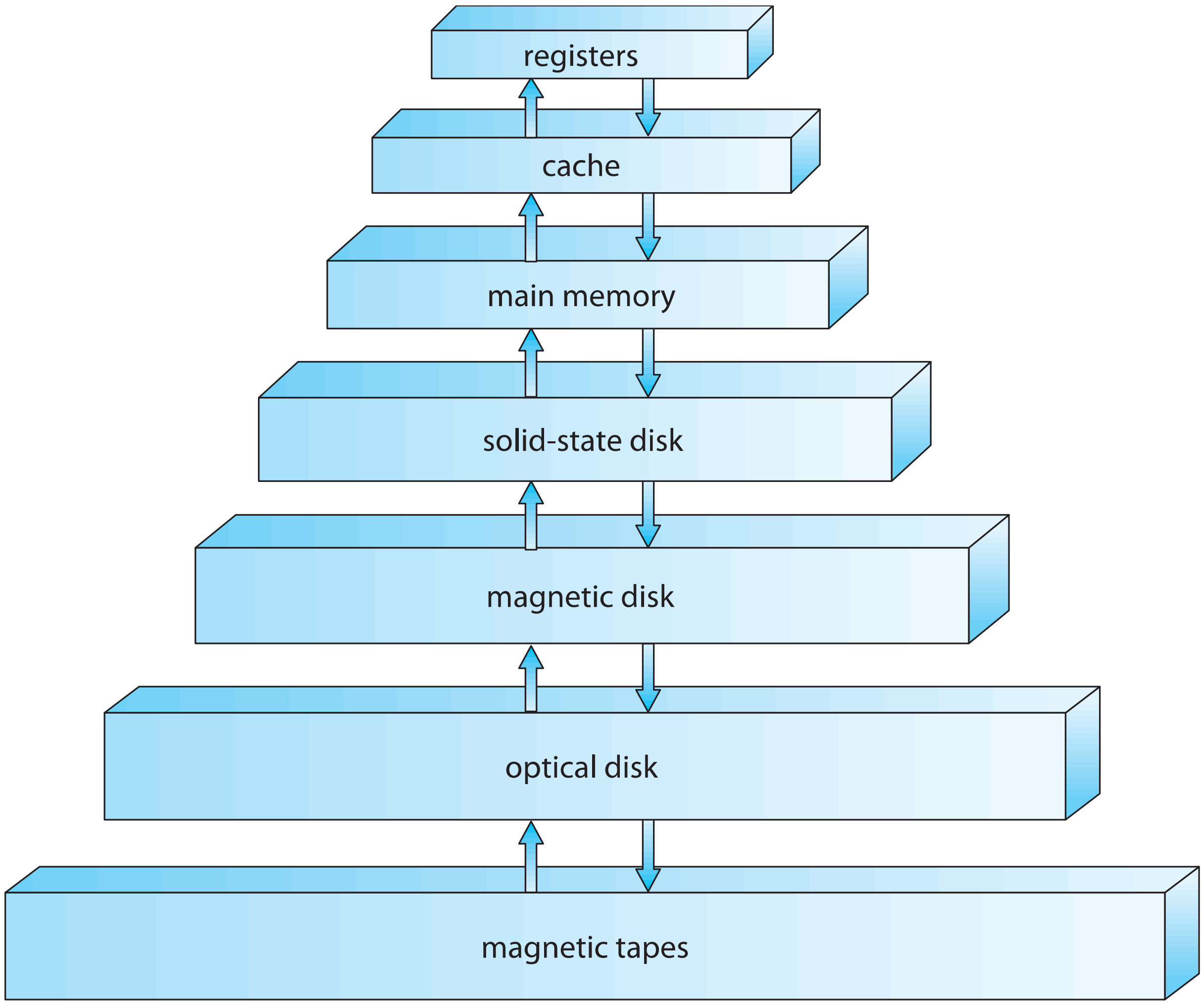
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - 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
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Performance of Various Levels of Storage

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

- Movement between levels of storage hierarchy can be explicit or implicit

Storage-Device Hierarchy



Caching

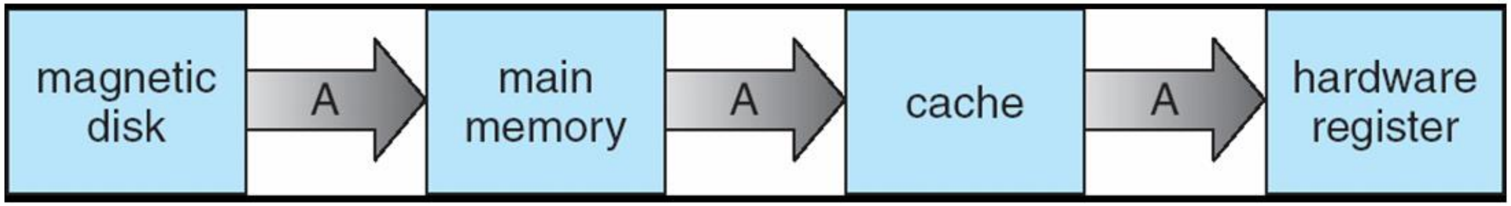
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (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

Useful Properties to Exploit

- Skew
  - 80% of time taken by 20% of code
  - 10% of memory absorbs 90% or references
  - Basis behind cache: put 10% in fast memory, 90% in slow memory, to give illusion of one big fast memory
- Past predicts future (a.k.a. temporal locality)
  - What is the best cache entry to replace?
  - If past = future, then least-recently-used should work well.
- Conflict between fairness and throughput
  - Higher throughput if we keep running the same process
  - But, fairness dictates that we periodically preempt CPU and give it to another process

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 exist
  - Various solutions covered in Chapter 17

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
- 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

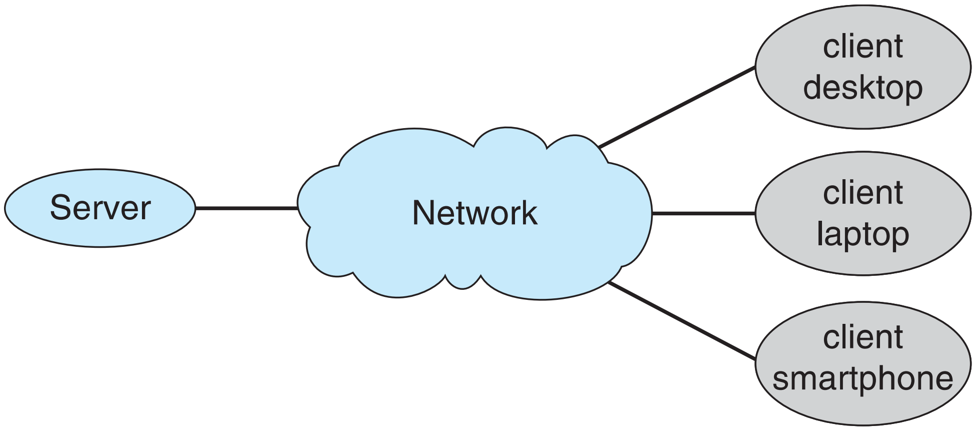
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders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

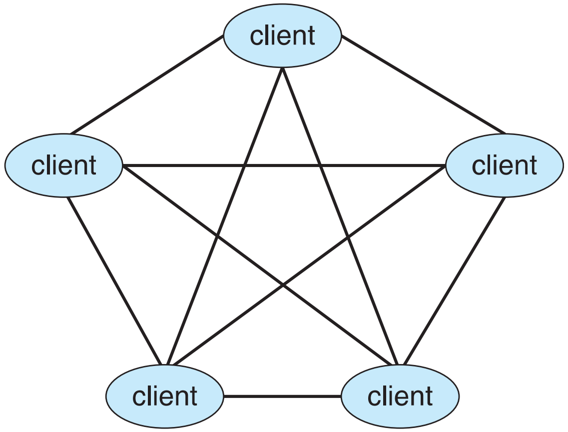
Computing Environments – Client-Server

- 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



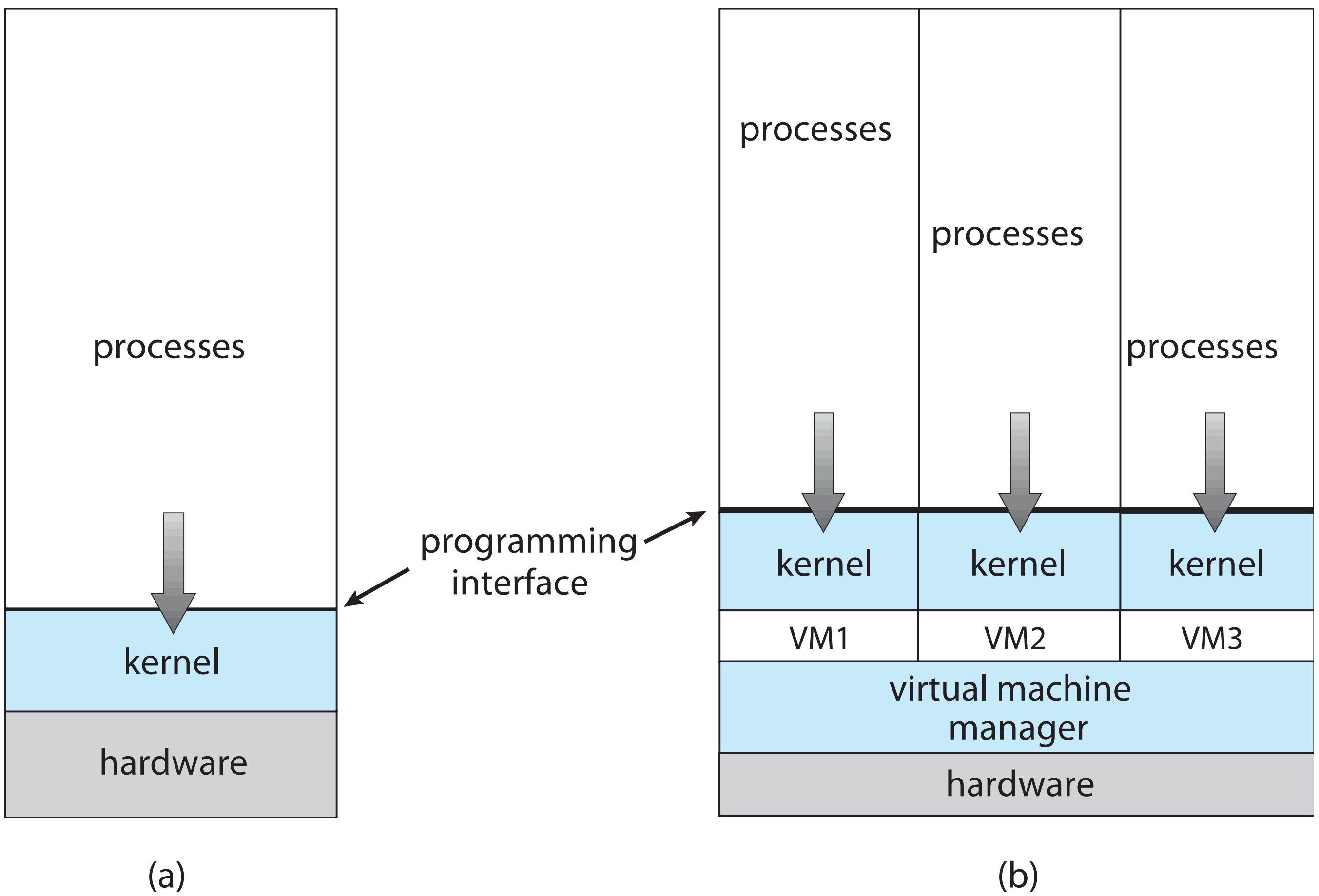
Computing Environments - Virtualization

- 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

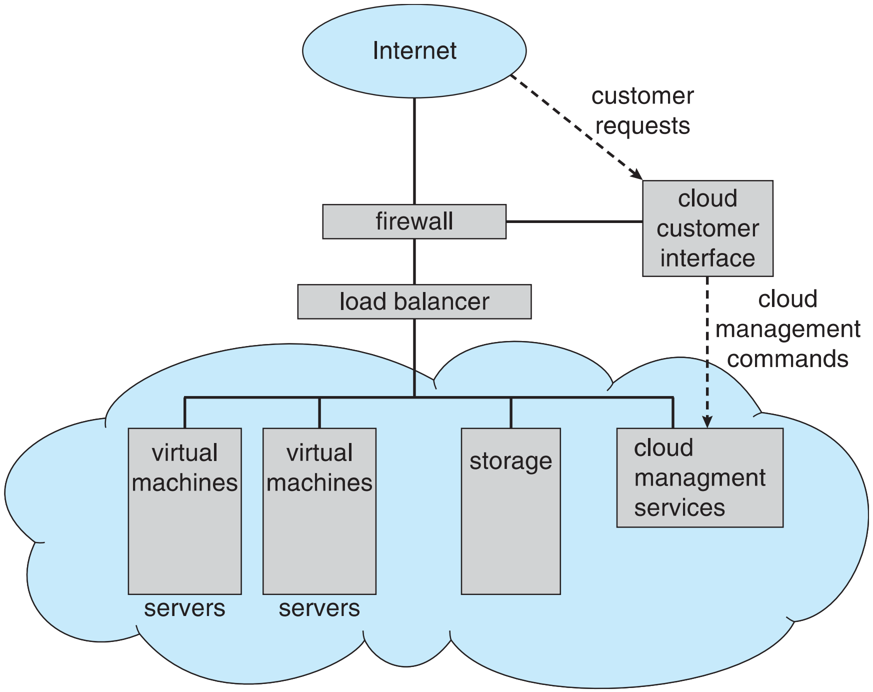


Computing Environments – Cloud Computing

- 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 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**), **Google Android**, **Pintos**, 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

Summary

- Course web page via K-State OnLine has all lecture notes, assignments, handouts, etc.
- Read Ch. 1 – OS Overview
- Obtain CIS Account
- Have a great first week!
- We’ll talk about Pintos on Wednesday.