Operating System

LIAW, RUNG-TZUO

Department of Computer Science and Information Engineering

Fu Jen Catholic University, Taiwan

Chapter 01 Introduction

An Introduction to Operating System

Outline

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed System
- Kernel Data Structures
- Computing Environment
- Free and Open-Source Operating Systems

Objectives

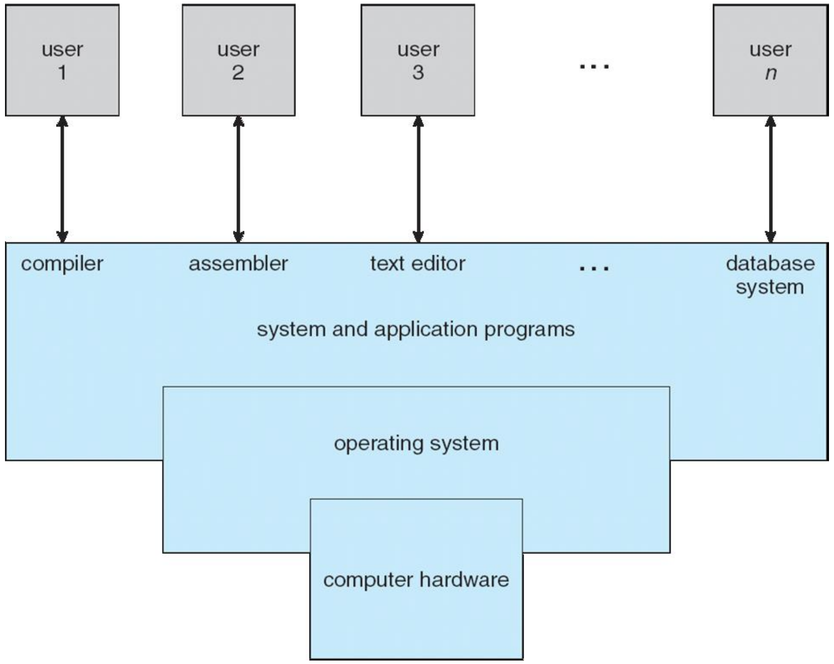
- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

Basics

- Operating system
  - A program that acts as an intermediary between a user of a computer and the computer hardware
  - Manage computer hardware
- 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

What Operating System Do

- A computer system can be divided roughly into four components:
  - ***Hardware*** – provides basic computing resources
  - ***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
  - ***Users*** – People, machines, other computers



What Operating System Do

- Depends on the point of view
- User View
  - PC
    - Ease of use
    - Performance
  - Mainframe (minicomputer)
    - Resource utilization
  - Workstations
    - Connected to networks of other workstations and servers
    - Individual usability vs. resource utilization
  - Mobile computers (smartphones and tablets)
    - Touch screen and voice recognition interface
    - Usability and battery life
  - Embedded computers (home devices, automobiles)
    - **no** user view
    - Run without user intervention

What Operating System Do

- System View
  - OS is a resource allocator
    - Manages all resources
    - Decides between conflicting requests for efficient and fair resource use
  - OS is a control program
    - Controls execution of programs to prevent errors and improper use of the computer
    - Especially I/O devices

What Operating System Do

- Defining Operating Systems
  - Kernel
    The one program running at all times on the computer
  - System program
    - Associated with the operating system but are not necessarily part of the kernel
  - Middleware
    - A set of software frameworks that provide additional services to application developers
    - iOS, Android
  - Application programs
    - Programs not associated with the operation of the system

What Operating System Do

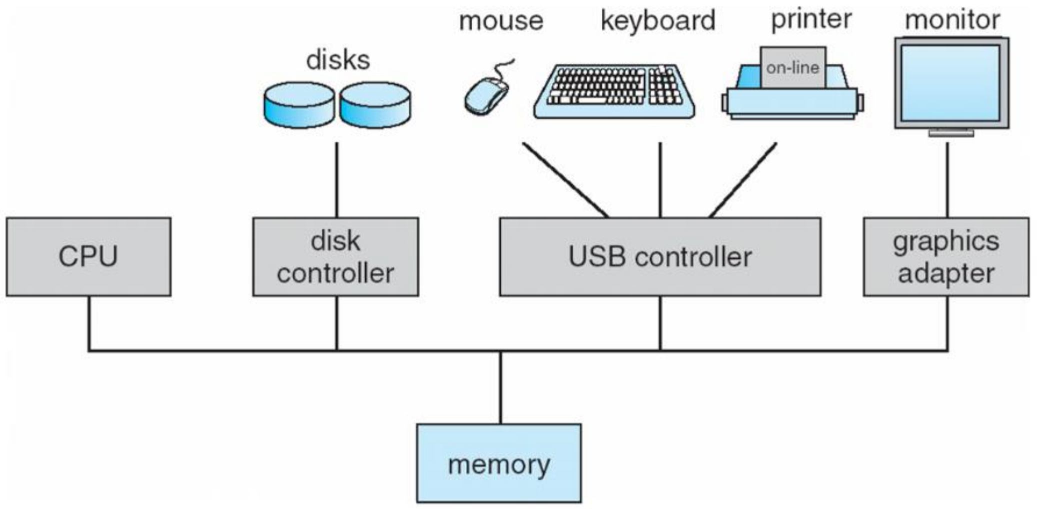
- Why study operating systems?
  - Almost all code runs on top of an operating system
  - Crucial to proper, efficient, effective, and secure programming
  - Useful to those who program them, write programs on them and use them

Exercises

- What are the three main purposes of an operating system?
- We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?

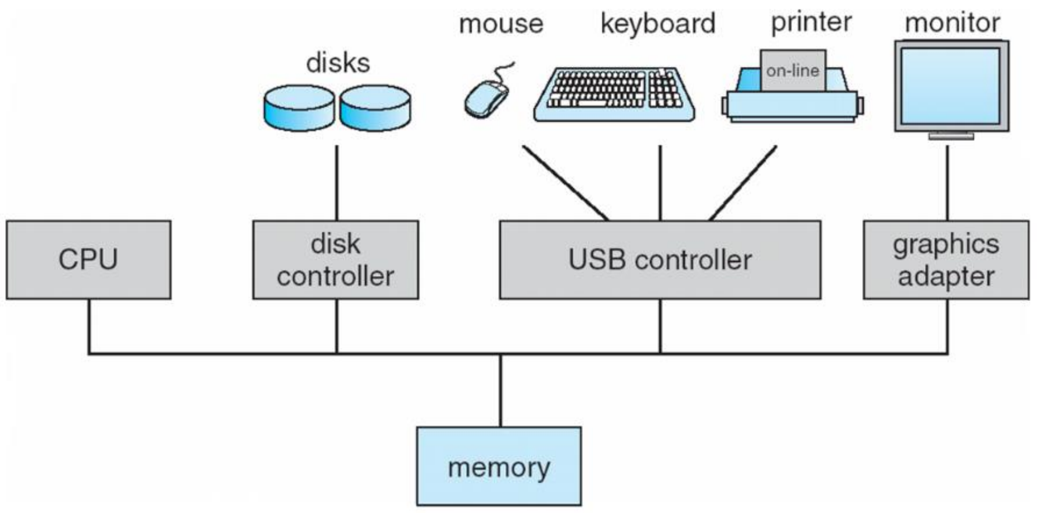
Computer System Organization

- A modern general-purpose computer
  - Include one or more CPUs and device controllers
  - Connect through common bus providing access to shared memory
  - One device driver for each device controller
    - Provide OS with a uniform interface to the device



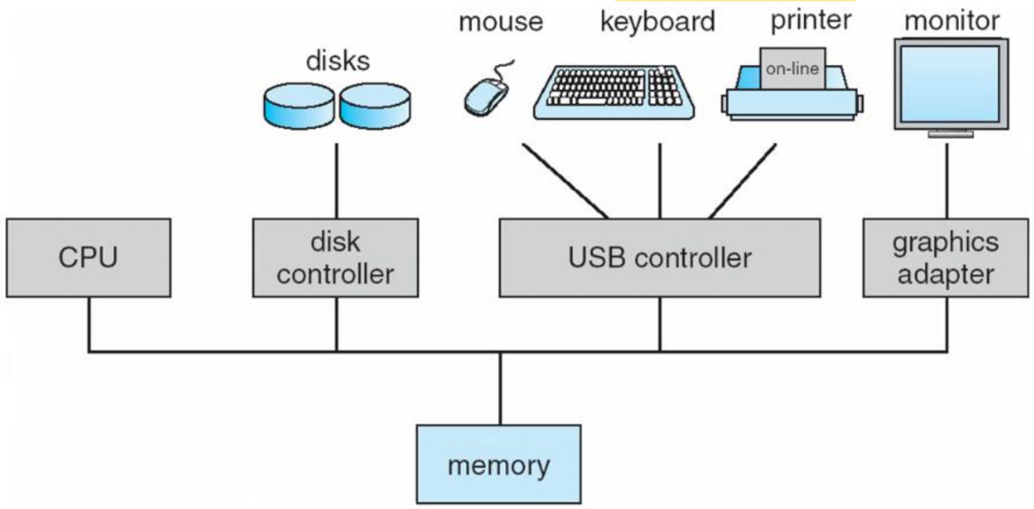
Computer System Organization

- I/O devices and the CPU can execute concurrently
  - CPU and device controller compete for memory cycles
  - 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



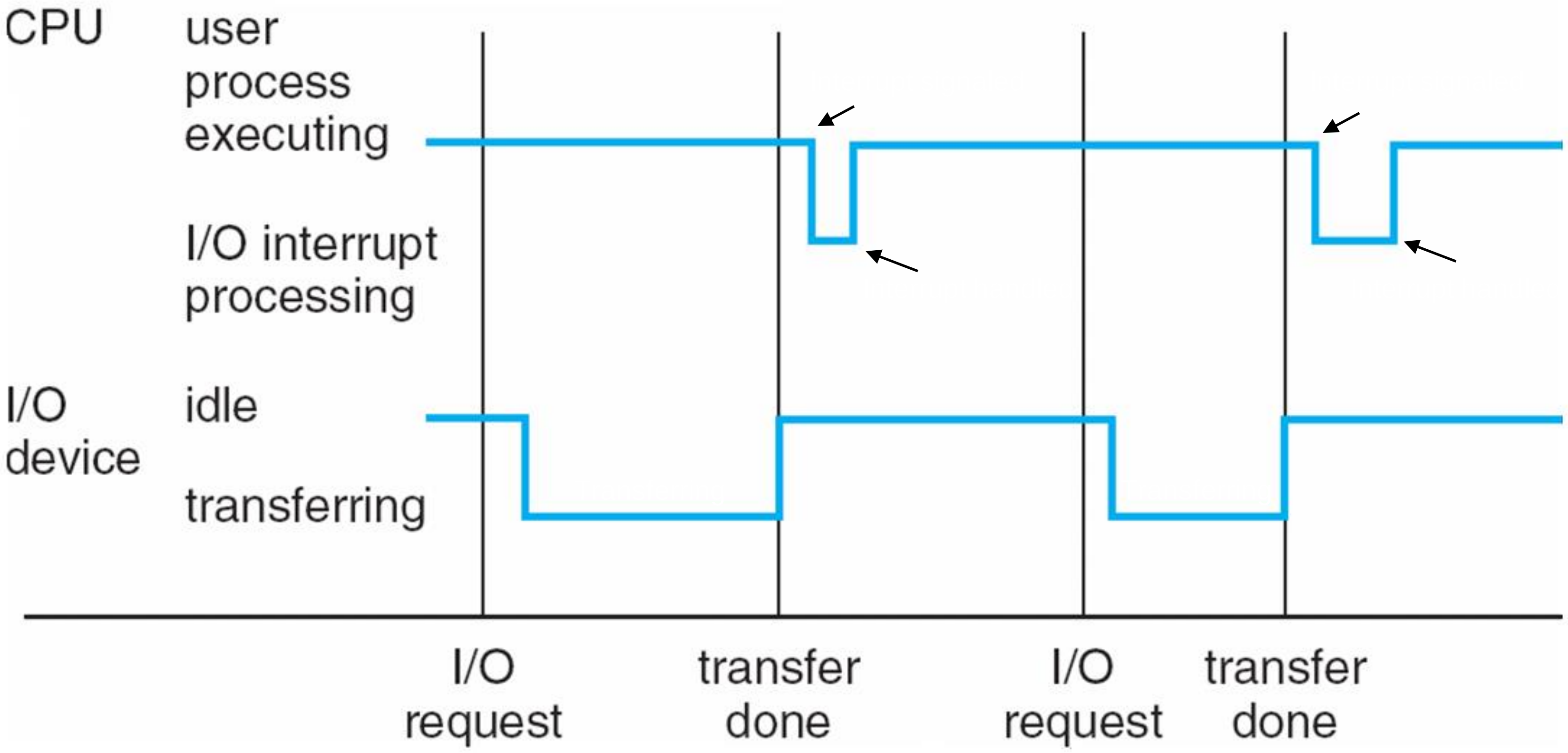
Computer System Organization

- Consider a program performing I/O
  - Device driver loads appropriate registers
  - Device controller examines registers for determining actions
  - Controller starts the transfer of data
  - Controller informs the device driver when complete
  - Device driver then gives control to operating system.
- How does the controller inform the device driver that it has finished its operation? Interrupt



Computer System Organization

- Interrupt Timeline



Computer System Organization

- The occurrence of an event is usually signaled by an interrupt
  - A key part of how operating systems and hardware interact
  - Software: trigger by executing system call (monitor call)
  - Hardware: send signal to CPU by system bus

Computer System Organization

- What happened when CPU is interrupted?
  - Stop what it is doing
  - Transfer execution to a fixed location
    - Contain the starting address where the service routine for the interrupt is located
  - Execute interrupt service routine
  - Resume the interrupted computation

Computer System Organization

- How to transfer control to the appropriate interrupt service routine?
  - Invoke a generic routine to examine the interrupt information
    - Call interrupt-specific handler
    - A table of pointers to interrupt routines can be used for speed
  - Interrupt vector
    - Stored in low memory
    - Addresses of the interrupt service routine for different devices
  - Using a unique index number given with the interrupt request to get the address of the interrupt service routine

Computer System Organization

- Implementation of Interrupt
  - Use wired interrupt-request line
  - Sense and detect signal
  - Read the interrupt number as an index
  - Jump to interrupt-handler routine
    - Save states it will be changing
    - Determine the cause
    - Processing
    - Restore states
    - Return the CPU to the execution state
      - Execute return_from_interrupt instruction

Computer System Organization

- Handling Interrupt
  - Device controller raise an interrupt
    - Assert a signal on the interrupt-request line
  - CPU catches the interrupt
  - CPU dispatches it to the interrupt handler
  - Interrupt handler clears the interrupt
    - Serve the device

  *Enable CPU to respond asynchronous event

Computer System Organization

- Advanced Interrupt Handling
  - In modern operating systems, we need
    - Ability to defer interrupt
    - Efficient dispatch
    - Multilevel interrupt

Computer System Organization

- Advanced Interrupt Handling
  - Provided by CPU and interrupt-controller hardware
    - Two interrupt request lines
      - Nonmaskable interrupt
      - Maskable interrupt
    - Interrupt chaining
      - Each element in the interrupt vector points to the head of a list of interrupt handlers
      - Interrupt handlers called one by one
      - Compromise between table size and inefficiency
    - Interrupt priority levels
      - Index of interrupt vector as priority
        - ✓ E.g. Intel processors with 256 entries
        - ✓ Nonmaskable (0-31) vs. maskbale (32-255)

Computer System Organization

- Storage structure
  - Main memory
    - Rewritable
    - Where program loaded for execution
    - Random access memory
    - Implemented in dynamic random-access memory (DRAM)
  - Electrically erasable programmable read-only memory (EEPROM)
    - Non-volatile
    - Low speed
    - E.g. bootstrap program, firmware, serial number, hardware information

Computer System Organization

- Storage Definitions and Notation
  - Bit
    - One of two values, 0 and 1.
  - Byte (8 bits)
    - The smallest convenient chunk of storage
  - Word
    - A given computer architecture's native unit of data
    - Made up of one or more bytes.
  - Computer storage
    - A kilobyte, or KB, is 1,024 bytes
    - A megabyte, or MB, is $1,024^2$ bytes
    - A gigabyte, or GB, is $1,024^3$ bytes
    - A terabyte, or TB, is $1,024^4$ bytes
    - A petabyte, or PB, is $1,024^5$ bytes
  - Computer manufacturers often round off these numbers
    - A megabyte is 1 million bytes and a gigabyte is 1 billion bytes.
    - Networking measurements are given in bits

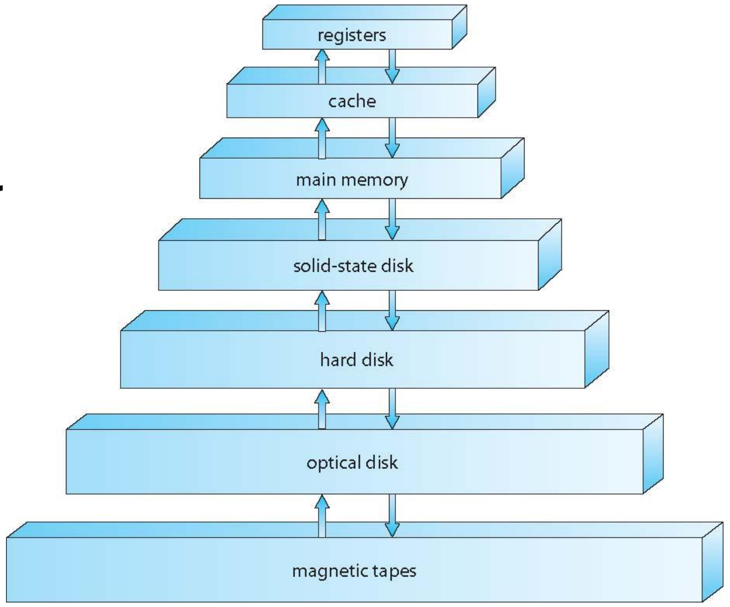
Computer System Organization

- Instruction–execution cycle in Von Neumann architecture
  - Fetch
  - Decode
  - Execute
- Program and data may not reside in memory permanently
  - Small main memory
  - Volatile

Computer System Organization

- Secondary Storage
  - Extension of main memory
  - Non-volatile
  - E.g. hard disk drive (HDD), nonvolatile memory device (NVM)
    - Can be the source and destination of processing

Computer System Orga



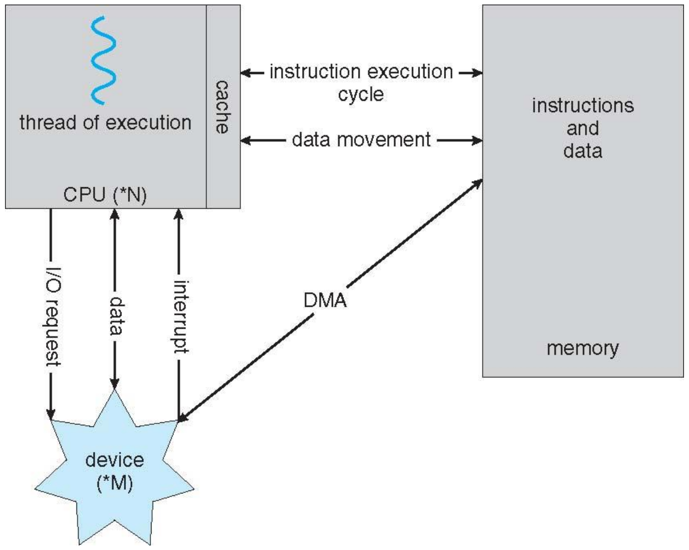
- Storage systems differ in
  - Speed
  - Size
  - Volatility
- Storage systems organized in hierarchy according to speed and size
  - Tertiary storage (Magnetic tape)
  - Semiconductor memory
- Volatile vs non-volatile
  - Volatile: registers, cache, main memory
  - Non-volatile: solid-state disk, hard disk, optical disk, magnetic tapes

Exercises

- Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

Computer System Organization

- I/O Structure
  - Direct Memory Access (DMA)
  - 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



Exercises

- Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load.
  - How does the CPU interface with the device to coordinate the transfer?
  - How does the CPU know when the memory operations are complete?
  - The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.

Computer-System Architecture

- Categorize roughly according to the number of general-purpose processors
  - Single-Processor systems
  - Multiprocessor systems

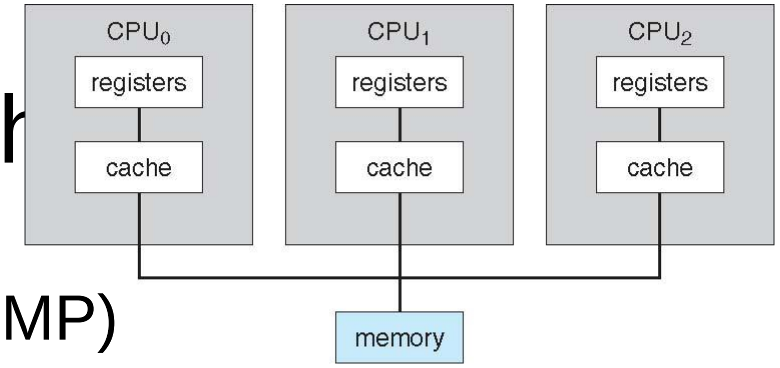
Computer-System Architecture

- Single-Processor Systems
  - A single general-purpose processor
    - One CPU with one core
  - Core as the component
    - Execute instructions
    - Store data locally in registers
  - Also have special-purpose processors
    - Device-specific processors
    - I/O processors
    - Run limited instructions
    - Do **not** run user processes
  - Nowadays only few are single-processor systems
    - With only one general-purpose CPU

Computer-System Architecture

- Multiprocessor systems
  - Advantages
    - Increased throughput
    - Economy of scale
    - Increased reliability
  - Two types:
    - Asymmetric Multiprocessing – each processor is assigned a specie task.
    - Symmetric Multiprocessing – each processor performs all tasks

Computer-System Arch



- Symmetric Multiprocessing (SMP)
  - Each processor performs all tasks
    - Operating system functions
    - User processes
  - Each process owns registers and cache
  - All processors share physical memory
  - Advantage
    - Processes can run simultaneously
  - Disadvantage
    - Need to control load balance
    - Avoid if processors share certain data structures

Exercises

- Many SMP systems have different levels of caches; one level is local to each processing core, and another level is shared among all processing cores. Why are caching systems designed this way?

- Consider an SMP system similar to the one we have shown. Illustrate with an example how data residing in memory could in fact have a different value in each of the local caches.

Computer-System Architecture

- Multicore System
  - Multiple computing cores reside on a single chip
  - More efficient than multi-chip
    - On-chip vs. between-chip communication
  - Less power than multi-chip

Computer-System Architecture

- Definitions of Computer System Components
  - CPU: The hardware that execute instructions.
  - Processor: A physical chip that contains one or more CPUs.
  - Core: The basic computation unit of the CPU.
  - Multicore: Including multiple computing cores on one CPU.
  - Multiprocessor: Including multiple processors

Computer-System Architecture

- Uniform Memory Access (UMA)
  - All processors share memory through system bus
  - Performance bottleneck
- Non-uniform Memory Access (NUMA)
  - Shared system interconnect
  - Scale effectively
  - Increased latency

Computer-System Architecture

- Blade Servers
  - Including multiple processor boards, I/O boards, and networking boards in one chassis
  - Each blade-processor board
    - boots independently
    - runs its own operating system
  - Consists of multiple independent multiprocessor systems

Computer-System Architecture

- Clustered Systems
  - Two or more individual systems
  - Loosely coupled
  - Connected via a local-area network (LAN) or a faster interconnect such as InfiniBand.
  - Provide high-availability service
    - Graceful degradation
    - Fault tolerant

Computer-System Architecture

- Clustered Systems
  - Structure
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Provide high-performance computing (HPC)
    - Significantly greater computational power
    - Applications must be written to use parallelization
  - Parallel clusters over a wide-area network (WAN)
    - Must provide access control over shared storage
    - Commonly known as a distributed lock manager (DLM)

Operating-System Operations

- Booting
  - Need an initial bootstrap program
    - Stored in firmware such as EEPROM
    - Locate and load operating-system kernel into memory
  - Load system daemons
    - System processes run the entire time the kernel is running
    - Provide service outsize the kernel
    - E.g. “systemd” in Linux

Operating-System Operations

- Trap (exception)
  - Software-generated interrupt
    - Error
    - Request from user program
  - Invoke by system call

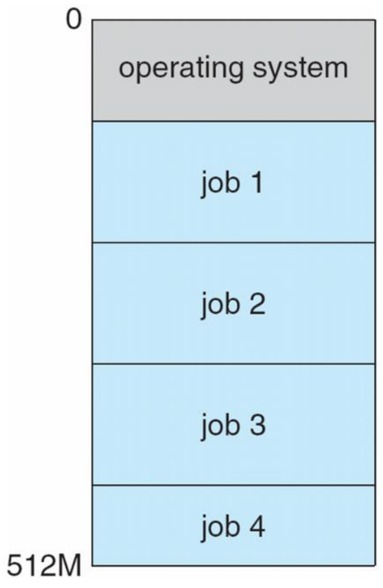
More Exercises

- How does an interrupt differ from a trap?

- Can traps be generated intentionally by a user program? If so, for what purpose?

Operating-System Operations

- Multiprogramming
  - Users want to run more than one program at a time
  - Keep CPU or I/O devices busy
  - Increases CPU utilization
  - Process as running program



Exercises

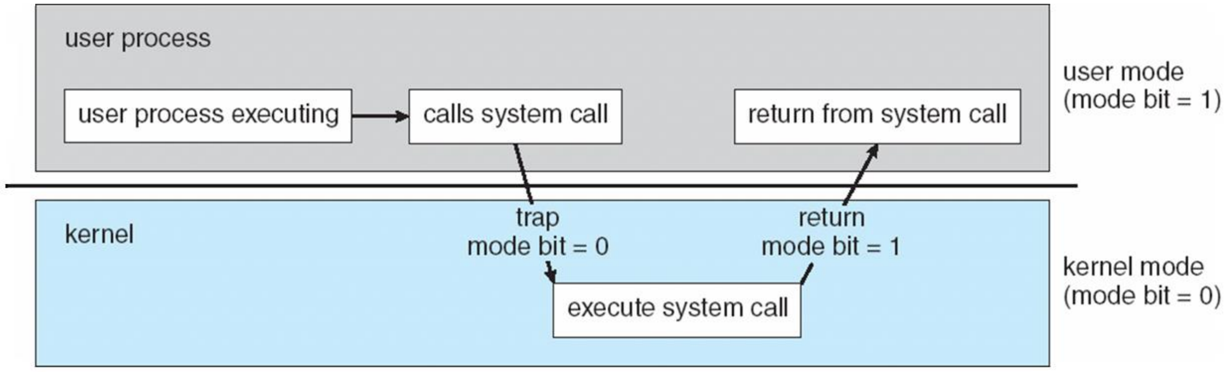
- Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.

Operating-System Operations

- Multitasking
  - A logical extension of multiprogramming
  - CPU switches frequently
  - Result in fast response time
  - Need memory management for multiple processes in memory
  - Need CPU scheduling for selecting among processes
  - Need virtual memory to ensure reasonable response time
    - Allow execution of process not completely in memory
    - Physical memory vs. logical memory

Operating-System Operations

- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. virtual machine manager (VMM) mode for guest VMs



Operating-System Operations.

- Timer to prevent 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 (privileged instruction)
  - When counter zero generate an interrupt
  - E.g. 10-bit counter with 1-millisecond clock allows interrupts at intervals from 1 to 1,024 milliseconds

Exercises

- How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security)?

- Which of the following instructions should be privileged?
  a. Set value of timer.
  b. Read the clock.
  c. Clear memory.
  d. Issue a trap instruction.
  e. Turn off interrupts.
  f. Modify entries in device-status table.
  g. Switch from user to kernel mode.
  h. Access I/O device.

Resource Management

- Process Management
  - A process is a program in execution.
    - 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

Resource Management

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Scheduling processes and threads on the CPUs
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication

Resource Management

- Memory Management
  - Memory = large array of words
    - Word = one or more bytes with single address
  - To execute a program
    - All (or part) of the instructions must be in memory
    - All (or part) of the data 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
    - Different methods are effective to different situations.
  - Memory management activities
    - Keeping track of which parts of memory are currently being used and which process is using them
    - Allocating and deallocating memory space as needed
    - Deciding which processes (or parts of processes) and data to move into and out of memory

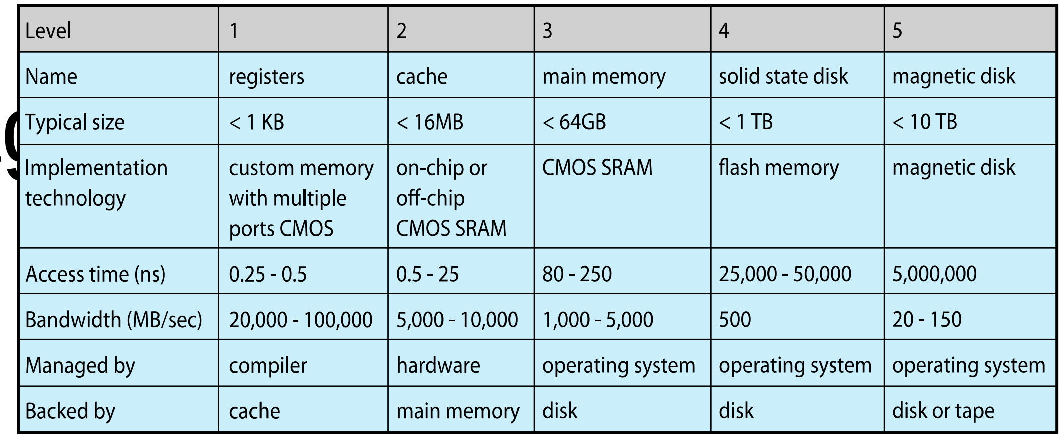
Resource Management

- File-System 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)
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files
    - Creating and deleting directories to organize files
    - Supporting primitives to manipulate files and directories
    - Mapping files onto mass storage
    - Backup files onto stable (non-volatile) storage media

Resource Management

- Mass-Storage Management
  - Usually HDDs or NVMs 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
  - OS activities
    - Mounting and unmounting
    - Free-space management
    - Storage allocation
    - Disk scheduling
    - Partitioning
    - Protection
  - Entire speed of computer operation hinges on disk subsystem and its algorithms
  - Some storage need not be fast
    - Tertiary storage includes optical storage, magnetic tape
    - Still must be managed – by OS or applications

Resource Manag

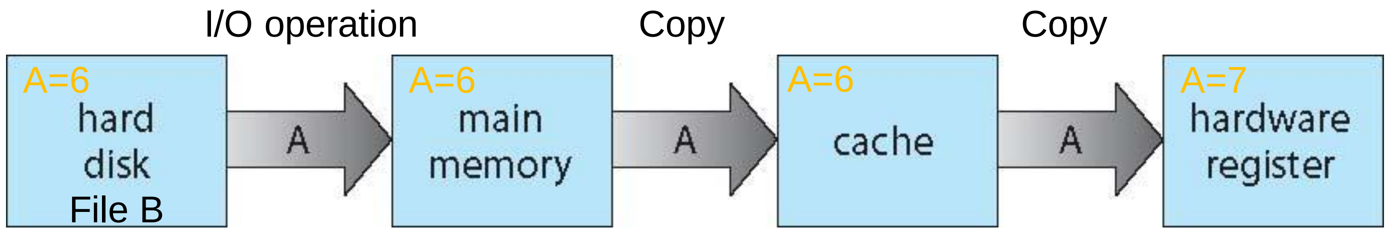


Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 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

- Cache Management
  - Caching
    - Small but faster than main memory
    - For the required information
      - Check if it is in cache
      - Yes: use it
      - No: put a copy in the cahce from main memory
  - High-speed cache: Internal programmable registers
    - E.g. index registers
  - Other caches are implemented totally in hardware.
    - E.g. Instruction and data caches
    - Out of the scope of this course

Resource Management

- Cache Management
  - The movement of information between levels of a storage hierarchy may be either explicit or implicit



  - 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

Exercises

- Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following processing environments:
  a. Single-processor systems
  b. Multiprocessor systems
  c. Distributed systems

Resource Management

- I/O System Management
  - One purpose of OS is to hide peculiarities of hardware devices from the user
  - I/O subsystem responsible for
    - A memory-management component that includes buffering, caching, and spooling
    - A general device-driver interface
    - Drivers for specific hardware devices

Protection and Security

- Protection
  - Access to data must be regulated for multiple-user computer systems
    - E.g. Memory-addressing hardware, timer, device-control registers
  - Any mechanism for controlling access of processes or users to resources defined by the OS
    - Means to specify controls
    - Enforce the controls
  - Improve reliability
    - Detect latent errors
    - Prevent contamination of healthy subsystem

Protection and Security

- Protection is not enough
  - E.g. password stolen
- **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 identifiers (user IDs, security IDs)
      - Name and associated number
      - One per user
      - Unique
    - User ID associated with all files, threads, and processes of that user to determine access control
    - Group identifier (group ID) allows set of users to be defined and controls managed
      - Owner can do all operations, but some users can only read on the file.
    - Privilege escalation allows user to change to effective ID with more rights
      - E.g. setuid to run program with the user ID of the owner of the file

Exercises

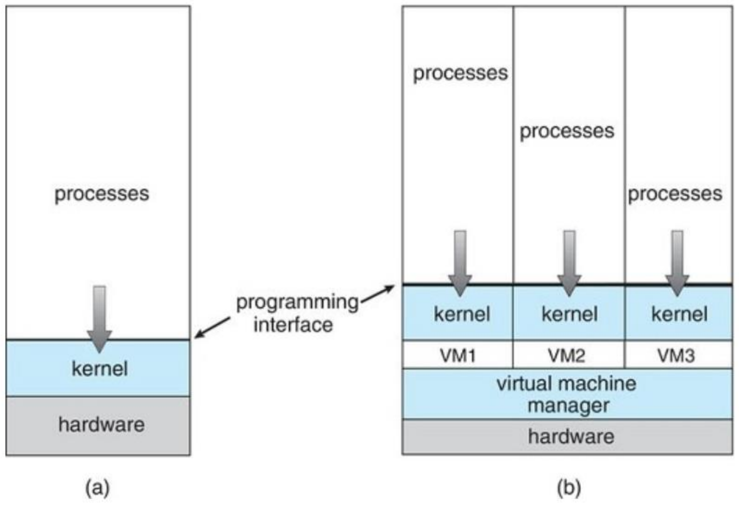
- Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

Virtualization

- Abstract the hardware of a computer into different execution environments
  - Each separate environment seems running on its own private computer
  - These environments have different individual operating systems
  - A user of a virtual machine can switch among the various operating systems
- Allows operating systems to run as applications within other operating systems
- Emulation used when source CPU type different from target type
  - E.g. Apple’s “Rosetta” from IBM Power PC to Intel x86
  - Every machine-level instruction from source system need to be translated to the one in destination system.
  - Slowest

Virtualization

- Virtualization – OS natively compiled for CPU, running guest OSes also natively compiled
  - IBM mainframe for multiple users
  - VMware as the virtual machine manager (VMM) in the host OS (Windows) for multiple guest copies of Windows
- Modern OSes can run multiple applications reliably. Do we still need virtualization?
  - MacBook Pro on the x86 CPU can run Windows 10 for windows application
  - Develop, test, and debug for different OSes on one server
  - Computing environments in data center



Distributed System

- Collection of separate, possibly heterogeneous, systems networked together
  - Increase computation speed, functionality, data availability, and reliability
  - Systems contain a mix of two modes:
    - Generalize network access as a form of file access
    - Invoke network function
    - E.g. file transfer protocol (FTP), network file system (NFS)

Distributed System

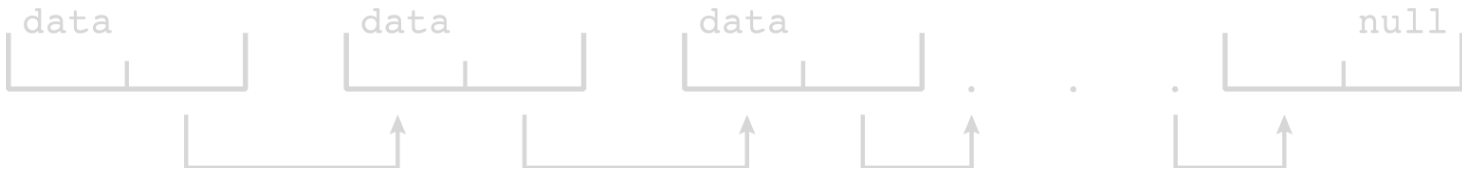
- Network is a communications path, TCP/IP most common
  - Local Area Network (LAN)
  - Wide Area Network (WAN)
  - Metropolitan Area Network (MAN)
  - Personal Area Network (PAN)
    - BlueTooth and 802.11 devices

Distributed System

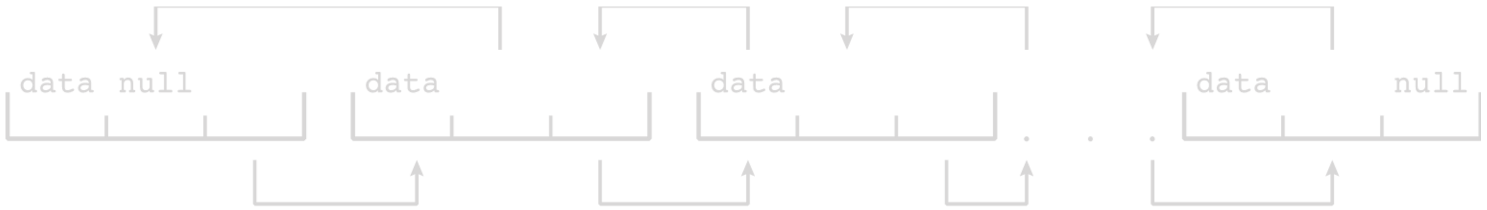
- Network Operating System provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

Kernel Data Structures

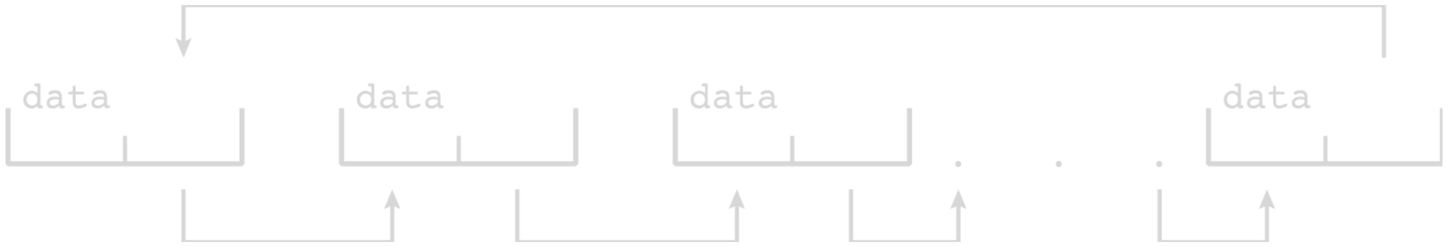
- Similar to standard programming data structures
- Array as simplest data structure
- Singly linked list



- Doubly linked list



- Circular linked list

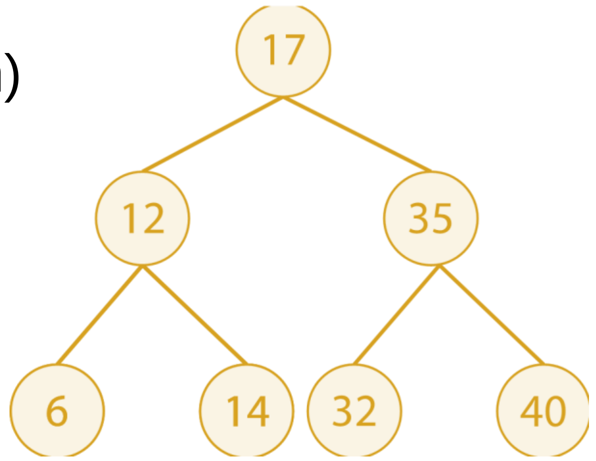


Kernel Data Structures

- Stack
  - Last-in-first-out (LIFO)
  - Insert on top as push
  - Remove on top as pop
  - E.g. function call stack
- Queue
  - First-in-first-out (FIFO)
  - Insert at tail as ENQUEUE
  - Delete at head as DEQUEUE
  - E.g. Ready queue

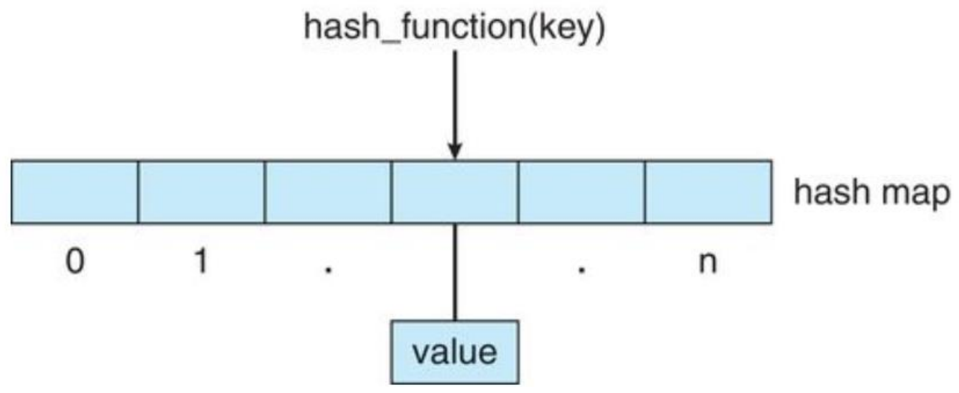
Kernel Data Structures

- Tree
  - Linked through parent-child relations
  - General tree: unlimited number of children
- Binary search tree
  - Binary tree
  - left <= right
  - Search performance is O(n)
  - Balanced binary search tree is O(lg n)
    - Red-black tree



Kernel Data Structures

- Hash function can create a hash map
  - Quickly retrieve data
  - Difficulty due to hash collision
    - Two unique inputs have the same result
  - Implementation
    - Map key:value using hash function



Kernel Data Structures

- Bitmap – string of n binary digits representing the status of n items
  - E.g. 0 0 1 0 1 1 1 0 1
    Resources 2, 4, 5, 6, and 8 are unavailable; resources 0, 1, 3, and 7 are available.
  - Bring space efficiency
  - Represent disk blocks in a disk drive

Computing Environments - Traditional

- 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
  - Portable and lightweight
  - Functionality between laptop is difficult to discern
- Mobile systems for not only web browsing but playing music and video, reading digital books, taking photos, …
- Use unique features to design application
  - E.g. global positioning system (GPS) chips, accelerometers, and gyroscopes
  - Allows new types of apps like ***augmented-reality***

Computing Environments - Mobile

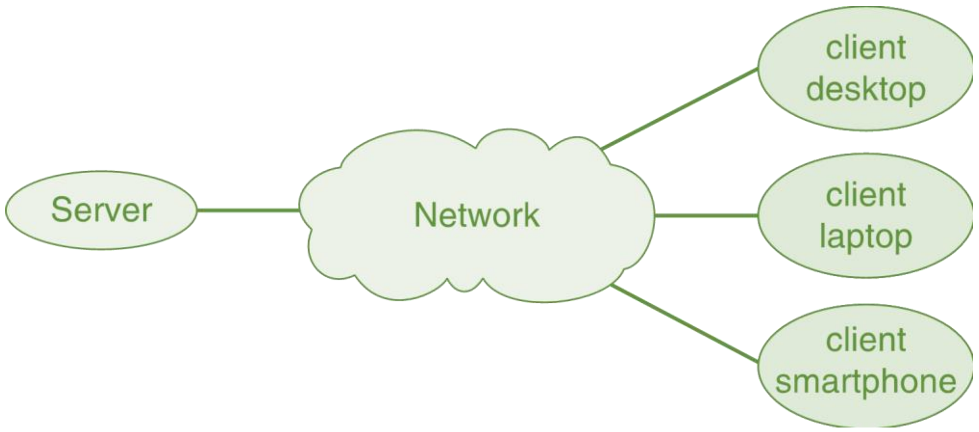
- Use IEEE 802.11 wireless, or cellular data networks for on-line services
- Limitation
  - Memory capacity
  - Processing speed
  - Secondary Storage
  - Power consumption
- Leaders are Apple iOS and Google Android

Exercises

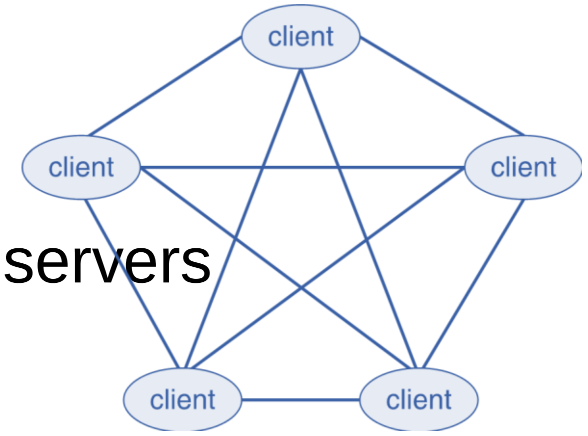
- Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.

Computing Environments – Client-Server

- Client-Server Computing
  - 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



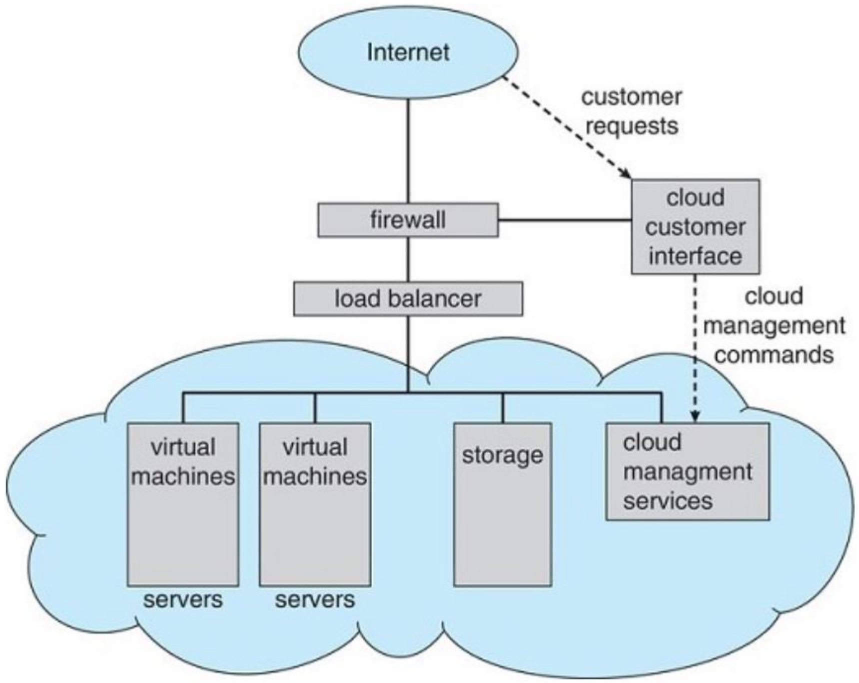
- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype

Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
- 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 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



Computing Environments – Real-Time Embedded Systems

- Most prevalent form of computers
- Vary considerably and use expanding
  - E.g. general purpose computers, hardware devices with special-purpose OS, hardware devices with application-specific integrated circuits (ASICs)
- Almost always run real-time operating systems
  - Has rigid time requirements
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Exercises

- What are some advantages of peer-to-peer systems over client–server systems?

- Describe some distributed applications that would be appropriate for a peer-to-peer system.

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

QA?