Chapter I: Overview

1.1: Introduction

GV: Nguyễn Thị Thanh Vân

Outline

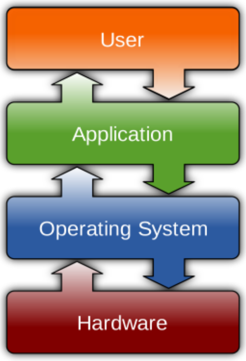
1. What Operating Systems Do
2. Computer-System Organization
3. Computer-System Architecture
4. Operating-System Operations
5. Resource Management
6. Security and Protection
7. Virtualization
8. Distributed Systems
9. Kernel Data Structures
10. Computing Environments
11. Free/Libre and Open-Source Operating Systems

Objectives

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments
- Provide examples of free and open-source operating systems

What is an Operating System?

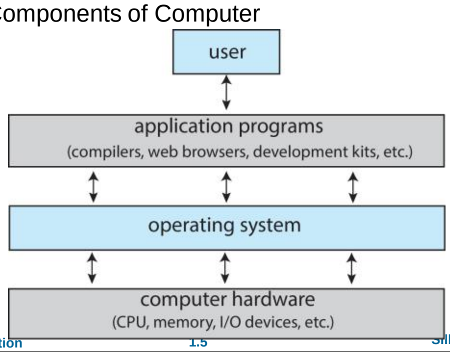
- A program that acts as an intermediary between a user of a computer and the computer hardware
- The **OS** controls the **hardware** and coordinates its use among the various **application** programs for the various **users**



- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources: CPU, memory, I/O devices
  - OS: Controls and coordinates use of hardware among various apps and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, games
  - Users: People, machines, other computers
- Abstract View of Components of Computer



What Operating Systems Do

- **User view**: Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization** (how various HW and SW resources are shared
- **System view**: The shared computer such as **mainframe** or **minicomputer** (many resources) must keep all users happy
  - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention

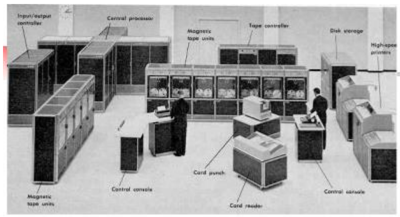
Operating System Definition

- "The one program running at all times on the computer" is the **kernel,** which is part of the operating system
- Everything else is either
  - A **system program** (ships with the operating system, but not part of the kernel) , or
  - An **application program**, all programs not associated with the operating system
- Today's OSes for general purpose and mobile computing also include **middleware** – a set of software frameworks that provide addition services to application developers such as databases, multimedia, graphics

History of Operating System

- The First Generation (1945–55): Vacuum Tubes 7
- The Second Generation (1955–65): Transistors and Batch Systems 8
- The Third Generation (1965–1980): ICs and Multiprogramming 9
- The Fourth Generation (1980–Present): Personal Computers 14
- The Fifth Generation (1990–Present): Mobile Computers 1

Transistors and Batch Systems – Gen2

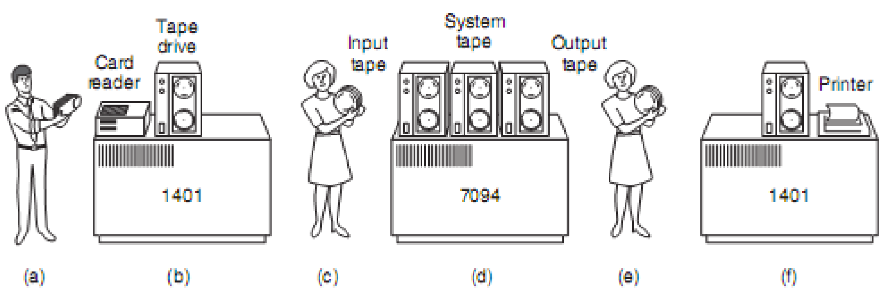


Mainframe computer in 1967

IBM System/360 Mainframe Computer

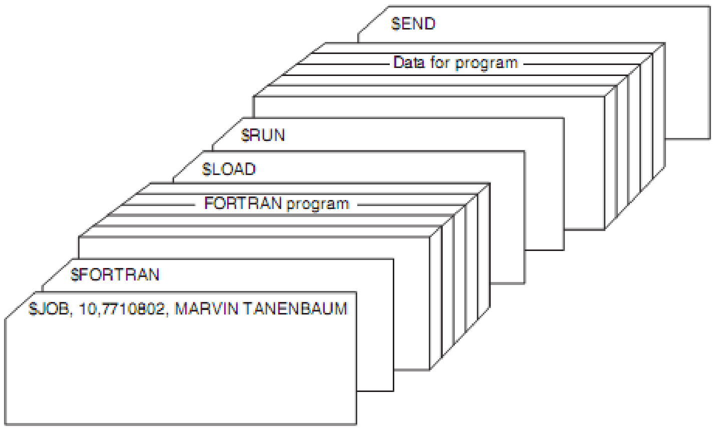
Modern Mainframe Computer

Transistors and Batch Systems



- An early batch system.
  - (a) Programmers bring cards to 1401.
  - (b) 1401 reads batch of jobs onto tape.
  - (c) Operator carries input tape to 7094.
  - (d) 7094 does computing.
  - (e) Operator carries output tape to 1401.
  - (f) 1401 prints output.

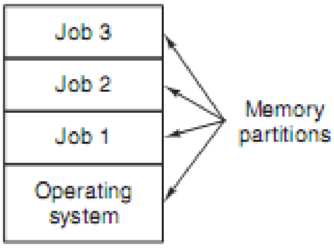
Transistors and Batch Systems



- Structure of a typical FMS job.
- They were largely programmed in FORTRAN and assembly language.
- Typical operating systems were **FMS** (the Fortran Monitor System) and IBSYS, IBM's operating system for the 7094.

Multiprogramming – Gen3

- partition memory into several pieces, with a different job in each partition
- While one job was waiting for I/O to complete, another job could be using the CPU.
- Some Oss
  - UNIX: BSD, SystemV
  - MINIX
  - Linux

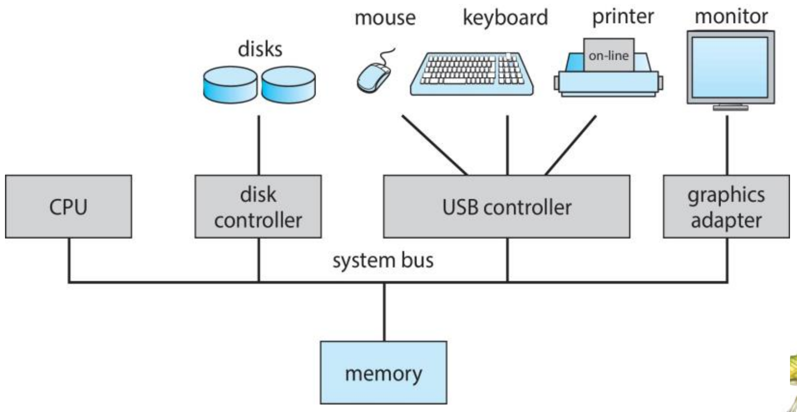


Gen4 & 5

- Personal computer
- Network OS: Windows NT, Server
- Distributed OS
- Mobile, PDA…

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 Organization

- I/O devices and the CPU can execute concurrently
- A particular device type (disk, audio, display) is be in charge by a device controller
- A **device controller**:
  - has a local buffer,
  - is responsible for moving the data between the peripheral devices that it controls and its local buffer storage
  - OSs have **device driver** for each device controller
- **Operation**: (The CPU and the device controllers can execute in parallel)
  - 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**

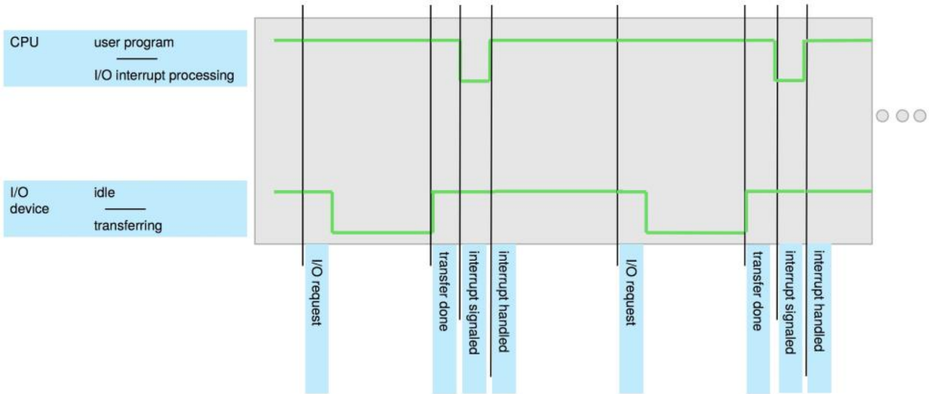
=> **Organization focusing on 3 key aspects: interrupt, storage & I/O structure**

Common Functions of Interrupts

- A interrupt is used for the **device controll**er inform the device driver that it has finished its operation (ex, transfer data from device to local buffer)
- Interrupts must be handled quickly, as they occur very frequently.
  - A table of pointers to interrupt routines can be used instead to provide the necessary speed
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Timeline

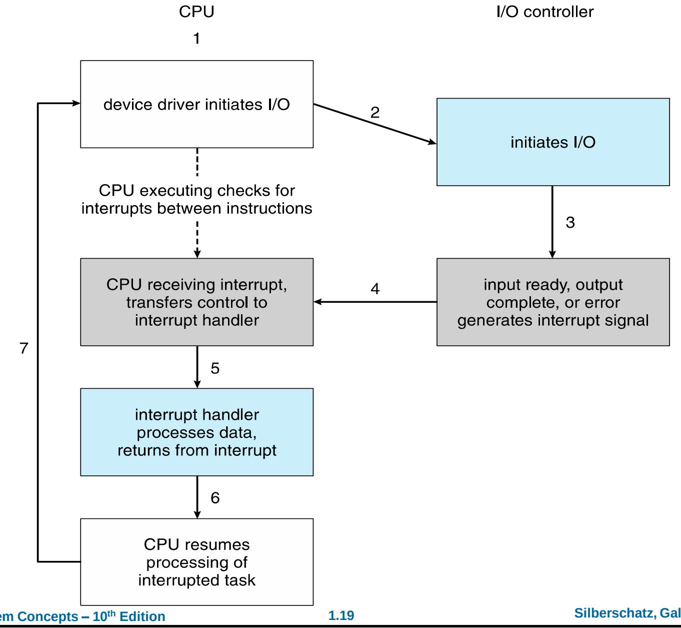
- A timeline of the operation:
  - the CPU is interrupted, it stops what it is doing
  - Immediately transfers execution to a fixed location.
  - The interrupt service routine executes; on completion,
  - the CPU resumes the interrupted computation



Interrupt Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred:
  - Overload (div/0)
  - Timer
  - I/O
  - Hardware failure
  - Trap (software interrupt)
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt-drive I/O Cycle



I/O Structure

Two methods for handling I/O

- 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 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

Storage Structure

- **Main memory** – only large storage media that the CPU can access directly
  - Typically, **volatile**
  - Typically, **random-access memory (RAM)** in the form of **Dynamic Random-access Memory** (**DRAM**)
- Secondary storage – extension of main **memory** that provides large **nonvolatile** storage capacity
- **Hard Disk Drives** (**HDD**) – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Non-volatile memory** (**NVM**) devices– faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular as capacity and performance increases, price drops

Storage Definitions and Notation Review

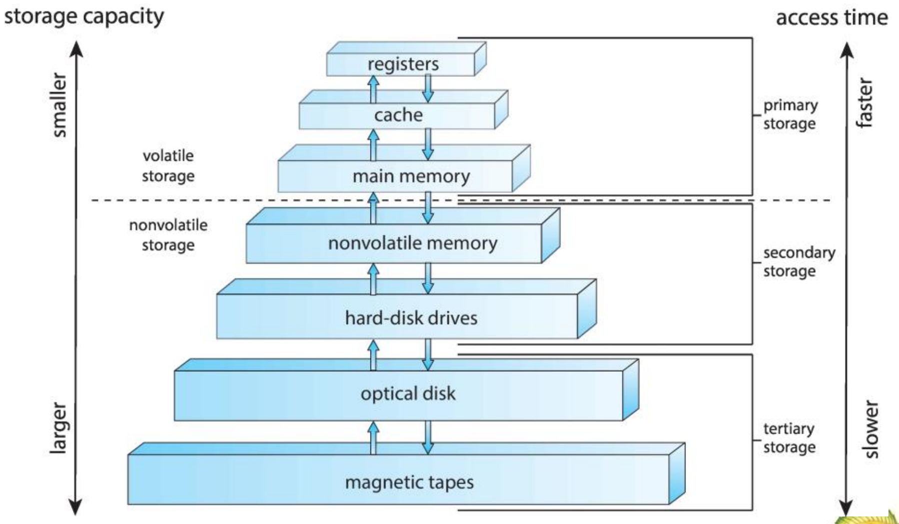
The basic unit of computer storage is the **bit** . A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, 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 $1{,}024^2$ bytes; a **gigabyte** , or GB , is $1{,}024^3$ bytes; a **terabyte** , or **TB** , is $1{,}024^4$ bytes; and a **petabyte** , or **PB** , is $1{,}024^5$ bytes. 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).

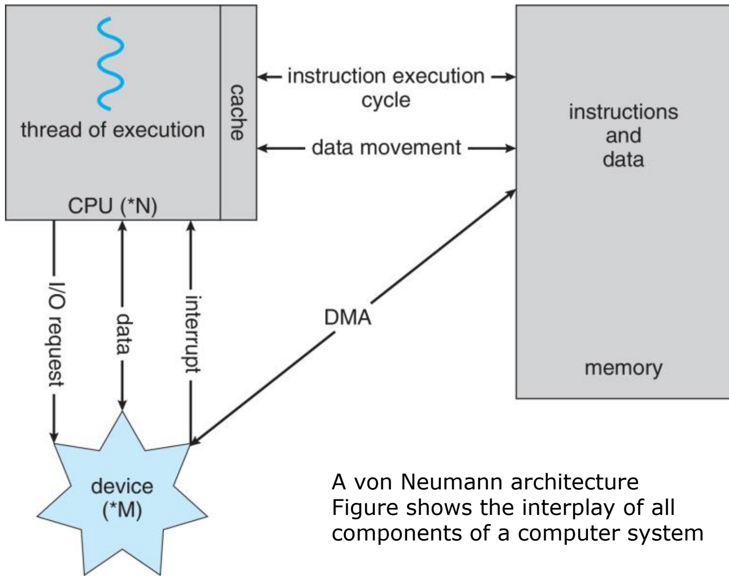
Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-Device Hierarchy



How a Modern Computer Works



A von Neumann architecture
Figure shows the interplay of all components of a computer system

Direct Memory Access Structure

- DMA is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory
  - 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 - the CPU is available to accomplish other work.
- Only one interrupt is generated per block, rather than the one interrupt per byte

Operating-System Operations

- Computer is powered up => it needs to have an initial program to run
- The first program to run: a **bootstrap program**,
  - typically stored within the computer hardware in **firmware:** EPROM
    - contains mostly static programs and data that aren't frequently used
    - is low speed,

  Ex, iPhone uses EEPROM to store serial numbers and hardware information about the device.

  - Initializes all aspects of system, from CPU registers to device controllers to memory contents
  - Loads the OS kernel and load it into memory

Operating-System Operations (cont)

- Kernel loads and executing,
  - it can start providing services to the system and its users
- Starts **system daemons** (services provided outside of the kernel)
- => the system is fully booted, and the system waits for some event to occur
  - Events are almost always signaled by the occurrence of an **interrupt**.
- Kernel **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap**):
    - Software error (e.g., division by zero)
    - Request for operating system service – **system call**
    - Other process problems include infinite loop, processes modifying each other or the operating system

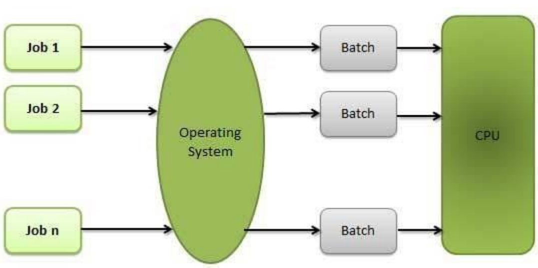
Multiprogramming (Batch system)

- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute (keep either the CPU or the I/O devices busy at all times)
  - the most important aspects of operating systems
- In a multiprogrammed system
  - a program in execution is termed a process
  - several processes in memory simultaneously
  - OS picks and begins to execute one of these processes.
  - Eventually, the process may have to wait for some task, such as an I/O operation, to complete
- Memory layout for a multiprogramming system:

Operating system
Process 1
Process 2
Process 3
Process 4

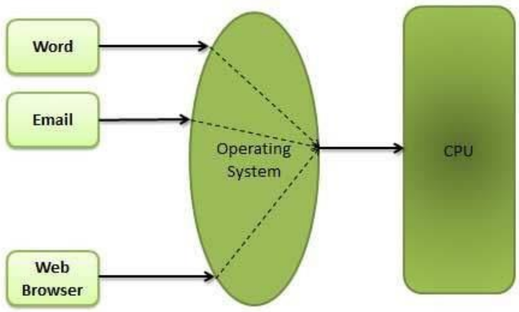
Batch Processing

- Batch processing is a technique in which an OS collects the programs and data together in a batch before processing starts.
- Advantages
  - takes much of the work of the operator to the computer.
  - Increased performance as a new job get started
- Disadvantages
  - Difficult to debug program. A job could enter an infinite loop.
  - Due to lack of protection scheme, one batch job can affect pending jobs



Multitasking (Timesharing)

- A logical extension of multiprogramming – the 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 1 program executing in memory, => called **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 run programs that are larger than actual physical memory



Dual-mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
- **Mode bit** provided by hardware to distinguish when system is running user code or kernel code.
  - When a user is running ➔ mode bit is "user"
  - When kernel code is executing ➔ mode bit is "kernel"

mot so huong dan duoc chi dinh nhu la mot dac quyen

- Some instructions designated as **privileged** – protect OS from errant users

mot so user khong duoc phep

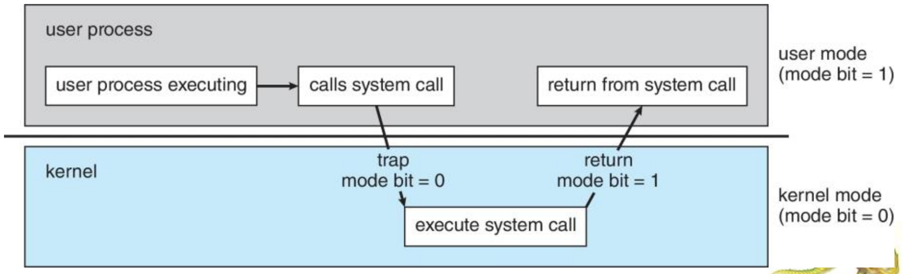
  - only executable in kernel mode
  - If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction

Dual-mode Operation (Cont.)

bao dam

- How do we guarantee that user does not explicitly set the mode bit to "kernel"?
  - When the system starts executing it is in "kernel mode"
  - When control is given to a user program: changes to "user mode".
  - When a user issues a system call it results in an interrupt, which trap to the operating system => the mode–bit is set to "kernel mode".

  khi nguoi dung thuc thi system call no se gay ra interrupt , dieu nay anh huong den he dieu hanh
- Transition from User to Kernel Mode:



Timer

vong lap vo tan process ton qua nhieu tai nguyen

- Timer to prevent infinite loop (or process hogging resources)
  - Timer is set to interrupt the computer after some time period cai dat interrupt sau mot khoang thoi gian
  - Keep a counter that is decremented by the physical clock dong ho dem nguoc
  - Operating system set the counter (privileged instruction) nguoi dung khong the set up
  - When counter zero generate an interrupt khi dong ho dem nguoc so 0 thuc thi interrrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

    set up truoc tien trinh de lay lai su kiem soat hoac tat chuong trinh khi chuong trinh vuot qua thoi gian quy dinh

Resource Management

- An operating system is a resource manager.
- The system's CPU, memory space, file-storage space, and I/O devices are among the resources that the operating system must manage.
- Resource Management
  - Process Management
  - Memory Management
  - File-System Management
  - Mass-Storage Management
  - Cache Management
  - I/O System Management

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity;*** process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- **Single-threaded** process has one **program counter** specifying location of next instruction to execute
  - The execution of a process must be sequential.
  - The CPU executes one instruction of the process after another, until the process completes
- **Multi-threaded** process has multiple program counters, each per thread
- Typically, system has many processes, some user, some OS 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

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

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)
- 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 directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually, disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

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

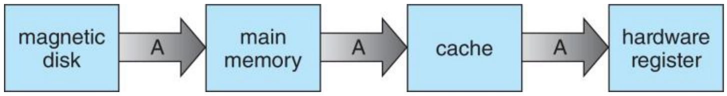
Characteristics of Various Types of Storage

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

Migration of data “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 19

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – 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

Protection

- 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

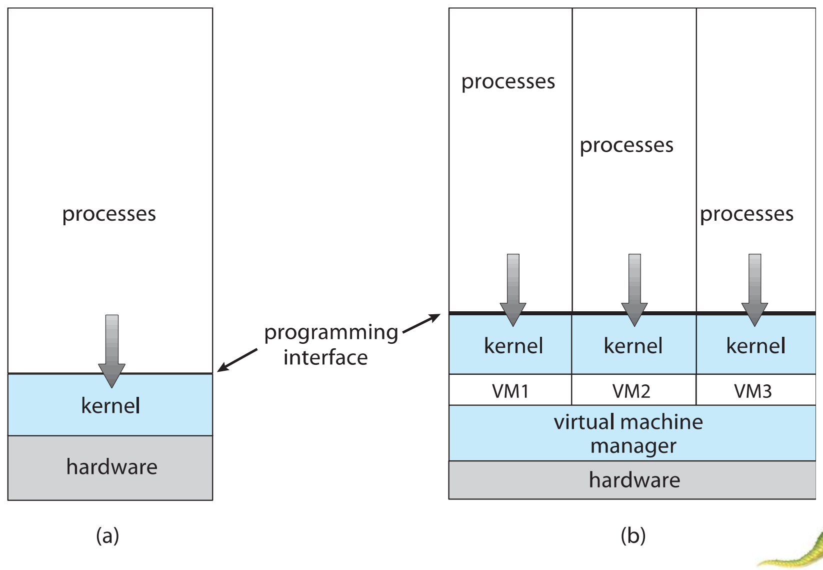
Virtualization

- 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** (virtual machine Manager) provides virtualization services

Virtualization (cont.)

- 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
  - Quality assurance 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)

Virtualization Illustration



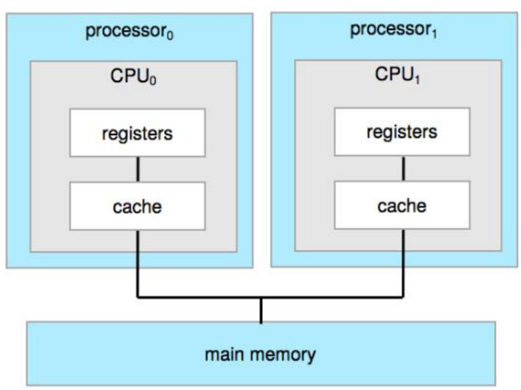
Distributed Systems

- 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

Computer-System Architecture

- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessor's** 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** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



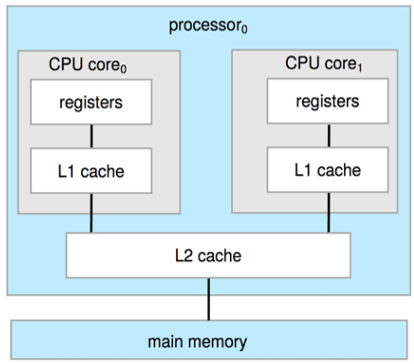
- All processors share physical memory over the system bus
- Benefit: many processes can run simultaneously
- Limit: one may be sitting idle while another is overloaded, resulting in inefficiencies

Dual-Core Design

- The **core** is the component that executes instructions and registers for storing data locally
- **Multicore** systems can be more eficient than multiple chips with single cores
  - because on-chip communication is faster than between-chip communication
  - one chip with multiple cores uses significantly less power than multiple single-core chips,
- Systems containing all chips
- Chassis containing multiple separate

Systems

- Ex, Dual-Core Design:
  - L1-cache (local cache): lower-level:
  - L2 cache (shared cache) higher-level



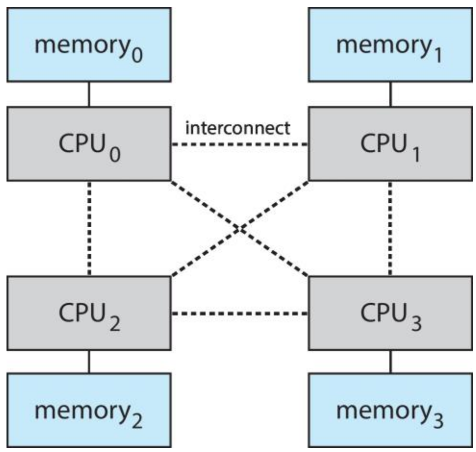
Note

DEFINITIONS OF COMPUTER SYSTEM COMPONENTS

- **CPU**—The hardware that executes 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 the same CPU.
- **Multiprocessor**—Including multiple processors.

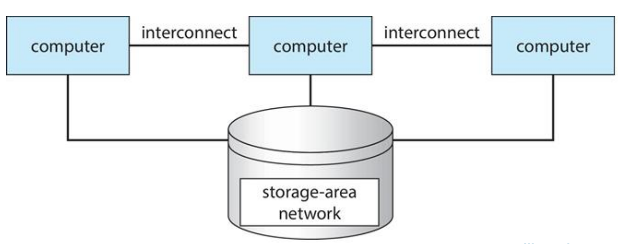
Although virtually all systems are now multicore, we use the general term *CPU* when referring to a single computational unit of a computer system and *core* as well as *multicore* when specifically referring to one or more cores on a CPU.

Non-Uniform Memory Access System



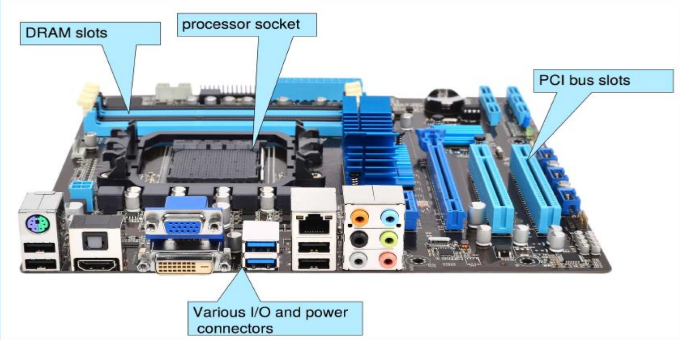
Clustered Systems

- 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



PC Motherboard

Consider the desktop PC motherboard with a processor socket shown below:



This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

Computing Environments

- Traditional
- Mobile
- Client Server
- Pear-to-Pear
- Cloud computing
- Real-time Embedded

Traditional

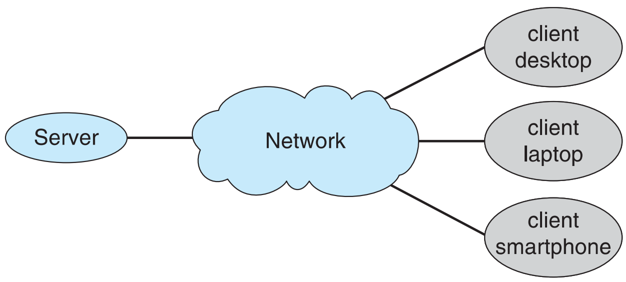
- Stand-alone general-purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Mobile Computing

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

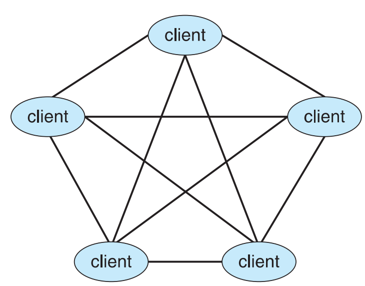
Client Server Computing

- 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



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



Cloud Computing

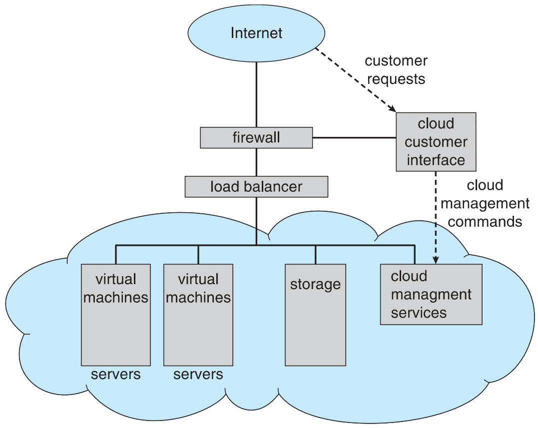
- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for its functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage

Cloud Computing – 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)

Cloud Computing (cont.)

- Cloud computing environments composed of traditional Oses plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited 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

Free and Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source** and **proprietary**
- Counter to the **copy protection** and **Digital Rights Management** (**DRM**) movement
- Started by **Free Software Foundation** (**FSF**), which has "copyleft" **GNU Public License** (**GPL**)
  - Free software and open-source software are two different ideas championed by different groups of people
    - http://gnu.org/philosophy/open-source-misses-the-point.html/
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

The Study of Operating Systems

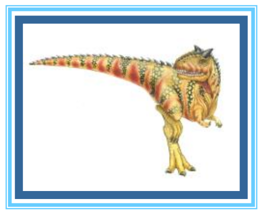
There has never been a more interesting time to study operating systems, and it has never been easier. The open-source movement has overtaken operating systems, causing many of them to be made available in both source and binary (executable) format. The list of operating systems available in both formats includes Linux, BUSD UNIX, Solaris, and part of macOS. The availability of source code allows us to study operating systems from the inside out. Questions that we could once answer only by looking at documentation or the behavior of an operating system we can now answer by examining the code itself.

Operating systems that are no longer commercially viable have been open-sourced as well, enabling us to study how systems operated in a time of fewer CPU, memory, and storage resources. An extensive but incomplete list of open-source operating-system projects is available from https://curlie.org/Computers/Software/Operating_Systems/Open_Source/

In addition, the rise of virtualization as a mainstream (and frequently free) computer function makes it possible to run many operating systems on top of one core system. For example, VMware (http://www.vmware.com) providesa free "player" for Windows on which hundreds of free "virtual appliances" can run. Virtualbox (http://www.virtualbox.com) provides a free, open-source virtual machine manager on many operating systems. Using such tools, students can try out hundreds of operating systems without dedicated hardware.

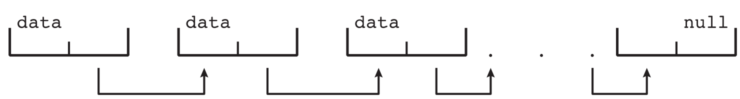
The advent of open-source operating systems has also made it easier to make the move from student to operating-system developer. With some knowledge, some effort, and an Internet connection, a student can even create a new operating-system distribution. Just a few years ago, it was difficult or impossible to get access to source code. Now, such access is limited only by how much interest, time, and disk space a student has.

End of Chapter 1.1

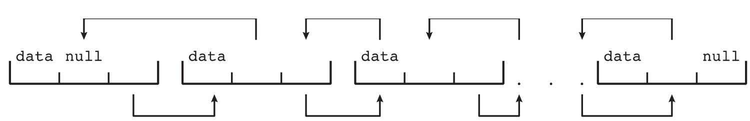


Kernel Data Structures

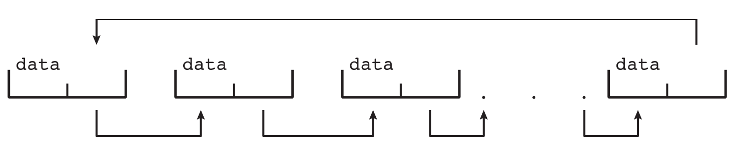
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

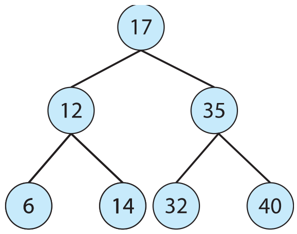


- ***Circular linked list***



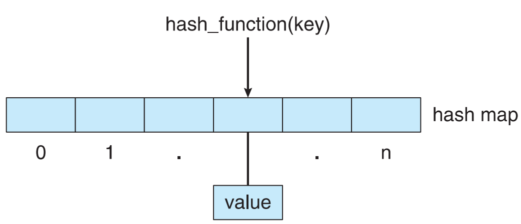
Kernel Data Structures

- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*



Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Characteristics of Various Types 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