

Chapter 1: Introduction





Chapter 1: Introduction

What Operating Systems Do
Computer-System Organization
Computer-System Architecture
Operating-System Structure
Operating-System Operations
Process Management
Memory Management
Storage Management
Protection and Security
Kernel Data Structures
Computing Environments
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





What is an Operating System?

A program that acts as an intermediary between a user of a computer and the computer hardware

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

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

- ▶ Word processors, compilers, web browsers, database systems, video games

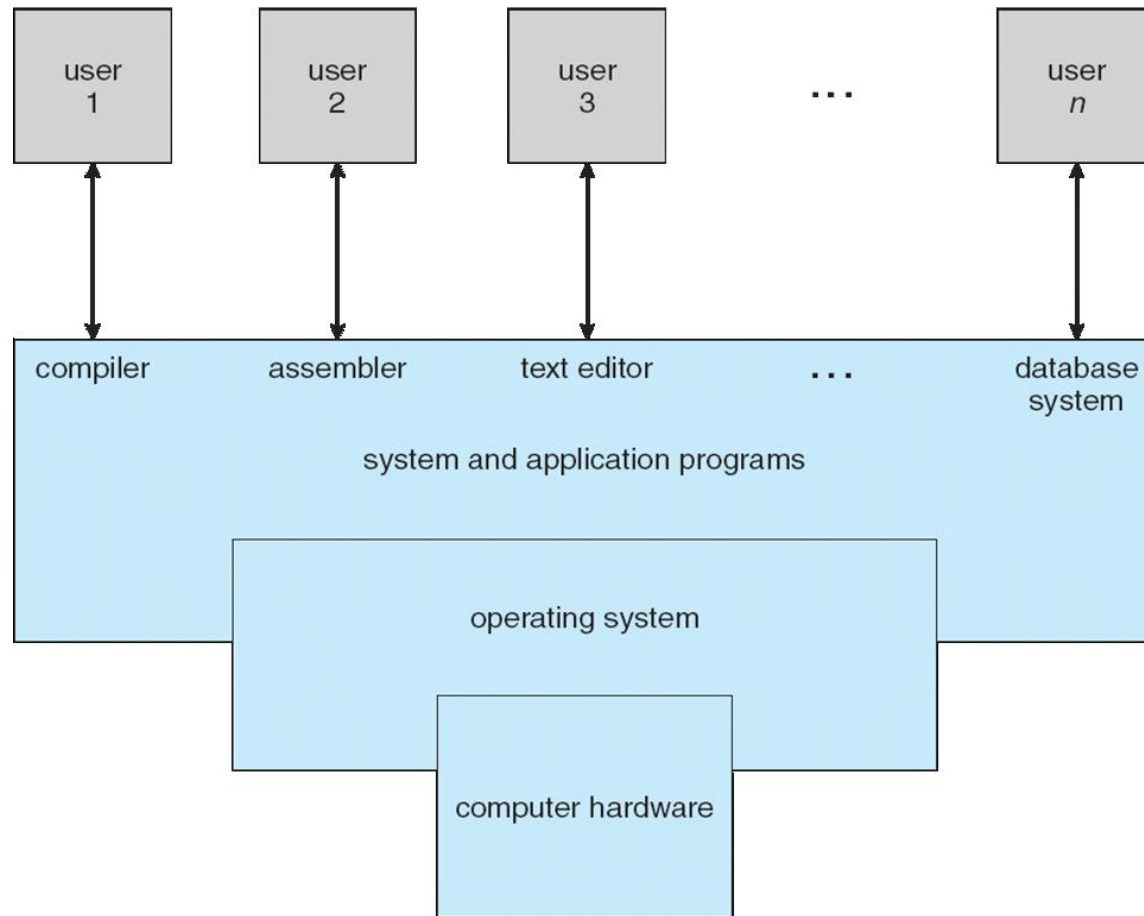
Users

- ▶ People, machines, other computers





Four Components of a Computer System





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 dedicated 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, such as embedded computers in devices and automobiles





Operating System Definition

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





Operating System Definition (Cont.)

No universally accepted definition

“Everything a vendor ships when you order an operating system” is good approximation

But varies wildly

“The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.





Computer Startup

bootstrap program is loaded at power-up or reboot

Typically stored in ROM or EPROM, generally known as **firmware**

Initializes all aspects of system

Loads operating system kernel and starts execution



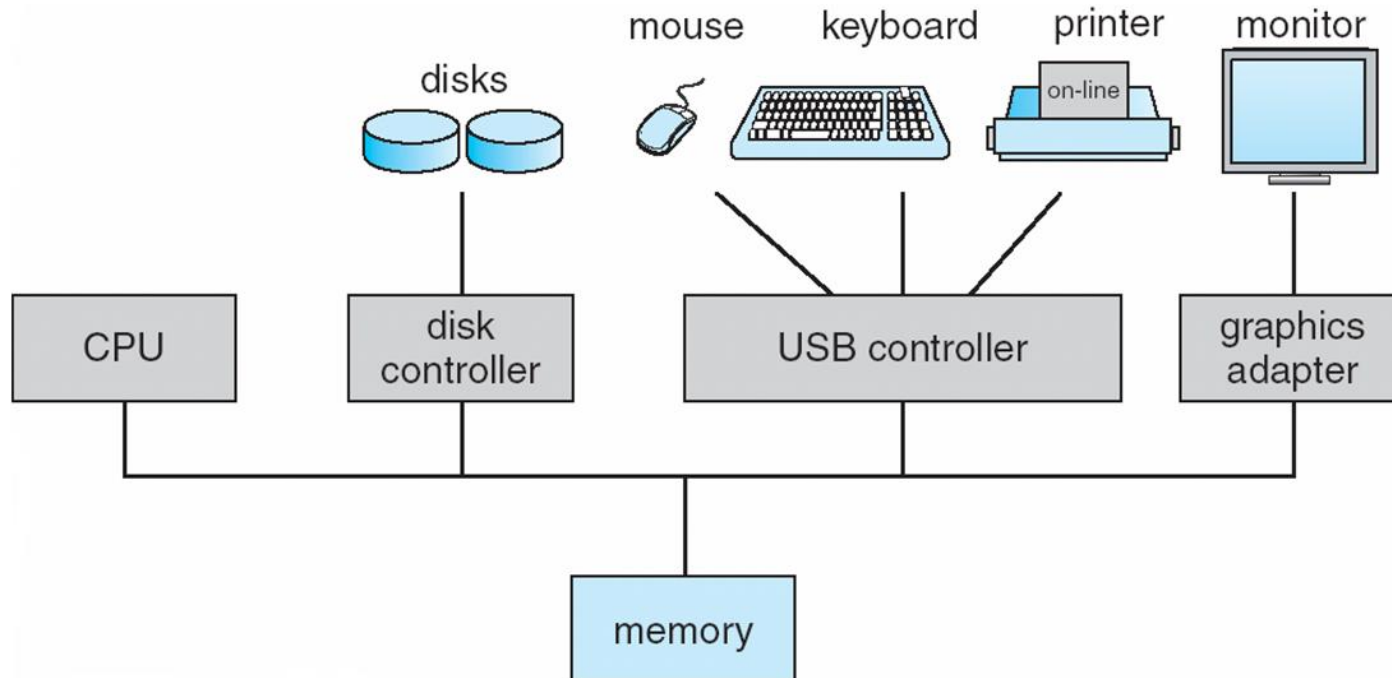


Computer System Organization

Computer-system operation

One or more CPUs, device controllers connect through common bus providing access to shared memory

Concurrent execution of CPUs and devices competing for memory cycles





Computer-System Operation

I/O devices and the CPU can execute concurrently

Each device controller is in charge of a particular device 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 **interrupt driven**





Interrupt Handling

The operating system preserves the state of the CPU by storing registers and the program counter

Determines which type of interrupt has occurred:

polling

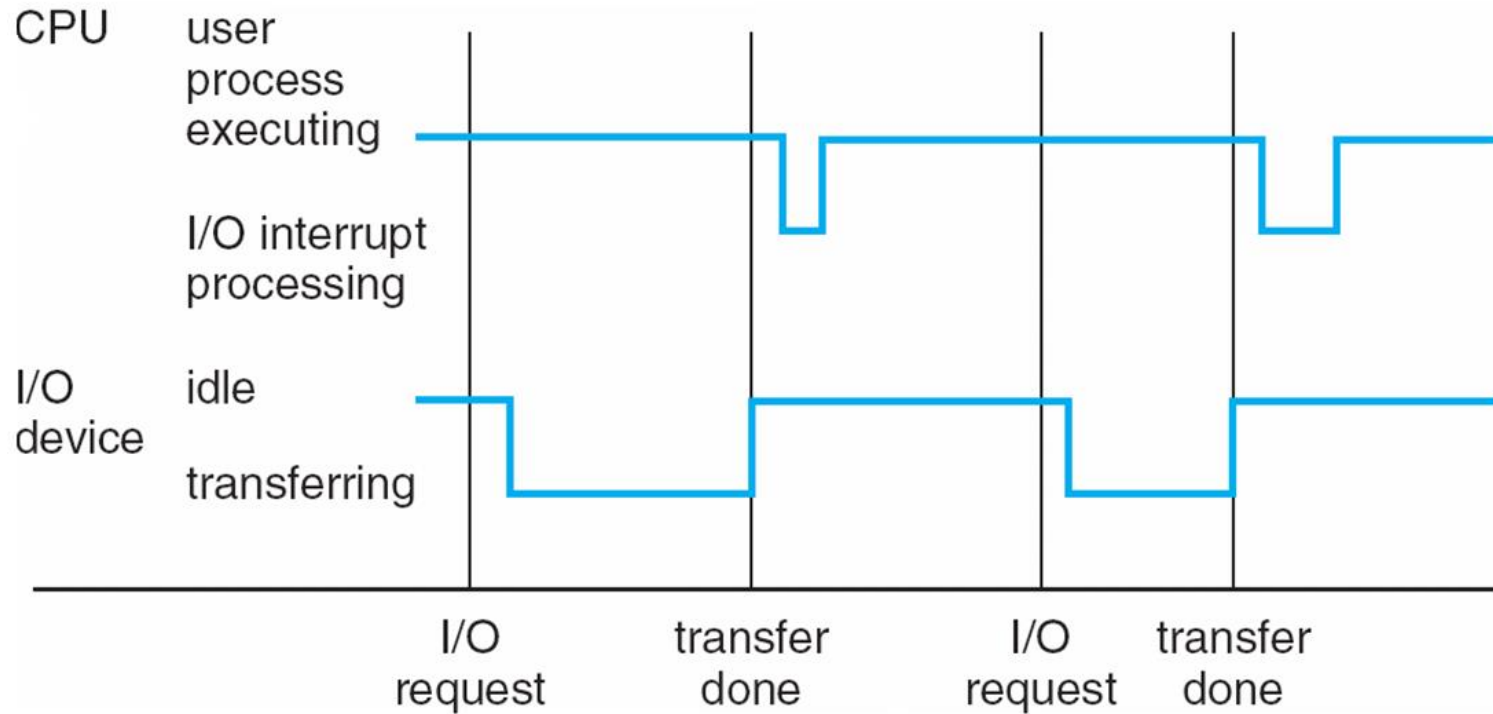
vectored interrupt system

Separate segments of code determine what action should be taken for each type of interrupt





Interrupt Timeline





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 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 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 1,024² bytes; a **gigabyte**, or **GB**, is 1,024³ bytes; a **terabyte**, or **TB**, is 1,024⁴ bytes; and a **petabyte**, or **PB**, is 1,024⁵ 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).





Direct Memory Access Structure

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





Storage Structure

Main memory – only large storage media that the CPU can access directly

Random access

Typically **volatile**

Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity

Magnetic disks – 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

Solid-state disks – faster than magnetic disks, nonvolatile

Various technologies

Becoming more popular





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

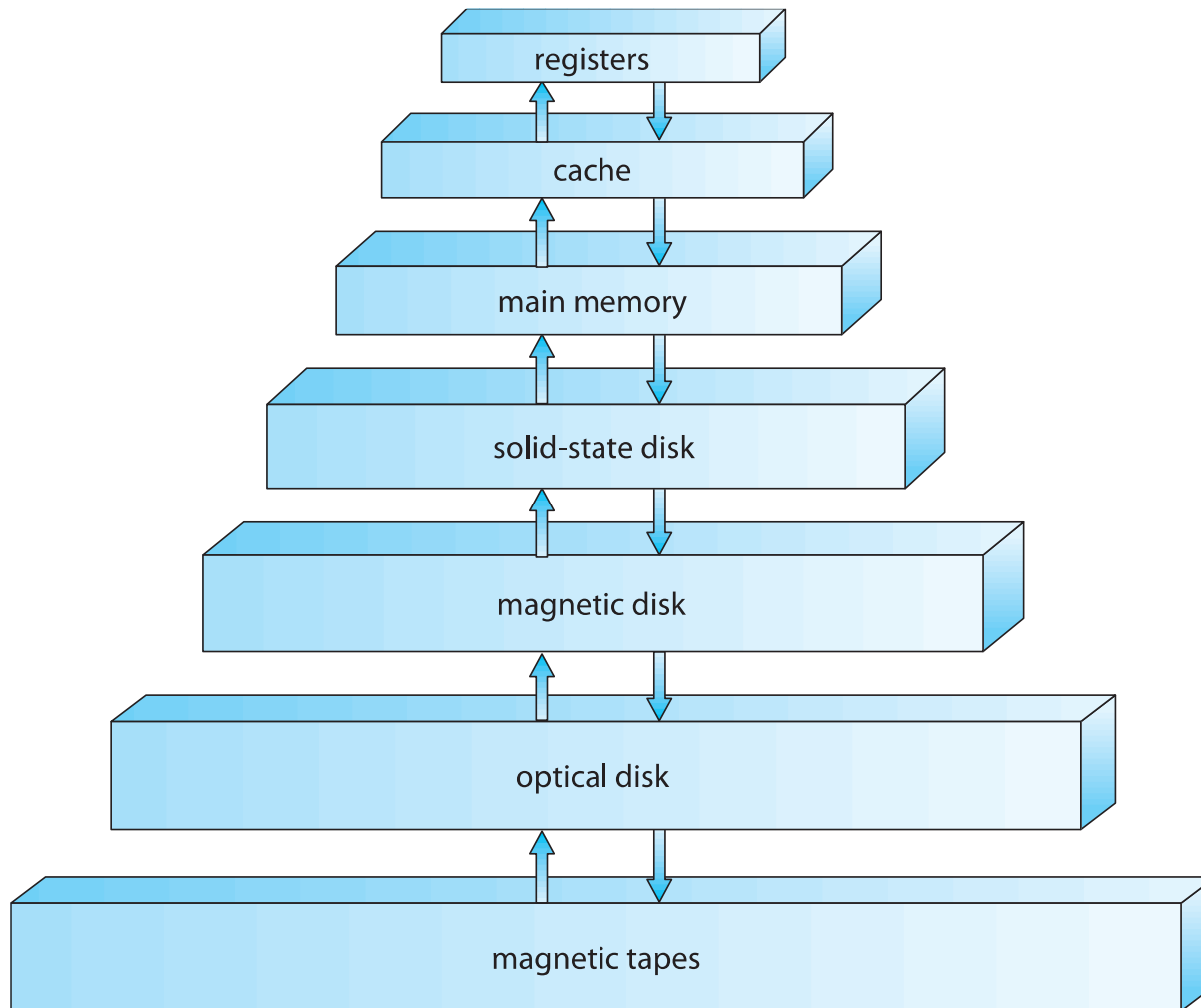
Device Driver for each device controller to manage I/O

Provides uniform interface between controller and kernel





Storage-Device Hierarchy





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





Computer-System Architecture

Most systems use a single general-purpose processor (PDAs through mainframes)

Most systems have special-purpose processors as well

Multiprocessors systems growing in use and importance

Also known as **parallel systems**, **tightly-coupled systems**

Advantages include:

1. **Increased throughput**
2. **Economy of scale**
3. **Increased reliability – graceful degradation or fault tolerance**

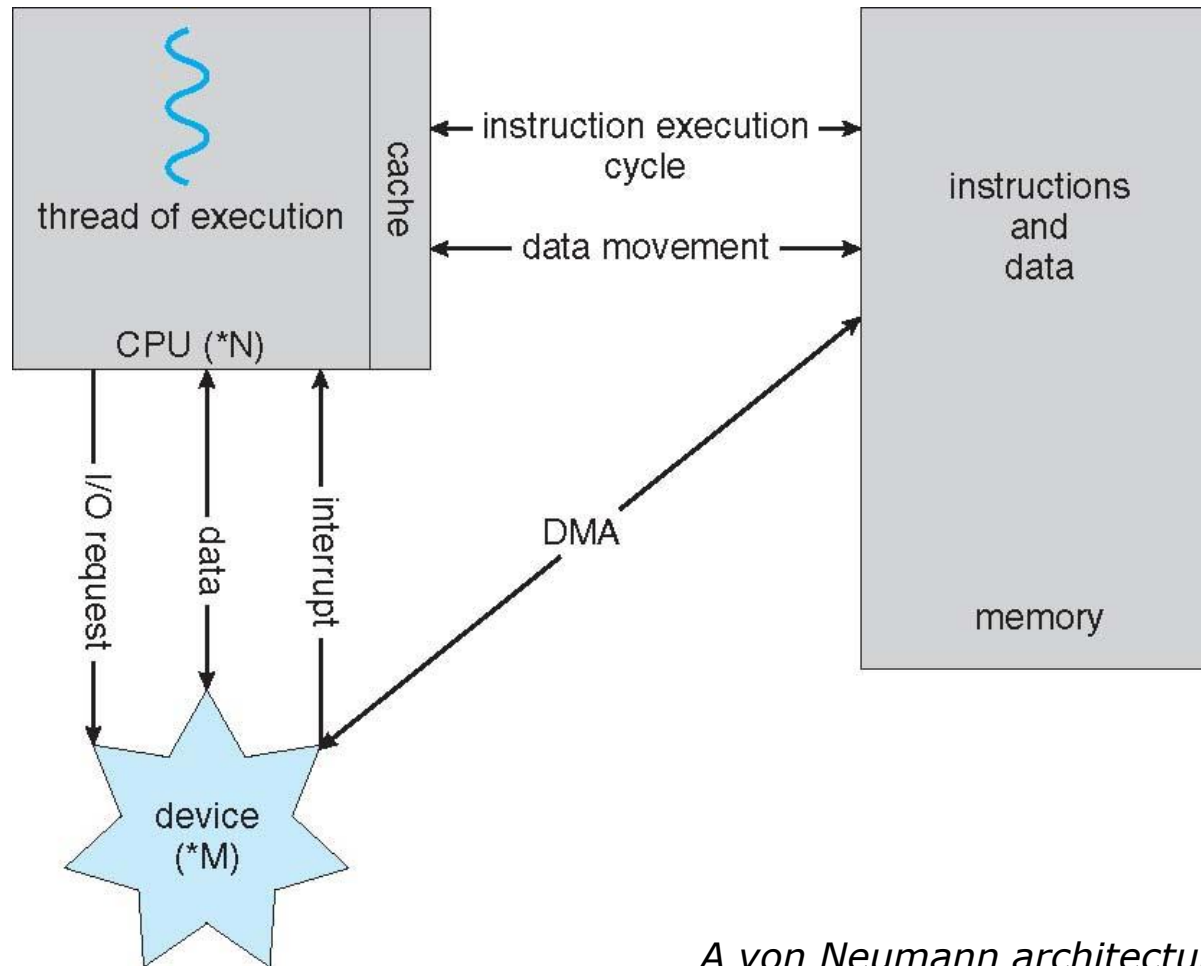
Two types:

1. **Asymmetric Multiprocessing**
2. **Symmetric Multiprocessing**





How a Modern Computer Works

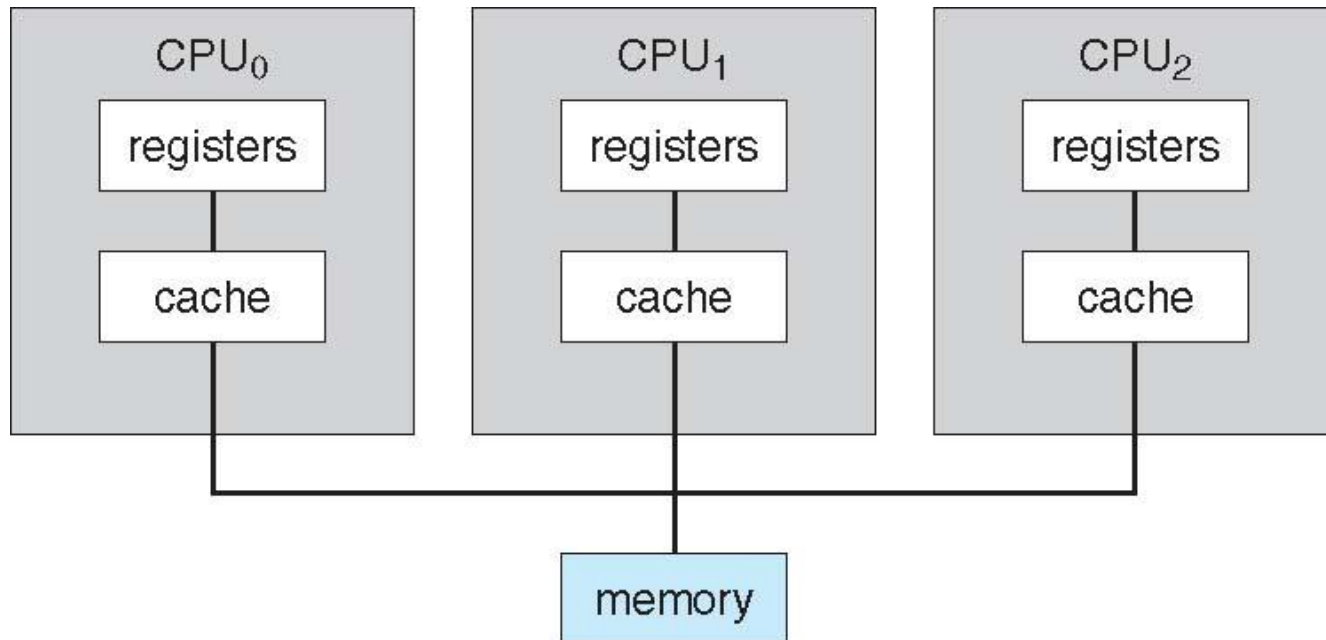


A von Neumann architecture





Symmetric Multiprocessing Architecture





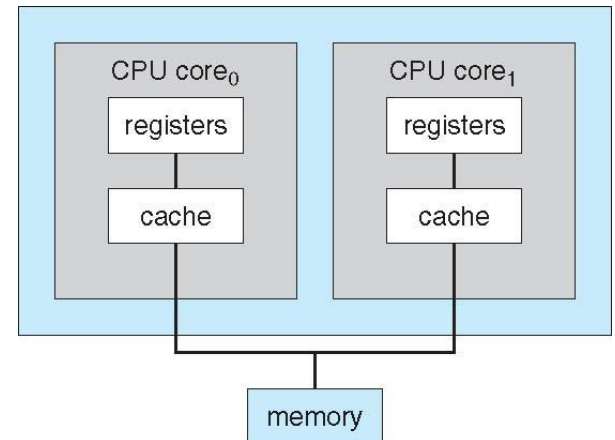
A Dual-Core Design

UMA and **NUMA**
architecture variations

Multi-chip and **multicore**

Systems containing all
chips vs. **blade servers**

Chassis containing multiple
separate systems





Clustered Systems

Like multiprocessor systems, but multiple systems working together

Usually sharing storage via a **storage-area network (SAN)**

Provides a **high-availability** service which survives failures

- ▶ **Asymmetric clustering** has one machine in hot-standby mode
- ▶ **Symmetric clustering** has multiple nodes running applications, monitoring each other

Some clusters are for **high-performance computing (HPC)**

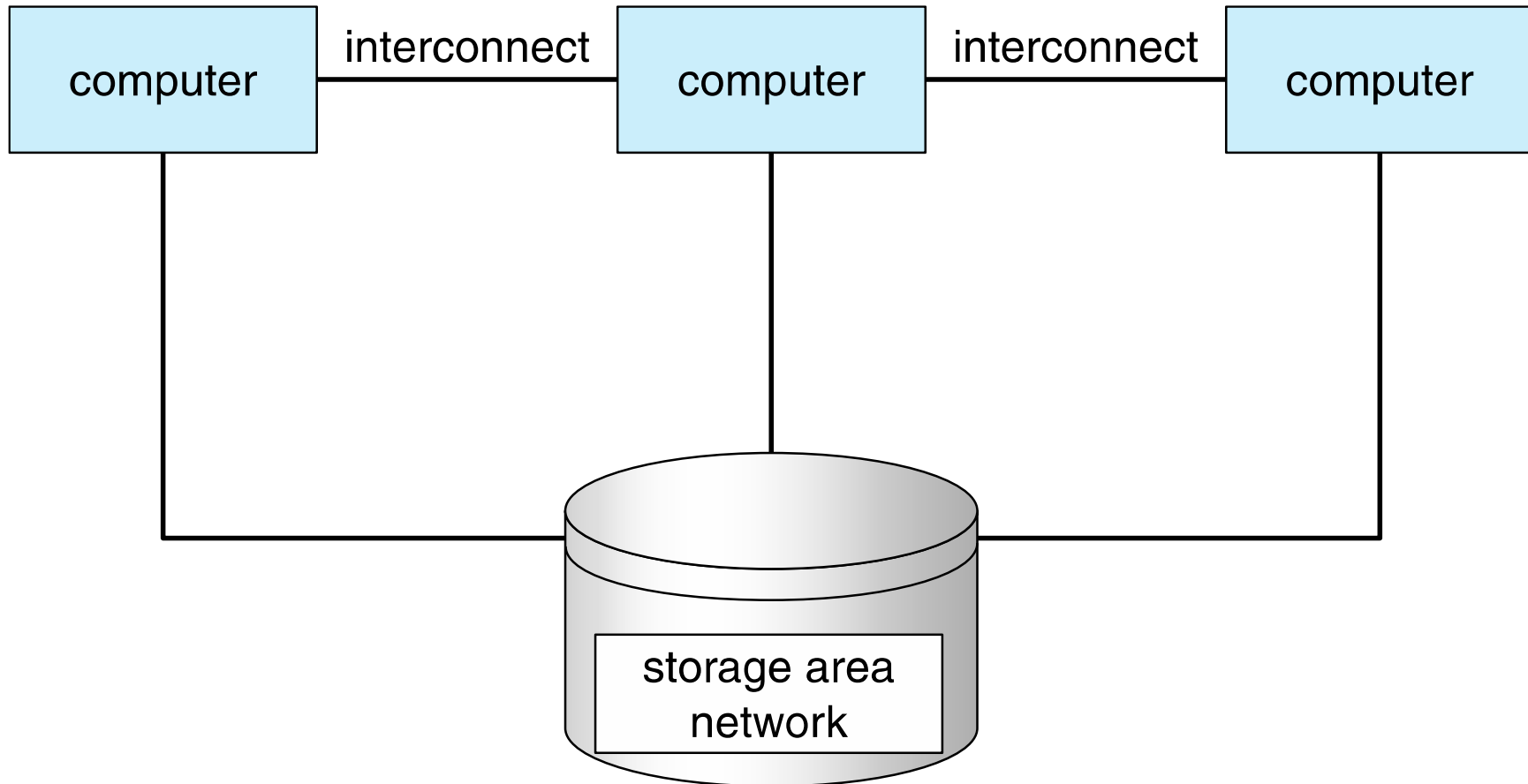
- ▶ Applications must be written to use **parallelization**

Some have **distributed lock manager (DLM)** to avoid conflicting operations





Clustered Systems





Operating System Structure

Multiprogramming needed for efficiency

Single user cannot keep CPU and I/O devices busy at all times

Multiprogramming organizes jobs (code and data) so CPU always has one to execute

A subset of total jobs in system is kept in memory

One job selected and run via **job scheduling**

When it has to wait (for I/O for example), OS switches to another job

Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

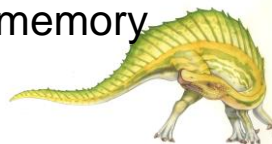
Response time should be < 1 second

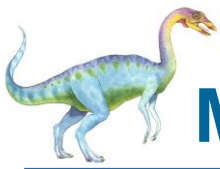
Each user has at least one program executing in memory \Rightarrow **process**

If several jobs ready to run at the same time \Rightarrow **CPU scheduling**

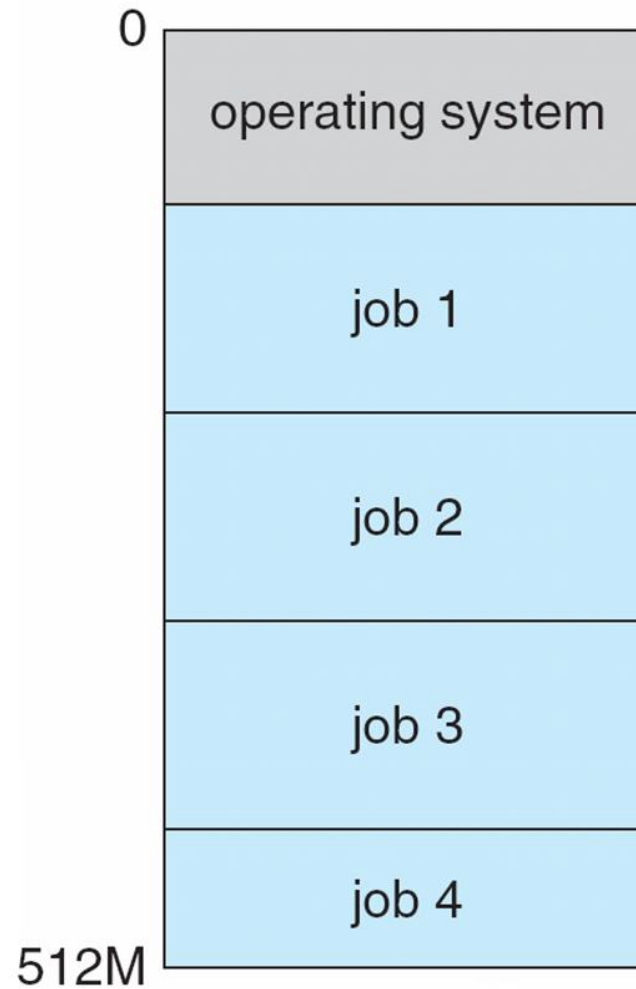
If processes don't fit in memory, **swapping** moves them in and out to run

Virtual memory allows execution of processes not completely in memory





Memory Layout for Multiprogrammed System





Operating-System Operations

Interrupt driven by hardware

Software error or request creates **exception** or **trap**

Division by zero, request for operating system service

Other process problems include infinite loop, processes modifying each other or the operating system

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**





Transition from User to Kernel Mode

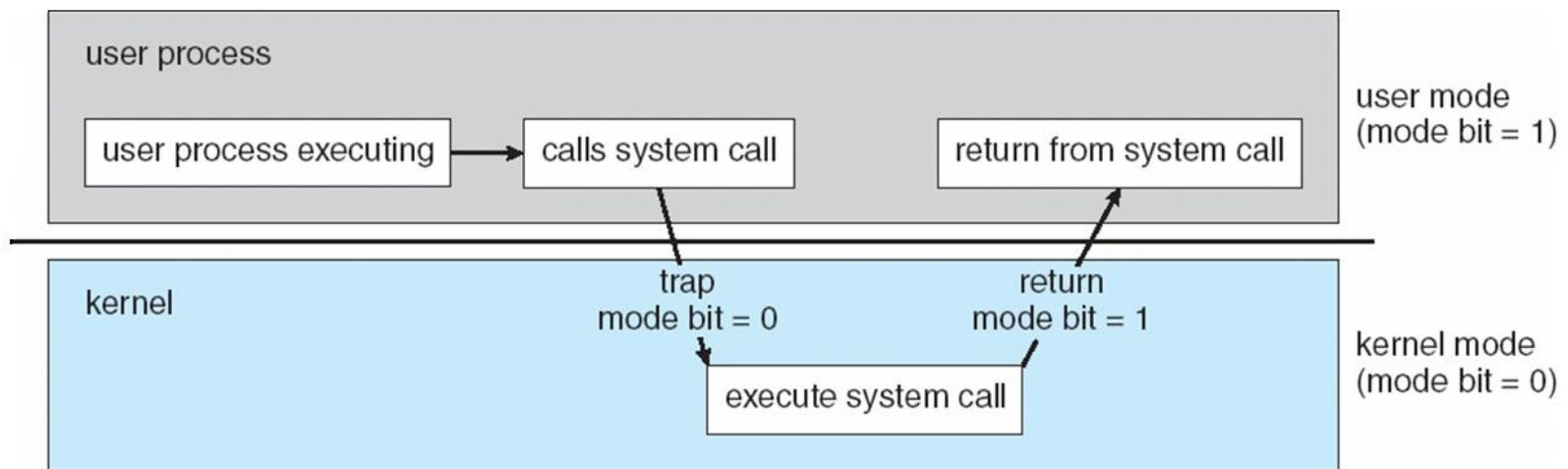
Timer to prevent infinite loop / process hogging resources

Set interrupt after specific period

Operating system decrements counter

When counter zero generate an interrupt

Set up before scheduling process to regain control or terminate program that exceeds allotted time





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





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

- Free-space management

- Storage allocation

- Disk scheduling

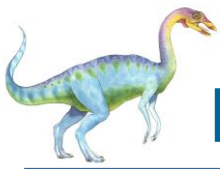
Some storage need not be fast

- Tertiary storage includes optical storage, magnetic tape

- Still must be managed – by OS or applications

- Varies between WORM (write-once, read-many-times) and RW (read-write)





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

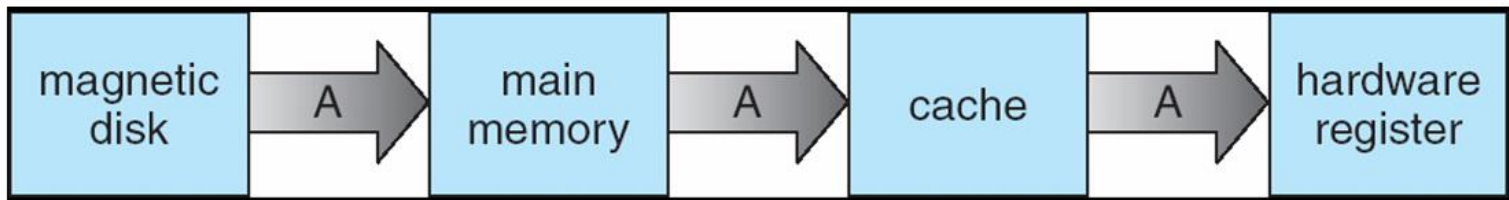
Movement between levels of storage hierarchy can be explicit or implicit





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





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 – 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

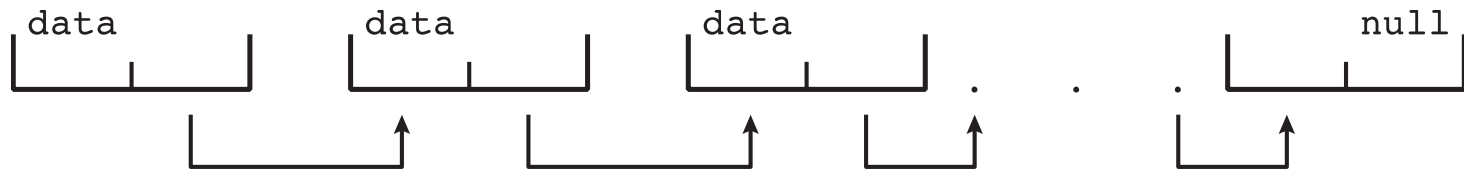




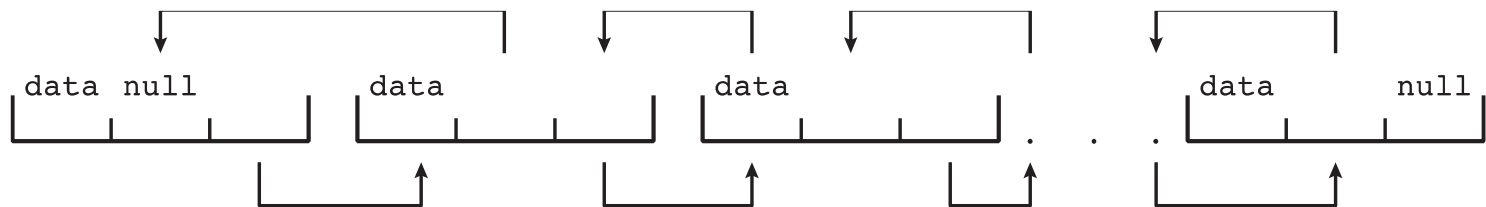
Kernel Data Structures

Many similar to standard programming data structures

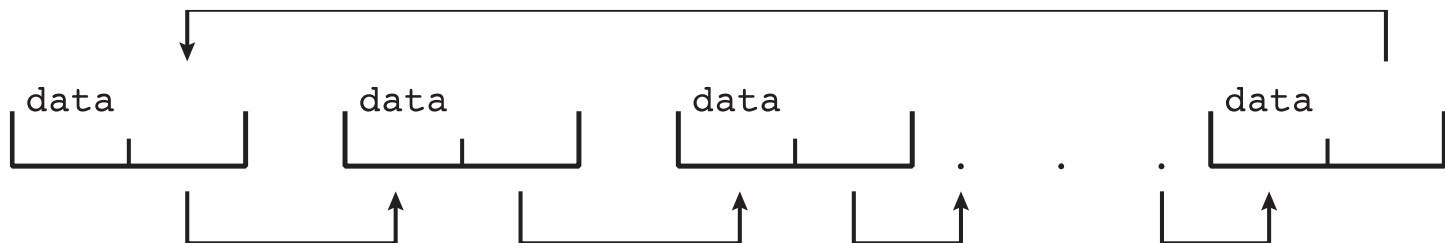
Singly linked list



Doubly linked list



Circular linked list





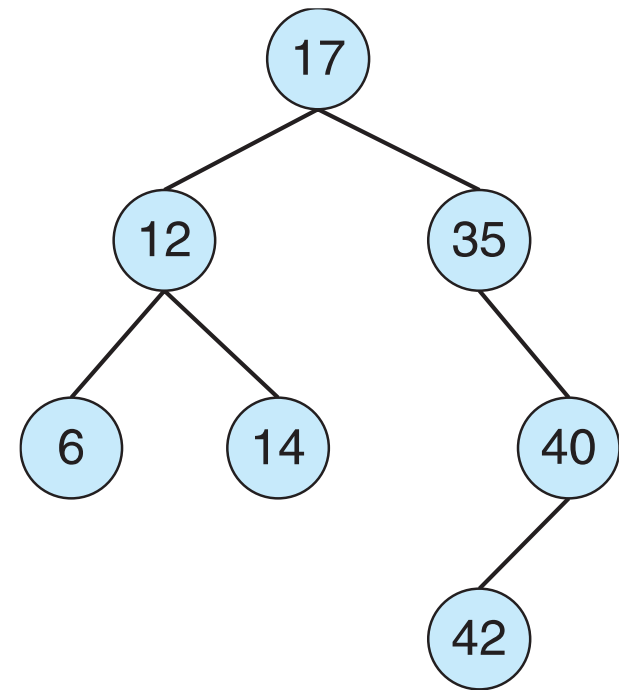
Kernel Data Structures

Binary search tree

left \leq right

Search performance is $O(n)$

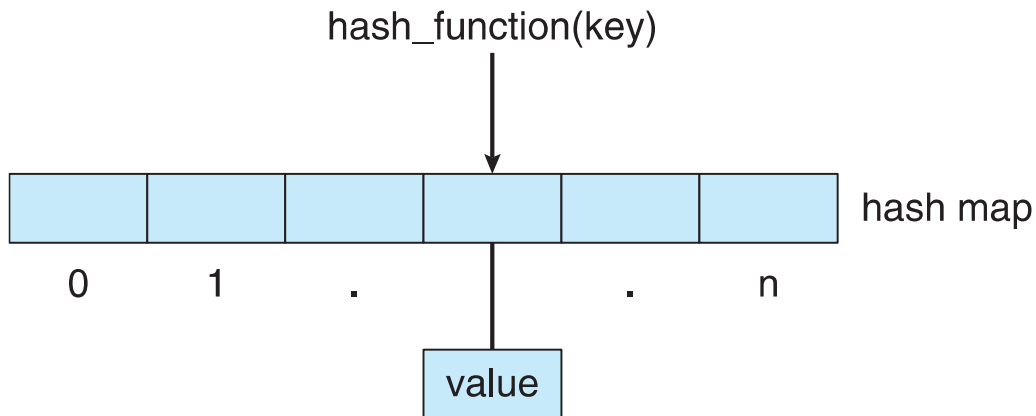
Balanced binary search tree is $O(\lg n)$





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>





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?

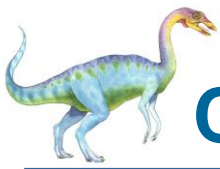
Extra feature – more OS features (GPS, gyroscope)

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





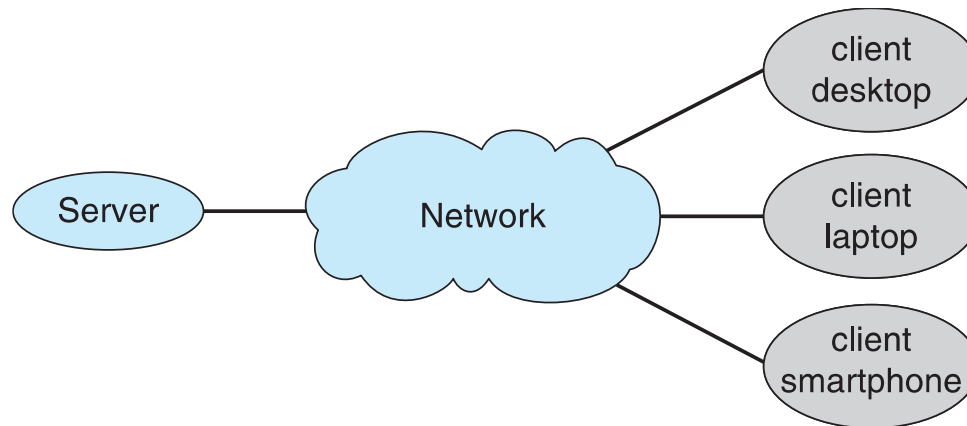
Computing Environments – Client-Server

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

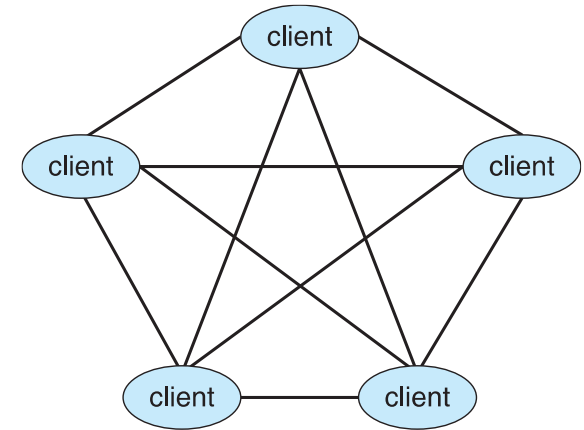
Instead all nodes are considered peers

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

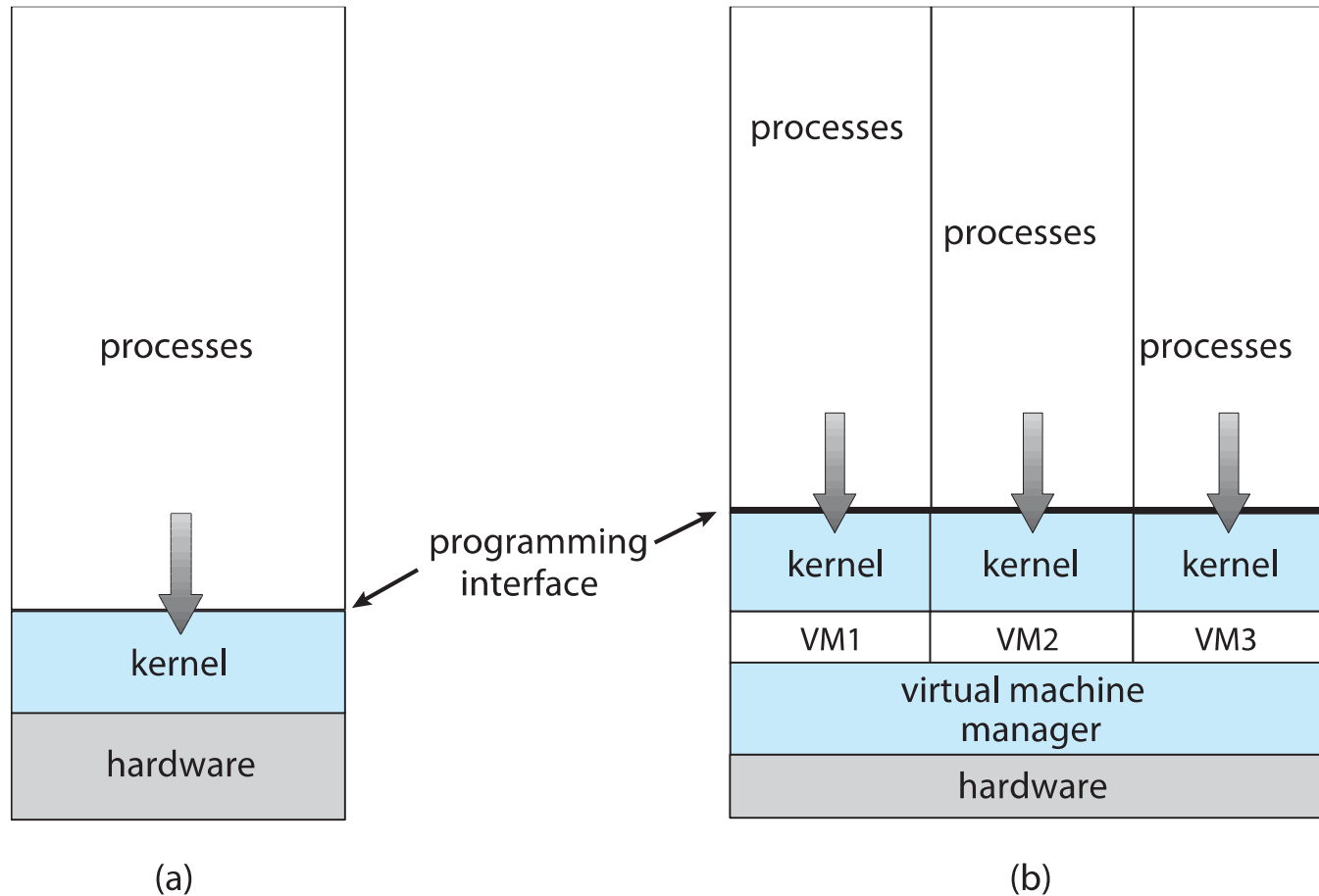
VMM can run natively, in which case they are also the host

- 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)



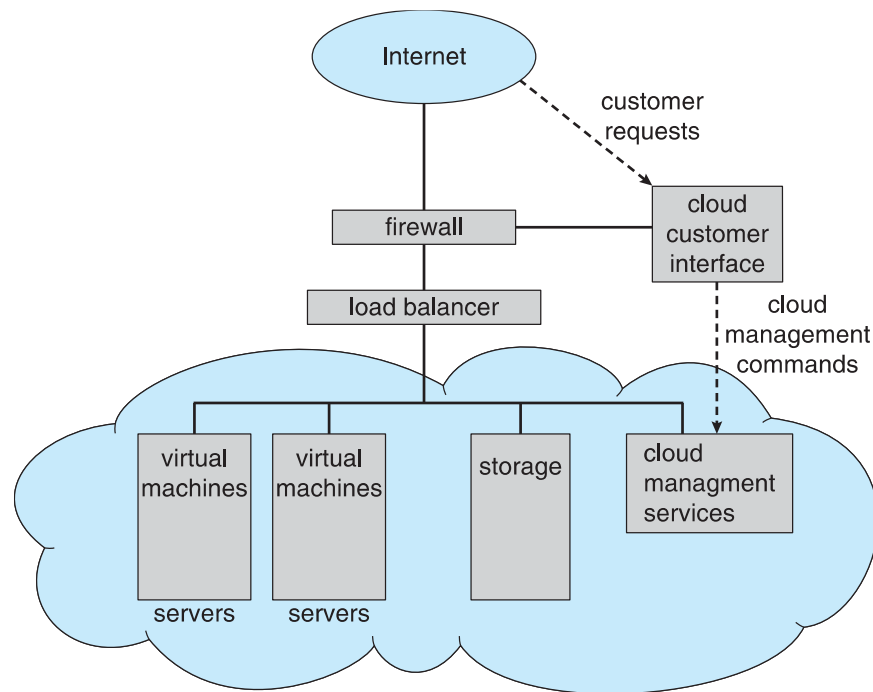


Computing Environments – Cloud Computing

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, 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





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

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

