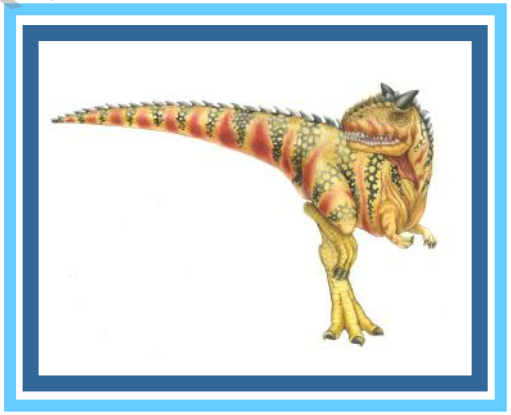
Chapter 1: Introduction



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

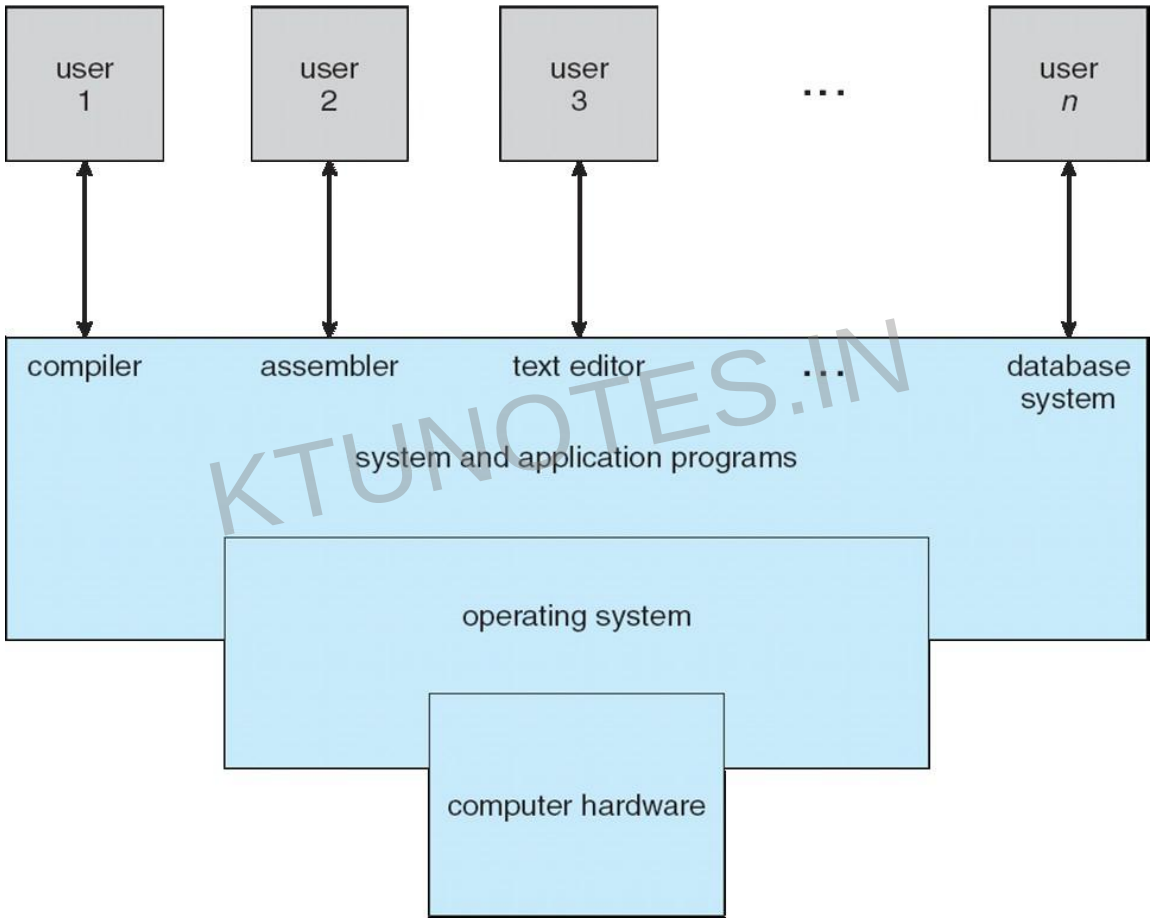
What is an Operating System?

- 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 dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, 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.)

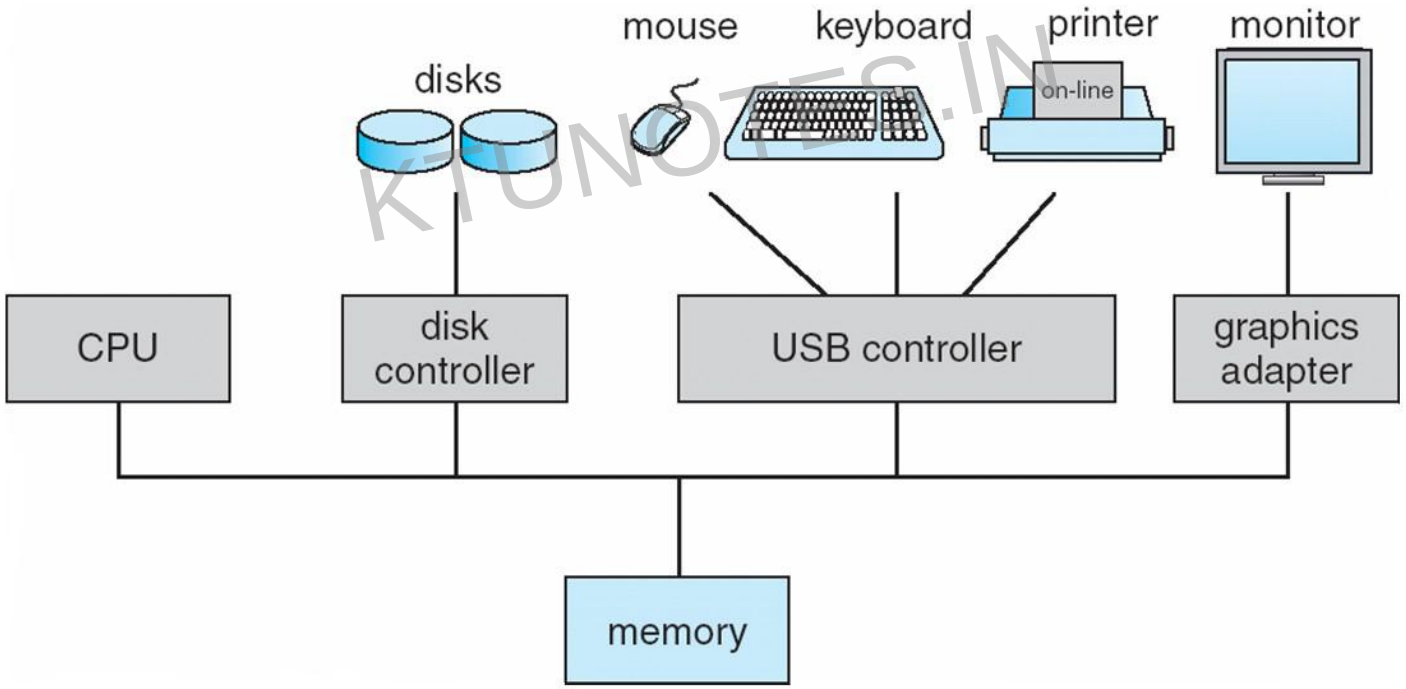
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or 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

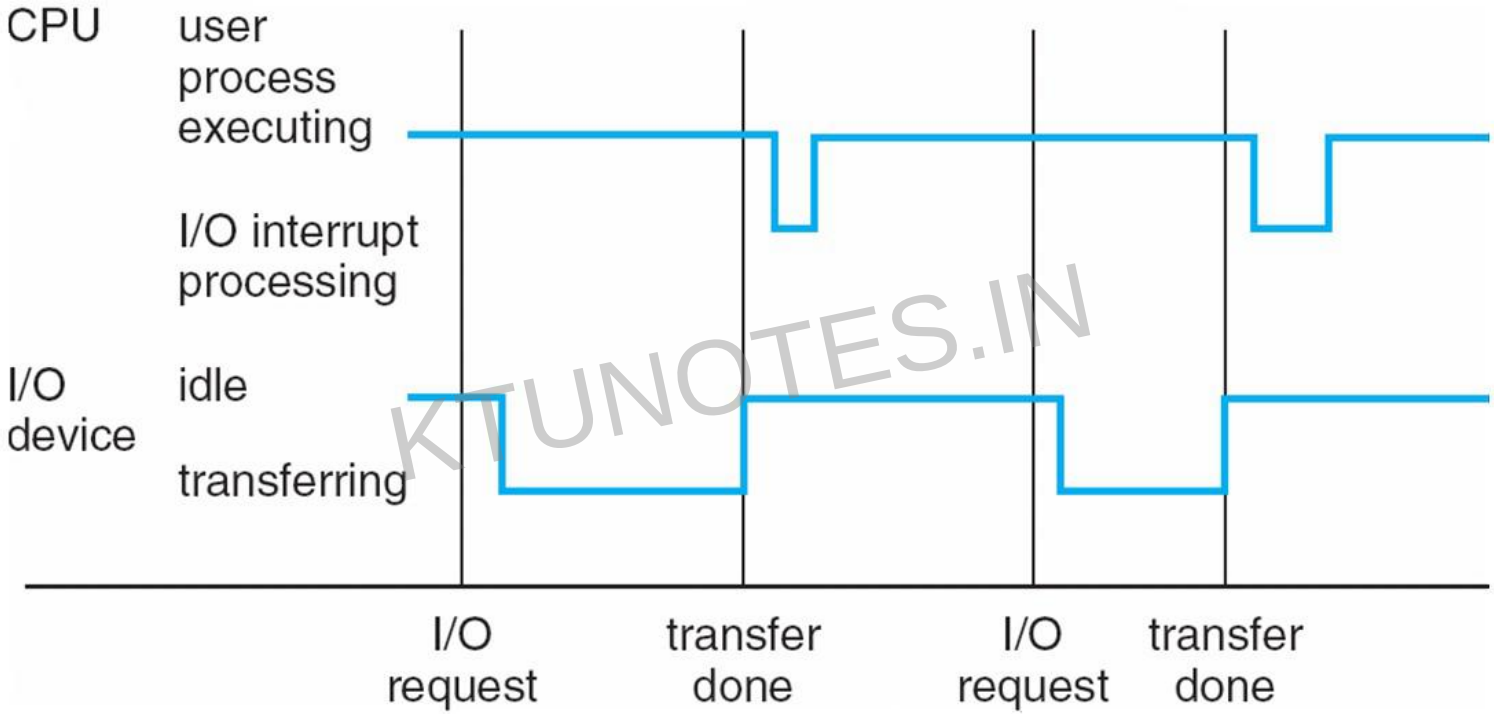
Common Functions of Interrupts

- 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 $1,024^2$ 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 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

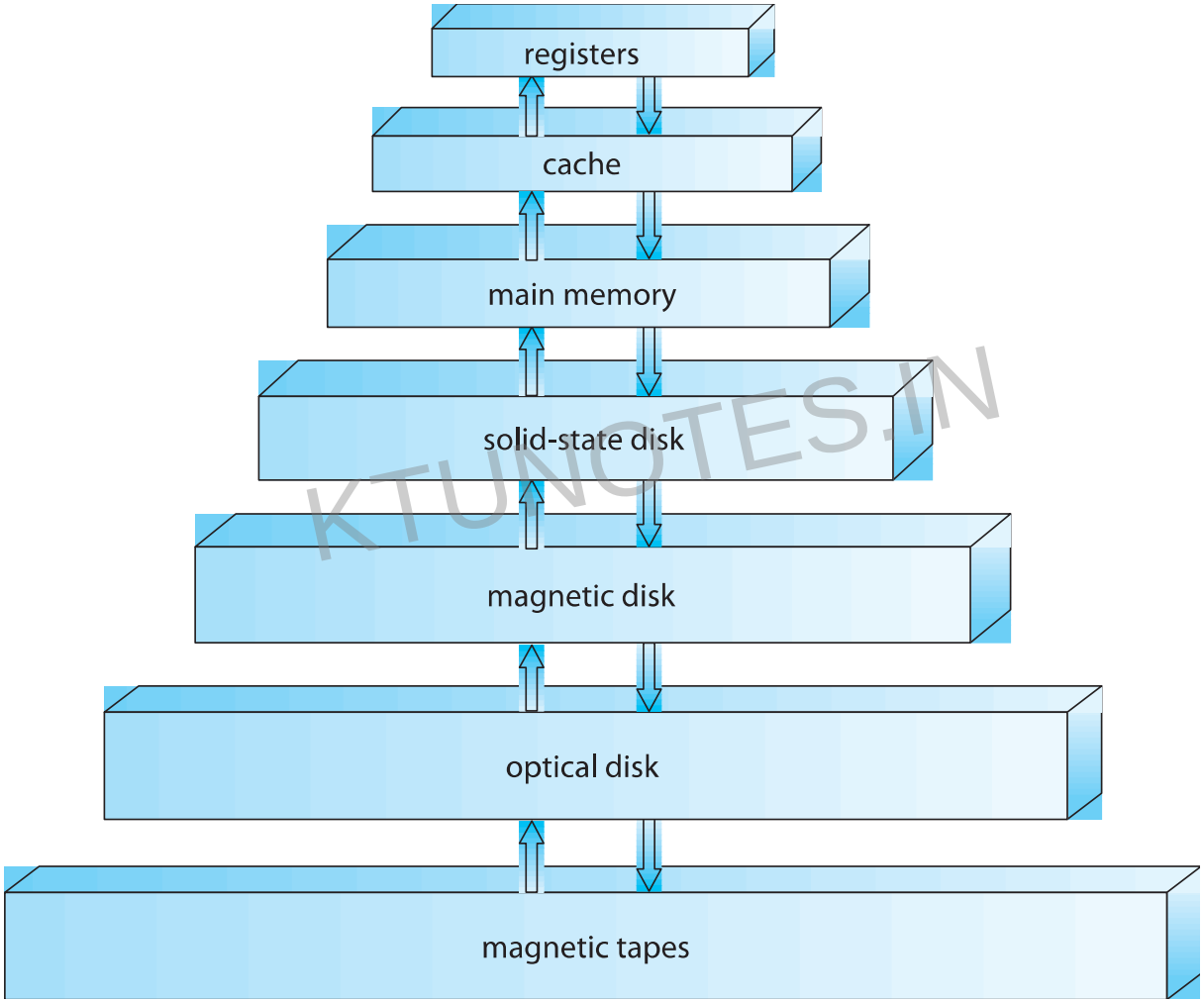
Storage Structure

- 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



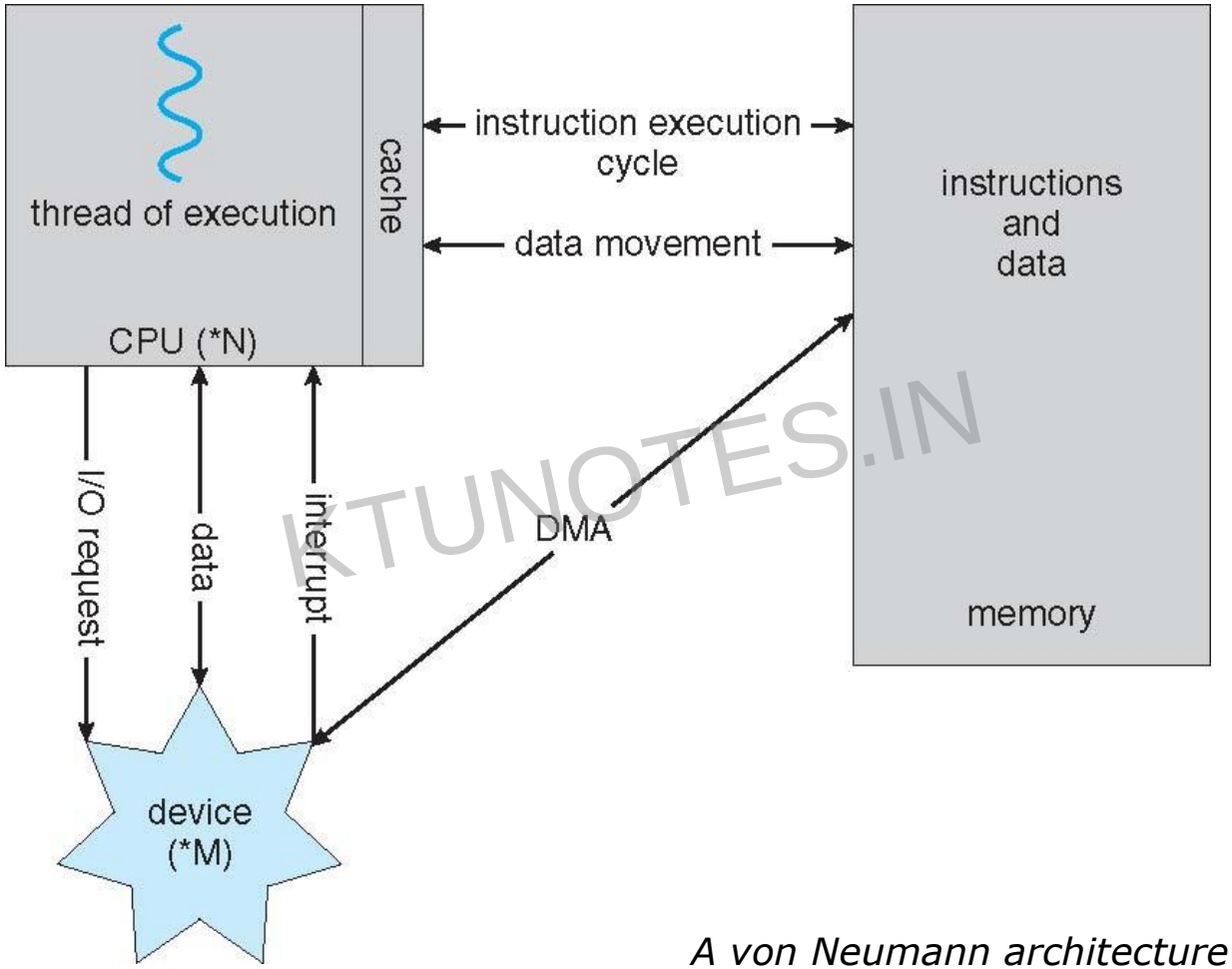
Caching

- 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

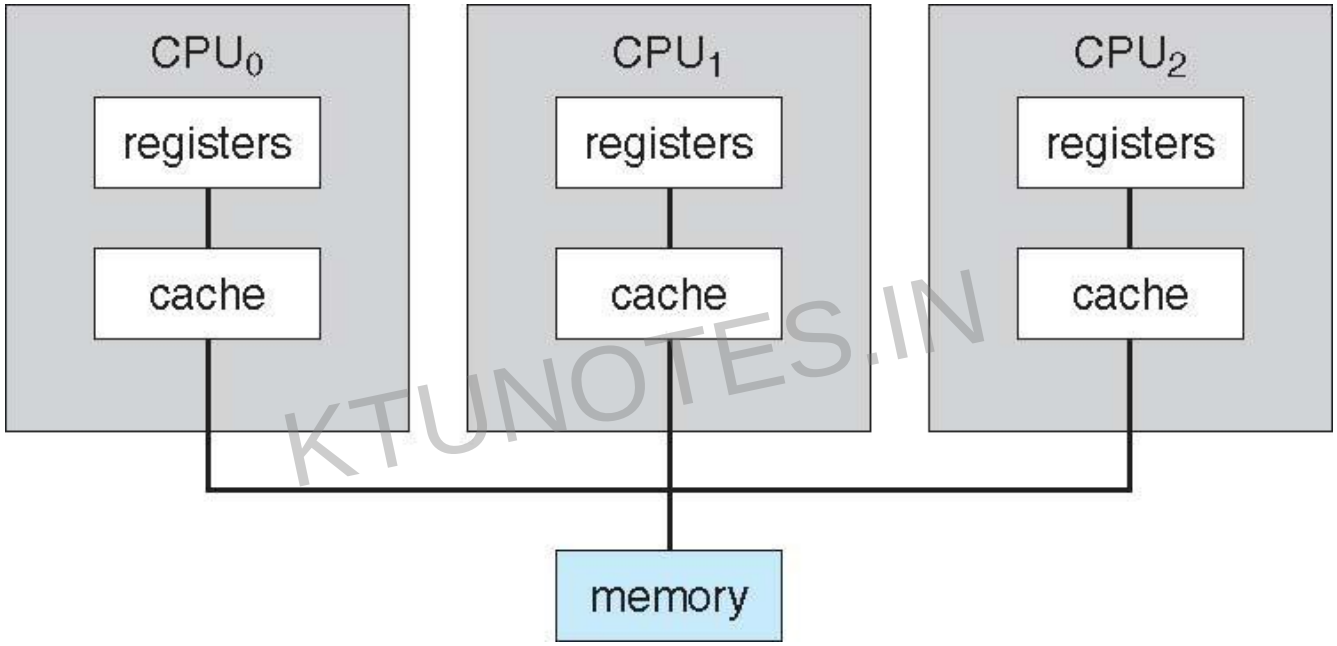
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased 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



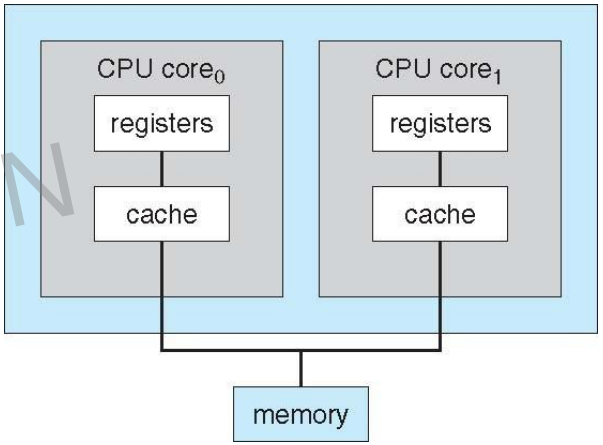
A von Neumann architecture

Symmetric Multiprocessing Architecture



A Dual-Core Design

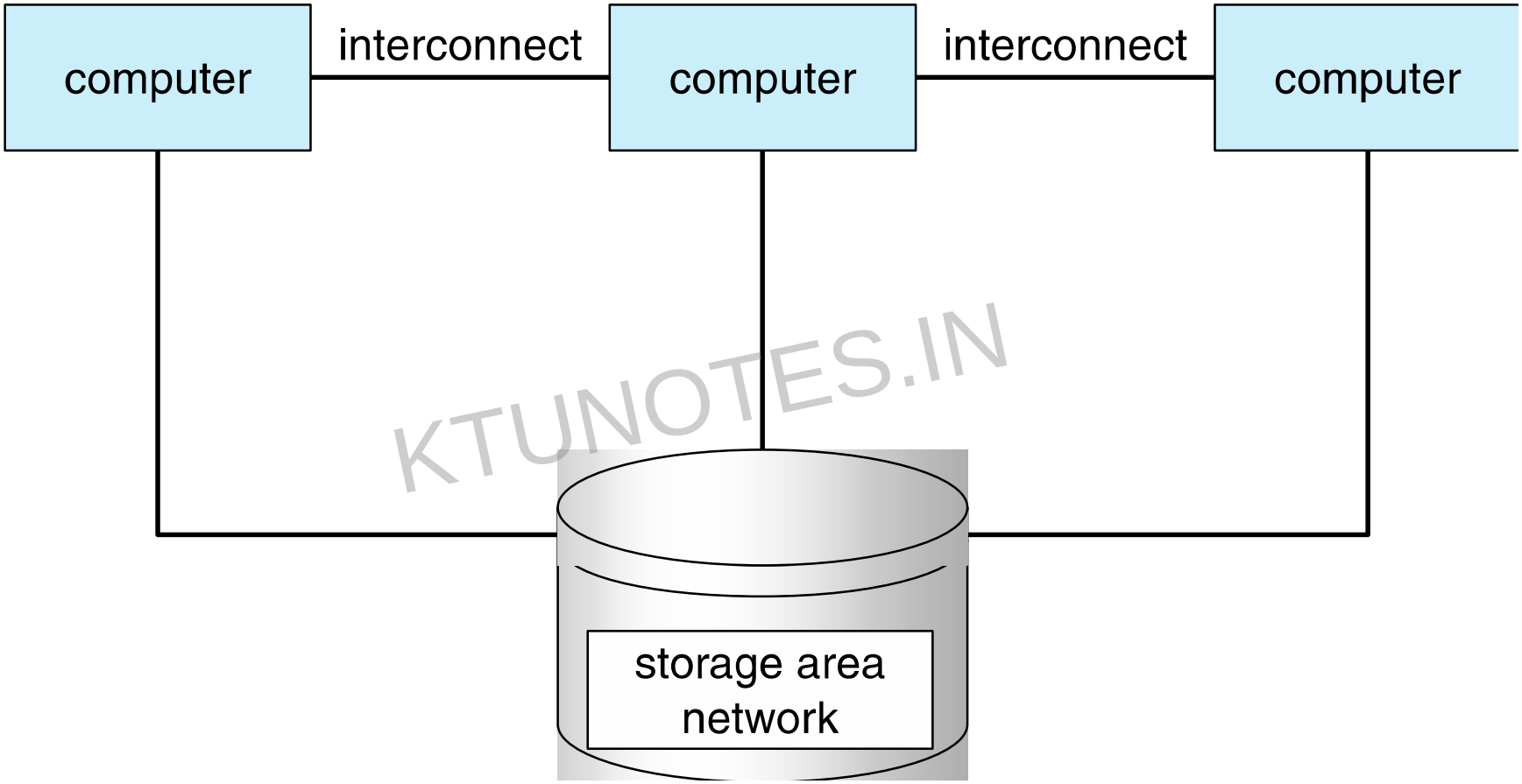
- **UMA** and **NUMA** architecture variations
- Multi-chip and **multicore**
- Systems containing all chips vs. **blade servers**
  - Chassis containing multiple separate systems



Clustered Systems

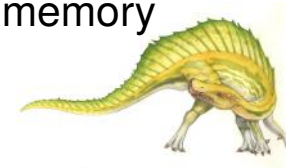
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** 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

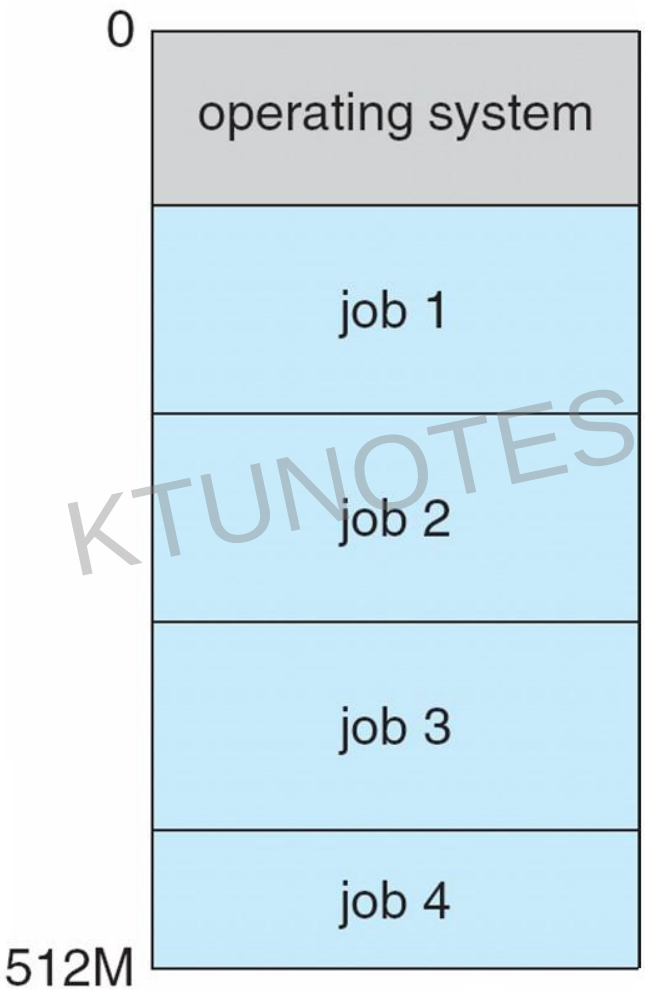


Operating System Structure

- **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 □**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 execution of processes not completely in memory



Memory Layout for Multiprogrammed System

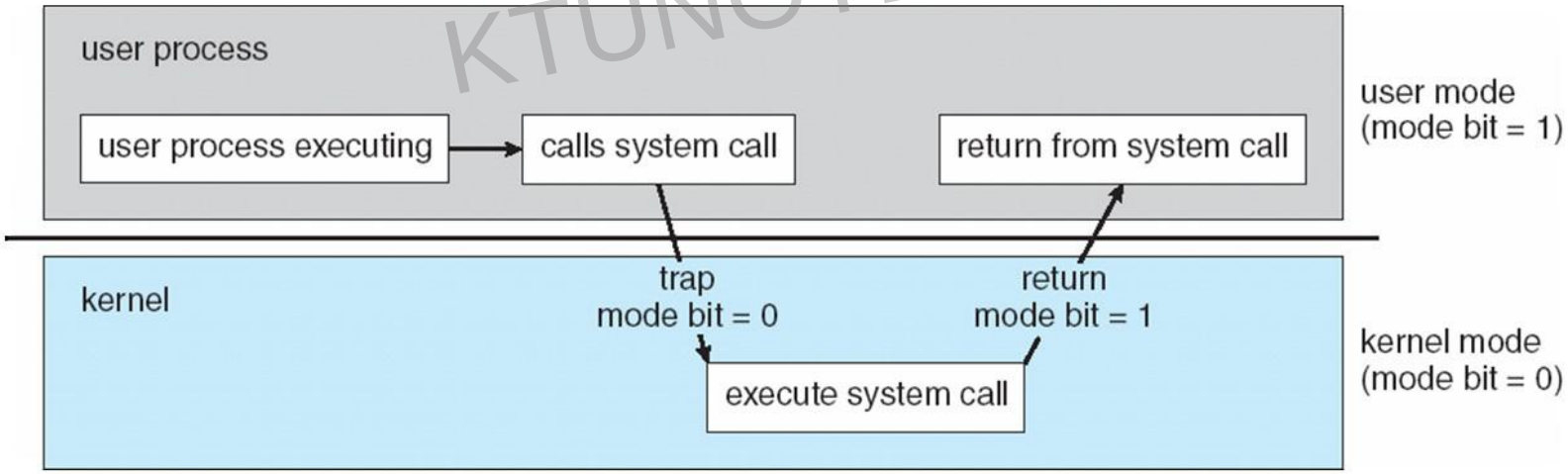


Operating-System Operations

- **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)

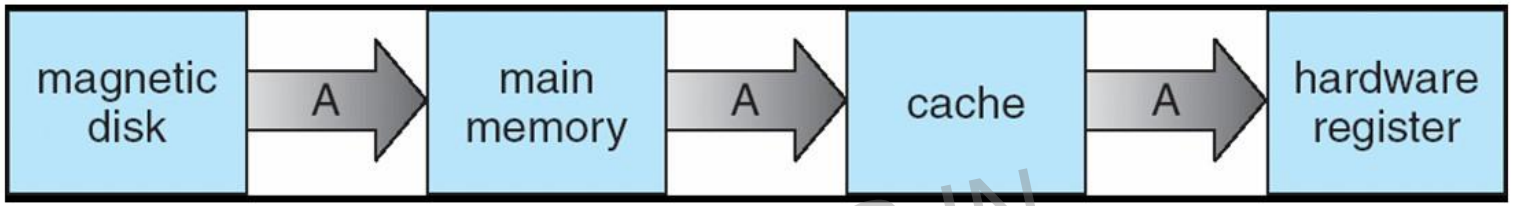
Performance of Various Levels of Storage

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

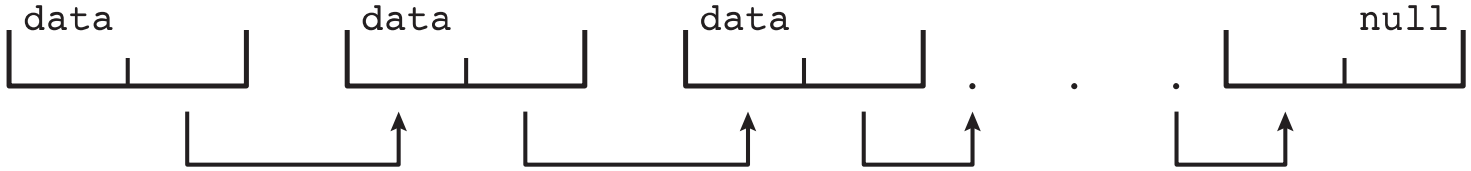
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver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

Protection and Security

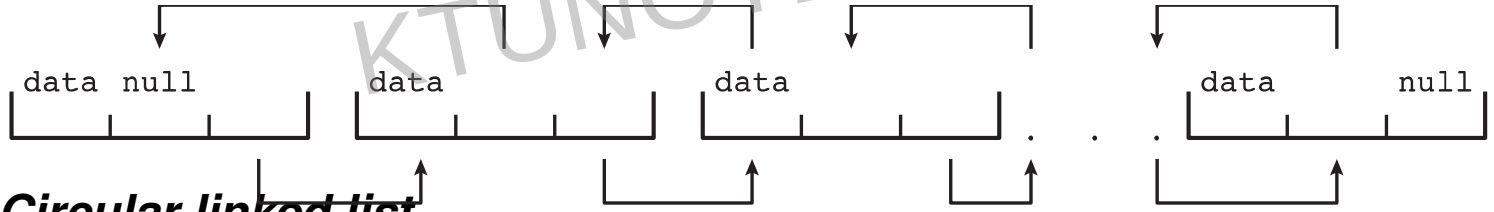
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first 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

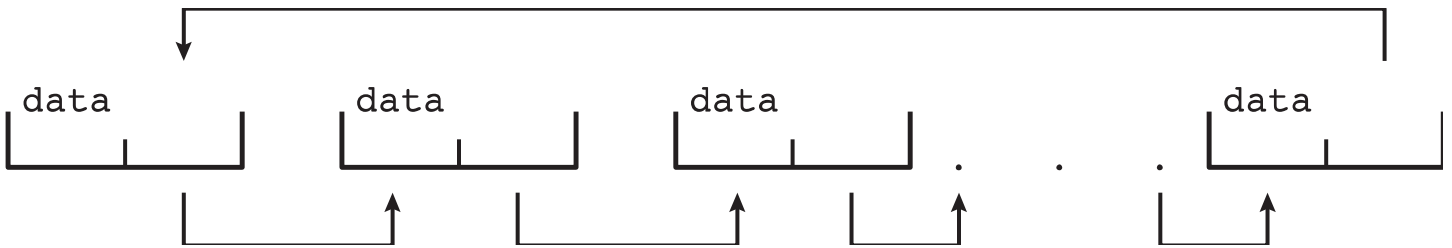
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

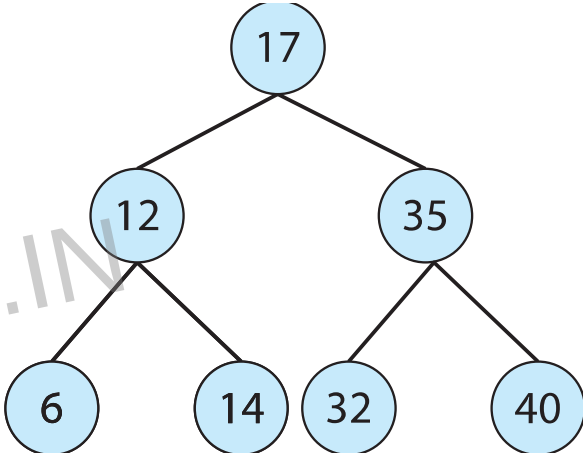


- ***Circular linked list***



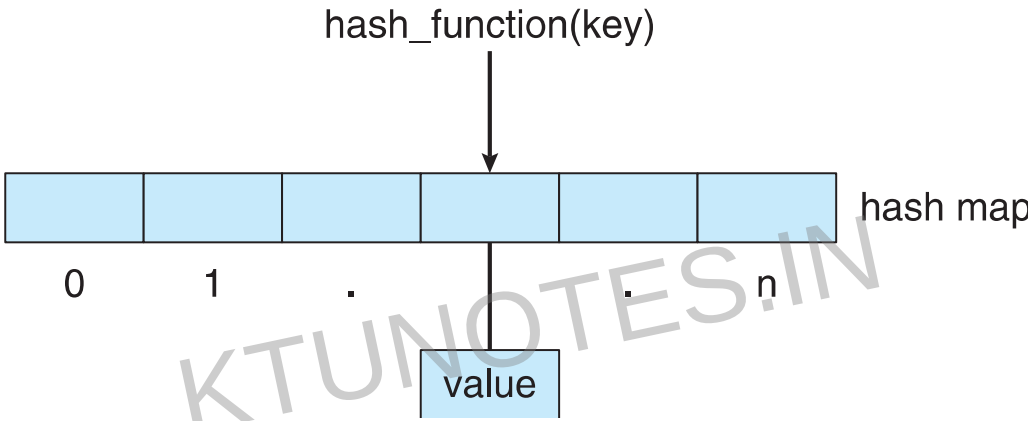
Kernel Data Structures

- **Binary search tree**
  left <= 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

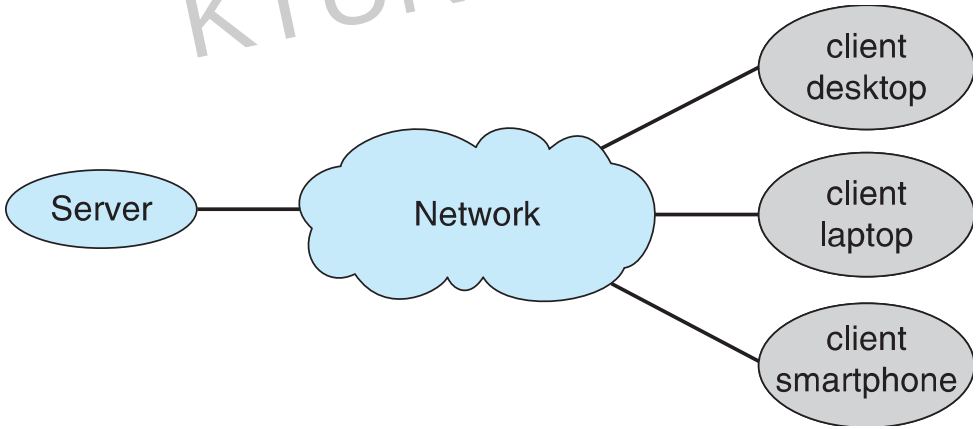
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders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

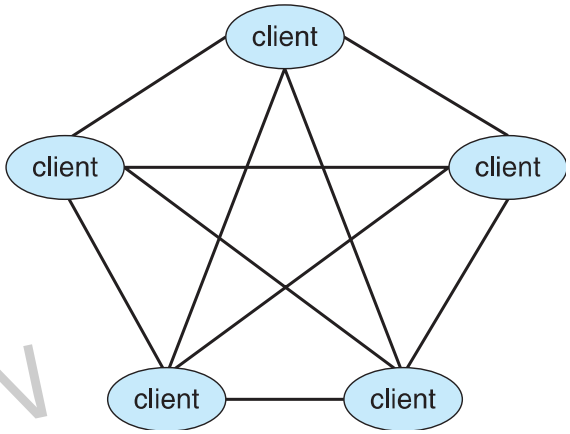
Computing Environments – Client-Server

- 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

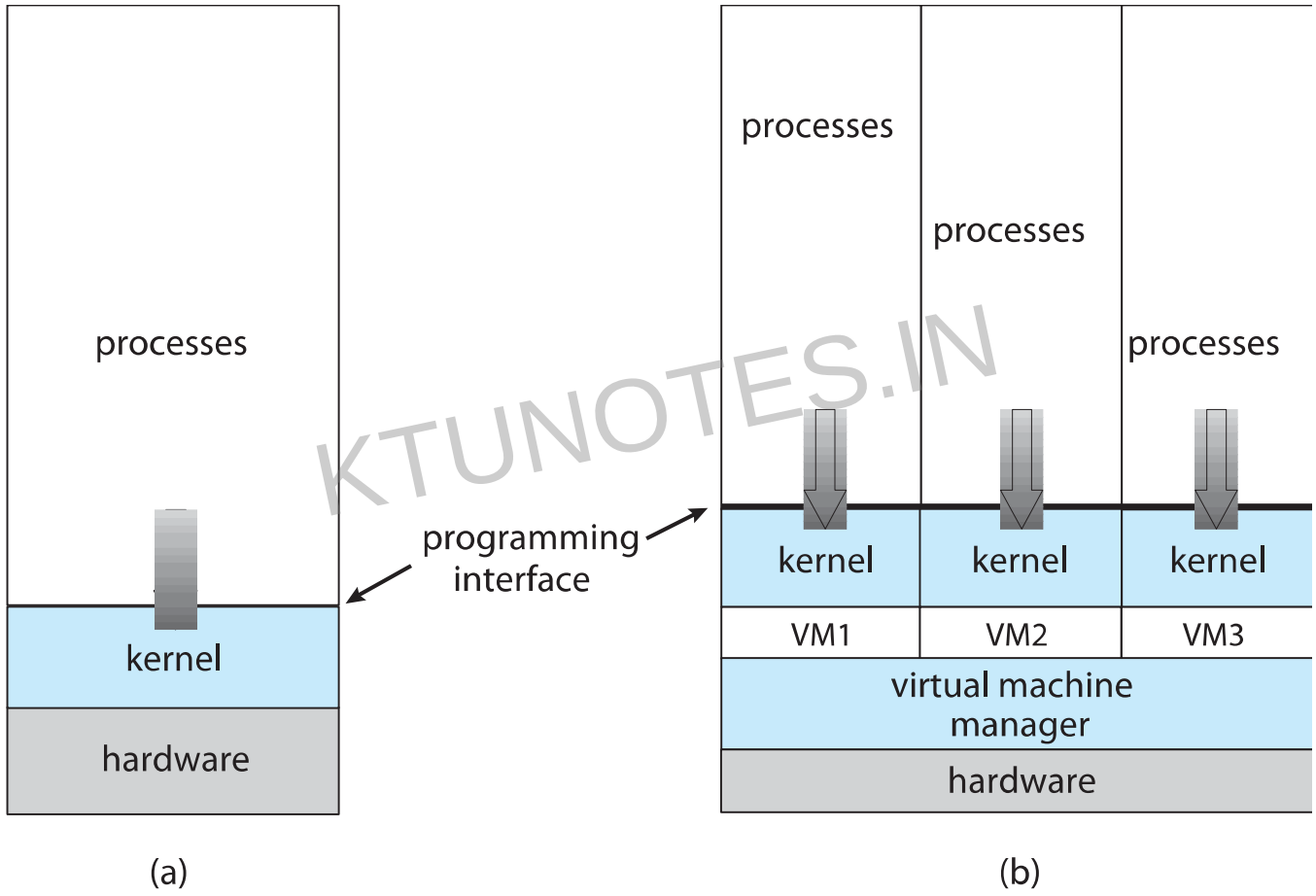
Computing Environments - Virtualization

- 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

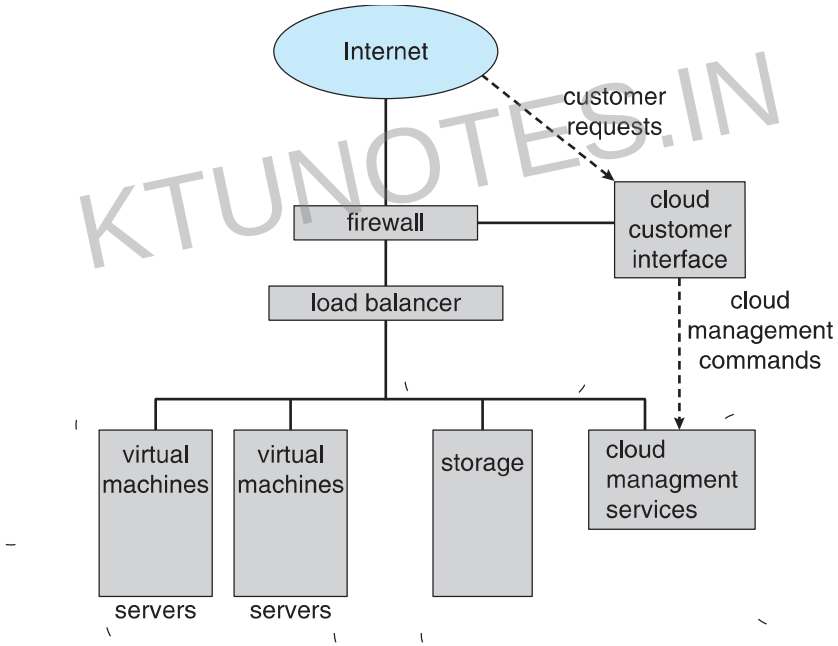


Computing Environments – Cloud Computing

- 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



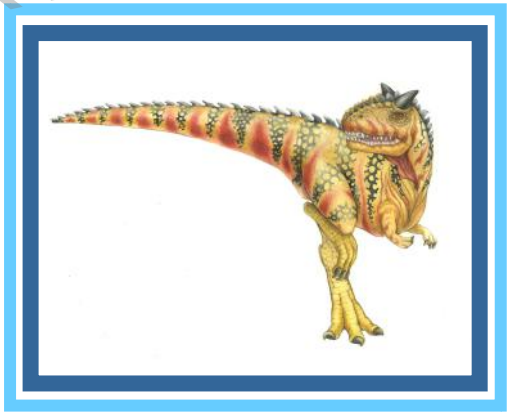
Computing Environments – Real-Time Embedded Systems

- 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

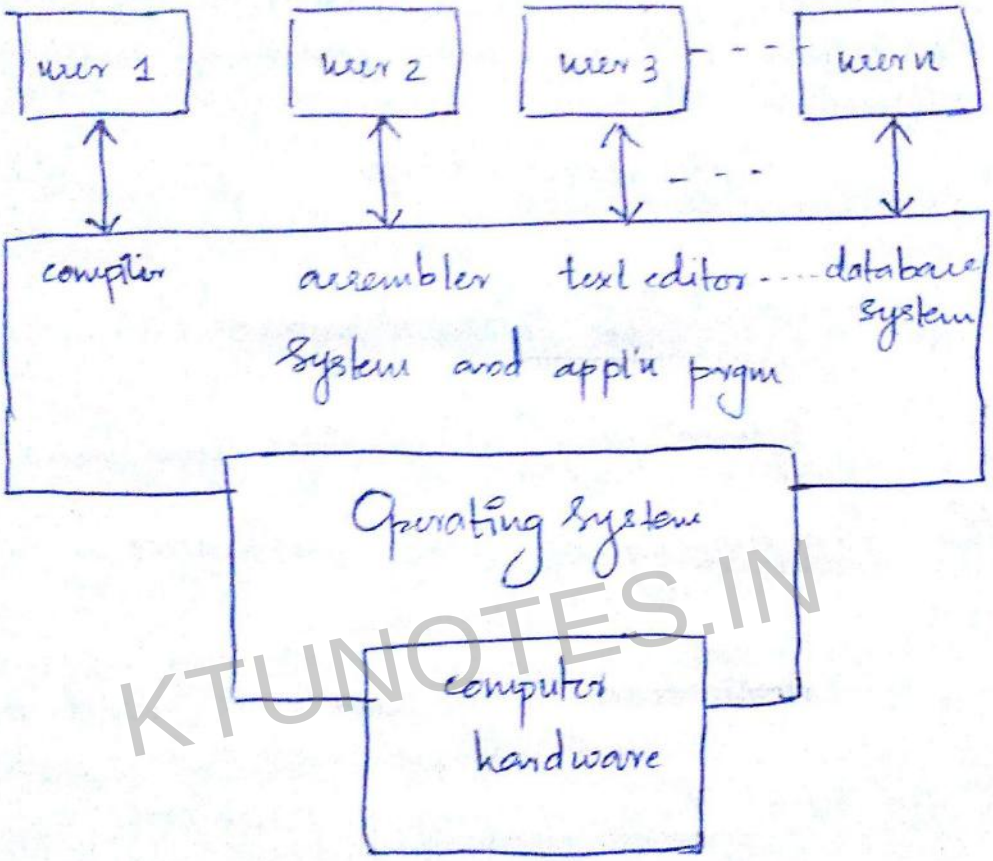
- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **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

End of Chapter 1



Module - 1

Operating System



Abstract views of component of a Computer System

OS

Operating System is a program that manages the computer hardware. It acts as an intermediary b/w the computer user and the computer hardware.

System program provides some service to some other softwares

Application softwares are used by users

eg: text editors

System and Application Program

System program provides some services for the functioning of other softwares.

Application programs are used by the users to solve their computing problems

"The core part of operating system is known as kernel."

functions As System View

1) It should act as a resource allocater
2) It should act as a control program

eg: It should avoid collision if we run multiprograms or multiple systems

Computer System Architecture

Depending on the number of processors used

1) Single processor computer

general purpose processor
special purpose processor

If there is only one general purpose processor we name it as a single purpose processor ~~also known as~~ ~~known as special~~

2) Multiprocessor system: Memory sharing

It is also known as parallel systems, tightly coupled system or multicore system

It has more than one or more general purpose processor.

It will get o/p faster

Advantages

i) Increased throughput (ie we get o/p faster) beg multiple processor are running.

ii) Less hardwares are used beg sharing hardware

iii) Economical and cost effective

iv) Increased the reliability (if one processor fails it will have other)

① Graceful degradation: the ability to continue providing service proportional to the level of surviving hardware is called graceful degradation.

② fault tolerant : System will continue working even after a failure occurs.

It can suffer a failure of any single component and still continue operation.

i) It should detect the fault

ii) Should find solutions to remove that fault

Multiprocessor System

Asymmetric multiprocessor system

⇒ There will be a controlling power

⇒ high performance

⇒ Here work allocate uniformly - if one processor is not working then the coming work will allocate to this processor.

⇒ Master - Slave Relationship

Symmetric multiprocessor (SMP)

To identify the master and slave, we use hardware / software.

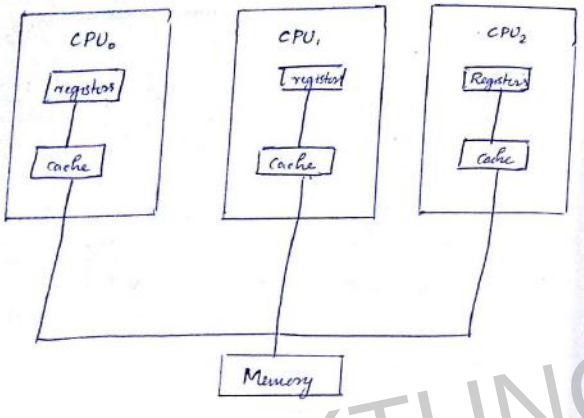
eg: Sun OS → version 4 (Asymmetric)

→ version 5 (SMP)

Here softwares are used to differentiate master and slave.

Multi-core processor

fig:- Symmetric Multiprocessor Architecture

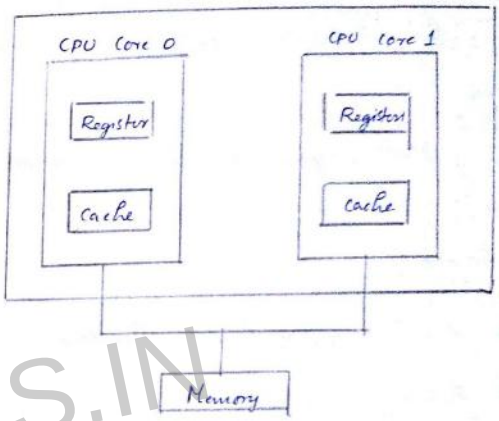


All processors share multiple processor so tightly coupled systems.

Registers are high speed memory for temporary storage.

Definition
Multicore processor - It includes multiple computing core on a single chip

fig:- A dual core design



Blade Servers

Here multiple processor board, I/O board and networking board are placed in the same frame. Here each blade processor board boots independently and runs its own operating system.

Advantage
Cost effective
Reduce power

3) Clustered systems: (loosely coupled)

Here individual systems are connected using wires

Multiple system are shared in single data base.

Advantage

⇒ It will be giving higher efficiency

⇒ Reliability (If failure occurs in one system, it will overcome by other systems)

⇒ Higher speed.

A cluster software is there which will control the all work.

Parallelization - It devides a program into seperate components that run in parallel on individual computers in the cluster. Once each computing node in the cluster has solved its portion of the problem, the results from all the nodes are combined into a final solution

Parallel Clusters

Multiple system will access the same storage i same database

advantage

⇒ It is easier if a modification occurs it will be done only in single storage or server

disadvantage

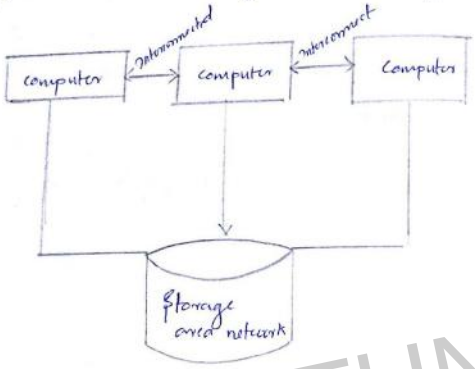
⇒ All system depend on single server so if that server stops due to any failure, the whole system became fail

⇒ Multiple system tries to access same thing or operation or data.

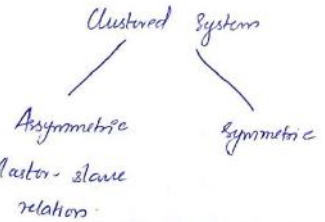
To avoid this, we introduce lock, so if the data is to be accessed, there should not be no lock, and if there is lock it should have to wait until the lock moves. This mechanism

is called Distributed lock Mechanism.

fig:- General Structure of a Clustered System



Each systems in this cluster systems have its own base memory. Here only data is sharing no memory sharing.



Parallel Clusters

Parallel cluster allows multiple host to acces the same data on shared storage. It requires the use of special versions of softwares and applications.

eg: Oracle Real Application Cluster

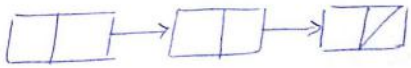
Kernel Data Structure

1a) List, stack, Queues

* Linked List

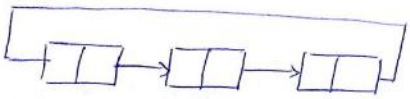
a) Singly Linked List
b) Doubly Linked List
c) Circular Linked List

a) Singly linked list



b) Doubly linked list

c) Circular linked list



* Stack

⇒ Stack is LIFO

⇒ Operations push and pop

⇒ It is used in function call (to store details regarding fn call)

⇒ function call related datas are stored in stack

* Queue

⇒ Queue is FIFO

eg's ⇒ Process scheduling decide in which order the program will execute when a multiprocess is working (procs - Round Robin)

⇒ Printer sharing is another example

2) Trees

⇒ Parent-child relationship

⇒ Heirarchical arrangement

⇒ Two types

a) General tree

b) Binary tree : left child and right child

Binary search tree

lc < parent < right child

3) Hash functions and maps

Hash function takes data as its i/p, performs a numeric operation on this data and returns a numeric value. This numeric value can then be used as an index into a table to

quickly retrieve the data.

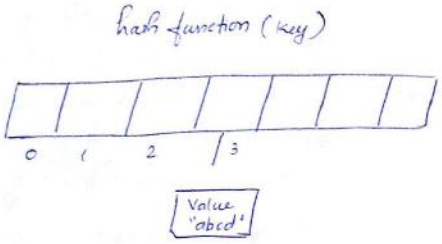


fig: hash maps

If user is the given username and abcd is the password given during sign up process. This password is stored in an index position using hash function. If we try to login our typed password is compared with stored password to check whether it is correct. If two things got same index then linked list is used.

4) Bit maps

It is a string of n binary digit. It is used to find out the availability of particular resources. Each bit position will correspond to particular resources.

eg: 11001001

1 → available
0 → unavailable

This method is used to reduce memory. This method provides the use of single bit to check the availability of resources.

Bit map is a string of n binary digits that can be used to represent the status of n items.

eg A disk drive might be divided into seven thousand individual units called disk blocks. Bit Map can be used to indicate availability of each disk block.

User Interface

a) Graphical user interface

- we object to interact with computers

b) Command Line interface (CLI)

- no graphics, only text commands
- always an interaction b/w user and computer systems

c) Batch Interface

- user command itself
- commands are given as a set of commands and when execution starts, all commands starting from first to last are executed with no interaction b/w the user and computer in b/w.

CLI

Command Interpreter (software)

↳ The function of command interpreter is to get and execute the next user specified command.

When multiple command line interpreter are found in a single OS, we call it as shells

Eg in LINUX, we have command shell

The two different methods for interpreting the commands by an command interpreter

1) The commands in itself contains the code for execution of command.

2) System program (provides services to some other)

=> here we will have seperate file for certain commands

Eg: In 1st method, the command create is a sub-program under the command program. But in the 2nd program, there will be a seperate file for the command create.

Most commands are implemented using system program.

GUI

Here the user employs a ~~or~~ mouse based window and menu system characterisation by a desktop.

- GUI was implemented first in Xerox Alto in 1973

Later in Apple Macintosh Aqua Version

- The Linux versions that uses GUI are

Linux versions
- common Desktop Environment (CDE)
- X - windows s/m
- K - Desktop Environment (KDE)
- GNOME Desktop

GESTURES : The interactions without the use of mouse

SHELL SCRIPT: Group or set of commands are stored in a file called shell

System calls provide an interface to the services made available by an OS

Application Programming Interface (API)

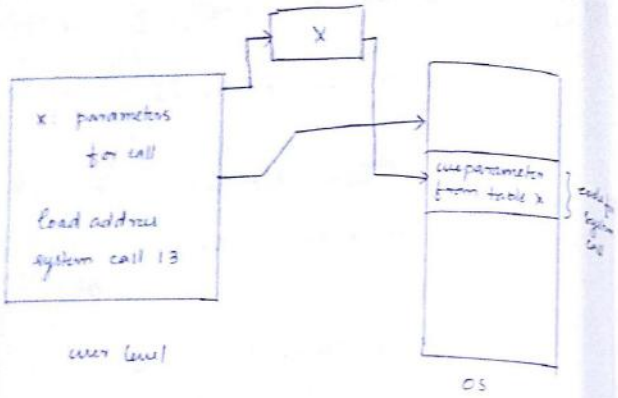
Set of functions that help the application programmer (a person who develops an application)

Create Process ()
↓ system calls
NT Create Process ()

eg: API Win 32

a) Registers are used to pass arguements at fn call
Registers are temporary storage

b) Tables
parameters are passed as tables or blocks
The address of these tables will be stored in registers



Types of System calls

1) Process control

A program in execution is called a process

i) end, abort

ii) load, execute: At the time of execution of a program when it needs to execute another program, the load and execute system call is used.

iii) create process, terminate process

When a program P_1 needs to execute another program P_2. There is program P_1 and P_2 needs to execute parallely then a create process (allocating memory) system call is used.

Terminate process is to stop that process & deallocate the memory.

iv) Get process attributes & set process attributes

To set certain values to attributes we use set process attributes.

To retrieve the attributes we use use get process attributes.

v) wait for time, wait for event, signal event

wait for time: when a program has to wait for a certain time.

When P_1 is a program that executes, it uses another program P_2 to read the i/p from the keyboard so P_1 stops its execution for program P_2 to return

data to P_1 so P_1 waits for an event and P_2 signals an event to P_1. acquire block, release lock.

If a process needs to share data, the process should aquire lock. After a process acquires a lock, it should release lock in order to share the data with other systems.

(2) File Management

* create file, delete file
* open, close
* read, write, repositions
* get file attribute, set file attribute

(3) Device Management

* Request device, release device
* Read, write, reposition
* get device attributes, set device attributes
* logically attach or detach devices.

(4) Information maintainance :

* get time or date, set time or date
* get system data, set system data
* read, write or repositions
* get process file, device attributes
* set process, file, device attributes

Program counter → shows the address of current instruction

time profile → shows the time taken by a set of instructions or set of system calls.

(5) communications :

Inter process communications (IPC)

Message Passing | Shared Memory

1. Message Passing:

Client	Server
Open connection()	wait for connection()
	accept connection()
read message()	
write message()	get hostid()
close connection()	get process id()

2. Shared Memory

Shared Memory create

Shared Memory Attach

3. Protection

set permission

get permission

allow user

deny user

Example of System calls

	Windows	Unix
Process control	Create process	fork()
	exit process	exit()
	wait for single object	wait()
File manipulation	Createfile()	open()
	Readfile()	read()
	write file()	write()
	Close Handle()	close()
Device	set (console Mode)	ioctl()
	Read Console()	read()
	write console()	write()
information maintenance	Get Current Process ID()	getpid()
	Set Timer()	alarm()
	Sleep()	sleep()

communication	Create Pipe()	pipe()
	Create File Mapping()	shmget()
	Map View File()	mmap()
Protection	SetFile Security()	chmod()
	Initialize Security Descriptor()	umask()
	Set Security Descriptor Group()	chown()

Operating System Design and Implementation

1) Defining the goals :

Requirements of OS

- user goals
  - => ease of use
  - => reliable
  - => fast
- system goals
  - => easy to design, implement and to maintain
  - => Easy to debugg

Goals should not unique for operating system

2) Mechanisms and policy

- "How to do" — eg: timer, counter (used to implement time-sharing)
- "What to do" — eg: time-sharing, batch-system, distributed system

Efficient OS should design is such a way that the mechanism to should insensitive to change in policies

eg: Micro kernel OS

eg: Solaris is an eg: for OS

↳ time sharing, batch processing all can be executed in this (implemented)

In order to implement this we use loadable table. (data in loadable table only changes only when policies changes)

3) Implementation

Two types Languages

(HLL) Highlevel Language – It is not machine independent

(LLL) lowlevel Language – Machine dependent or hardware dependent if it is design for particular machine then it cannot be use in another machine

we develop OS using assembly Language

1st OS use HLL is Master Control Program (MCP) and written in Language ALGOL

next OS developed is MULTICS and programming language is PL/1

HLL advantage

i) Portable due to machine independent
ii) easier to use
iii) easier to debugg

HLL disadvantage

i) Reduced speed
ii) It needs more memory

System Programs (System Utilities)

Provide convenient environment for program development and execution.

categories of System Programs

1) File management : File delete, copy, remove, rename all this comes under this
2) Status information : No of running process currently available memory, date, time,
3) File modification : commands for searching, copy etc
4) Programming language support : Interpreters, compilers, assemblers
5) Programming loading & execution : loading to primary memory. For this we

use absolute loader, overlay loader, relocatable loader, linkage editor.

c) communications : system programs used for communication among process, user, computer systems.

Operating System Structure

1) Monolithic System :

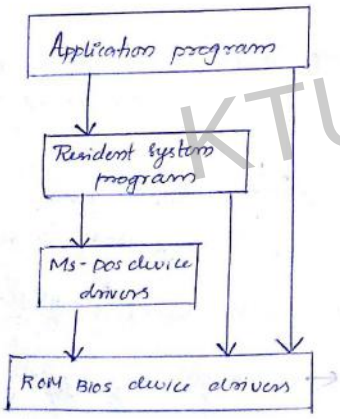


fig: MS-DOS structure

D Monolithic systems :

disadvantage : no differentiation b/w OS and functionality (there is no interface) so errors can occur or OS failures can occur. difficult to find errors

appln program directly call OS. It doesn't provide information hiding

Advantage : It is simple

OS is written as a collection of procedures linked together into a single large executable binary program.

2) layered approach

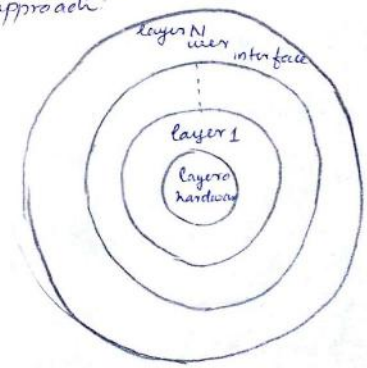


fig: A layered OS

Here it is divided into different layers.

inner - layer - computer hardware
outer - layer - user interface

Here each layer is an object and we provide neural data for each object

object
data / operation

Advantage:

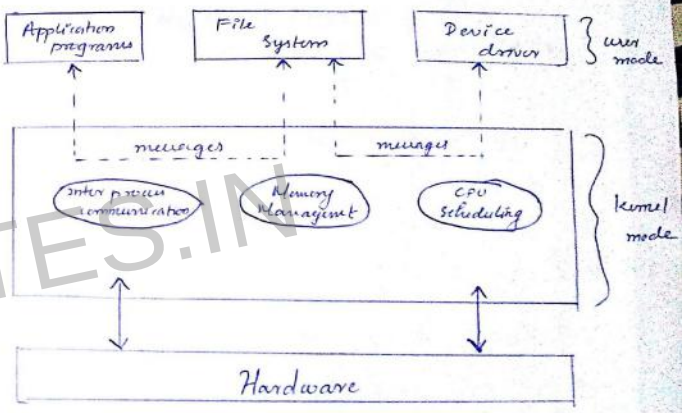
→ Information hiding is possible
→ debugging is easy and also error verification
⇒ Simpler construction or simpler implementation

Disadvantage:

⇒ designing or defining each layer is difficult
⇒ efficiency will be less, it takes more time (beg it has to pass through many layers before it reaches the desired layer)

eg: THE system (Batch system)
MULTICS

3) Microkernel approach



Only through kernel appln program can access file system only through messages

Advantages:

⇒ we can more functionalities to os without changing kernel program as user-level programs

=> Modification to kernel program is easy.

=> Portability is easier in case of microkernel bcz kernel programs are easier to change in that low-level programs is again little.

=> More secure and reliable.

eg: Tru64 UNIX system (implement this microkernel system)

4) Modules

dynamic calling or run time calling - loading kernel only and others are load at different time (like run time or booting time)

loadable modules can approach any module or layer.

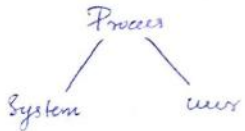
Here each is categorized as loadable modules

Here directly communications with each other.

eg: Solaris.

Module - 2

Process Management



Double clicking an icon is a method for converting program to an executable process.

Command line or by using commands we can convert system programs to process

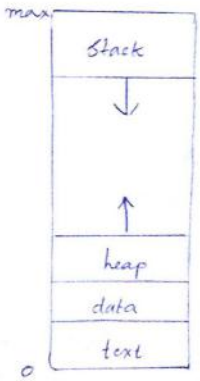


fig: Process in Memory

Program code of a process is known as text section

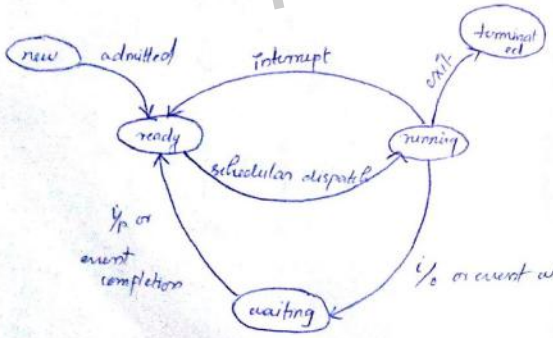
Variables used in program - data section

Memory allocated to a process dynamically - heap (Run time allocation)

Stack - Memory allocated during starting time.

text is common during the overall program when no multiple process is running in the same program.

Process state

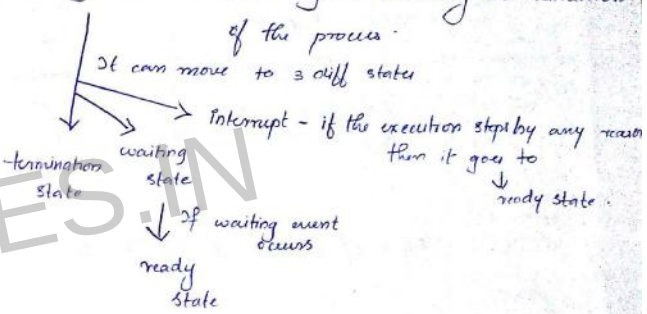


Process state is the current activity of that process

new - process is currently created or being created.

Ready - process is ready to execute
↓ when scheduler dispatches

running state - scheduling are executing the instructions of the process.
It can move to 3 diff states



new - process is being created.

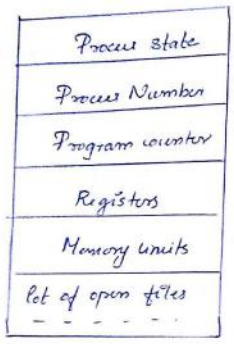
Ready - process waiting to be assigned to a process

running - instructions are being executed

waiting - the process is waiting for some event to occur.

terminated - process has finished execution

Process Control Block (PCB) or Task control Block



If a prog... goes to waiting sta... and the PCB is used to again get how the things to again star... the waiting proces...

PCB

Block of memory used to identify a process

Process number - It is also known as process id
every process has a unique id

program counter - ~~Stored of~~ stores the address of next executable process or instructions

Registers :

Types
1) Accumulators
2) Index Registers
3) stack pointers
4) General purpose registers

⇒ CPU scheduling information are also ~~stated~~ stored in a process control block.

⇒ Memory management info (ie values related memory mngmt) are also stored in PCB
↳ organizing the memory

⇒ Accounting info also stored in PCB
↳ { Amount of cpu time used
No of job or process
time limits etc..

⇒ IO status info also stored in PCB
↳ { how much I/P O/P devices
List of IO devices allocated to process
List of open files etc