**UNIT -1**

**COMPUTER SYSTEM AND OPERATING SYSTEM OVERVIEW**

**INTRODUCTION & SYSTEM STRUCTURES:**

An Operating System is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.

**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

**What Operating System Do:**

##### Computer System Structure

Computer system can be divided into four components  Hardware – provides basic computing resources

CPU, memory, I/O devices  Operating system

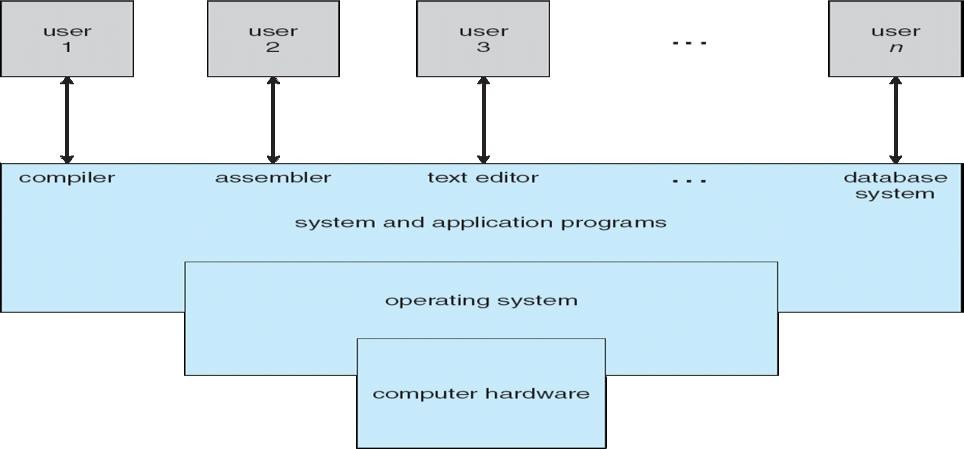
Controls and coordinates use of hardware among various applications and users

 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



**User View:**

* The goal is to maximize the work that the user is performing (OS designed most for **ease of use**)
* **Resource utilization-**Sharing various hardware and software resources
* Through **terminal,** a user can connect to a mainframe or minicomputer and others can access same computer through other terminals
* In still other cases, users sit at workstations connected to networks of other workstations and severs

**System View:**

OS is a **resource allocator**

Manages all resources

Decides between conflicting requests for efficient and fair resource use OS is a **control program**

Controls execution of programs to prevent errors and improper use of the computer No universally accepted definition

Everything a vendor ships when you order an operating system” is good approximation But varies wildly

**Defining Operating Systems:**

 “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 System Organization**

**Computer-System Operation:**

##### Computer Startup

**Bootstrap program** is loaded at power-up or reboot

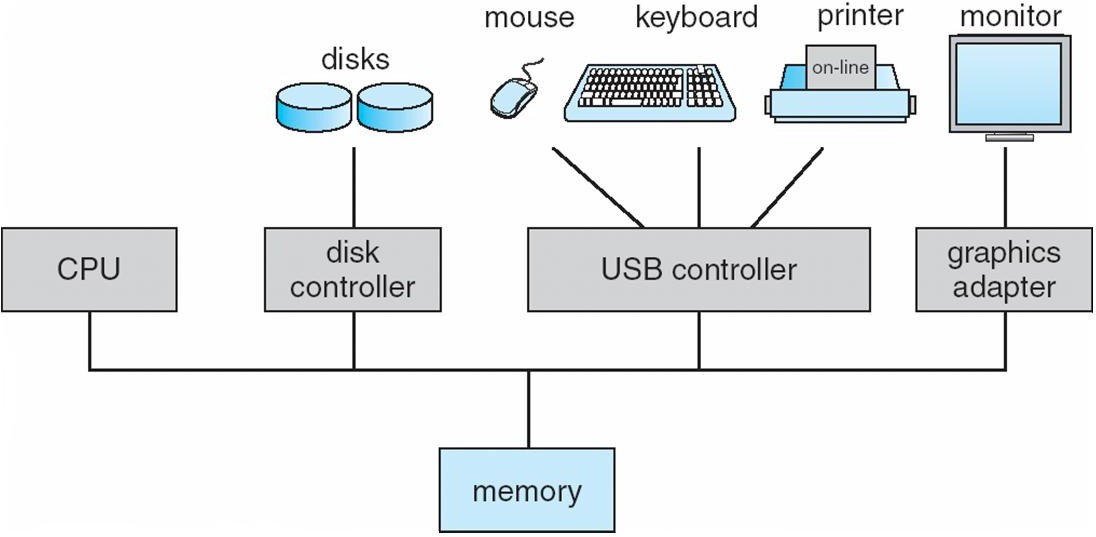
Typically stored in ROM or EPROM, generally known as **firmware**

Initializes all aspects of system

Loads operating system kernel and starts execution

Computer-system 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

 Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt* and A *trap* is a software-generated interrupt caused either by an error or a user request

 An operating system is **interrupt driven**

##### Interrupt Handling

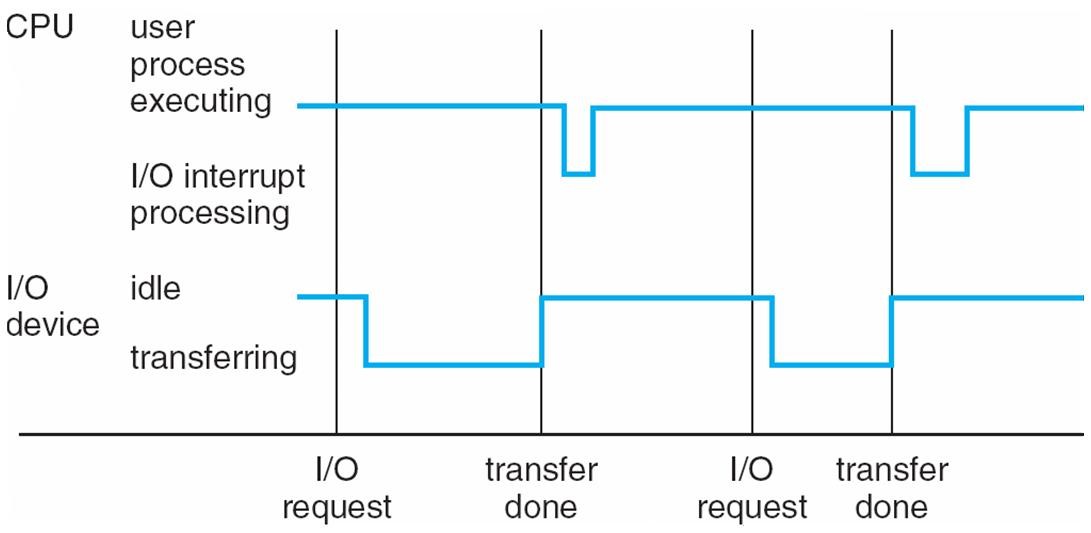
The operating system preserves the state of the CPU by storing registers and the program counter Determines which type of interrupt has occurred:

##### polling

* **vectored** interrupt system

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

##### Interrupt Timeline



##### Storage Structure:

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

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

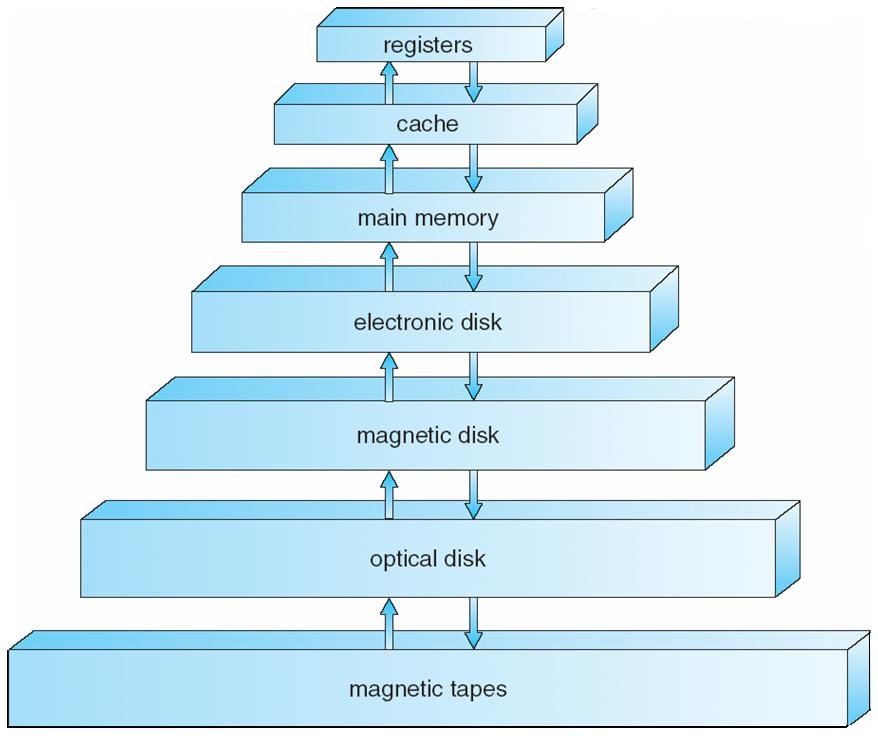
##### Storage Hierarchy

Storage systems organized in hierarchy Speed

Cost Volatility

**Caching** – copying information into faster storage system; main memory can be viewed as a last *cache* for

secondary storage



##### 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

**I/O Structure:**

* Storage is only one of many types of I/O devices within a computer.
* CPU and multiple devices are attached to one another
* 7 or more devices attached to SCSI(Small Computer System Interface)
* Multiple devices have device controller, device controller maintains local buffer
* Operating System have a device driver for each device controller
* Device driver loads data in device controller and device controller starts data transfer
* After completion of data transfer, it informs to device driver and device driver returns control to Operating System

##### Direct Memory Access Structure

* For small amount of data interrupt driven I/O is fine
* 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

**How a Modern Computer Works**

##### D:\OS\download.png

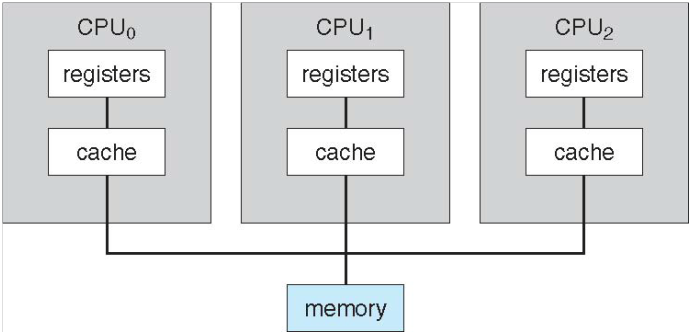
##### Computer-System Architecture

* Most systems use a single general-purpose processor (PDAs through mainframes), then the system is single-processor system
* On a **single-processor system**, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes
* 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

1. Increased reliability – graceful degradation or fault tolerance Two types
   1. Asymmetric Multiprocessing 2.Symmetric Multiprocessing



##### Fig: Symmetric Multiprocessing architecture

##### A Dual-Core Design

**Clustered Systems:**

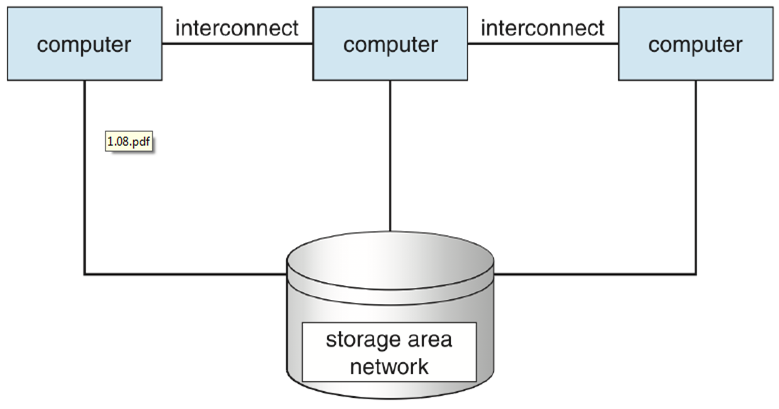
Like multiprocessor systems, but multiple systems working together Usually sharing storage via a storage-area network (SAN)

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

****

**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

Since, main memory is too small, the jobs are kept initially in job poll

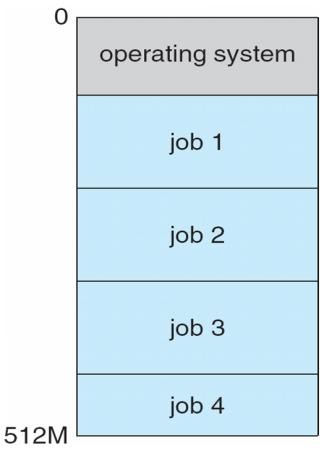
* A subset of total jobs in system is kept in memory (i.e) job pool

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-**The program loaded into memory and executing]

If several jobs ready to run at the same time

* [CPU scheduling-It makes decision to send a job]
* If processes don’t fit in memory(i.e) time sharing system not have proper response time then, swapping moves them in and out of main memory
* Virtual memory allows execution of processes not completely in memory

Fig: Memory Layout for Multiprogrammed System

##### 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



**Distributed Systems**

* A **distributed system** is a collection of physically separate, possibly heterogeneous, computer systems that are networked *to* provide the users with access to the various resources that the system maintains.
* Accessing a shared resource increases computation speed, functionality, data availability, and reliability.
* Systems contain two modes- FTP(File Transfer Protocol) and NFS(Network File System)
* These protocols affect the system’s utility and popularity
* **Network** is a communication path between two or more systems. Distributed systems depend on networking for their functionality.
* Networks vary by the protocols-TCP/IP
* Networks are characterized based on the distances between their nodes.
* A **Local Area Network** (LAN) is short distance and fast
* A **Wide Area Network** (WAN) is slower and long distance.
* A **metropolitan-area network** (MAN) is a computer network that interconnects users with computer resources in region larger than that covered by a large local area network but smaller than the area covered by a wide area network.
* **Small Area Network** has a example of Bluetooth and 802.11 devices use wireless technology to communicate over a distance of several feet
* A **network operating system** is an operating system that provides features such as file sharing across the network and that includes a communication scheme that allows different processes on different computers to exchange messages.

**Special-Purpose Systems**

* Vary from the general-purpose systems discussed so far
* Real-time embedded systems:
* Found on omnipresent embedded computers
  + - * VCRs, cars, phones, microwaves
* Very specific tasks, little or no user interface
* Vary considerably (general-purpose OS with special-purpose applications, hardware devices with special-purpose embedded OS, hardware device with application-specific integrated circuits (ASICs) that perform task without an OS
* Embedded systems almost always real-time
  + - * Rigid time requirements placed on operation of processor or data flow
* Multimedia data includes audio and video files as well as conventional files:
  + Multimedia data must be delivered (streamed) according to certain time restrictions
* Handheld Systems:
  + PDAs and cell phones
  + Use special-purpose embedded operating systems
  + Many physical device limitations (user interface, storage, performance)

**Computing Environments Traditional Computing:**

They are Blurring  Office environment

PCs connected to a network, with servers providing file and print services.

* + - Terminals attached to minicomputers providing a ways to access these computing environments

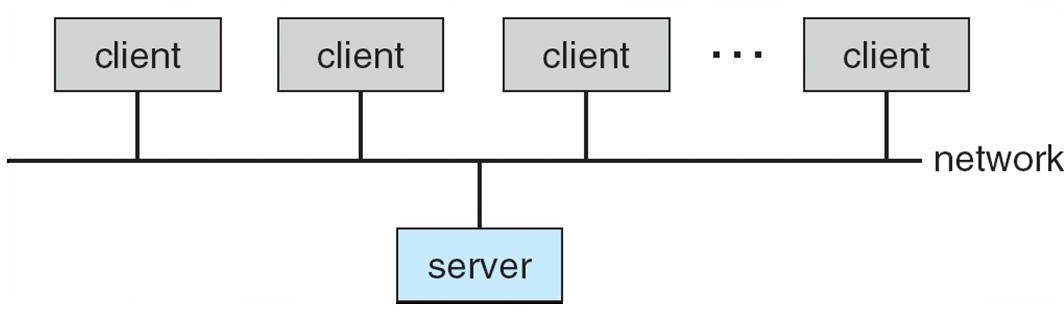
 Remote systems access and portability options are allowed to same resources

* + - Companies establish which provide Web accessibility to their internal servers
* Home networks
  + - Most users had a single computer with a slow modem connection to the office
    - Today, network-connection speeds available at great cost are relatively inexpensive, giving home users more access to more data
* Some homes have firewall to protect their networks from security breaches
* In previous century, systems were either batch or interactive.
* Batch systems processed jobs in bulk, with predetermined input (from files or other sources of data).
* Interactive systems waited for input from users.
* To optimize this system, Time-sharing systems used a timer and scheduling algorithms

**Client-Server Computing:**

* Terminals connected to centralized systems are now being supplanted by PCs

 More pcs are connected to centralized systems increases user functionality, so Many systems now acts as **servers**, responding to requests generated by **clients**



**Compute-server** provides an interface to client to request services (i.e. database)



**File-server** provides interface for clients to store and retrieve files

##### Peer-to-Peer Computing:

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 (audio files and songs)* and *Gnutella (file sharing)*

##### Web-Based Computing:

Web has become ubiquitous PCs most prevalent devices

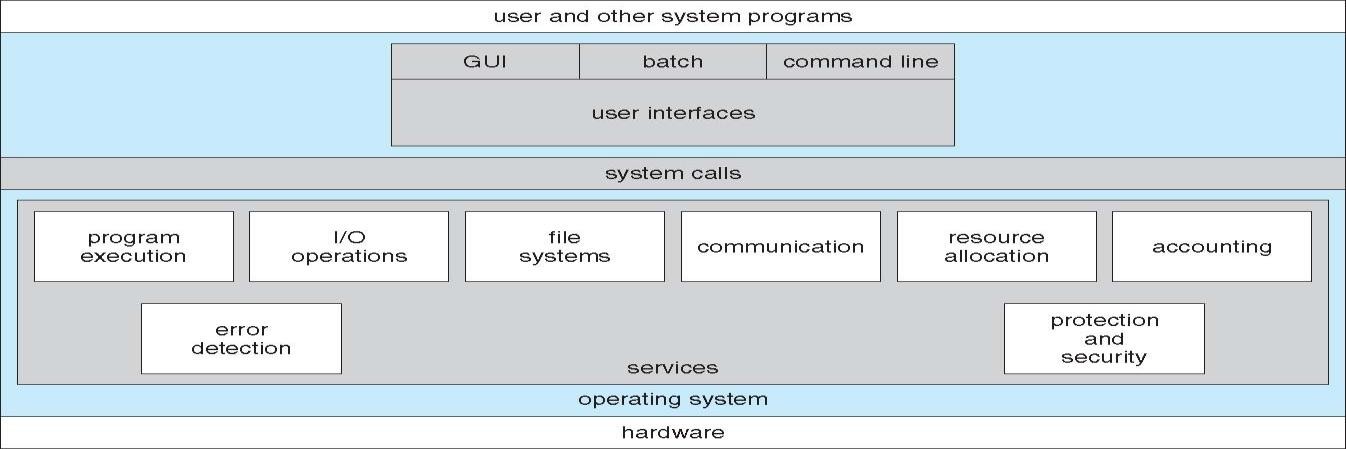
More devices becoming networked to allow web access

New category of devices to manage web traffic among similar servers: **load balancers**

Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

##### Operating System Services

##### A View of Operating System Services



**Operating System Services**

One set of operating-system services provides functions that are helpful to the user: User interface - Almost all operating systems have a user interface (UI)

 Varies between DTrace Command-Line (CLI)-text commands and method for entering, Graphics User Interface (GUI), Batch-commands which are entered into the files and executing those files

* Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)

I/O operations - A running program may require I/O, which may involve a file or an I/O device

 File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management

 Communications – Processes may exchange information, on the same computer or between computers over a network Communications may be via shared memory or through message passing (packets moved by the OS)

 Error detection – OS needs to be constantly aware of possible errors. It may occur in the CPU and memory hardware, in I/O devices, in user program For each type of error, OS should take the appropriate action to ensure correct and consistent computing Debugging facilities can greatly enhance the user’s and programmer’s abilities to efficiently use the system

 Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing

 **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them. Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code

* **Accounting -** To keep track of which users use how much and what kinds of computer resources

 **Protection and security -** The owners of information stored in a multiuser or networked computer

system may want to control use of that information, concurrent processes should not interfere with each other

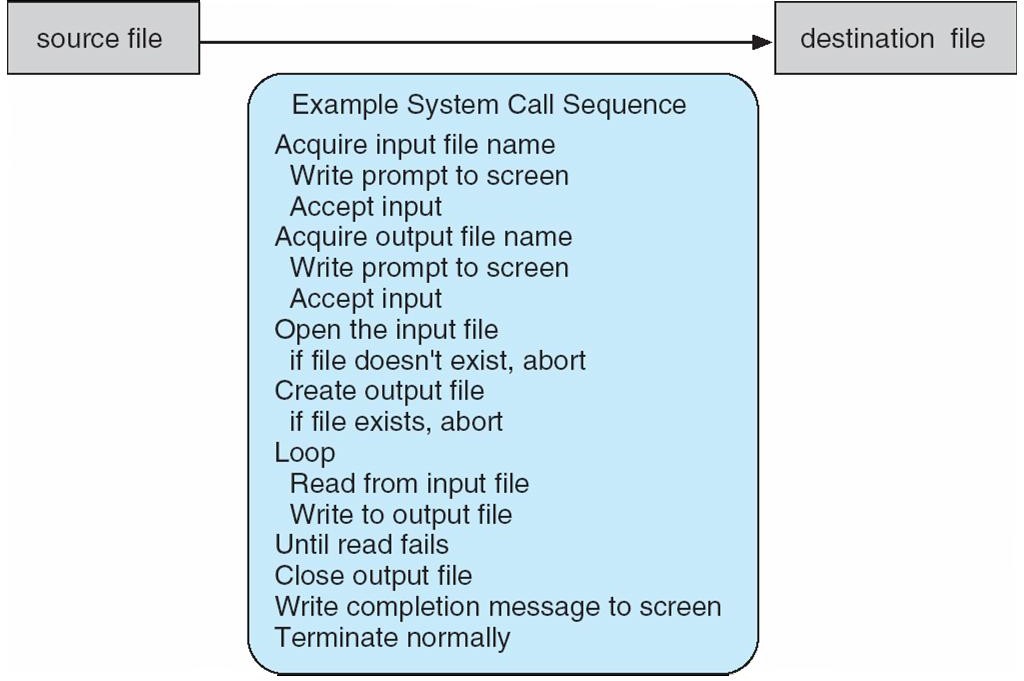
* **Protection** involves ensuring that all access to system resources is controlled
* **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
* If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

**System Calls**

* System calls provide interface to the services made by OS
* Typically written in a high-level language (C or C++)

##### Fig: Example of how System Calls are used

##### System call sequence to copy the contents of one file to another file

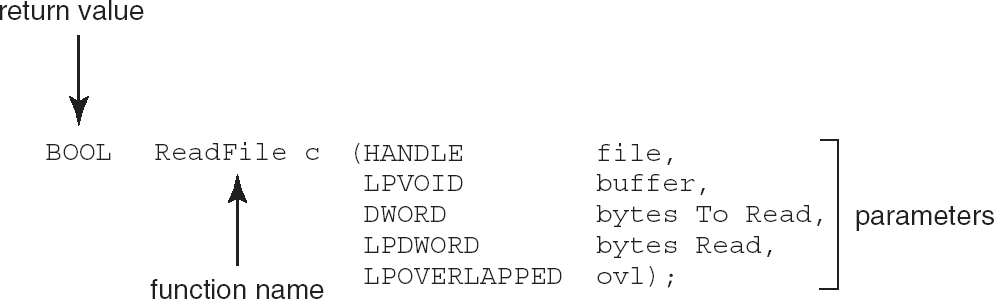


In the above example, a simple program is making use of OS more times; systems execute thousands of system calls per second.

* Mostly programs accessed by an Application Programming Interface (API) rather than a direct system call. Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
* Why use APIs rather than system calls?(Note that the system-call names used throughout this text are generic example)
* An application programmer designing a program using an API can expect her program to compile and run on any system that supports the same API. Actual system calls can often be more detailed and difficult to work with than the API available to an application programmer
* System calls – driven by ease of implementation, API – driven by ease of use

Fig: Example of Standard API

Consider the ReadFile() function in the Win32 API—a function for reading from a file



A description of the parameters passed to ReadFile()

HANDLE file—the file to be read

LPVOID buffer—a buffer where the data will be read into and written from DWORD bytesToRead—the number of bytes to be read into the buffer LPDWORD bytesRead—the number of bytes read during the last read LPOVERLAPPED ovl—indicates if overlapped I/O is being used

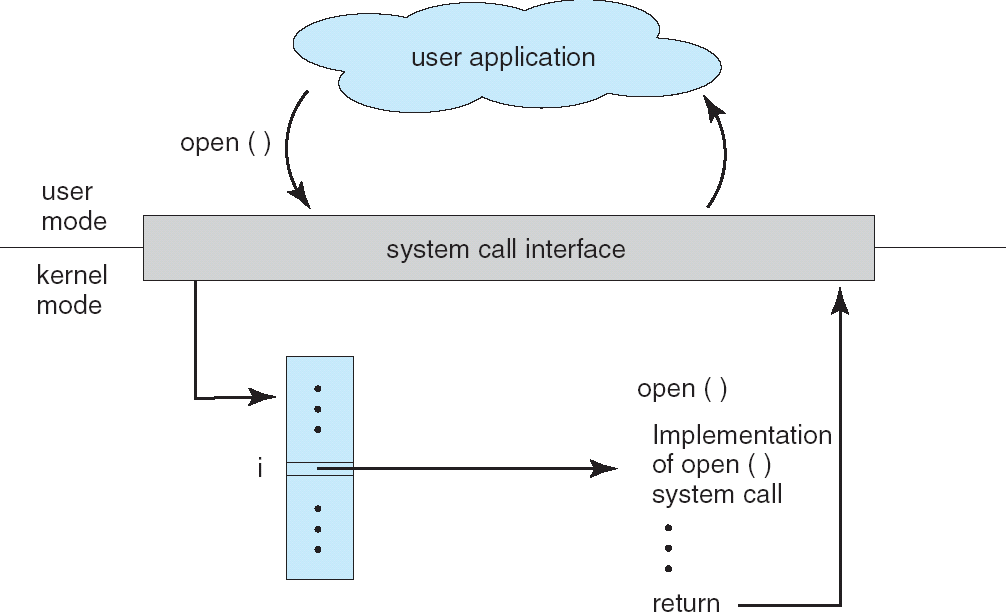
##### System Call Implementation

* Typically, a number associated with each system call
  + System-call interface maintains a table indexed according to these Numbers
* The system call interface intercepts function calls in the API & invokes intended system call in OS kernel and returns status of the system call and any return values
* The caller need know nothing about how the system call is implemented
  + Rather, it needs to obey API and understand what OS will do as a result call
  + Most details of OS interface hidden from programmer by API

Managed by run-time support library (set of functions built into libraries included with compiler)

##### API – System Call – OS Relationship

**Standard C Library Example**



**System Call Parameter Passing**

* System calls occur in different ways, more information is required than simply identity of desired system call
* Exact type and amount of information vary according to OS and call

Three general methods used to pass parameters to the OS

* Simplest: pass the parameters in registers

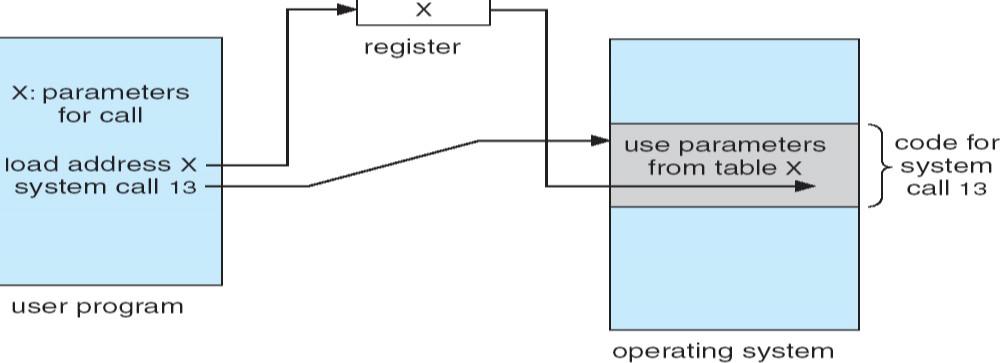
 In some cases, may be more parameters than registers

* Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register

 This approach taken by Linux and Solaris

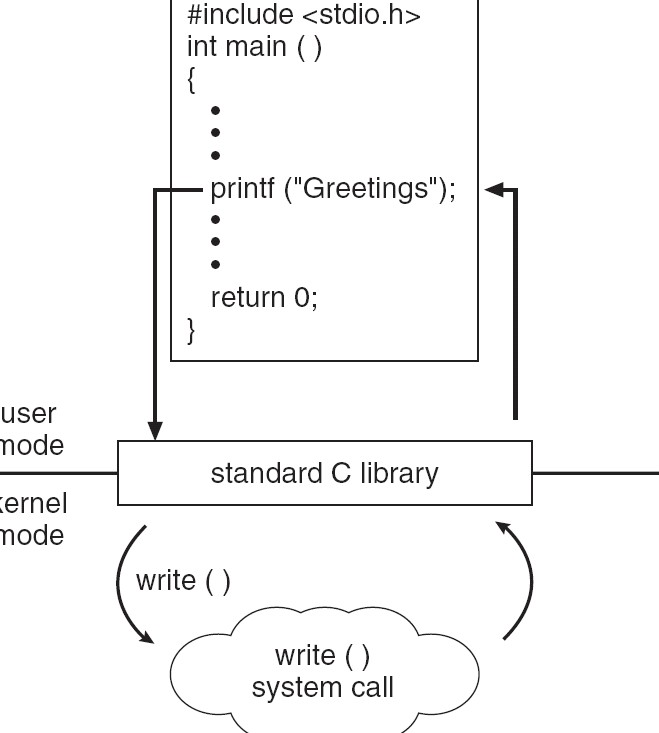
* Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system
* Block and stack methods do not limit the number or length of parameters being passed

##### Fig: Parameter Passing via Table



**Types of System Calls**

* **Process control** - A running program needs to be able to halt its execution either normally (end) or abnormally (abort). If a system call is made to terminate the currently running program abnormally, or any problem occurs and causes an error trap, a dump of memory is sometimes taken and an error message generated.
* The dump is written to disk and may be examined by **debugger**-a system program designed to aid the programmer in finding and correcting bug
  + A **control card** is a batch-system concept. It is a command to manage the execution of a process.



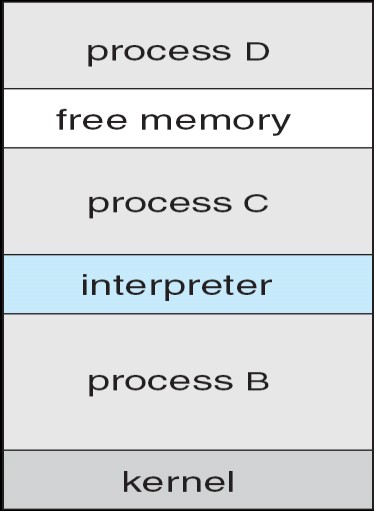
**MS-DOS execution**



(a) At system startup (b) running a program

* The MS-DOS operating system is an example of a single-tasking system. It has a command interpreter that is invoked when the computer is started (Figure (a)).
* Because MS-DOS is single-tasking, it uses a simple method to run a program and does not create a new process. It loads the program into memory, writing over most of itself to give the program as much memory as possible (Figure (b)).

##### Fig: FreeBSD Running Multiple Programs



* FreeBSD is an example of a multitasking system. When a user logs on to the system the shell of the user's-choice is run.
* This shell is similar to the MS-DOS shell.
* However, since FreeBSD is a multitasking system, the command interpreter may continue running while another program is executed (Figure). To start a new process a shell executes a fork() system call. Then, the selected program is loaded into memory via an exec() system call, and the program is executed
* **File management**- identify several common system calls dealing with files. At least two system calls, get file attribute and set file attribute, are required for this function
* **Device management**- The various resources (main memory(H/W), disk drives(H/W), access to files(S/W)) controlled by the operating system can be thought of as devices.
* **Information maintenance** - the operating system keeps information about all its processes, and system calls are used to access this information. Generally, calls are also used to reset the process information (get process attributes and set process attributes)
* **Communications**- There are two common models of interprocess communication: the message-passing model and the shared-memory model.
  + Message-passing model- In Message-passing model, the communicating processes exchange messages with one another to transfer information
  + In shared-memory model, processes use shared memory create and shared memory attach system calls to create and gain access to regions of memory owned by other processes.
* **Protection**- Protection provides a mechanism for controlling access to the resources provided by a computer system. Typically, system calls providing protection include set permission and get permission, which manipulate the permission settings of resources such as files and disks.

Types of system calls.

Process control

o end, abort

o load, execute

o create process, terminate process

o get process attributes, set process attributes

o wait for time

o wait event, signal event

o allocate and free memory

File management

o create file, delete file

o open, close

o read, write, reposition

o get file attributes, set file attributes

Device management

o request device, release device

o read, write, reposition

o get device attributes, set device attributes

o logically attach or detach devices

Information maintenance

o get time or date, set time or date

o get system data, set system data

o get process, file, or device attributes

o set process, file, or device attributes

Communications

o create, delete communication connection

o send, receive messages

o transfer status information

o attach or detach remote devices

##### Examples of Windows and Unix System Calls



**Operating System Generation**

* Operating systems are designed to run on any of a class of machines
* The system must be configured for each specific computer site a process sometimes known as system generation (SYSGEN)
* This SYSGEN program reads from a given file, or asks the operator of the system for information concerning the specific configuration of the hardware system or probes the hardware directly to determine what components are there.
* The following kinds of information must be determined.
* What CPU is to be used?
* How will the boot disk be formatted?
* How much memory is available?
* What devices are available?
* What operating-system options are desired, or what parameter values are to be used?
* The major differences among different approaches are the size and generality of the generated system and the ease of modifying it as the hardware configuration changes.
* Consider the cost of modifying the system to support a newly acquired graphics terminal or another disk drive.
* Booting – starting a computer by loading the kernel
* *Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution