Topics Covered

- What is Operating System
- Abstract view of Operating System
- Goals and functions of Operating System
- Evolution of Operating System
- Multiprogramming Operating System
- Multitasking/Time Sharing Operating System
- Multiprocessing Processing Operating System
- Real Time Operating System
- Distributed Operating System

What is Operating System

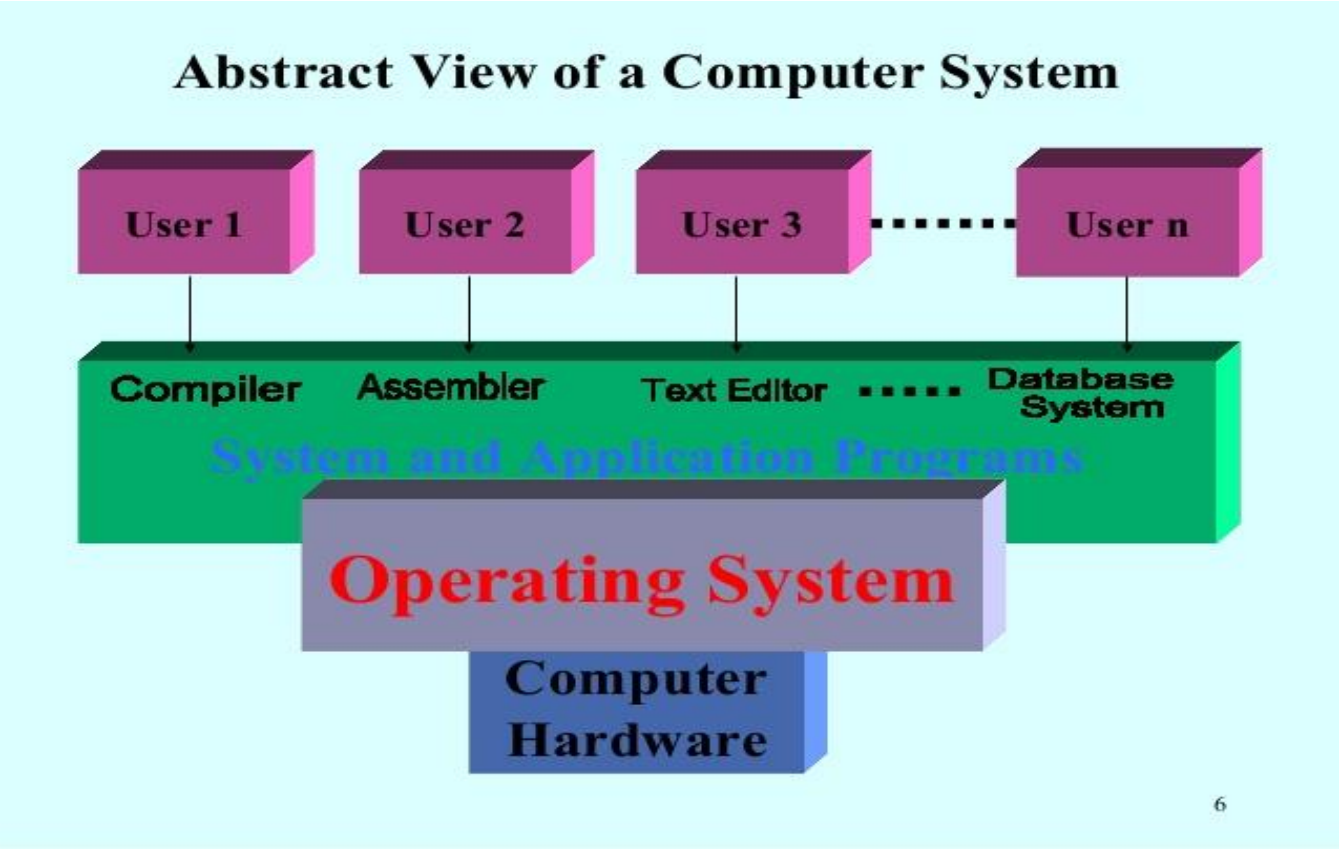
- Whatever used as an interface between the user and the core machine is OS.
    - E.g. steering of car, switch of the fan etc., Buttons over electronic devices.
- Question comes why we need an operating system?
    - To enable everybody to use h/w In a convenient and efficient manner
- Definition of Operating System
    - **There is no exact or precise definition for OS but we can say, "A program or System software"**
        - Which Acts as an intermediary between user & h/w
        - Resource Manager/Allocator - Manage system resources in an unbiased fashion both h/w (mainly CPU time, memory, system buses) & s/w (access, authorization, semaphores) and provide functionality to application programs. OS controls and coordinates the use of resources among various application programs.
        - OS provides platform on which other application programs can be installed, provides the environment within which programs are executed.
- For personal computers - Microsoft windows (82.7%), Mac (13.23), Linux (1.57)
- For mobile phones - Android (87.5), IOS (12.1) etc.

Q An operating system is: **(NET-DEC-2006)**

(A) Collection of hardware components
(B) Collection of input-output devices
(C) Collection of software routines
(D) All the above

Answer: C

Abstract view of operating system



- Computer hardware – CPU, memory units, i/o devices, system bus, registers etc. provides the basic computing resources.
- OS - Control and coordinates the use of the hardware among the various applications programs.
- System and Applications programs - Defines the way in which these resources are used to solve the computing problems of the user.
- User

Goals and Functions of operating system

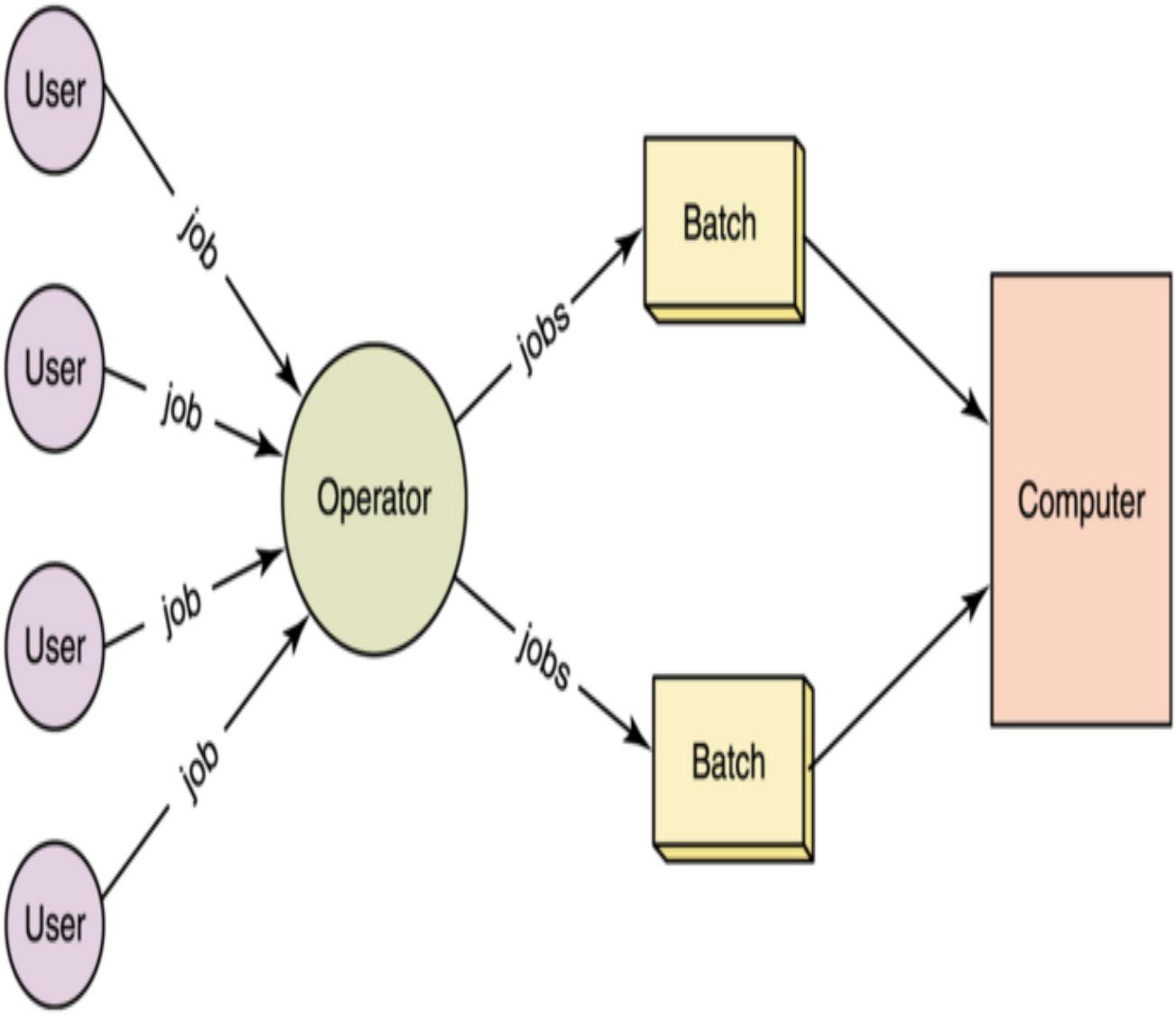
- Goals are the ultimate destination, but we follow functions to implement goals
- Goals
  - Primary goals (Convenience / user friendly)
  - Secondary goals (Efficiency (Using resources in efficient manner) / Reliability / maintainability)
- **Functions of operating system**
  - Process management
  - Memory management
  - I\O device management
  - File management
  - Network management
  - Security & Protection

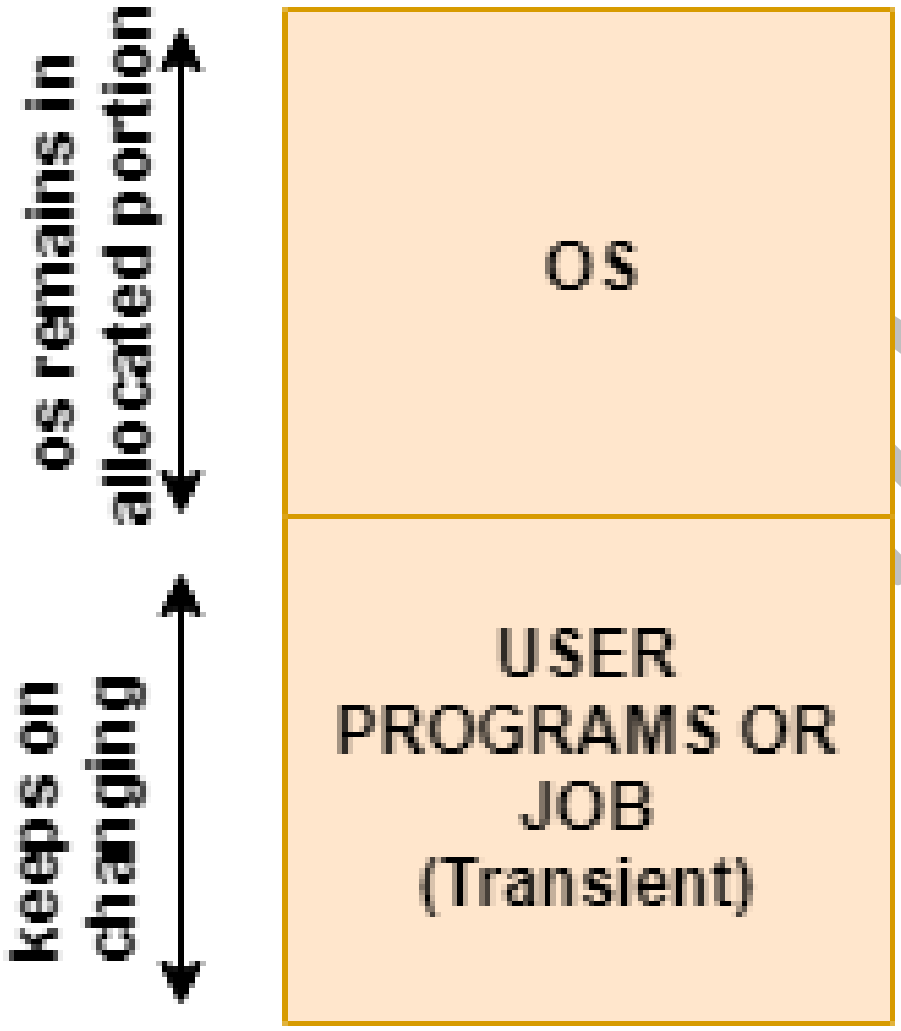
Evolution of Operating System

- Early computers were not interactive device, there user use to prepare a job which consist three parts
    - Program
    - Control information
    - Input data
- Only one job is given input at a time as there was no memory(secondary memory), computer will take the input then process it and then generate output.
- Common input/output device were punch card or tape drives.
- So, these devices were very slow, and processor remain ideal most of the time.

Batch Operating System

- To speed up the processing job with similar types (programming language) were batched together and were run through the processor as a group (batch).
- In some system grouping is done by the operator while in some systems it is performed by the 'Batch Monitor' resided in the low end of main memory)
- Then jobs (as a deck of punched cards) are bundled into batches with similar requirement.
- Then the submitted jobs were 'grouped as FORTRAN jobs, COBOL jobs etc.





Advantage

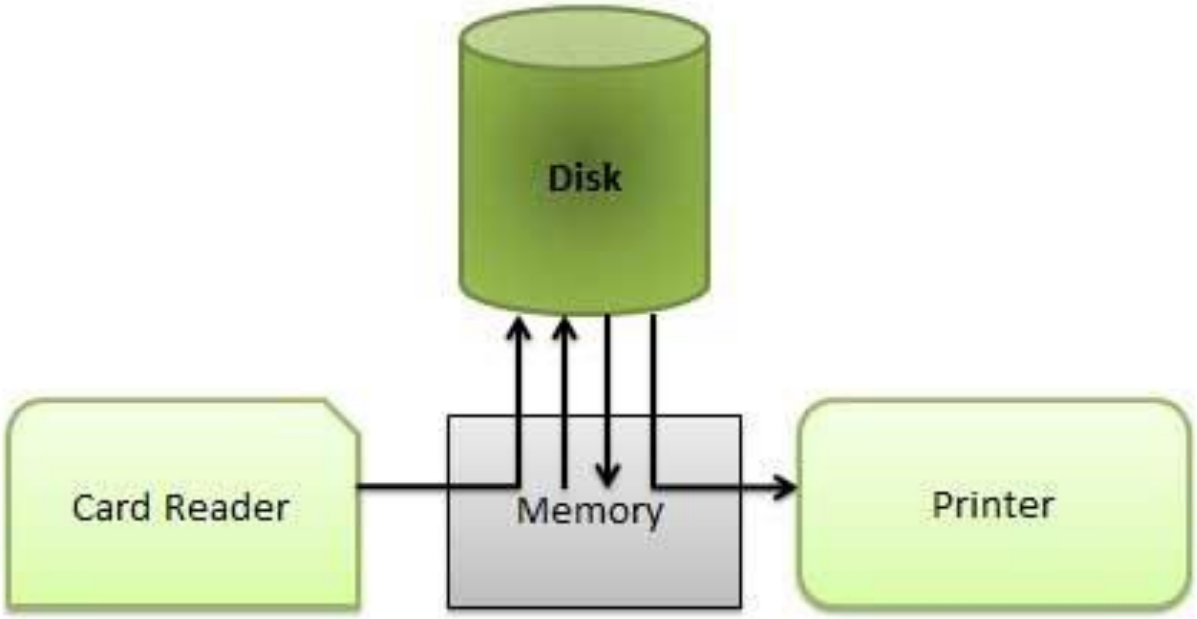
- The batched jobs were executed automatically one after another saving its time by performing the activities (like loading of compiler) only for once. It resulted in improved system utilization due to reduced turnaround time.
- Increased performance as a new job get started as soon as the previous job is finished, without any manual intervention

Disadvantage

- Memory limitation – memory was very limited, because of which interactive process or multiprogramming was not possible

Spooling

- **(**"Spool" is technically an acronym for simultaneous peripheral operations online.**)**
- In a computer system peripheral equipment, such as printers and punch card readers etc, are very slow relative to the performance of the rest of the system. Spooling is useful because devices access data at different rates.
- Spooling is a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.
- Generally, the spool is maintained on the computer's physical memory, buffers or the I/O device-specific interrupts.



- The most common implementation of spooling can be found in typical input/output devices such as the keyboard, mouse and printer. For example, in printer spooling, the documents/files that are sent to the printer are first stored in the memory. Once the printer is ready, it fetches the data and prints it.
- A spooler works by intercepting the information going to the printer, parking it temporarily on disk or in memory. The computer can send the document information to the spooler at full speed, then immediately return control of the screen to you.
- The spooler, meanwhile, hangs onto the information and feeds it to the printer at the slow speed the printer needs to get it. So if your computer can **spool,** you can work while a document is being printed.
- Even experienced a situation when suddenly for some seconds your mouse or keyboard stops working? Meanwhile, we usually click again and again here and there on the screen to check if its working or not. When it actually starts working, what and wherever we

pressed during its hang state gets executed very fast because all the instructions got stored in the respective device's spool.

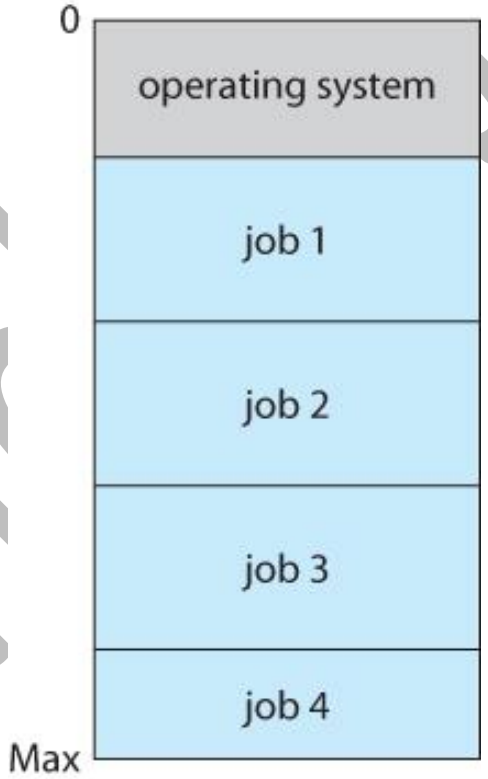
- Spooling is capable of overlapping I/O operation for one job with processor operations for another job. i.e. multiple processes can write documents to a print queue without waiting and resume with their work.

Q Which of the following is an example of a spooled device? **(GATE-1996) (1 Marks)**
(A) a line printer used to print the output of a number of jobs
(B) a terminal used to enter input data to a running program
(C) a secondary storage device in a virtual memory system
(D) a graphic display device
Answer: (A)

Q which of the following is an example of a spooled device? **(Gate-1998) (1 Marks)**
(a) The terminal used to enter the input data for the C program being executed
(b) An output device used to print the output of a number of jobs so
(c) The secondary memory device in a virtual storage system
(d) The swapping area on a disk used by the swapper
Answer (b)

Multiprogramming Operating System

- A single program cannot, in general, keep either the CPU or the I/O devices busy at all times. The basic idea of multiprogramming operating system is it keeps several jobs in main memory simultaneously.
- The jobs are kept initially on the disk in the **job pool**. This pool consists of all processes residing on disk awaiting allocation of main memory.
- The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete.
- In a non-multi-programmed system, the CPU would sit idle and wait but in a multi-programmed system, the operating system simply switches to, and executes, another job. When ***that*** job needs to wait, the CPU switches to ***another*** job, and so on.
- Eventually, the first job finishes waiting and gets the CPU back. So, conclusion is as long as at least one job needs to execute, the CPU is never idle.



- **Advantage**
  - High and efficient CPU utilization.
  - Less response time or waiting time or turnaround time
  - In most of the applications multiple tasks are running and multiprogramming systems better handle these type of applications
  - Several processes share CPU time
- **Disadvantage**
  - It is difficult to program a system because of complicated schedule handling.
  - To accommodate many jobs in main memory, complex memory management is

required.

Q Which of the following features will characterize an OS as multi-programmed OS? **(NET-DEC-2012) (Gate-2002- 1 Marks)**

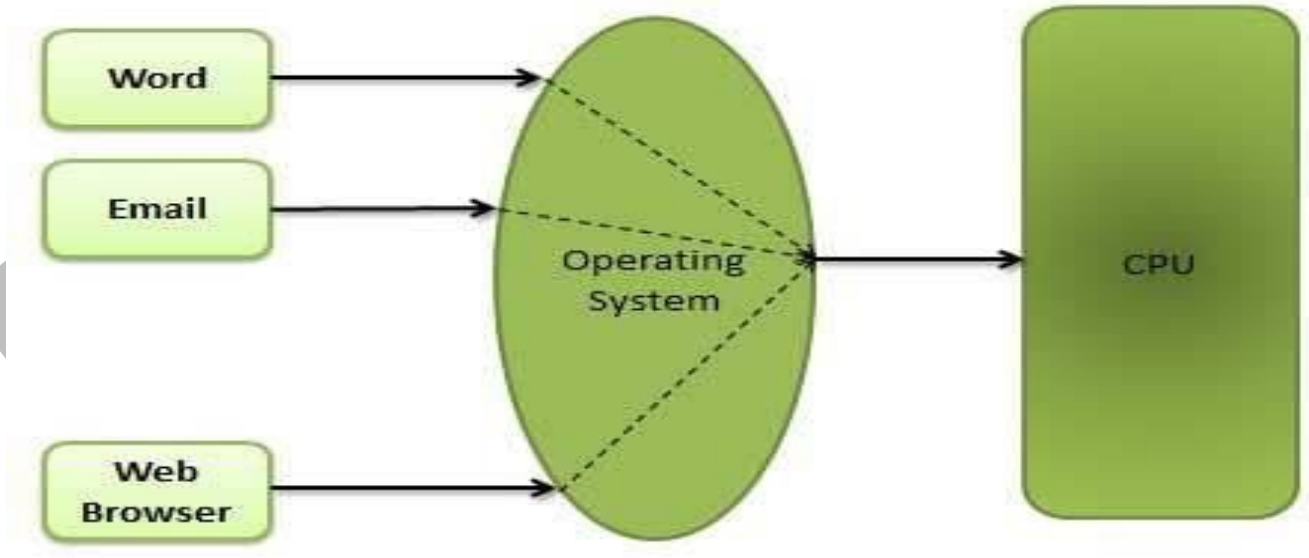
(a) More than one program may be loaded into main memory at the same time.
(b) If a program waits for certain event another program is immediately scheduled.
(c) If the execution of a program terminates, another program is immediately scheduled.

(A) (a) only
(B) (a) and (b) only
(C) (a) and (c) only
(D) (a), (b) and (c) only

Ans: d

Multitasking Operating system / Time sharing / Multiprogramming with round robin / fair share

- In the modern operating systems, we are able to play MP3 music, edit documents in Microsoft Word, surf the Google Chrome all running at the same time. (by context switching, the illusion of parallelism is achieved)
- A task in a multitasking operating system is not a whole application program but it can also refer to a "thread of execution" when one process is divided into sub-tasks.
- For multitasking to take place, firstly there should be multiprogramming i.e. presence of multiple programs ready for execution. And secondly the concept of time sharing.
- Time sharing (or multitasking) is a logical extension of multiprogramming, it allows many users to share the computer simultaneously. the CPU executes multiple jobs (May belong to different user) by switching among them, but the switches occur so frequently that, each user is given the impression that the entire computer system is dedicated to his/her use, even though it is being shared among many users.
- In multitasking, time sharing is best manifested because each running process takes only a fair quantum of the CPU time. minimal lag in overall performance and without affecting the operations of each task.



Q If you want to execute more than one program at a time, the systems software that are used must be capable of: **(NET-DEC-2005)**

(A) word processing
(B) virtual memory
(C) compiling
(D) multitasking

Answer:

Multiprocessing Operating System/ Tightly coupled system

- Multiprocessor Operating System refers to the use of two or more central processing units (CPU) within a single computer system. These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems
- So, in multiprocessing, multiple concurrent processes each can run on a separate CPU, here we achieve a true parallel execution of processes. Parallel processing is the ability of the CPU to simultaneously process incoming jobs.
- Multiprocessing becomes most important in computer system, where the complexity of the job is more, and CPU divides and conquers the jobs. Generally, the parallel processing is used in the fields like artificial intelligence and expert system, image processing, weather forecasting etc.
- **Multiprocessing can be of two types**
    - Symmetric multiprocessing - In a symmetric multi-processing, a single OS instance controls two or more identical processors connected to a single shared main memory. Most of the multi-processing PC motherboards utilize symmetric multiprocessing. Here each processor runs an identical copy of operating system and these copies communicate with each other. SMP means that all processors are peers; no boss–worker relationship exists between processors. Windows, Mac OSX and Linux.
    - Asymmetric - This scheme defines a master-slave relationship, where one processor behaves as a master and control other processor which behaves as slaves. For e.g. It may require that only one particular CPU respond to all hardware interrupts, whereas all other work in the system may be distributed equally among CPUs; or execution of kernel-mode code may be restricted to only one particular CPU, whereas user-mode code may be executed in any combination of processors. Multiprocessing systems are often easier to design if such restrictions are imposed, but they tend to be less efficient than systems in which all CPUs are utilized. A boss processor controls the system, other processors either look to the boss for instruction or have predefined tasks. This scheme defines a boss–worker relationship. The boss processor schedules and allocates work to the worker processors.
- **Advantage of multiprocessing**
    - Increased Throughput - By increasing the number of processors, we expect to get more work done in less time. The speed-up ratio with N processors is not N, however; rather, it is less than N. When

multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly. This overhead, plus contention for shared resources, lowers the expected gain from additional processors. These types of systems are used when very high speed is required to process a large volume of data.

- Economy of Scale - Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. If several programs operate on the same set of data, it is cheaper to store those data on one disk and to have all the processors share them than to have many computers with local disks and many copies of the data.
- Increased Reliability (fault tolerance) - If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fail, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether
- Increased reliability of a computer system is crucial in many applications. The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation. Some systems go beyond graceful degradation and are called fault tolerant, because they can suffer a failure of any single component and still continue operation.

- **Disadvantages**
  - It's more complex than simple operating systems.
  - It requires context switching which may impacts performance.
  - If one processor fails then it will affect in the speed
  - large main memory required

Q In which of the following, ready to execute processes must be present in RAM? **(NET-DEC-2008)**

(A) multiprocessing
(B) multiprogramming
(C) multitasking
(D) all of the above

Answer: D

Real time Operating system

- A real time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail. Very fast and quick respondent systems. Valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.
- RTOS intended to serve real-time applications that process data as it comes in, typically without buffer delays. Are used in an environment where a large number of events (generally external) must be accepted and processed in a short time.
- Little swapping of programs between primary and secondary memory. Most of the time, processes remain in primary memory in order to provide quick response, therefore, memory management in real time system is less demanding compared to other systems.
- For example, a measurement from a petroleum refinery indicating that temperature is getting too high and might demand for immediate attention to avoid an explosion. Airlines reservation system, Air traffic control system, Systems that provide immediate updating, Systems that provide up to the minute information on stock prices, Defense application systems like as RADAR.
- There are three different types of RTOS which are following
    - **Soft real-time operating system** - The soft real-time operating system has certain deadlines, may be missed and they will take the action at a time t=0+. The soft real-time operating system is a type of OS and it does not contain constrained to extreme rules. The critical time of this operating system is delayed to some extent. The examples of this operating system are the digital camera, mobile phones and online data etc.
    - **Hard real-time operating system** - This is also a type of OS and it is predicted by a deadline. The predicted deadlines will react at a time t = 0. Some examples of this operating system are air bag control in cars, anti-lock brake, and engine control system etc.

Distributed OS

- A distributed system is a collection of processors or loosely coupled nodes that do not share memory or a clock. Instead, each node has its own local memory. The nodes communicate with one another through various networks, such as high-speed buses and the Internet.
- The nodes in a distributed system may vary in size and function. They may include small microprocessors, personal computers, and large general-purpose computer systems.
- There are four major reasons for building distributed systems: resource sharing, computation speedup, reliability, and communication.
- Resource Sharing
    - If a number of different sites (with different capabilities) are connected to one another, then a user at one site may be able to use the resources available at another. In general, resource sharing in a distributed system provides mechanisms for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, using remote specialized hardware devices (such as a supercomputer), and performing other operations.
- Computation Speedup
    - If a particular computation can be partitioned into sub computations that can run concurrently, then a distributed system allows us to distribute the sub computations among the various sites. The sub computations can be run concurrently and thus provide computation speedup. In addition, if a particular site is currently overloaded with jobs, some of them can be moved to other, lightly loaded sites. This movement of jobs is called load sharing or job migration. Automated load sharing, in which the distributed operating system automatically moves jobs, is not yet common in commercial systems.
- Reliability
    - If one site fails in a distributed system, the remaining sites can continue operating, giving the system better reliability. If the system is composed of multiple large autonomous installations (that is, general-purpose computers), the failure of one of them should not affect the rest. If, however, the system is composed of small machines, each of which is responsible for some crucial system function (such as the web server or the file system), then a single failure may halt the operation of the whole system. In general, with enough redundancy (in both hardware and data), the system can continue operation, even if some of its sites have failed. The failure of a

site must be detected by the system, and appropriate action may be needed to recover from the failure. The system must no longer use the services of that site. In addition, if the function of the failed site can be taken over by another site, the system must ensure that the transfer of function occurs correctly. Finally, when the failed site recovers or is repaired, mechanisms must be available to integrate it back into the system smoothly.

- Communication
    - When several sites are connected to one another by a communication network, users at the various sites have the opportunity to exchange information. Such functions include file transfer, login, mail, and remote procedure calls (RPCs).
- The advantage of a distributed system is that these functions can be carried out over great distances. Two people at geographically distant sites can collaborate on a project, for example. By transferring the files of the project, logging in to each other's remote systems to run programs, and exchanging mail to coordinate the work, users minimize the limitations inherent in long- distance work.
- The advantages of distributed systems have resulted in an industry-wide trend toward downsizing. Many companies are replacing their mainframes with networks of workstations or personal computers. Companies get a bigger bang for the buck (that is, better functionality for the cost), more flexibility in locating resources and expanding facilities, better user interfaces, and easier maintenance.

Q Match the following: **(NET-June-2015)**

List - I	**List - II**
(a) Spooling	**(i)** Allows several jobs in memory to improve CPU utilization
(b) Multiprogramming	**(ii)** Access to shared resources among geographically dispersed computers in a transparent way
(c) Time sharing	**(iii)** Overlapping I/O and computations
(d) Distributed computing	**(iv)** Allows many users to share a computer simultaneously by switching processor frequently

codes:

	(a)	**(b)**	**(c)**	**(d)**
1)	iii	i	ii	iv
2)	iii	i	iv	ii
3)	iv	iii	ii	i
4)	ii	iii	iv	i

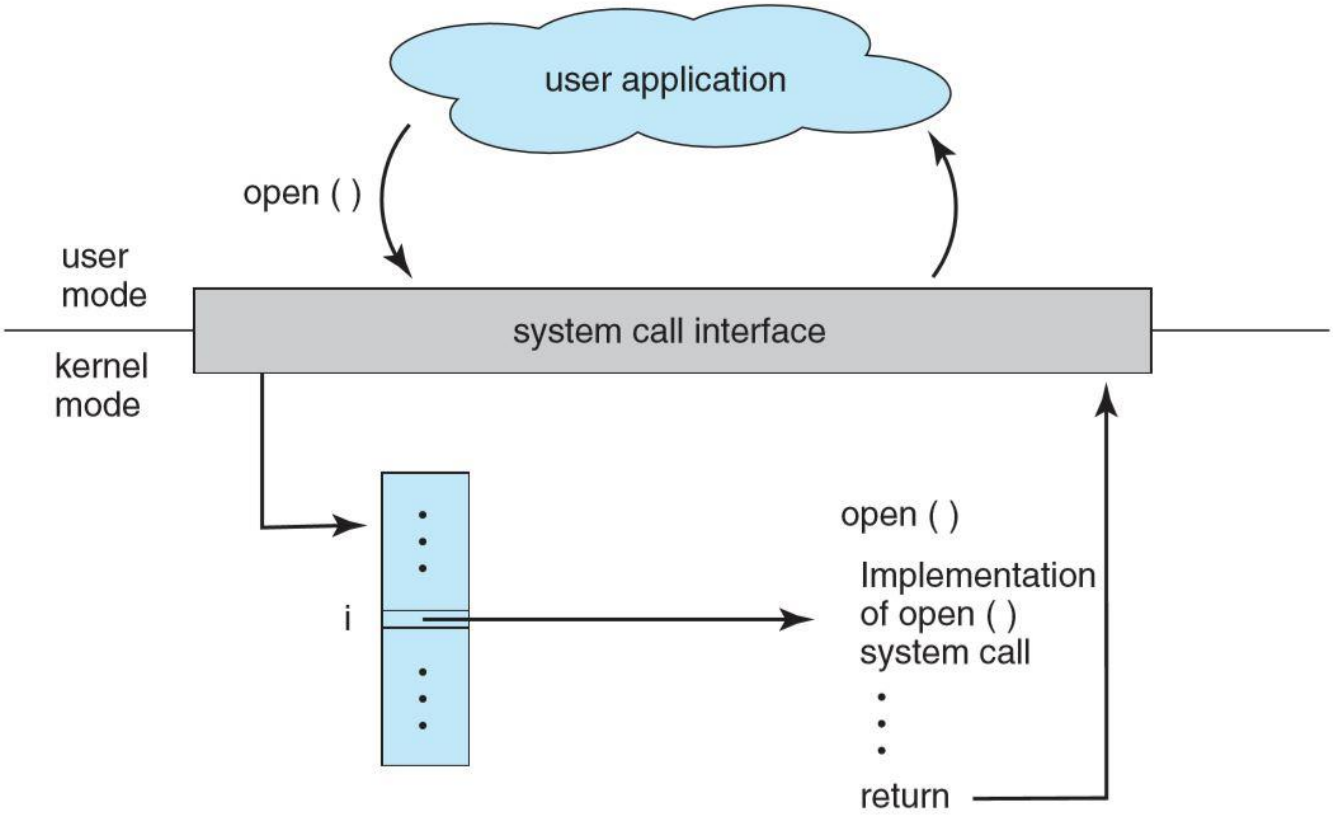
Ans: 2

User and Operating-System Interface

- There are several ways for users to interface with the operating system. Here, we discuss two fundamental approaches.
- One provides a command-line interface, or command interpreter, that allows users to directly enter commands to be performed by the operating system.
- The other allows users to interface with the operating system via a graphical user interface, or GUI.

System call

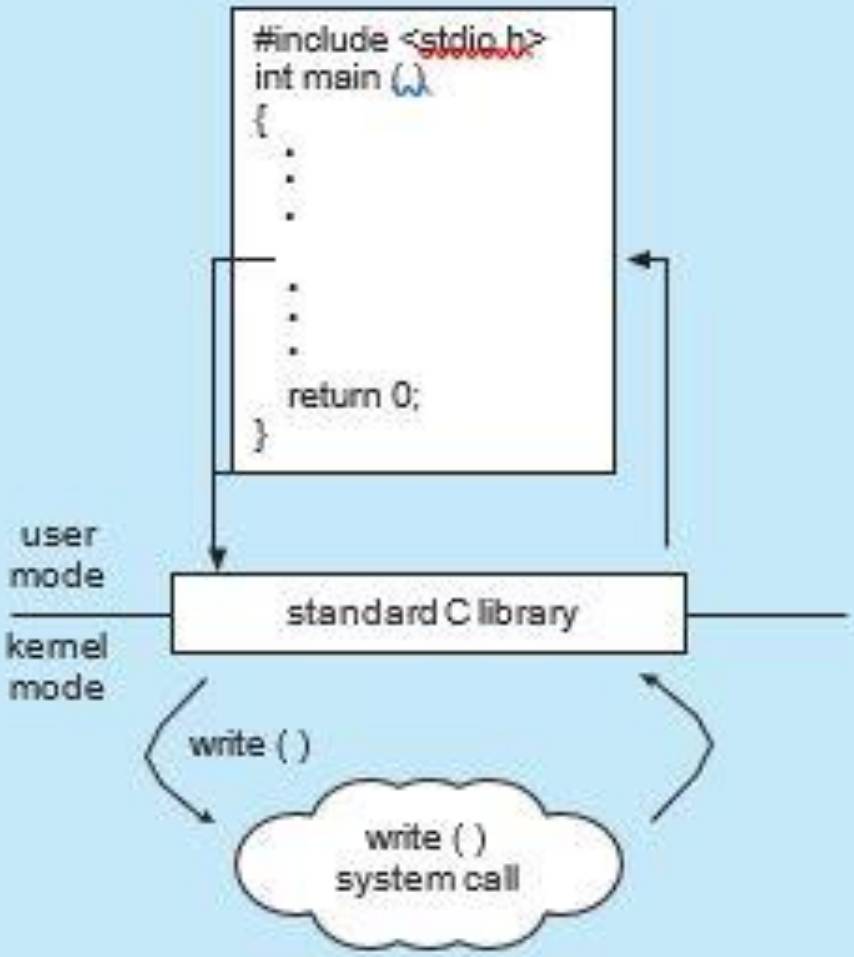
- System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf.
- System calls provide an interface to the services made available by an operating system. These calls are generally available as routines written in C and C++, although certain low-level tasks (for example, tasks where hardware must be accessed directly) may have to be written using assembly-language instructions.
- The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect. Three of the most common APIs available to application programmers are the Windows API for Windows systems, the POSIX API for POSIX-based systems (which include virtually all versions of UNIX, Linux, and Mac OS X), and the Java API for programs that run on the Java virtual machine.



- The caller need know nothing about how the system call is implemented or what it does during execution. Rather, the caller need only obey the API and understand what the operating system will do as a result of the execution of that system call.

EXAMPLE OF STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the printf( ) statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the write( ) system call. The C library takes the value returned by write( ) and passes it back to the user program. This is shown below:



.

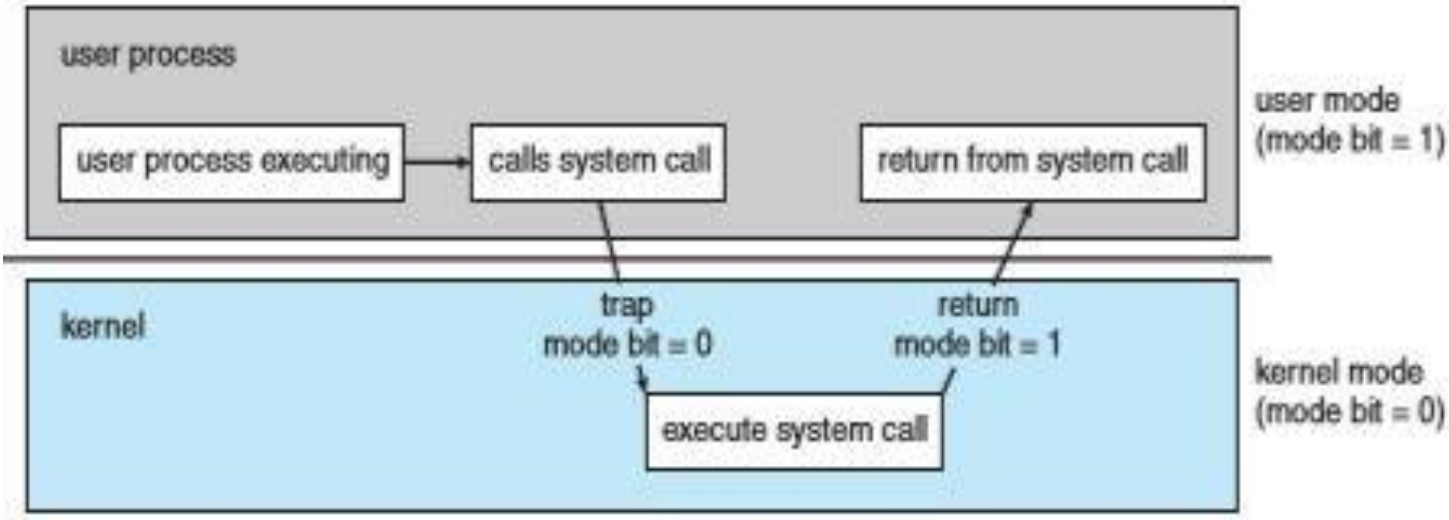
Types of System Calls - System calls can be grouped roughly into six major categories: process control, file manipulation, device manipulation, information maintenance, communications, and protection.

- Process control
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information

Mode bit

- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user- defined code.
- At the very least, we need two separate *modes* of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode). A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the operating system (via a system call), the system must transition from user to kernel mode to fulfill the request.

- At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0).
- The hardware allows privileged instructions to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system. The instruction to switch to kernel mode is an example of a privileged instruction. Some other examples include I/O control, timer management, and interrupt management.



User and Operating-System Interface

- there are several ways for users to interface with the operating system. Here, we discuss two fundamental approaches. One provides a command-line interface, or command interpreter, that allows users to directly enter commands to be performed by the operating system. The other allows users to interface with the operating system via a graphical user interface, or GUI.

- Command Interpreters - Some operating systems include the command interpreter in the kernel. Others, such as Windows and UNIX, treat the command interpreter as a special program that is running when a job is initiated or when a user first logs on (on interactive systems). On systems with multiple command interpreters to choose from, the interpreters are known as shells. For example, on UNIX and Linux systems, a user may choose among several different shells, including the *Bourne shell*, *C shell*, *Bourne-Again shell*, *Korn shell*, and others.

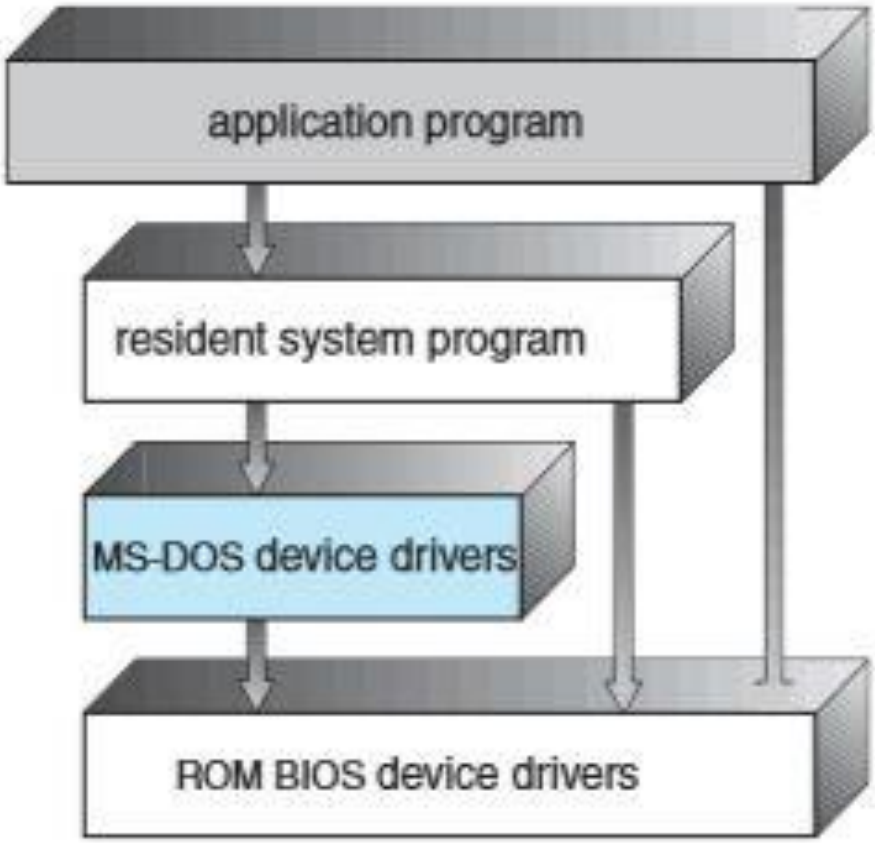
- Graphical User Interfaces - A second strategy for interfacing with the operating system is through a user- friendly graphical user interface, or GUI. Here, users employ a mouse-based window- and-menu system characterized by a desktop. The user moves the mouse to position its pointer on images, or icons, on the screen (the desktop) that represent programs, files, directories, and system functions. Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory—known as a folder —or pull down a menu that contains commands.

- Because a mouse is impractical for most mobile systems, smartphones and handheld tablet computers typically use a touchscreen interface. Here, users interact by making gestures on the touchscreen—for example, pressing and swiping fingers across the screen.

- The choice of whether to use a command-line or GUI interface is mostly one of personal preference. System administrators who manage computers and power users who have deep knowledge of a system frequently use the command-line interface. For them, it is more efficient, giving them faster access to the activities they need to perform. Indeed, on some systems, only a subset of system functions is available via the GUI, leaving the less common tasks to those who are command-line knowledgeable.

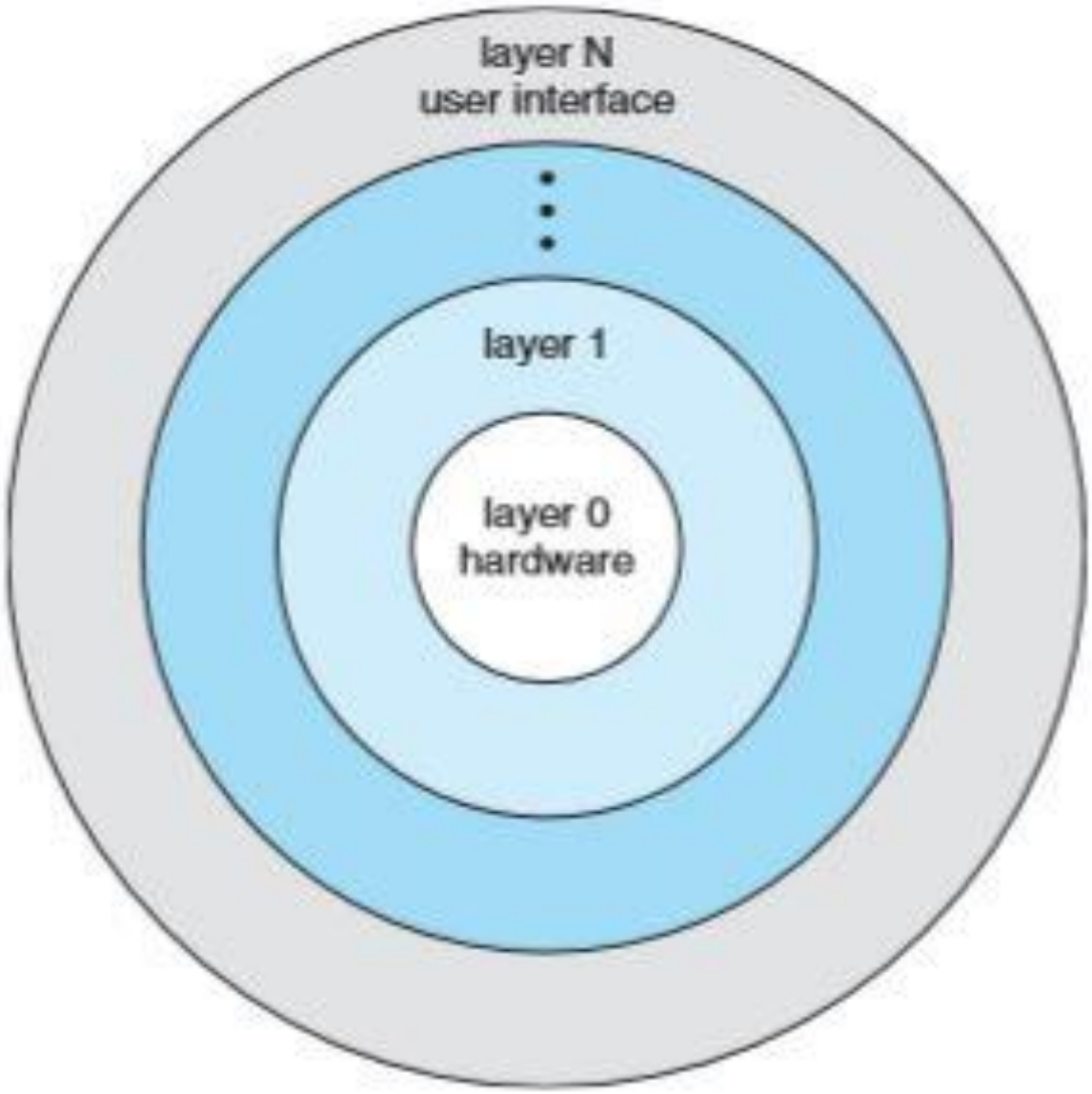
Structure of Operating System

- A common approach is to partition the task into small components, or modules, rather than have one monolithic system. Each of these modules should be a well-defined portion of the system, with carefully defined inputs, outputs, and functions.
- Simple Structure - Many operating systems do not have well-defined structures. Frequently, such systems started as small, simple, and limited systems and then grew beyond their original scope. MS-DOS is an example of such a system.



MS-DOS layer structure.

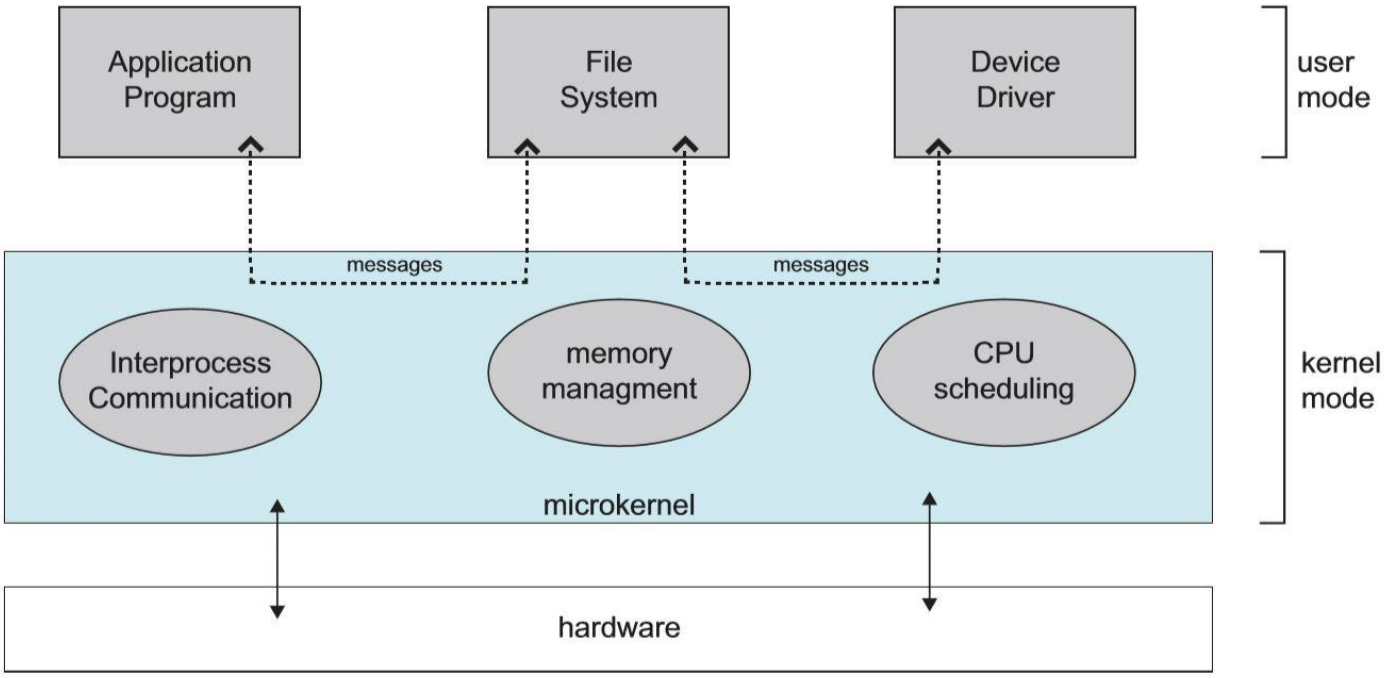
- Layered Approach - With proper hardware support, operating systems can be broken into pieces. The operating system can then retain much greater control over the computer and over the applications that make use of that computer. Implementers have more freedom in changing the inner workings of the system and in creating modular operating systems.
- Under a top-down approach, the overall functionality and features are determined and are separated into components. Information hiding is also important, because it leaves programmers free to implement the low-level routines as they see fit.
- A system can be made modular in many ways. One method is the layered approach, in which the operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer *N*) is the user interface.



A layered operating system.

Micro-Kernel approach

In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel** approach. This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel.



One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel. When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.