Operating System

Subject code : CSC 404

Subject In-charge

Nidhi Gaur
Assistant Professor
email: nidhigaur@sfit.ac.in

Operating System Overview

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware

OPERATING SYSTEMS

- The operating system controls all the software programs and allows hardware to work properly.

Types of operating systems:

Microsoft Windows

Mac OS

Linux

Handheld OS (PDA's MP3's, Cell Phones, etc.

MS-DOS = text interface ("old school" way of input)

GUI = graphical user interface that allows for point and click

Name 6 types of application software and describe what each one allows you to do.

What types of software program controls all the other software on your computer?

Operating System Objectives

- **C**onvenience
  - Makes the computer more convenient to use
- **E**fficiency
  - Allows computer system resources to be used in an efficient manner
- **A**bility to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with service

Layers of Computer System

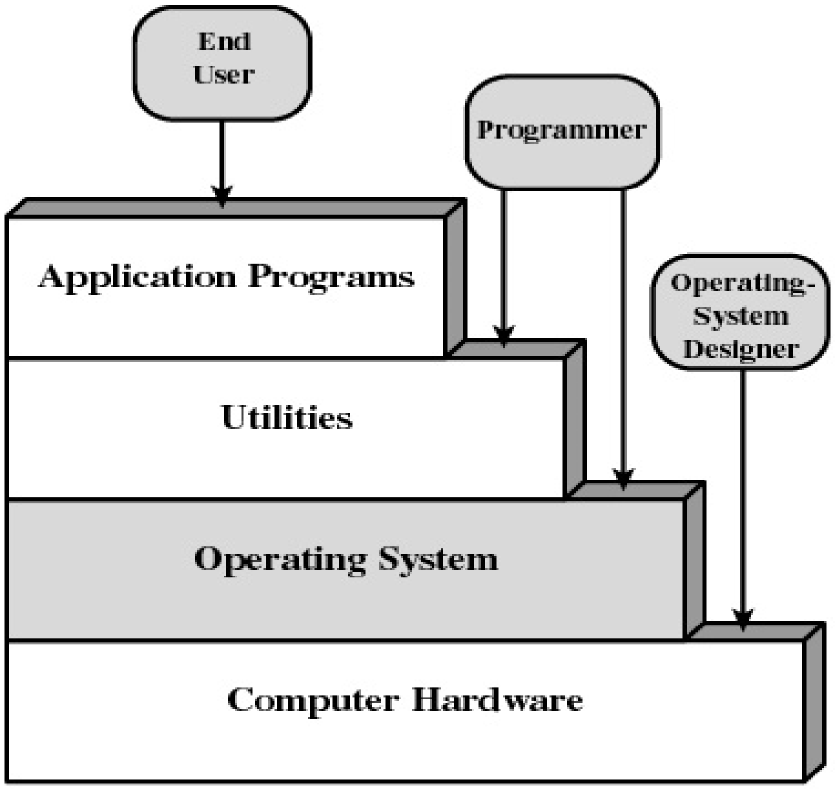


Figure 2.1 Layers and Views of a Computer System

OS as a user/computer interface

- H/w & S/w used in providing applications to user can be viewed in layered fashion.
- End user is not concerned with details of computer H/w.
- End user views computer in terms of a set of applications
- If one has to develop an application program, a set of **system program** is provided, some of these are **called utilities**– these implement functions that assist in program creation, management of files and control of I/O devices.

Example: **antivirus software, backup software, text editor.**

OS as a user/computer interface

- **Most important system program is O.S.**
- **O.S. masks the details of hardware from programmer with convenient interface for using system.**

Services Provided by the Operating System

- Program development
  - Editors and debuggers to assist programmer in creating programs.
- Program execution

  -- instruction and data must be loaded in main memory, I/O devices & files must be initialized.
- Access to I/O devices

  -- each I/O device requires its own peculiar set of instructions, control signals.

OS provides a uniform interface that hides these details.

- Controlled access to files

  -- protection mechanism in case of multiple users.
- System access

  --controlled access to system and system resources

Services Provided by the Operating System

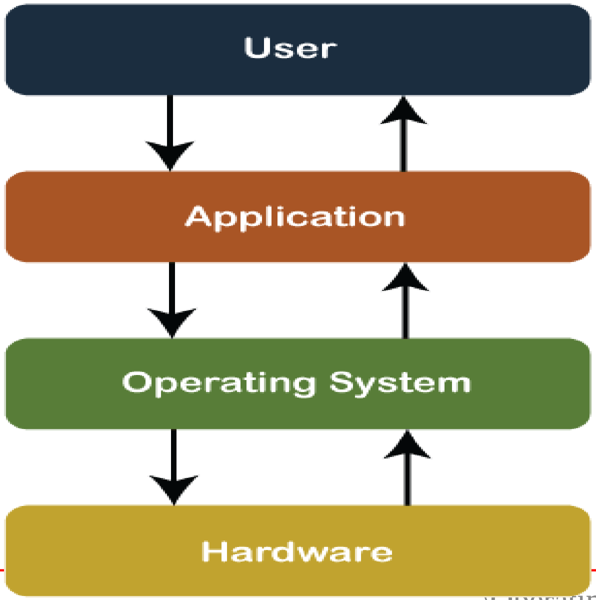
- Error detection and response
  - internal and external hardware errors
    - memory error
    - device failure
  - software errors
    - arithmetic overflow

Services Provided by the Operating System

- Accounting
  - collect statistics of usage of resources
  - monitor performance
  - used to anticipate future enhancements
  - used for billing users

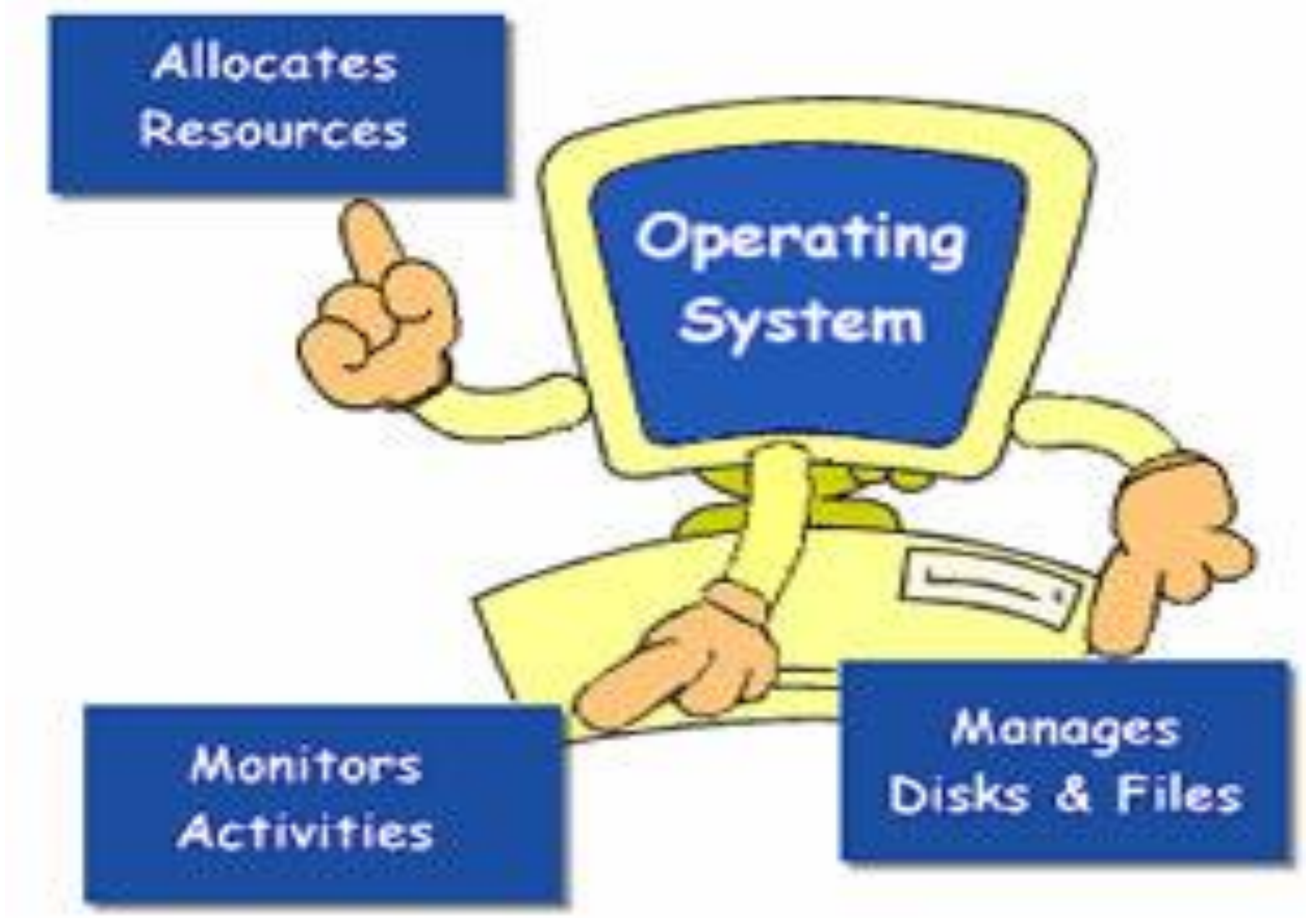
OS as a resource manager

- A computer is a set of resources.
- OS is responsible for managing resources.
- OS controls the storage and processing.
- OS directs the processor in the use of other system resources & in timing of its execution of other programs.



OS as a resource manager

- An operating is a software program that manages and controls the execution of application programs, software resources and computer hardware.
- It also helps manage the software/hardware resource, such as file management, memory management, input/ output and many peripheral devices like a disk drive, printers, etc.
- Processor stops executing OS in order to execute other programs and then gives the control back to the OS.


Allocates Resources
Operating System
Monitors Activities
Manages Disks & Files

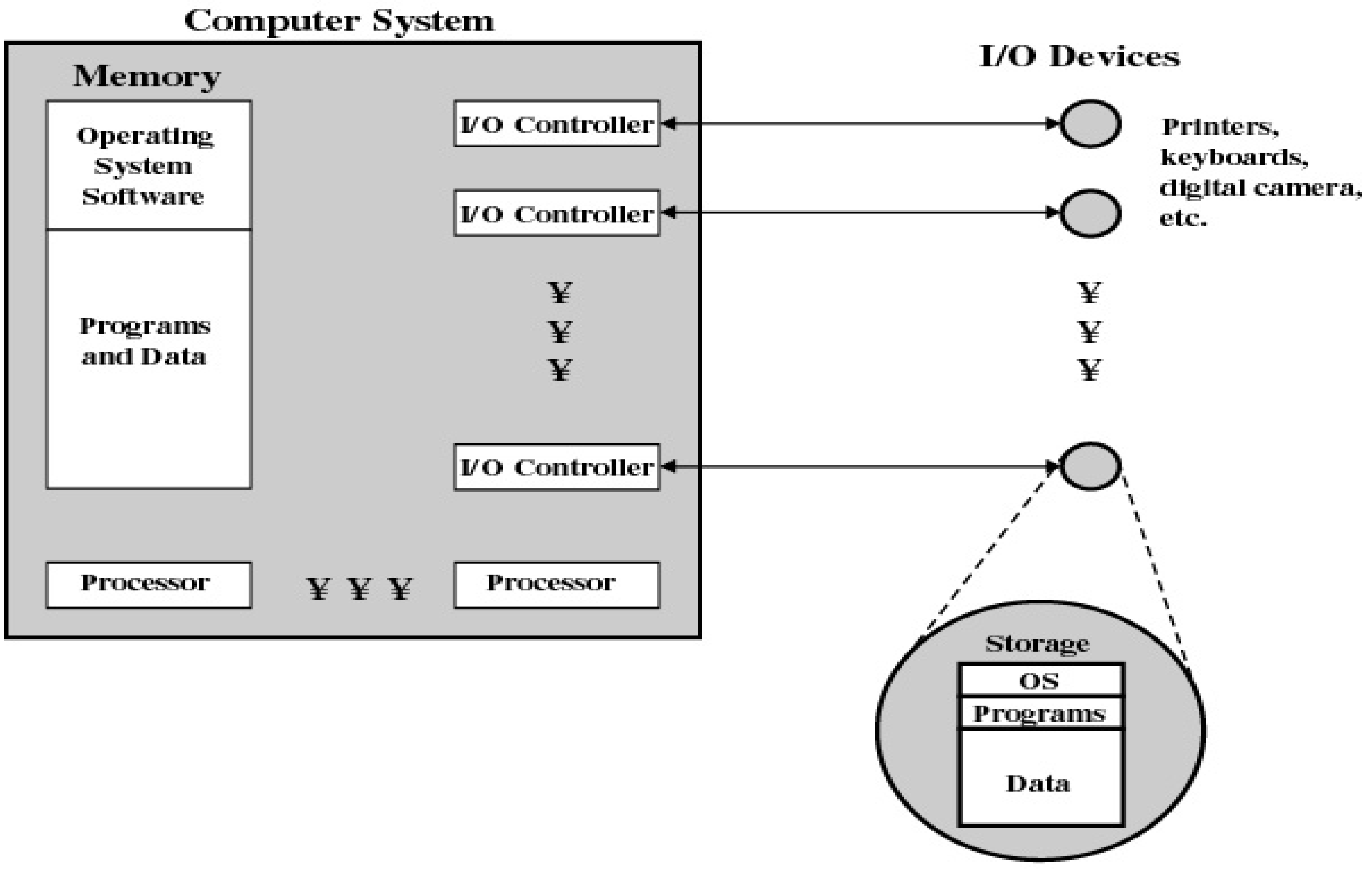


Figure 2.2 The Operating System as Resource Manager

Evolution of Operating Systems

- Early Systems (1950)
- Simple Batch Systems (1960)
- Multiprogrammed Systems (1970)
- Time-Sharing and Real-Time Systems (1970)
- Personal/Desktop Computers (1980)
- Multiprocessor Systems (1980)
- Networked/Distributed Systems (1980)
- Web-based Systems (1990)

Evolution of OS

Serial processing

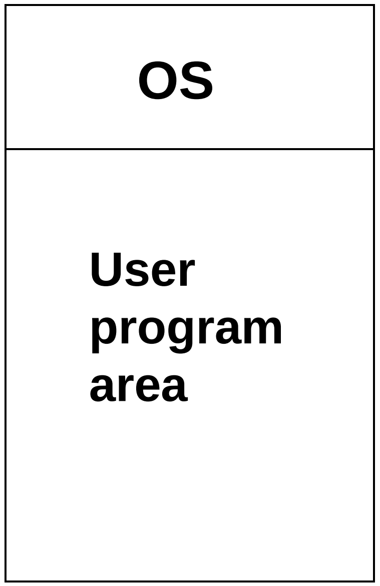
- The earliest computers from late 1940s to mid 1950s had no operating systems.
- The programmer interacted directly with hardware.
- Programs were loaded in machine code.
- Machines were run from console.
- Output appear on printer.

Types of OS

- In **Batch processing** same type of jobs batch *(BATCH- a set of jobs with similar needs)* together and execute at a time.
- The OS was simple, its major task was to transfer control from one job to the next.
- The job was submitted to the computer operator in form of punch cards. At some later time the output appeared.
- Common Input devices were card readers and tape drives.

Batch Processing (Contd…):

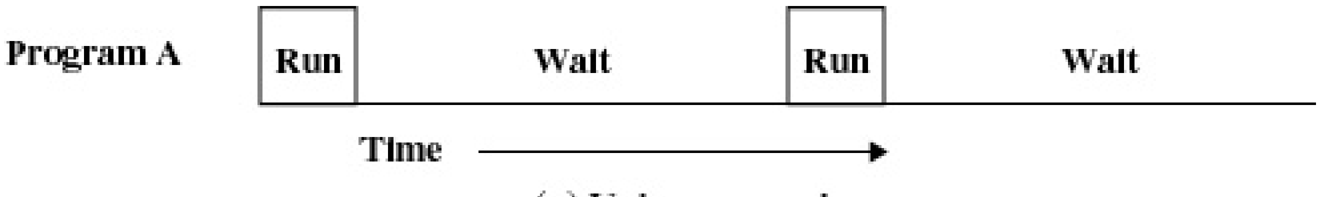
- Users did not interact directly with the computer systems, but prepared a job (comprising of the program, the data, & some control information).



Types of OS

Uniprogramming

- Processor must wait for I/O instruction to complete before proceeding

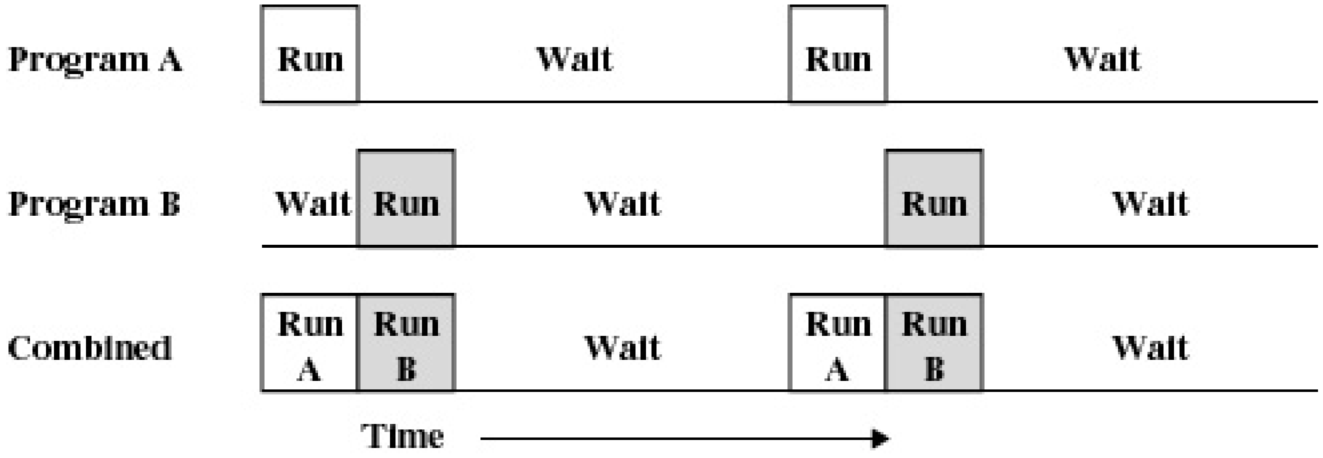


(a) Uniprogramming

Types of OS

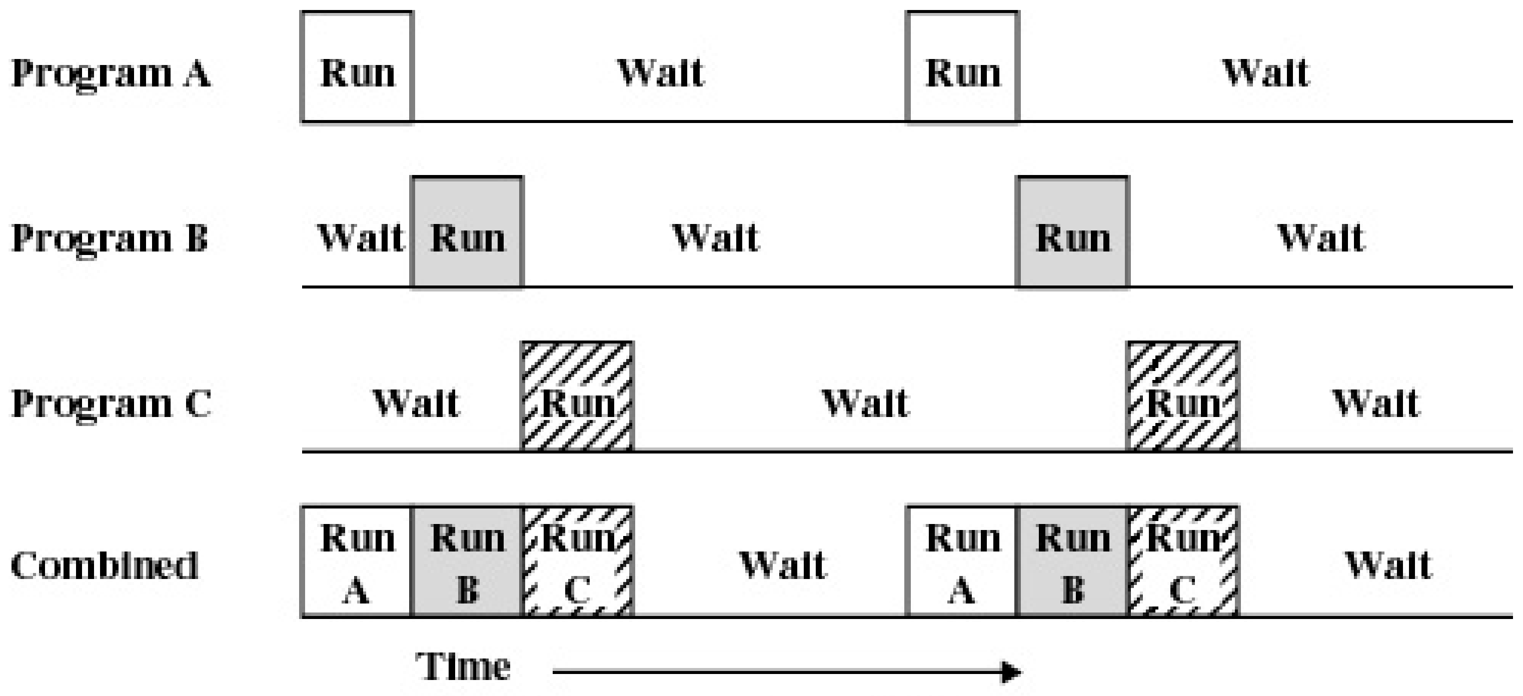
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

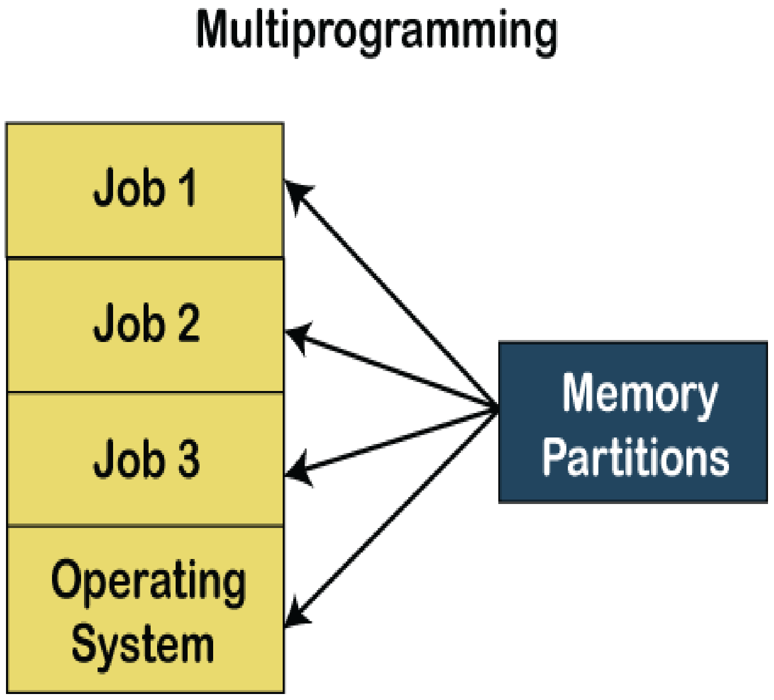
Multiprogramming



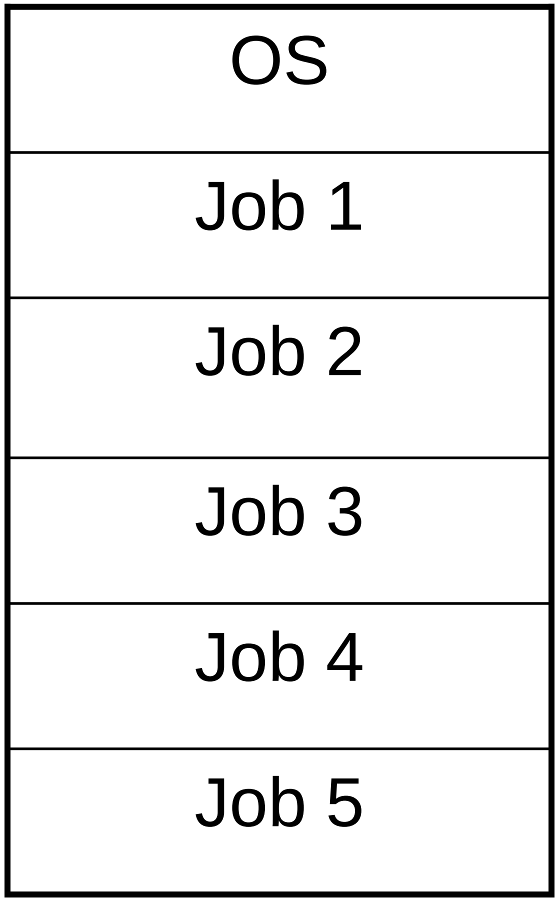
(c) Multiprogramming with three programs

Multiprogramming:

- Multiprogramming is a technique to execute number of programs simultaneously by a single processor.
- In Multiprogramming, number of processes reside in main memory at a time.
- The OS picks and begins to executes one of the jobs in the main memory.
- If any I/O wait happened in a process, then CPU switches from that job to another job.
- Hence CPU in not idle at any time.



Multiprogramming (Contd…):



- Figure shows the layout of multiprogramming system.
- The main memory consists of 5 jobs at a time, the CPU executes one by one.

Advantages:

- Efficient memory utilization
- Throughput increases
- CPU is never idle, so performance increases.

Types of OS

- **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

Types of OS:

Operating System can also be classified as,-

- **Single User Systems**
- **Multi User Systems**

Single User Systems:

- Provides a platform for only one user at a time.
- They are popularly associated with Desktop operating system which run on standalone systems where no user accounts are required.
- Example: DOS, Windows 95

Multi-User Systems:

- Provides regulated access for a number of users by maintaining a database of known users.
- Refers to computer systems that support two or more simultaneous users.
- Another term for *multi-user* is *time sharing*.
- Ex: All mainframes   are multi-user systems.

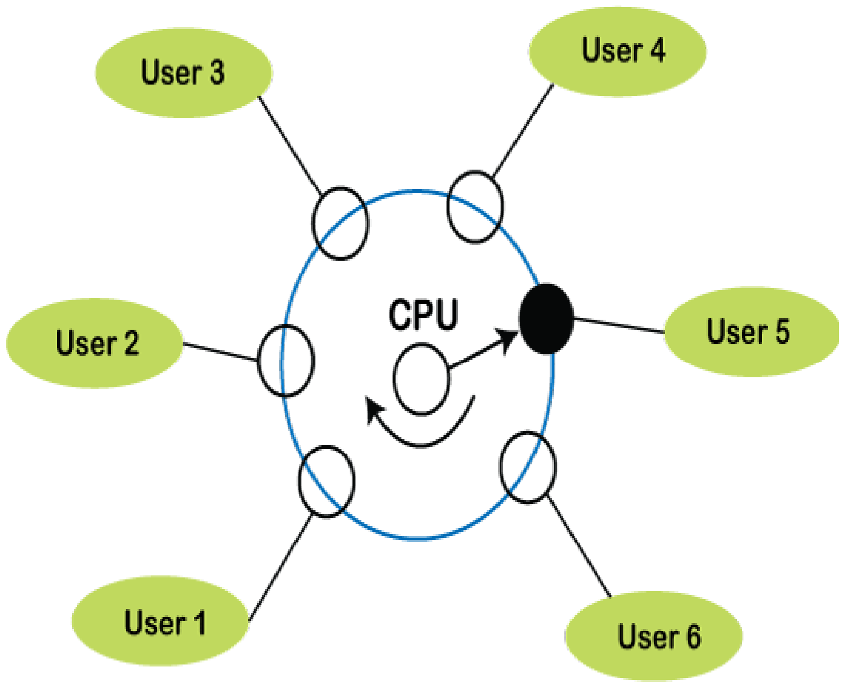
Example: Unix, Linux

Types of OS

- **Timesharing** 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

Time Sharing

- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals.



Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	**Time Sharing**
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Types of OS

Multiprocessor OS

- **Symmetric multiprocessing(SMP):--**

✓there are multiple processors

✓Share the same main memory, I/O facilities, interconnected by bus

✓All processors can perform the same functions, hence symmetric.

Advantage of SMP over uniprocessor:

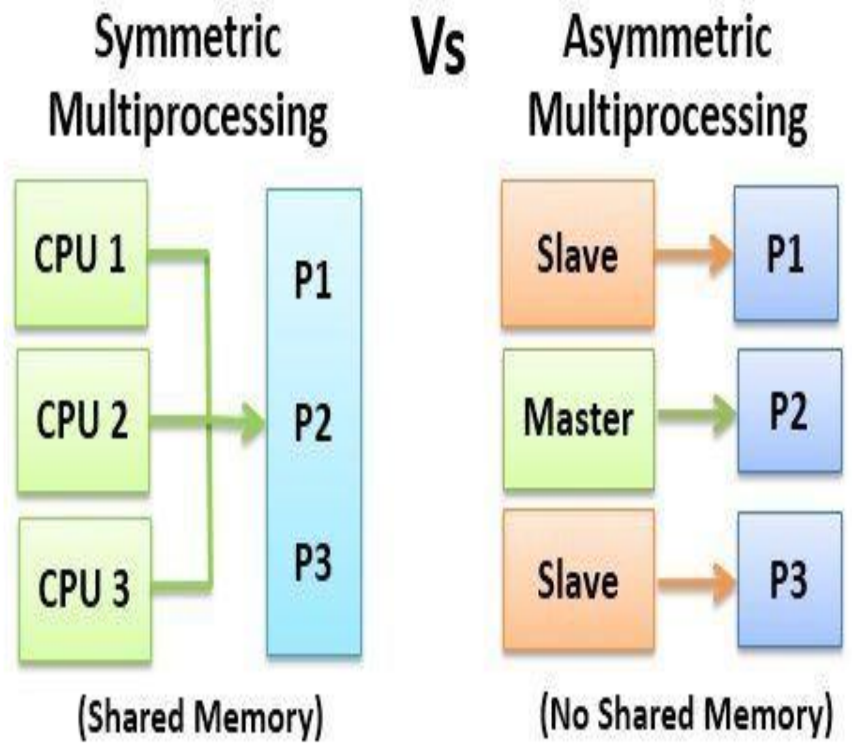
Performance: parallel

Availability: fault tolerance

Incremental growth : additional processor

Scaling: range of products by vendor

- **Asymmetric multiprocessor(ASMP)**



Which one of the following error will be handled by the operating system?

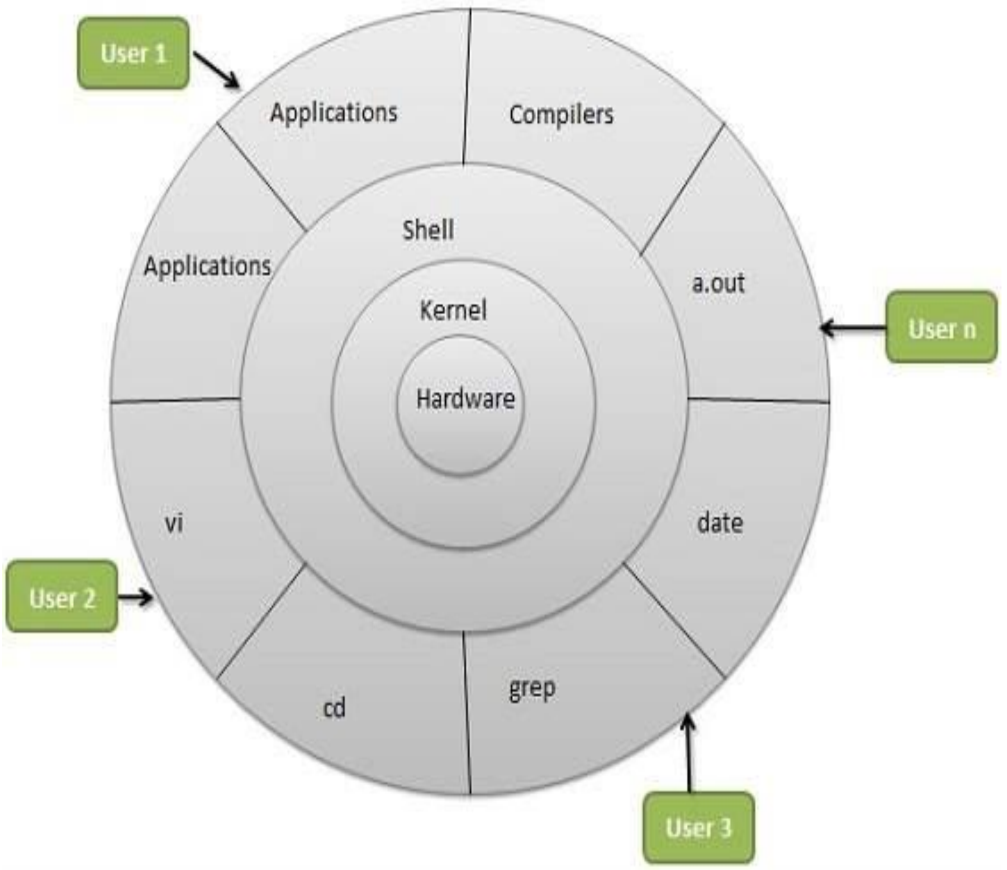
a) power failure
b) lack of paper in printer
c) connection failure in the network
d) all of the mentioned

Answer: d

Shell, Terminal and Kernel

LINUX

- Hardware is surrounded by the operating-system
- Operating system consist of kernel
- Comes with a number of user services and interfaces
  - shell
  - C compiler



Shell

- The **shell** is the outermost layer of the **operating system**.
- Interface to the **operating system** is called a **shell**.
- The shell is a program that takes commands from the keyboard and gives them to the operating system to perform

Shell

Shells incorporate a programming language to control processes and files, as well as to start and control other programs.

A shell manages the user–system interaction by prompting users for input, interpreting their input, and then handling an output from the underlying operating system

Shell

- On most Linux systems a program called **bash** (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, **sh**, written by Steve Bourne) acts as the shell program.
- Besides **bash**, there are other shell programs available for Linux systems. These include: **ksh**, **tcsh** and **zsh**.

What's a "Terminal?"

- It's a program called a *terminal emulator*. This is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators we can use. These might include

gnome-terminal, **konsole**, **xterm**

User Operating System Interface

- OS shells use either a command-line interface(CLI) or graphical user interface (GUI)
- Command Line Interface (CLI) or **command interpreter** allows direct command entry
    - Primarily fetches a command from user and executes it

Kernel

- Portion of operating system that is in main memory
- Contains most-frequently used functions
- Also called the **nucleus**
- “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program or an application program.

Kernel

The main functions of the Kernel are the following:

- Manage RAM memory, so that all programs and running processes can work.
- Manage the processor time, which is used by running processes.
- Manage access and use of the different peripherals connected to the computer.

User mode vs kernel mode

- A processor in a computer running Windows has two different modes: **user mode and kernel mode.**
- The processor switches between the two modes depending on what type of code is **running** on the processor.
- **Applications** run in user mode, and core operating system components run in kernel mode
- In **Kernel mode**, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address.
- Kernel mode is also known as **privileged mode or master mode**

User mode vs kernel mode

- User programs and kernel functions are executed in their respective spaces allotted in main memory.
- But it is obvious that user mode programs need to execute some privileged operations.
- The processor prevents the direct access to kernel mode functions, user mode programs must use an interface.
- This interface is called **system call**

System calls and their types

System Call

- A **system call** is a way for programs to interact with the operating system.
- A computer program makes a **system call** when it makes a request to the operating **system's** kernel.
- **System call** provides the services of the operating system to the user programs via Application Program Interface(API).

System call example

Copy contents from source to destination file.

Source file
Destination file

System call example

- Writing a simple program to read data from one file and to copy contents to another file. The first input is name of files : input and output file. (interactive or mouse based)
- Once two file names are obtained, the program must open the input file and create the output file. It requires system calls.
- There are possible error conditions for operations.(input file doesn’t exist, protected against access)
- If input file exist create an output file.
- If output file already exist, ask to replace or terminate the program.

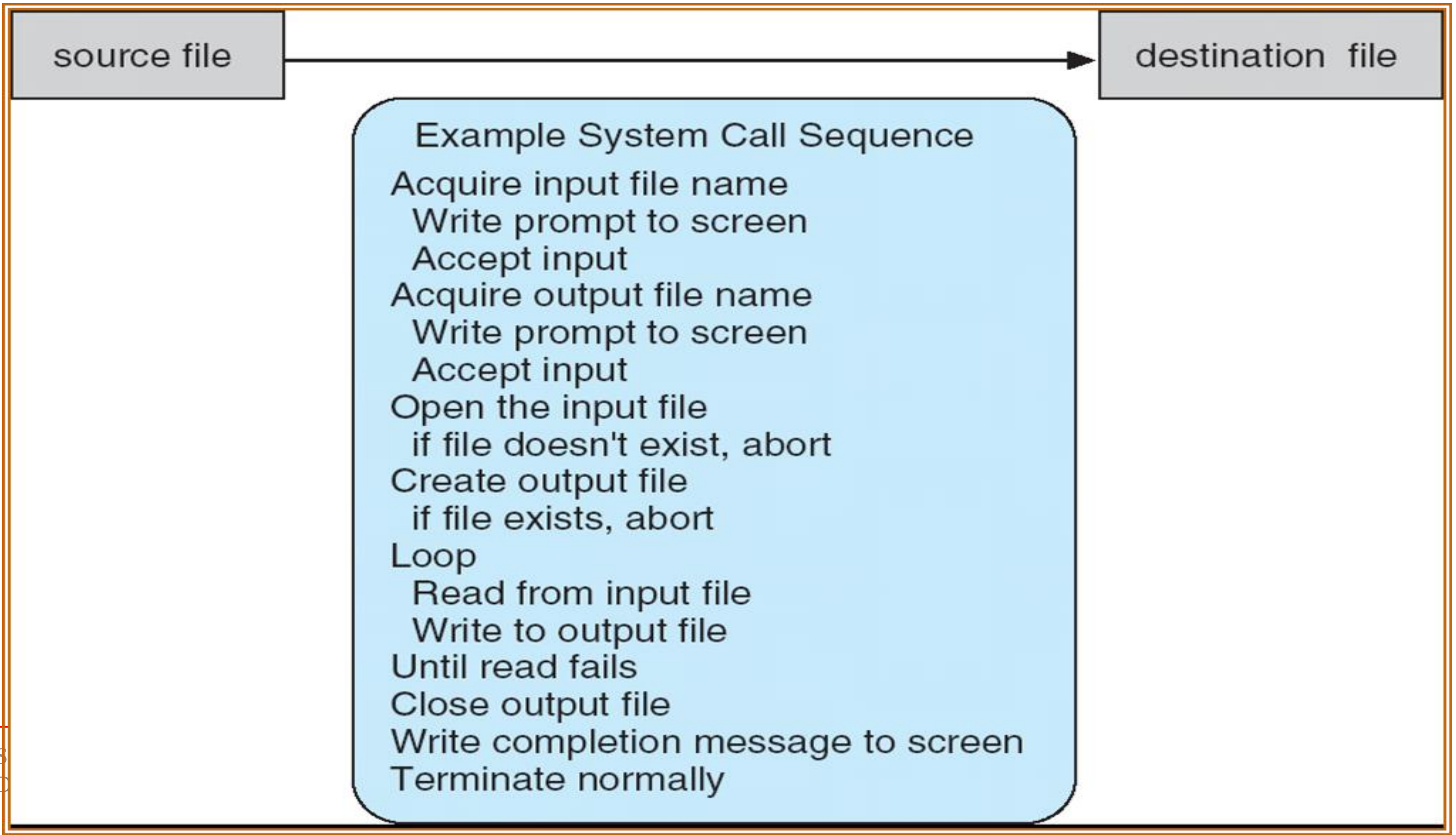
System call example

- When both files are setup we enter a loop that reads from input file(system call) and writes to output file(system call).
- On input the program may find that end of file has reached or some hardware failure in read.
- The write operation may encounter various errors(no more disk, printer out of paper).
- When entire file is copied, program may close both files(system call), write message to console or window(system call) and finally terminate normally (final system call).

System Calls

- **Programming interface to the services provided by the OS**
- Typically written in a high-level language (C or C++)

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



```
Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

System Calls Contd…

- Many System calls are required even to execute simple programs (around thousands of system calls per second)
- But application programmers develop programs according to Application Programming Interface(API).

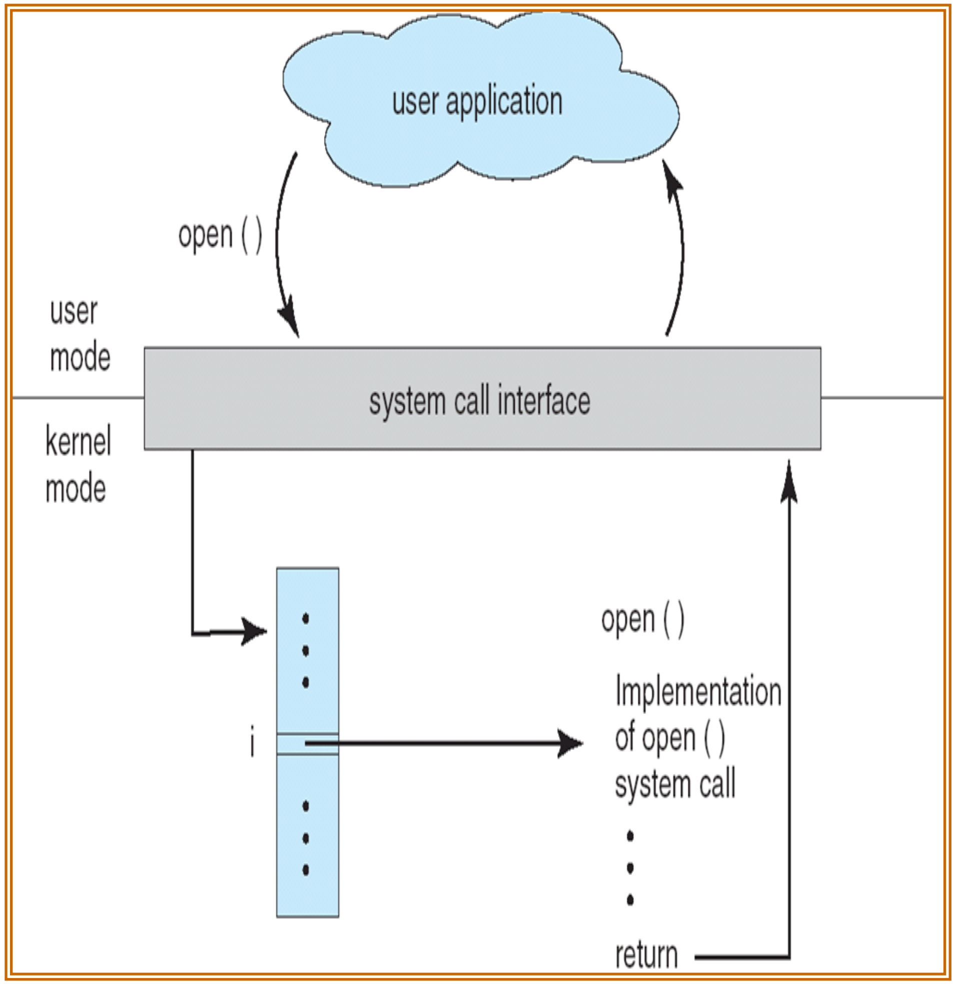
What is API? API specifies a set of functions that are available to an application programmer along with parameters passed and return values

Example of API

- Win32 API for windows systems
- Java API for java virtual machine
- POSIX API for linux , unix
- There are benefits working with APIs
  - Program portability
  - API are more easier to work with
  - There is a correlation between a function in API and an associated system call

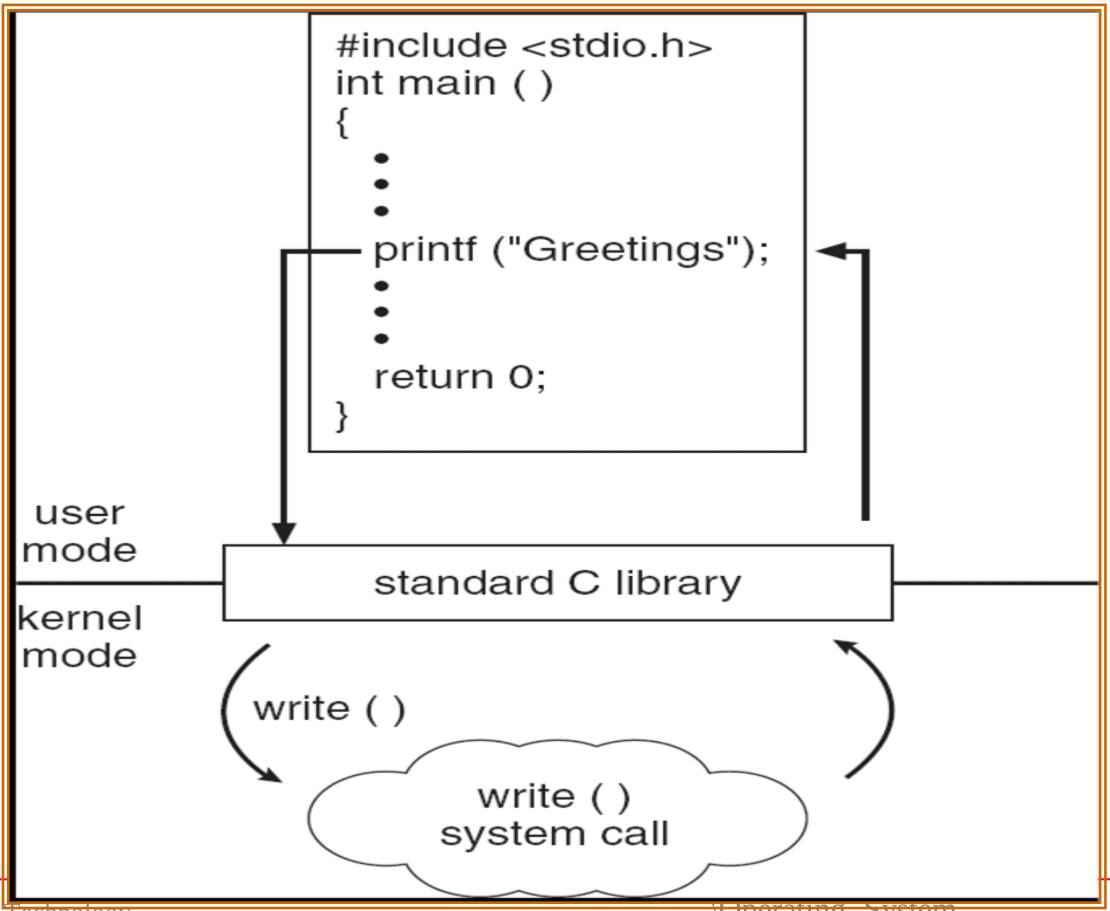
API – System Call – OS Relationship

- System call Interface(SCI) intercepts function calls in the API and invokes the necessary system calls within the OS
- SCI invokes the intended system call in the OS kernel and returns the status of the system call and any return values



Standard C Library Example

- C program invoking printf() library call, which calls write() system call



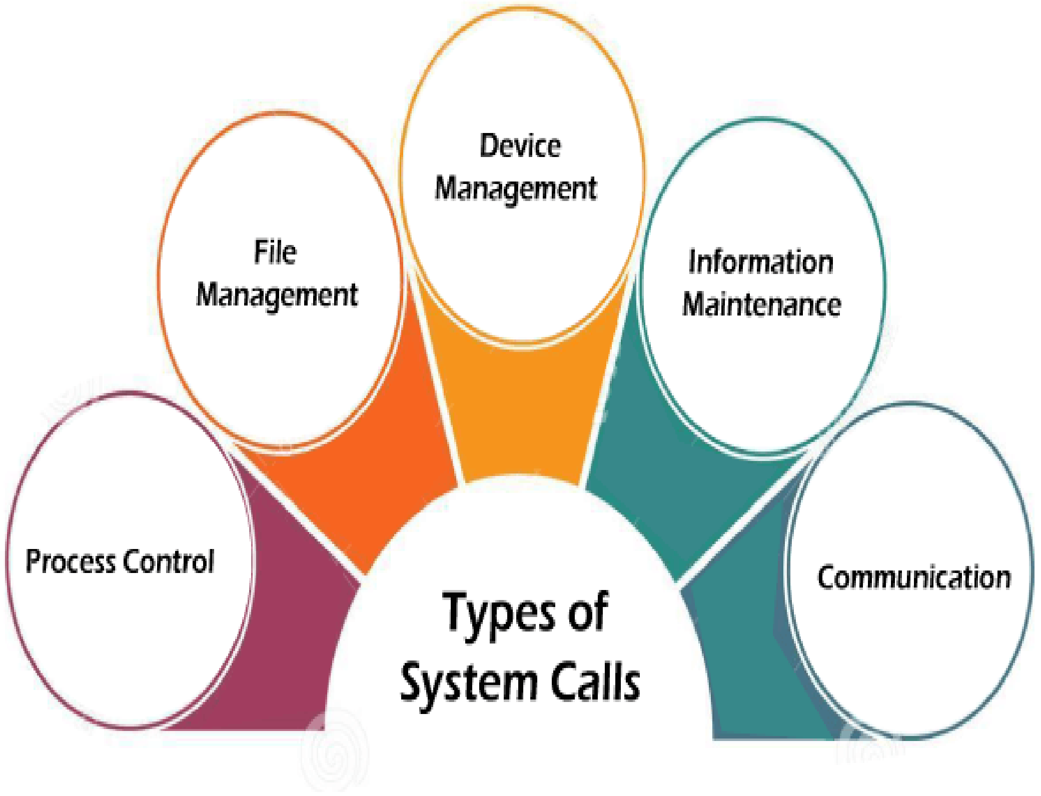
```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```

System Call Implementation

- A number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The caller needs to know nothing about how the system call is implemented
  - Just 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)

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection



Process Control

- Process control
  - Load, execute
  - Create process, terminate process
  - Get process attributes, set process attributes
  - Wait for time
  - Wait event, signal event
  - Acquire and release lock

File Management

- Create file, delete file
- Open , close
- Read , write , reposition
- Get file attributes, set file attributes
  - File attributes: file name, file type, protection codes, file move, file copy.

Device Management

- Request device, release device
- Read , write , reposition
- Get device attributes, set device attributes

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

Debugging and maintain time profile of a program.

Communication

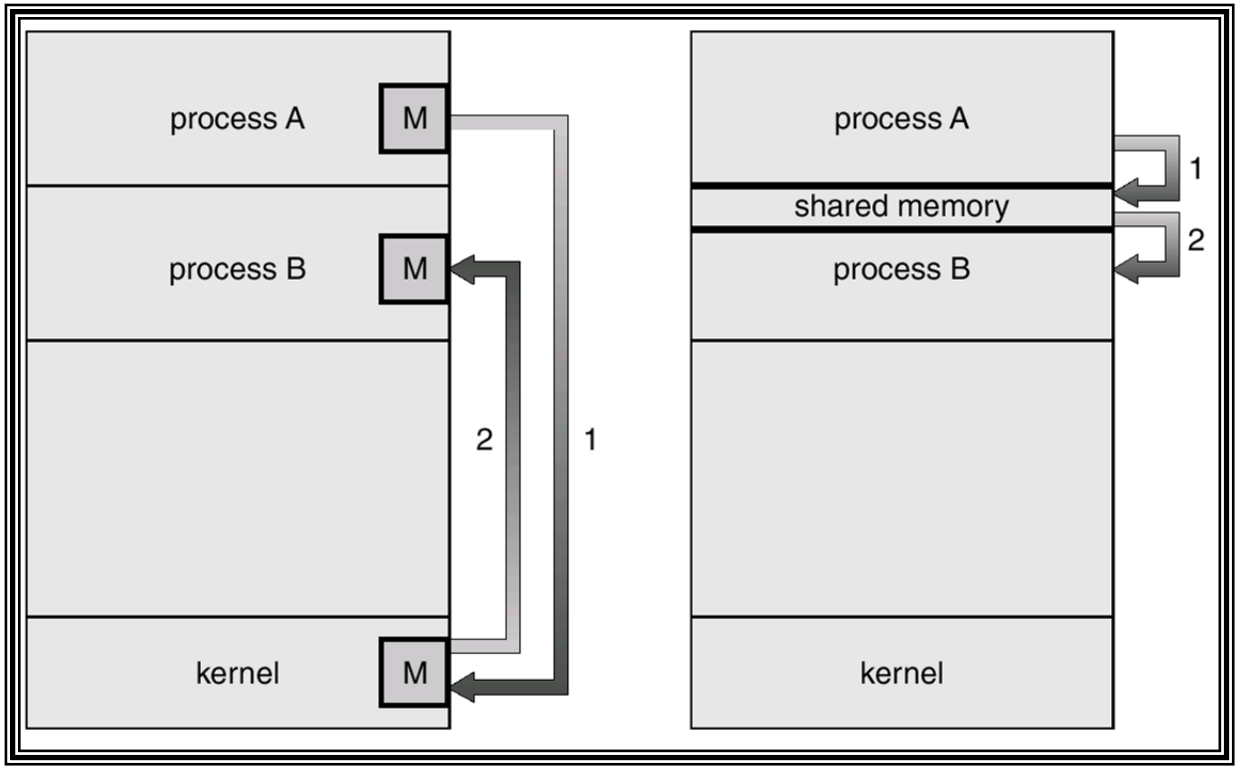
Message passing and shared memory model

- Create , delete communication connection
- Send , receive messages
- Transfer status information
- Attach or detach remote devices

Message passing is used to transfer small amount of data

Communication Models

- Communication may take place using either message passing or shared memory.



Msg Passing

Shared Memory

Protection

- Set permission, get permission
- Allow user and deny user system calls

- "At best, 'communication' is the name for those practices that compensate for the fact that we can never be each other."

  ~John Durham Peters (1999, p. 268)

OS Structure

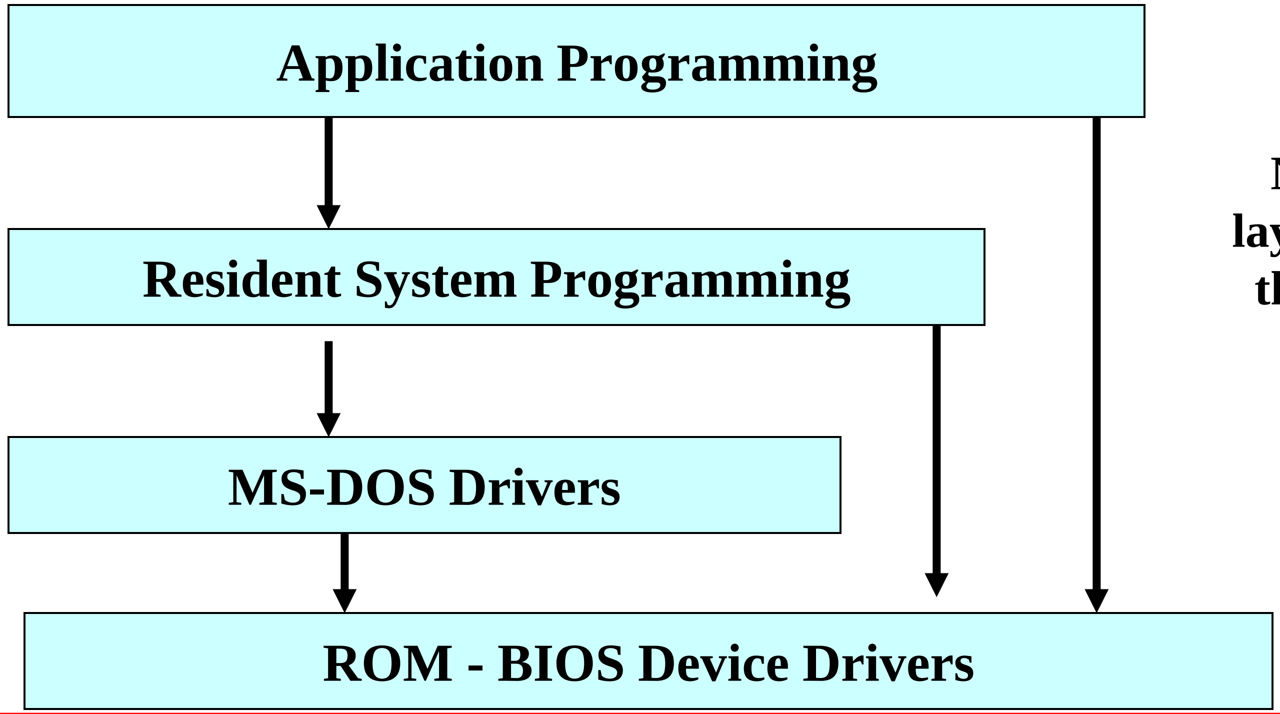
MS-DOS System Structure (Simple structure)

- MS-DOS – written to provide the most functionality in the least space
  - not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

OPERATING SYSTEM STRUCTURES

How An Operating System Is Put Together

A SIMPLE STRUCTURE: Example of MS-DOS.



Note how all layers can touch the hardware.

OS Structure

There are different types of Operating Systems commonly used

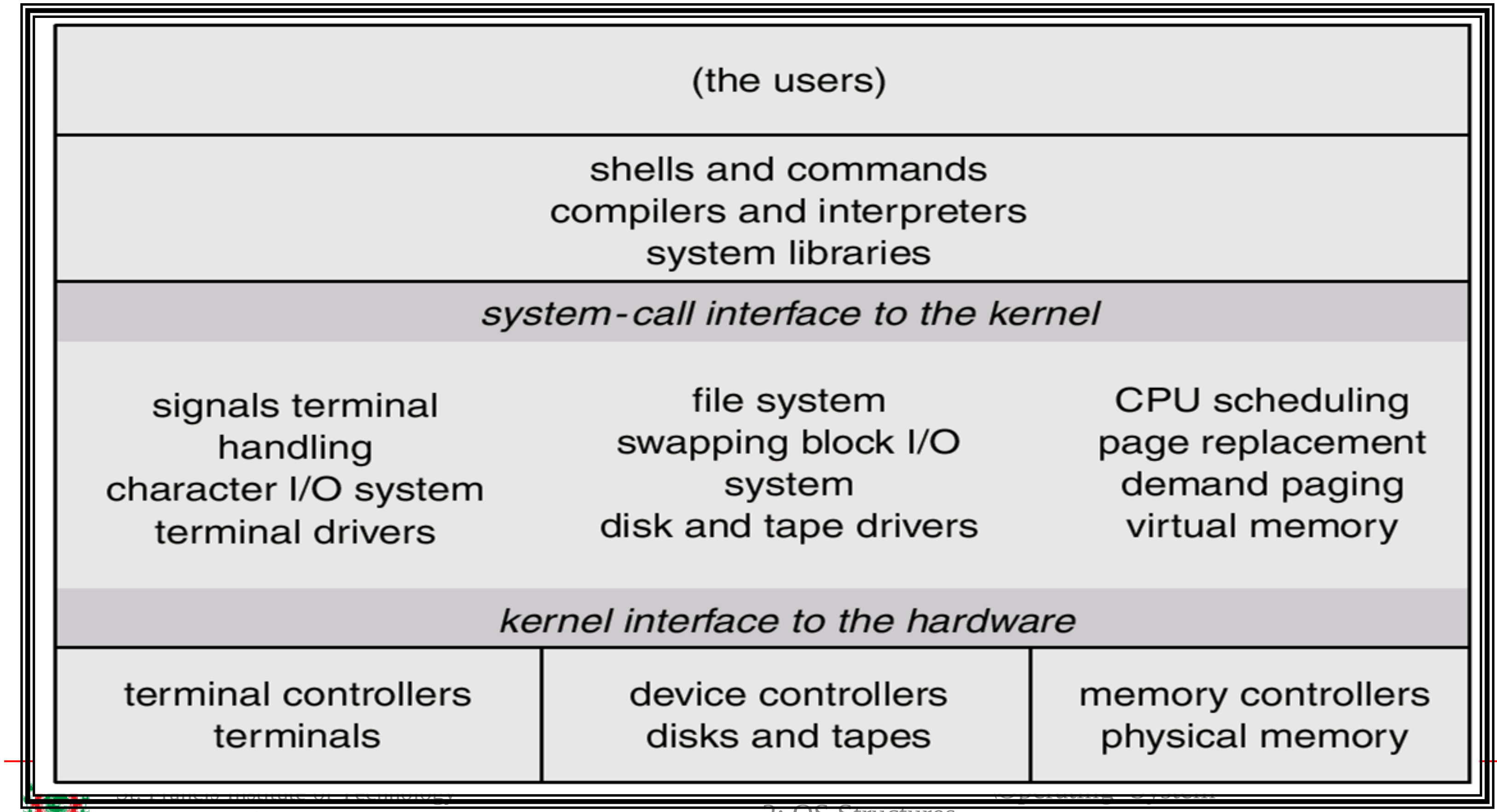
- **Monolithic**
- **Layered**
- **Modular**
- **Microkernel**

UNIX System Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.
- The UNIX OS consists of two separable parts.
  - Systems programs
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

OPERATING SYSTEM STRUCTURES

Monolithic structure: Example of earlier UNIX.



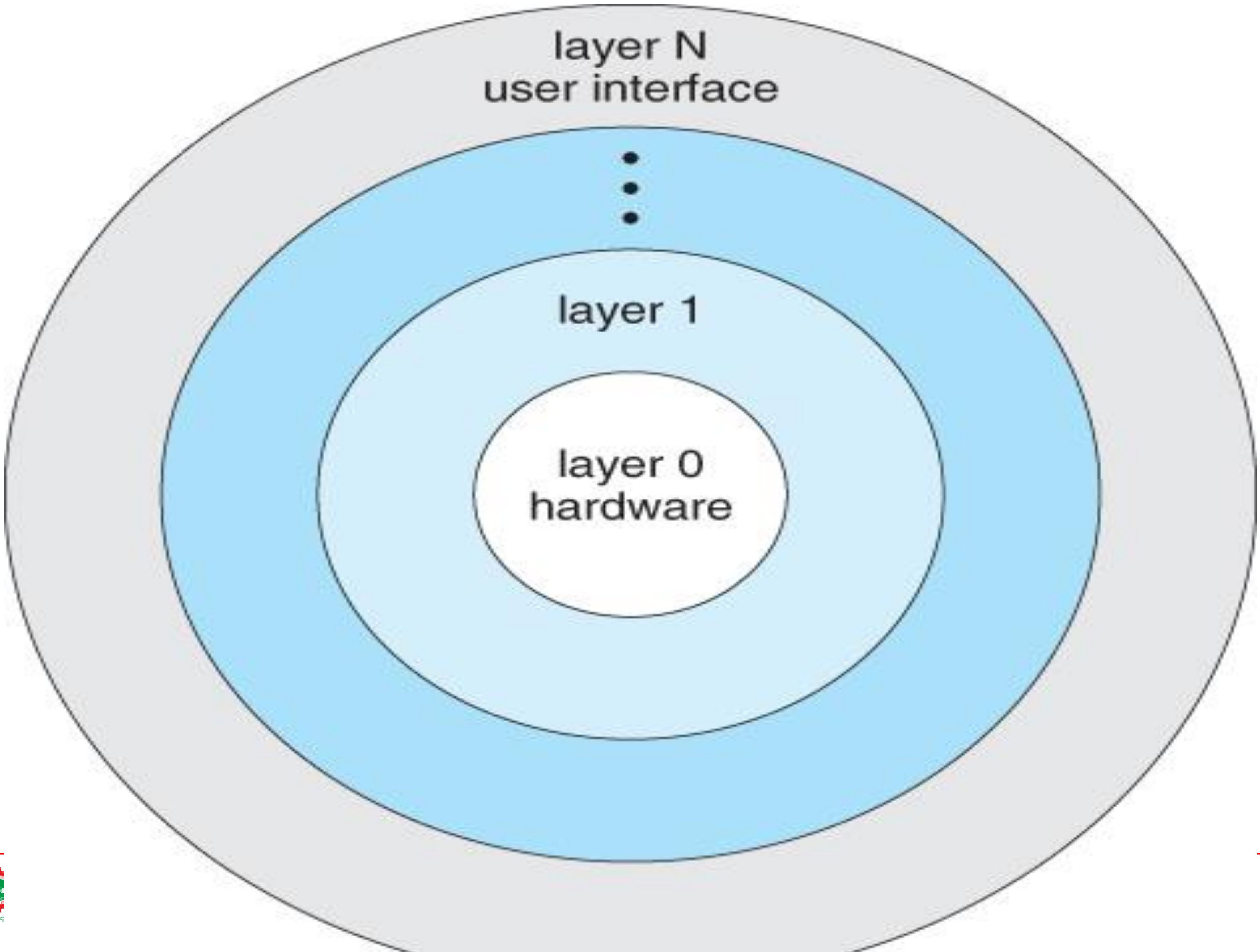
Monolithic

- All OS services operate in kernel space
- Good performance
- Disadvantages
  - Dependencies between system component
  - Complex & huge (millions of lines of code)
  - Larger size makes it hard to maintain
- E.g. Microsoft windows, Unix, DOS

Layered Approach

- The operating system is divided into a number of layers (levels).
- The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- A layer does not need to know how these operations are implemented; it needs to know only what these operations do.
- Hence, each layer hides the existence of certain data structures, operations, and hardware from higher-level layers.

Layered Structure



Layered structure

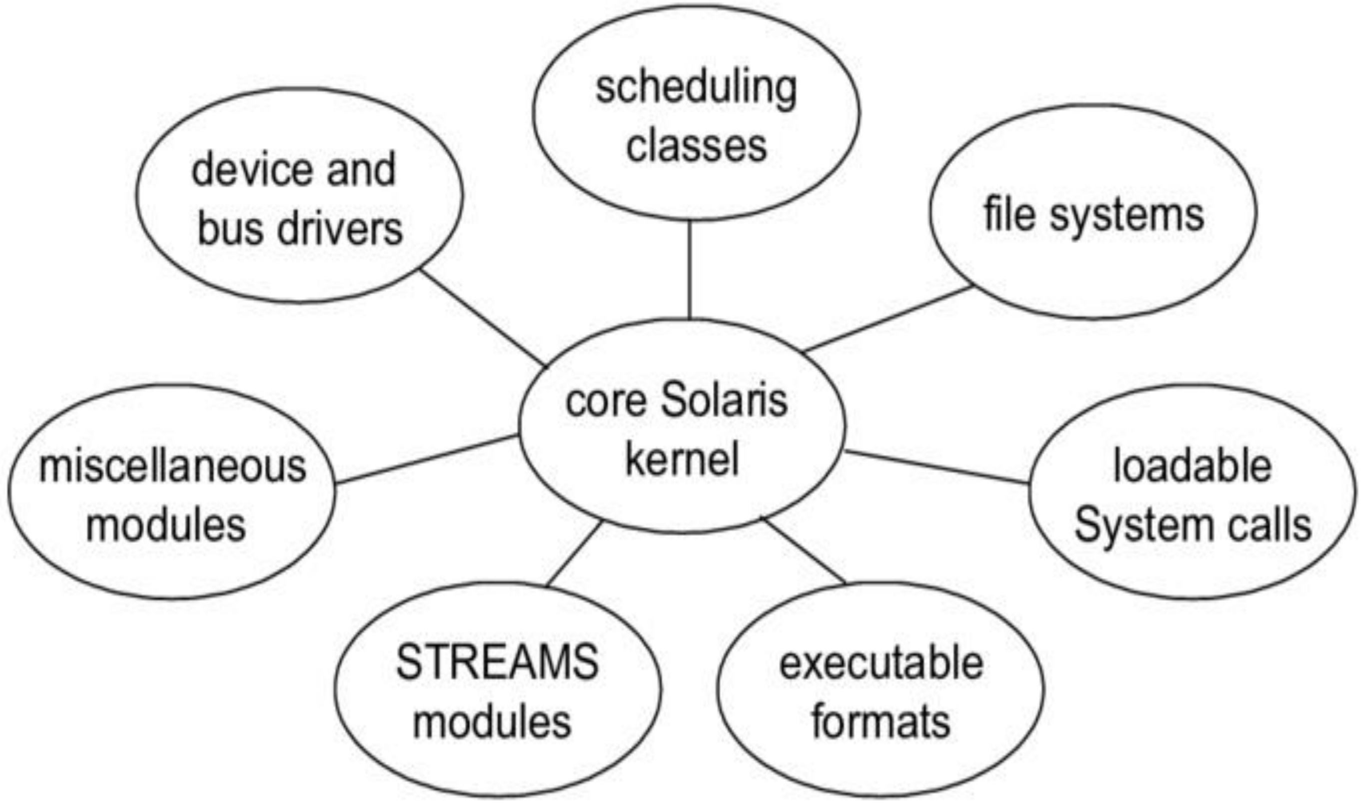
Advantage:

- Easy to debug. The first layer can be debugged without any concern for the rest of the system.
- Hardware layer is protected from the user interface

Disadvantage:

- The layers are selected so that each uses functions (operations) and services of only lower-level layers so designing layers is difficult.
- Not very efficient as request issued by one layer need to pass through all the layers in between.

Modular approach



Modular approach

- Functions are dynamically loaded to the kernel as and when required.
- It resembles the layered system but it is flexible than layered structure.
- Any module can call any other module.
- Each layer has protected interface.

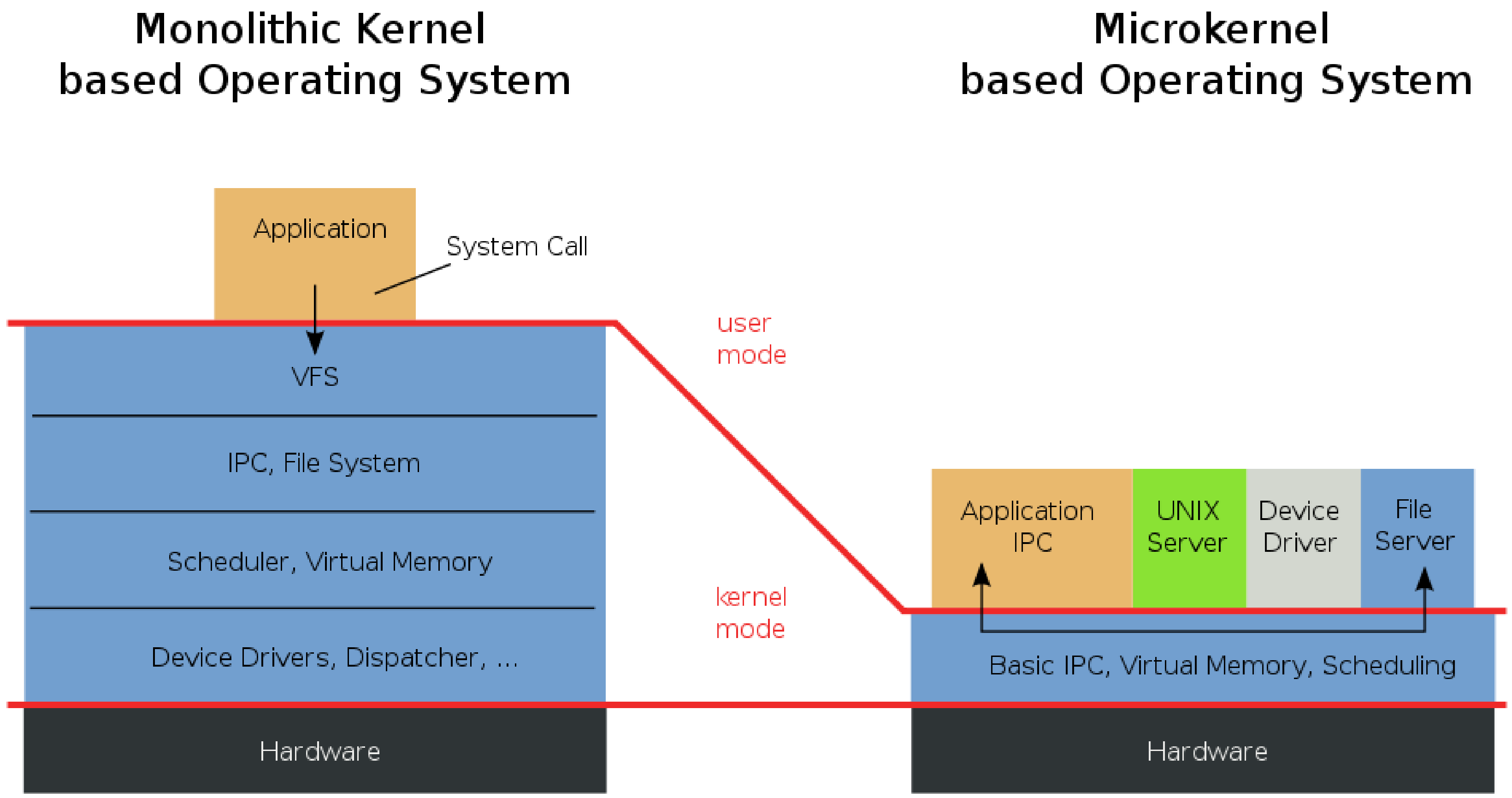
Kernels

- Different Types of Kernel Designs
  - Monolithic kernel
  - Microkernel
  - Hybrid Kernel

Microkernels

- Minimalist approach
  - IPC, virtual memory, thread scheduling are in kernel space.
- Put the rest into user space
  - Device drivers, networking, file system, user interface
- Less services in kernel space
- Only communication and message passing by kernel

Monolithic Kernels VS Microkernels



Microkernel System Structure

- Moves as much from the kernel into "*user*" space.
- Communication takes place between user modules using message passing.
- Benefits:

  - easier to extend a microkernel

  - easier to port the operating system to new architectures

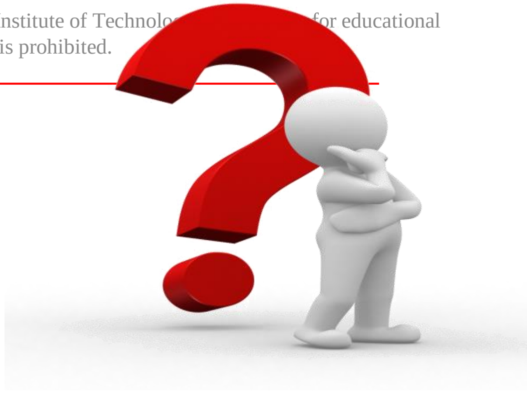
- Disadvantages
  - Lots of system calls and context switches
- E.g. Mach, Symbian, Minix, K42

Hybrid Kernels

- Combine the best of both worlds
  - Speed and simple design of a monolithic kernel
  - Modularity and stability of a microkernel
- E.g. Windows NT, NetWare.

Summary: Kernels

- Monolithic kernels
  - Advantages: performance
  - Disadvantages: difficult to debug and maintain
- Microkernels
  - Advantages: more reliable and secure
  - Disadvantages: more overhead
- Hybrid Kernels
  - Advantages: benefits of monolithic and microkernels
  - Disadvantages: same as monolithic kernels



Supervisor state is

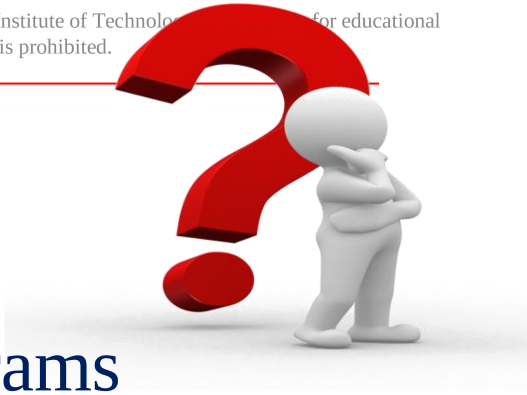
A. never used

B. entered by programs when they enter the processor

C. required to perform any I/O

D. only allowed to the operating system

E. None of the above



Which of the following functions is(are) performed by the loader

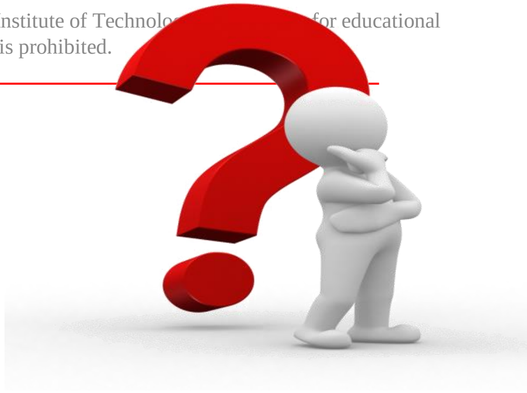
A. allocate space in memory for the programs and resolve symbolic references between object decks

B adjust all address dependent locations, such as address constants, to correspond to the allocated space.

C. physically place the machine instructions and data into memory.

D.All of the above

E.None of the above



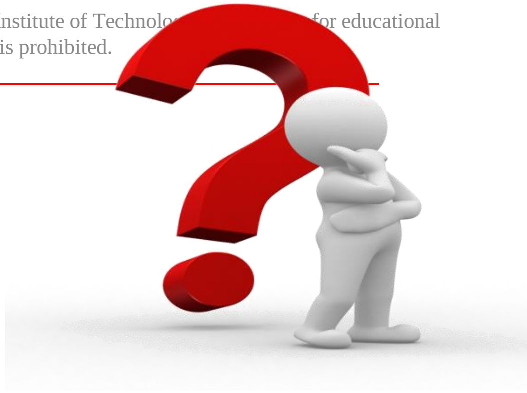
What is operating system?

a) collection of programs that manages hardware resources

b) system service provider to the application programs

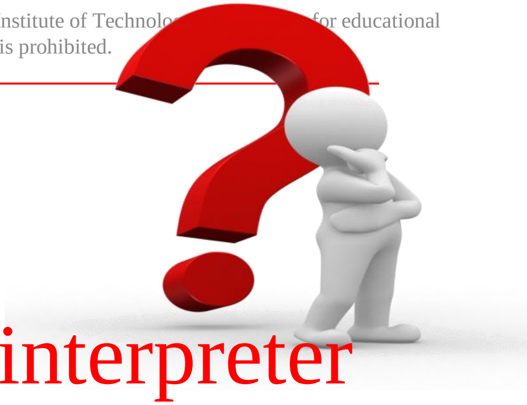
c) link to interface the hardware and application programs

d) all of the mentioned



To access the services of operating system, the interface is provided by the

a) system calls
b) API
c) library
d) assembly instructions



The main function of the command interpreter is

a)to get and execute the next user-specified command

b) to provide the interface between the API and application program

c) to handle the files in operating system

d) none of the mentioned

UNIX

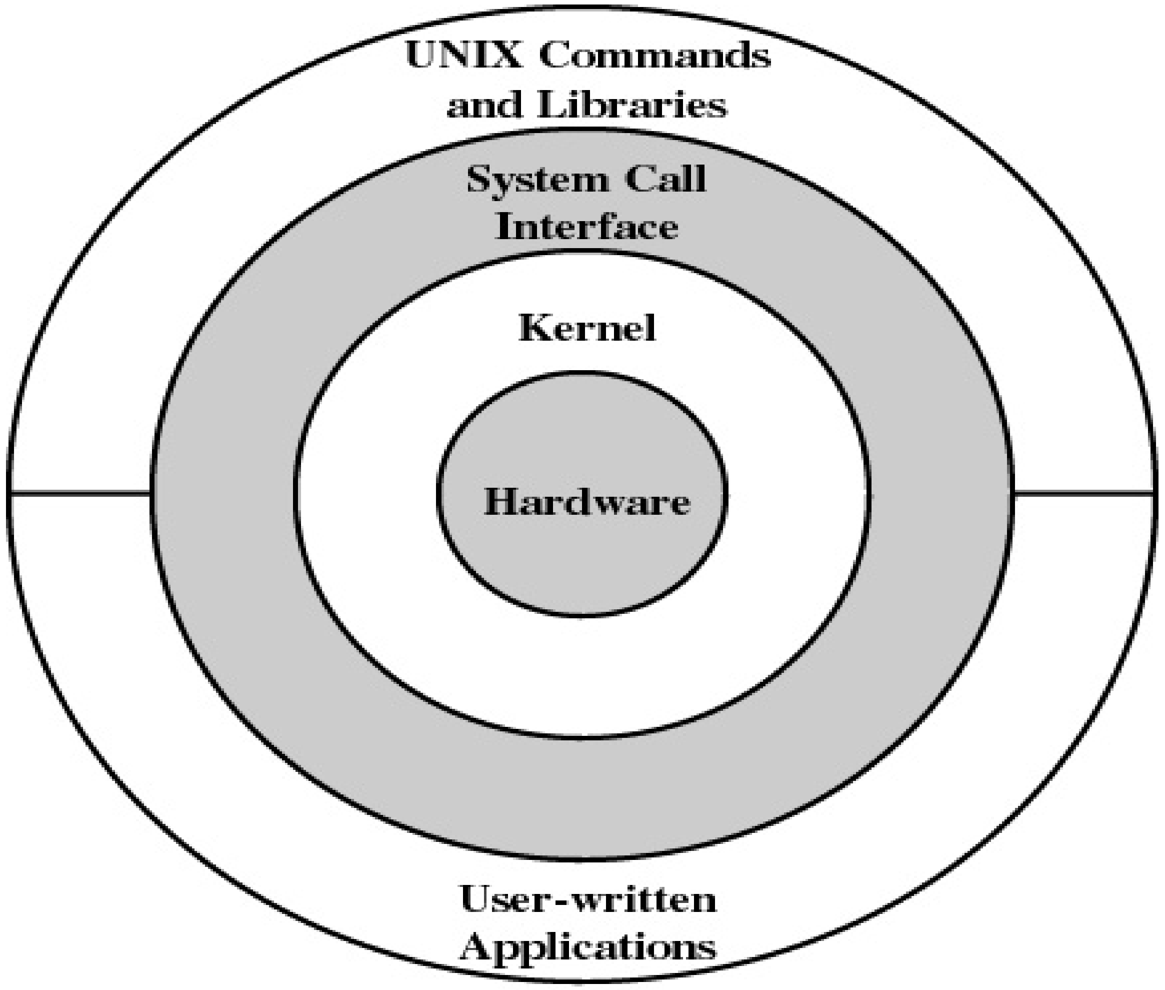


Figure 2.15 General UNIX Architecture