MODULE 1

What is an Operating System?

A program that acts as an intermediary (interface) 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

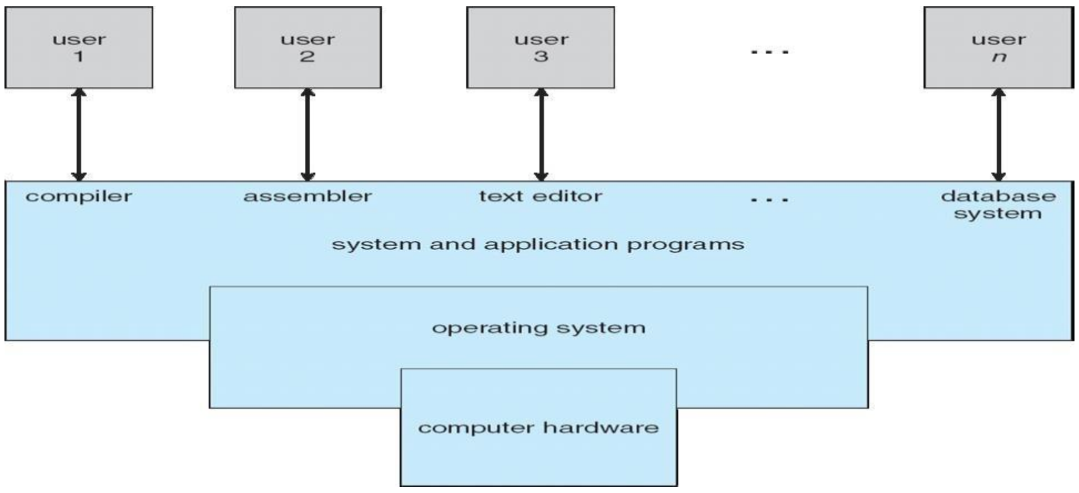
- **Hardware -** provides basic computing resources CPU, memory, I/O devices
- **Operating system-** Controls and coordinates use of hardware among various applications and users
- **Application programs -** define the ways in which the system resources are used to solve the computing problems of the users -Word processors, compilers, web browsers, database systems, video games
- **Users -** People, machines, other computers.

Role of operating System with User and System View points

Operating system can be explored from two view points

- User View
- System View

The user’s view of the computer varies according to the interface being use



Single User: Most users sit in front of a PC, consisting of a monitors, keyboard, mouse and system unit .Here the goal is to maximize the work.

Mainframe: In other cases user sits at a terminal connected to a mainframe or minicomputer .Users are accessing the same through other terminals. Resource sharing is main goal here.

Handheld Computer: Many types of handheld computer are used for easy of use. Some are connected to network through wired or wireless media embedded computers are used to run without user intervention.

Embeded Computers are used to run without user intervention.

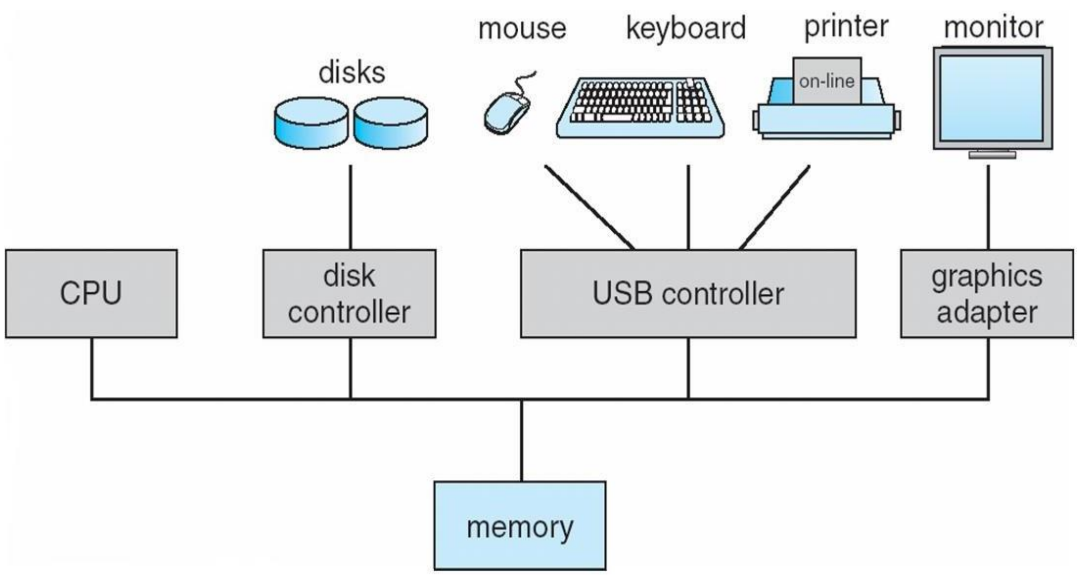
System View:

Resource Allocator: From the computer point of view operating system is viewed as a resource allocator . OS acts as a manager of resources like CPU, memory, files etc.

Control Program: OS also viewed as a control program it manages the execution of userof computers.

Computer System Organization

Computer System Operation:



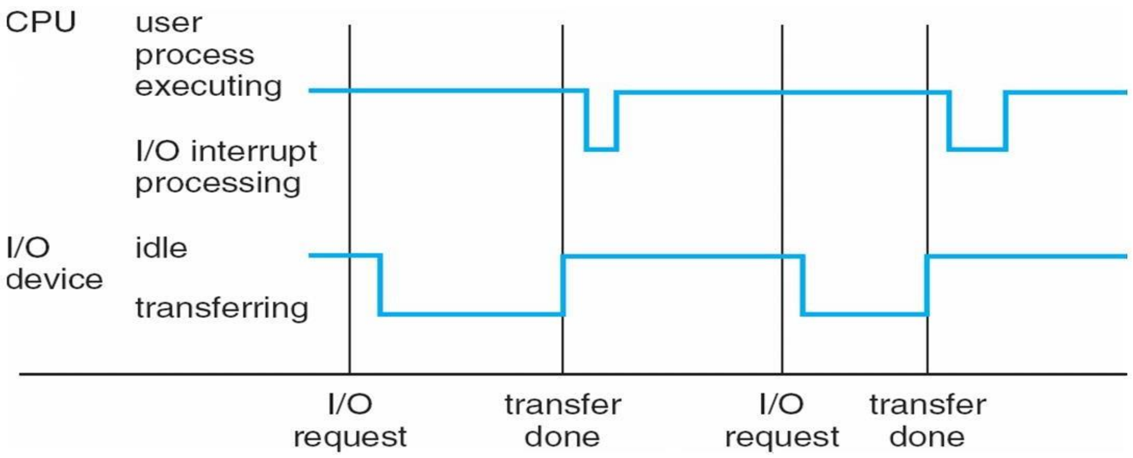
- ➢ One or more CPUs, device controllers connect through common bus providing access to shared memory
- ➢ Concurrent execution of CPUs and devices competing for memory cycles.
- ➢ I/O devices and the CPU can execute concurrently.
- ➢ Each device controller has a local buffer.
- ➢ CPU moves data from/to main memory to/from local buffers.
- ➢ Device controller informs CPU that it has finished its operation by causing an interrupt.
- ➢ For a computer to start running **bootstrap program** is required. It initializes all event to occur.
- ➢ The occurrence of an event is usually signaled by an **Interrupt** from either the hardware or the software.
  - ✓ Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
  - ✓ Software may trigger an interrupt executing a special operation called a **System call** (also called a **monitor call**).

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*. 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:
- Separate segments of code determine what action should be taken for each type of interrupt.



Interrupt Timeline for Single process to do output

Storage structures

Computer programs must be in main memory or Random-Access Memory or RAM. It is the only large storage media that the CPU can access directly.

Programs and data must reside in main memory, this is not possible because

- ✓ Main memory is too small
- ✓ Main memory is volatile storage.

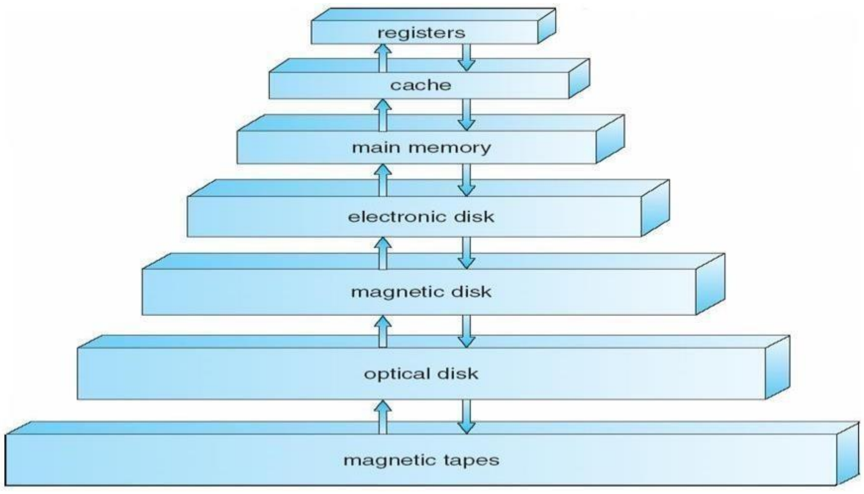
Thus, to hold large quantities of data most computer system provides secondary storage as an extension of main memory. Most common secondary-storage device is a magnetic disk, which provides storage for both programs and data.

Storage system are organized based on

- ✓ Speed
- ✓ Cost
- ✓ Volatility

Higher levels are expensive but fast.

Caching:



Information is copied from slower to faster storage temporarily. When we need a particular piece of information, we first check whether it is in the cache. If it is, we use the information directly from the cache. If it is not, we use the information from the source, putting a copy in the cache under the assumption that we will need it again soon. Information is copied from slower to faster storage temporarily.

In addition, internal programmable registers, such as index registers, provide a high-speed cache for main memory. The programmer (or compiler) implements the registerallocation and register-replacement algorithms to decide which information to keep in registers and which to keep in main memory.

I/O Structure:

- Computer system consists of CPU's and multiple device controllers that are connected through a common bus.
- Operating systems have a device driver for each device controller which presents auniform interface to the device to the rest of the operating system.

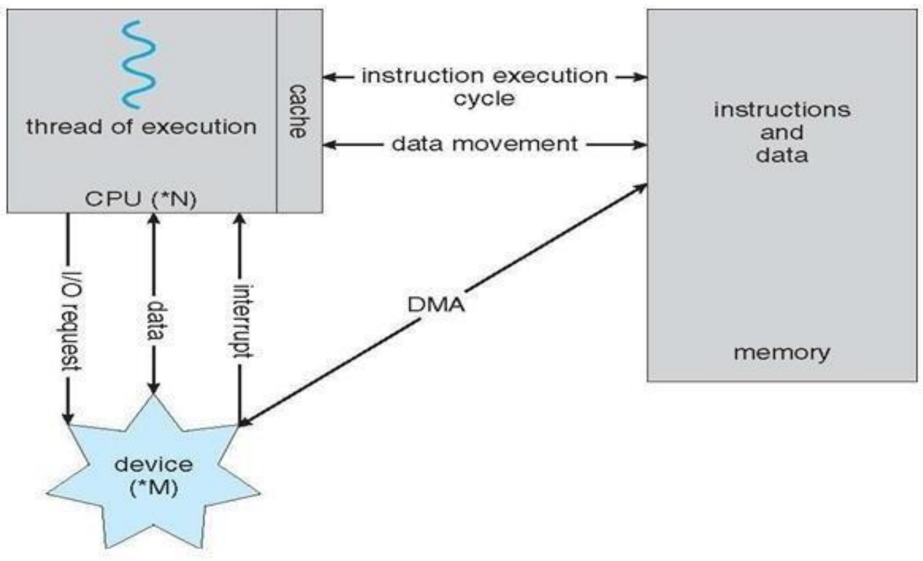
Interrupt-Driven I/O

- To start an I/O operation, the device driver loads the appropriate registers withinthe device controller.
- The device controller examines the contents of these registers, and starts the transfer of data from the device to its local buffer.
- Once the transfer of data is complete, the device controller informs the device driver via an interrupt. Device driver returns the control to operating system along with status information.

Direct Memory Access (DMA)

Interrupt-Driven I/O produce high overhead for bulk data movement like disk I/O. To solve this problem DMA is used.

After setting up buffers, pointers and counters for the I/O device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block to tell the device driver that the operation has completed at this time CPU is available to perform other work



Computer System Architecture

Three different types of Architecture are:-

- Single-processor systems
- Multi-processor systems
- Clustered systems

Single Processor system:

On a single processor system, there is one main CPU capable of executing a general-purpose instruction set.

Multi processor system (Parallel systems or Tightly coupled systems):

Such systems have two or more processors in close communication sharing the computer bus and memory.

Multiprocessor systems have three main advantages-

- Increased throughput
- Economy of scale.
- Increased reliability.

Two types of multi-processor systems are-

- **Symmetric Multi processors (SMP):**In symmetric multi processing, each processors runs an identical copy of OS and they communicate with one another as needed. All the CPU shares the common memory.SMP means all processors are peers i.e. no master slave relationship exists between processors. Each processor concurrently runs a copy of OS.
- **Asymmetric Multiprocessing:** Each processor is assigned a specific task. A master process controls the system. Other processors look to the master for instruction. This scheme defines master-slave relationship.

The differences between symmetric & asymmetric multi processing may be result of either H/w or S/w. Special H/w can differentiate the multiple processors or the S/w can be written to allow only master & multiple slaves.

Advantages of Multi Processor Systems:

- **Increased Throughput:** By increasing the Number of processors we can get more work done in less time. When multiple process co-operate on task, a certain amount of overhead is incurred in keeping all parts working correctly.
- **Economy of Scale:** Multi processor system can save more money than multiple single processor, since they share peripherals, mass storage & power supplies. If many programs operate on same data, they will be stored on one disk & all processors can share them instead of maintaining data on several systems.
- **Increased Reliability:** If a program is distributed properly on several processors, than the failure of one processor will not halt the system but it only slows down.

Operating System Structure

Multiprogramming: It increases CPU utilization by organizing jobs (code and data)so that the CPU always has one to execute.

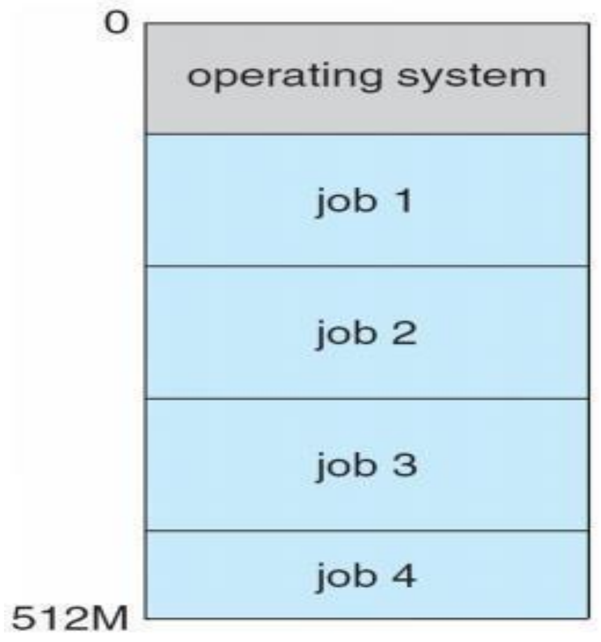


Fig shows memory layout for a multiprogramming system.

- In multiprogramming system a subset of total jobs in system is kept in memory.
- One job is selected and run via job scheduling.
- When it has to wait for I/O, OS switches to another job.
- No user interaction with computer system.

Advantages:

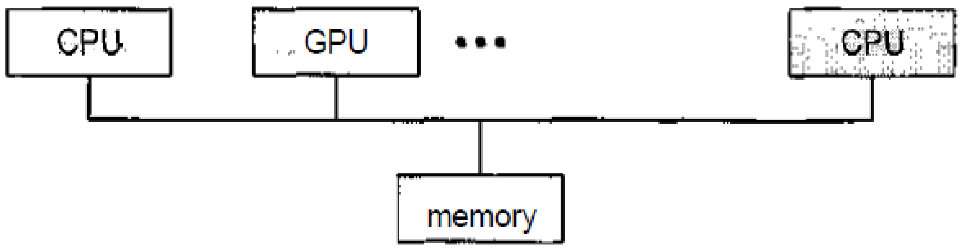
- ➢ High and efficient CPU utilization.
- ➢ Many programs are allocated to CPU almost simultaneously.

Disadvantages:

- ➢ CPU scheduling is required.
- ➢ To accomadate many jobs in memory, memory management is required.

Time sharing (Multi Tasking):

- It is a logical extension of multiprogramming in which CPU switches jobs so frequently that users can interact with each job while it is running.(interactive computing)
- Response time should be less than 1 second.
- Each user has at least one program executing in memory process.
- CPU scheduling is used to select a job from the job pool.
- 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.



Operating System Operations

- Modern operating systems are interrupt driven.
- Events are signaled by the occurrence of an interrupt or a trap(Trap is a software generated interrupt. ex: divide by zero. It is also called as exception.)

Dual mode

Operation allows OS to protect itself and other system components.Two different modes are

1. User mode
2. Kernel mode.

Hardware provides mode bit to switch between the modes.

- It provides ability to distinguish when system is running in user mode or kernel mode.

- Some instructions are designed as privileged & only executable in kernel mode.
- System call changes the mode to kernel mode.

Timer

- Timer is used to prevent a user program from getting stuck in an infinite loop.
- Timer can be set to interrupt the computer after a specified period.
- We can use the timer to prevent a user program from running too long.

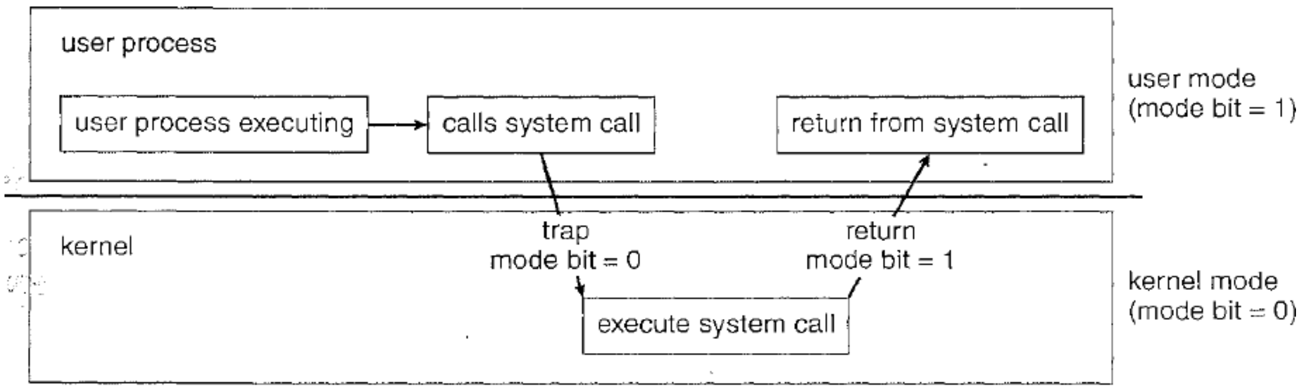


Figure 1.10 Transition from user to kernel mode.

Kernel Mode:

Normally in system booting hardware starts in kernel mode and after starting the Operating System it starts the user application.

Privilage (Work):

- Handling Interrupt.
- To switch user to kernel
- Input-Output Management.

Process Management

A program does nothing unless its instructions are executed by a CPU. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process. A system task, such as sending output to a printer, can also be a process.

A process is a program in execution. It is the unit of work within the system. Program is a passive entity. Process is an active entity.

- ❖ Process needs resources to accomplish its task Ex: CPU, Memory, I/O.
- ❖ Process termination requires reclaim of any reusable resources.
- ❖ Single-threaded process has one program counter specifying the location of next instruction to execute.
- ❖ Process executes instructions sequentially, one at a time, until completion
- ❖ Typically system has many processes, some user, some operating system running concurrently onone or more CPUs.
- ❖ Multi0threaded process has one program counter per thread.

Process Management Activities

The Operating system is responsible for the following activities in connection with process management.

- ❖ Creating and deleting both the user and system processes.
- ❖ Suspending and resuming process.
- ❖ Providing mechanisms for process synchronization and process communication.
- ❖ Providing mechanism for deadlock handling.

Memory Management

- ➢ Main memory is central to the operation of a modern computer system .is a repository of quickly accessible data shared by the CPU and I/O devices.
- ➢ Instructions must be in memory for the CPU to execute them.
- ➢ For a program to be executed, it must be mapped to absolute addresses and loaded into memory. As the program executes, it accesses program instructions and data from memory by generating these absolute addresses. After program termination, its memory space is declared available, and the next program can be loaded and executed.

Memory Management Activities

- Keeping track of which parts of memory are currently being used and by whom.
- Deciding which processes and data to move into and out of memory.
- Allocating and de-allocating memory space as needed.

Storage Management

- The operating system provides a uniform, logical view of information storage.
- The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, called as file.
- The operating system maps files onto physical media and accesses these files via the storage devices.

File system Management

- Files usually organized into directories.
- Access control mechanisms are used to determine who can access what

File System Management Activities

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (nonvolatile) storage media.

Mass Storage Management

- Main memory is too small to accommodate all data and programs, the computer system must provide secondary storage to back up main memory.
- Usually disks used to store data that does not fit in main memory or data that must be kept for a long time.
- Entire speed of computer operation hinges on disk subsystems and its algorithm

Mass storage management Activities

- Free space Management.

- Storage Allocation.
- Disk scheduling.

Protection and security

Protection: Any Mechanism for controlling access of process or users to resources defined by the operating system.

Security: Defense of the system against internal and external attacks-like denial-of-service, viruses, identity theft, theft of service etc.

Protection and security require the system to be able to distinguish among all its users.

- User identifiers 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 allows set of users to be defined and also associated with each process.
- User sometimes needs to *escalate privileges* to gain extra permissions for an activity. Operating system provides various methods to allow privilege escalation. Ex: set-uid attribute on UNIX

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

Access to a shared resource increases computation speed, functionality, data availability, and reliability.

Special-Purpose Systems

Object of these computer system are to deal with limited computation domains.

Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence.

- Little user interface
- Prefer to spend their time in monitoring and managing hardware devices such as automobile engines and robotic arms.
- Embedded systems almost always run **real-time operating systems.**
- A real-time system is used when rigid time requirements have been placed on the operation of a processor

Multimedia Systems

Multimedia data (audio, video) along with conventional data (text file, word processing systems) must be handled efficiently to satisfy the user requirements.

Handheld Systems

Handheld systems include personal digital assistants (PDAs), such as Palm and Pocket-PCs, and cellular telephones etc. Many of which use special-purpose embedded operating system.

Because of their size most handheld devices will be having small amount of memory, slow processors and display screen. Programmer must consider these limitations while designing applications for handheld devices.

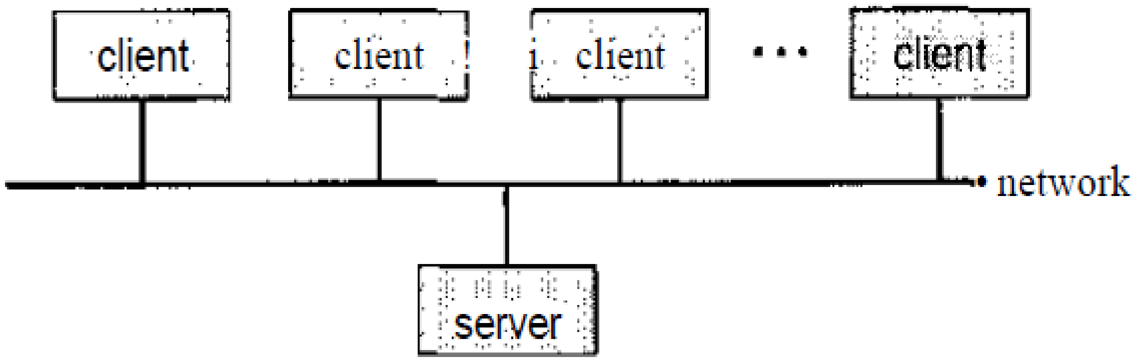
Computing Environments

Traditional Computing

Common office environment uses traditional computing normal PC is used **in traditional computing.**

Client-Server Computing

The file-server system provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers.'



Peer-to-Peer Computing

In this model, clients and servers are not distinguished from one another all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service.

Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck; but in a peer-to-peer system, services can be provided by several nodes distributed throughout the network.

Web-Based Computing

Web-Based computing has wide range of access devices like PDA’s mobiles, PC’s and workstations. It focuses on networking. Faster network connectivity is provided to wired and wireless terminals.

Operating System Services

Operating system services helpful to users are classified as

a) User interface

All operating systems have a user interface. Several forms of UI are

1. Command- line interface
2. Batch interface

Graphical User Interface(GUI)

b) Program Execution

System must be able to load the program into memory and to run that program. System must end the program execution.

c) I/O operations

Operating systems must provide a way to perform Input/output operations.

d) File-System Manipulation

Programs need to read and write files and directories. Some programs include permission management to allow or deny access to files based on file ownership.

One process needs to exchange information with other process.

Communications must be implemented via shared memory or message passing.

e) Error Detection

Errors may occur in different units of computer system. For each type of error OS should take the appropriate action to ensure correct and consistent computing.
OS functions used to ensure the efficient operations of the system are

❖ **Resources Allocation**

When multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

❖ **Accounting**

We must keep track of which user use how much and what kind of resources.

❖ **Protection and Security**

When several separate processes execute concurrently, it should not be possible for one process to interfere with the others.
Protection ensures that all access to system resources is controlled.

Security enforce each user to authenticate himself by means of a password to gain access to system resources.

User Operating System Interface
Two types are CLI and GUI

Command Interpreters
Command interpreter is a special program that is running when a job is initiated or when a user first logs on.
On a system with multiple command interpreters it is referred as shells.

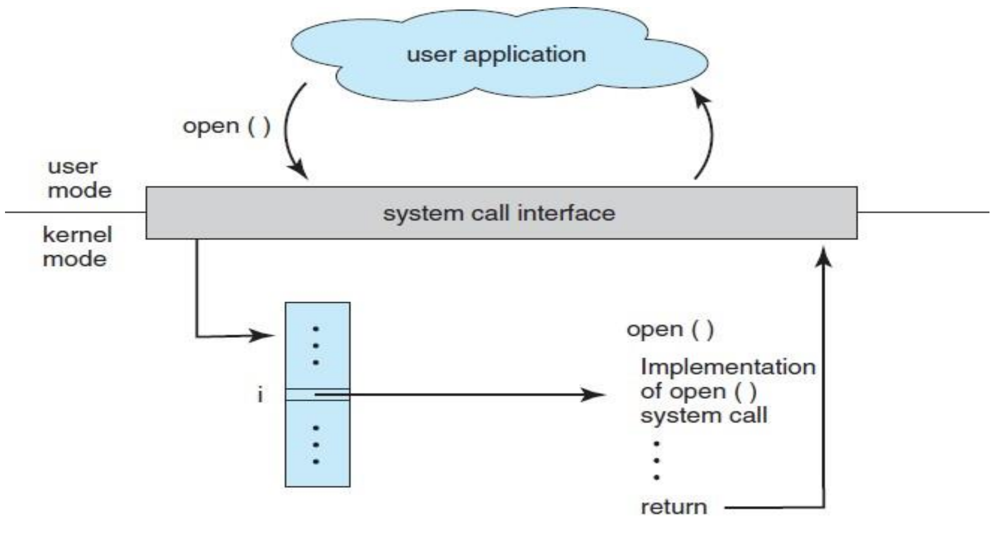
Ex: In UNIX→ Bourne Shell, KornShell etc.
Main function of the command interpreters is to execute user-specified command. Ex: DOS commands.

Graphical User Interface
GUI provides a mouse based window and menu system as an interface.

Ex: In desktop mouse is moved to position its pointer on images or icons on the screen. Depending on mouse pointer location respective program is invoked.

System calls
System calls provide an interface to the services made available by the operating system. These calls are generally available as functions written in c and C++
The run-time support system provides a system call interface that serves as the link to system calls made available by the operating system.

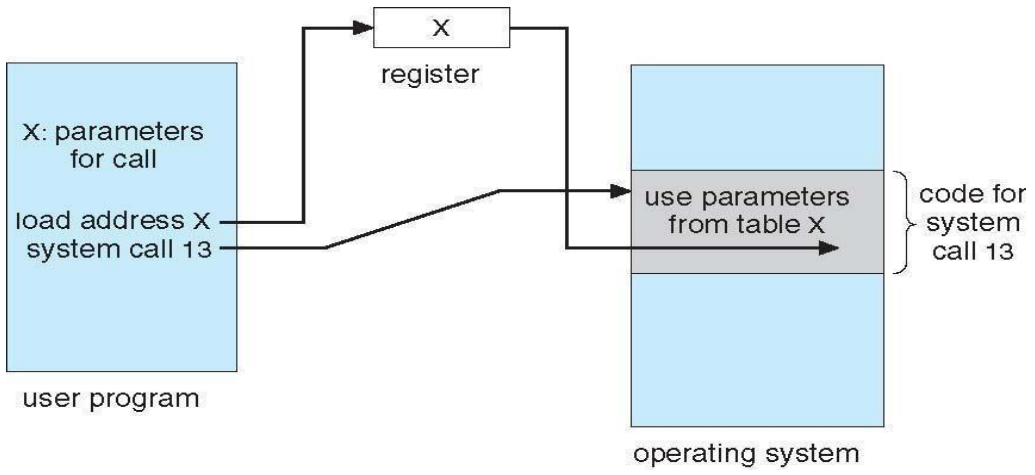
A number is associated with each system call and system call interface involves the intended system call in the operating system kernel and returns the status of the system call. Thus, most of the details of the operating system interface are hidden from the programmer



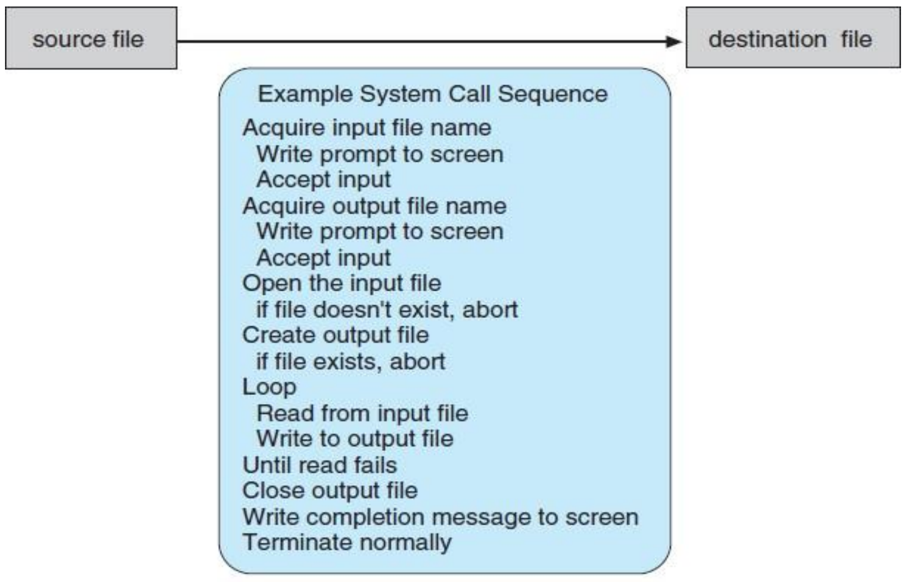
The handling of a user application invoking the open() system call.

Three general methods are used to pass parameters to the operating system.

- Pass the parameters in registers.
- Passing the address of the block or table.
- Parameters are pushed into the stack by program and popped off the stack by the OS.



Passing of parameters as a table.



Example of how system calls are used.

An example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file

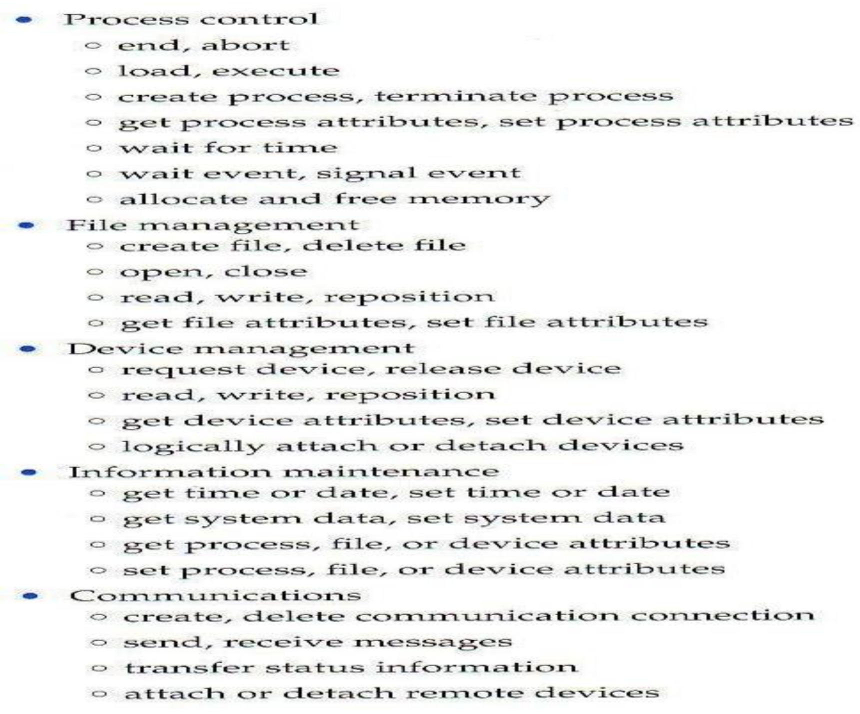
- There are number of system calls used to finish this task. The first system call is to write a message on the screen (monitor). Then to accept the input filename. Then another system call to write message on the screen, then to accept the output filename.
- When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access. In these cases, the program should print a message on the console (another system call) and then terminate abnormally (another system call) and create a new one (another system call).
- Now that both the files are opened, we enter a loop that reads from the input file (another system call) and writes to output file (another system call).
- Finally, after the entire file is copied, the program may close both files (another system call), write a message to the console or window (system call), and finally terminate normally (final system call).
- Most programmers do not use the low-level system calls directly, but instead use an

"Application Programming Interface", API.

- Instead of direct system calls provides for greater program portability between different systems. The API then makes the appropriate system calls through the system call interface, using a system call table to access specific numbered system calls.
- Each system call has a specific numbered system call. The system call table (consisting of system call number and address of the particular service) invokes a particular service routine for a specific system call.
- The caller need know nothing about how the system call is implemented or what it does during execution.

Types of System Calls

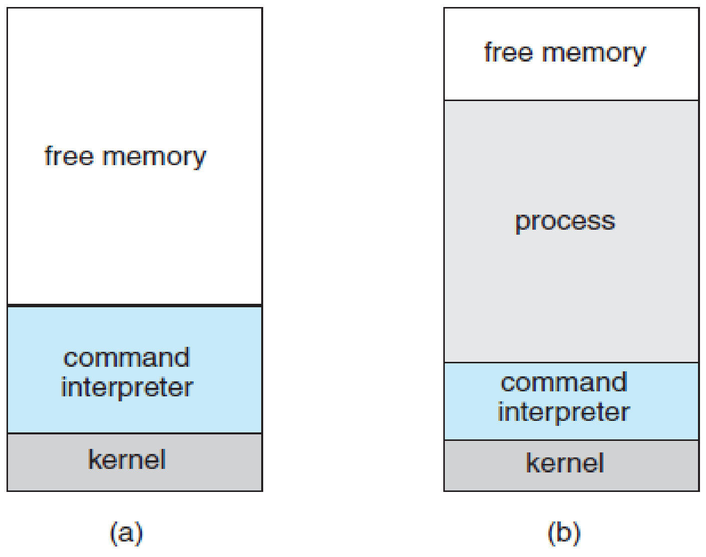
- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection.



- 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
  - attach or detach remote devices

Process Control

- Running programs can be halted by 2 methods
    - Normal
    - Abnormal.
- Operating system transfer control to the command interpreter in the normal or abnormal conditions.
- Some operating system allow control cards to indicate special recovery action in case an error occurs.
- Load and execute system calls are used by process to execute the programs.
- Get process attributes and set process attributes system calls are used to determine and reset the attributes of a job.
- Debugger is used to help the programmer in finding and correcting errors
- Two different types of process control methods are-
    - multitasking and single tasking.
- Ms-Dos operating system is an example of a single tasking system, which has a command interpreter that is involved when computer is started. ms-dos does not create a new process while running one process.



MS-DOS execution. (a) At system startup. (b) Running a program.

- Ms-Dos loads the program into memory writing over most of itself to give the program as much memory as possible.
- Ms-Dos does not have multitasking capabilities. Free BSD is an example of multitasking system.
- In free BSD, the command interpreter may continue running while another program is executed. Fork system call is used to create new process.

File Management

Files are created and deleted with the help of system calls. It require name of thefile with file attributes for creating and deleting files. Other operations performed on file is read, write and update. Close system call is used to close a file.

The file management functions of OS are –

- File management system calls include create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes.
- After **creating** a file, the file is **opened**. Data is **read** or **written** to a file.
- The file pointer may need to be **repositioned** to a point.
- The file **attributes** like filename, file type, permissions, etc. are set and retrieved usingsystem calls.
- These operations may also be supported for directories as well as ordinary files.

Information Maintenance

- Information maintenance system calls include calls to get/set the time, date, system data, and process, file, or device attributes.
- These system calls care used to transfer the information between user and the OS. Information like current time & date, no. of current users, version no. of OS, amount of free memory, disk space etc are passed from OS to the user.

Device Management

System calls are also used for accessing a device. When more than one user is accessing a device, request is made before use of the device. After using the device user must release the device. Release system call is used to free the device.

- Ms-Dos and UNIX merge the I/O devices and files to form file-device structure.

In this structure I/O devices are identified by special file names.

- In multiprogramming systems, after a process uses the device, it has to be returned to OS, so that another process can use the device.
- Devices may be physical (e.g. disk drives), or virtual / abstract (e.g. files, partitions, and RAM disks).

Communication

Communication system calls create/delete communication connection, send/receive messages, transfer status information, and attach/detach remote devices.

The **message passing** model must support calls to:

- Identify a remote process and/or host with which to communicate.
- Establish a connection between the two processes.
- Open and close the connection as needed.
- Transmit messages along the connection.
- Wait for incoming messages, in either a blocking or non-blocking state.
- Delete the connection when no longer needed.

The **shared memory** model must support calls to:

- Create and access memory that is shared amongst processes (and threads.)
- Free up shared memory and/or dynamically allocate it as needed.
- Message passing is simpler and easier, (particularly for inter-computer

communications), and is generally appropriate for small amounts of data. It is easy to implement, but there are system calls for each read and write process.

- Shared memory is faster, and is generally the better approach where large amounts of data are to be shared. This model is difficult to implement, and it consists of only few system calls

Protection

- Protection provides mechanisms for controlling which users / processes have access to which system resources.
- System calls allow the access mechanisms to be adjusted as needed, and for non-privileged users to be granted elevated access permissions under carefully controlled temporary circumstances.
- File Management- Programs to create, delete, copy, rename, print, list and generally manipulate files and directories.
- Status Information- Utilities to check on the date, time ,number of users, processes running ,data logging, etc. System registries are used to store and recall configuration information for particular applications.
- File Modification- ex: Text Editors and other tools which can change file contents.
- Programming Language Support- Ex: Compilers, Linkers, debuggers, profilers, assemblers, library archive management, interpreters for common languages.
- Program Loading and execution: Loaders, dynamic loaders, overly loaders, etc., as well as interactive debbugers.
- Communications: Programs for providing connectivity b/w processes and users, including mail, web browsers, remote logins, file transfers, and remote command execution.

OS Design and Implementation

Design Goals

At the highest level, the design of the system will be affected by the choice of hardware and type of the system like batch, time-shared etc.

Requirements can be divided into two basic groups:-

- User goals
- System goals

According to user the system must be-

- Convenient to use
- Easy to learn and use
- Reliable, safe and fast.

According to system designer the system must be-

- Easy to design, implement and maintain.
- It should be flexible, reliable &error free.

Mechanisms and Policies

- Policy and mechanism are separated from one another.
- Mechanism determine how to do something.
- Policies determine what will be done.
- Separation of policy and mechanism is important for flexibility.

Implementation

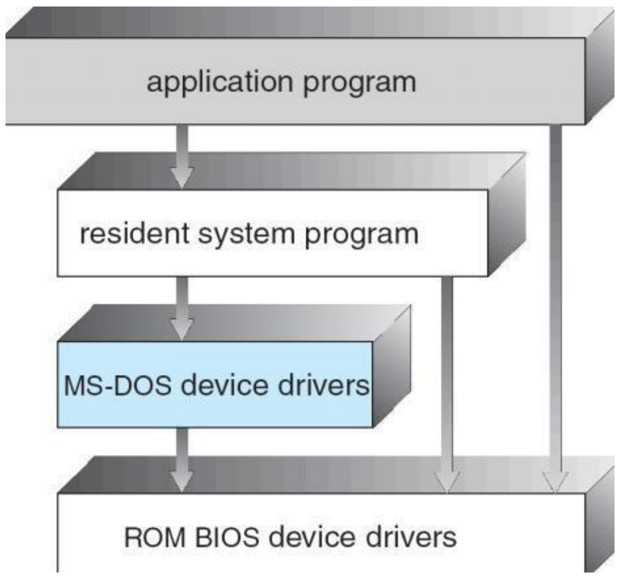
Operating systems are written in higher level language like C or C++.

The advantage of using higher level language is code can be written faster, is easy to understand and debug.

The only possible disadvantage of implementing a OS in higher level language are reduced speed and increased storage requirement.

Operating System Structure

Simple Structure

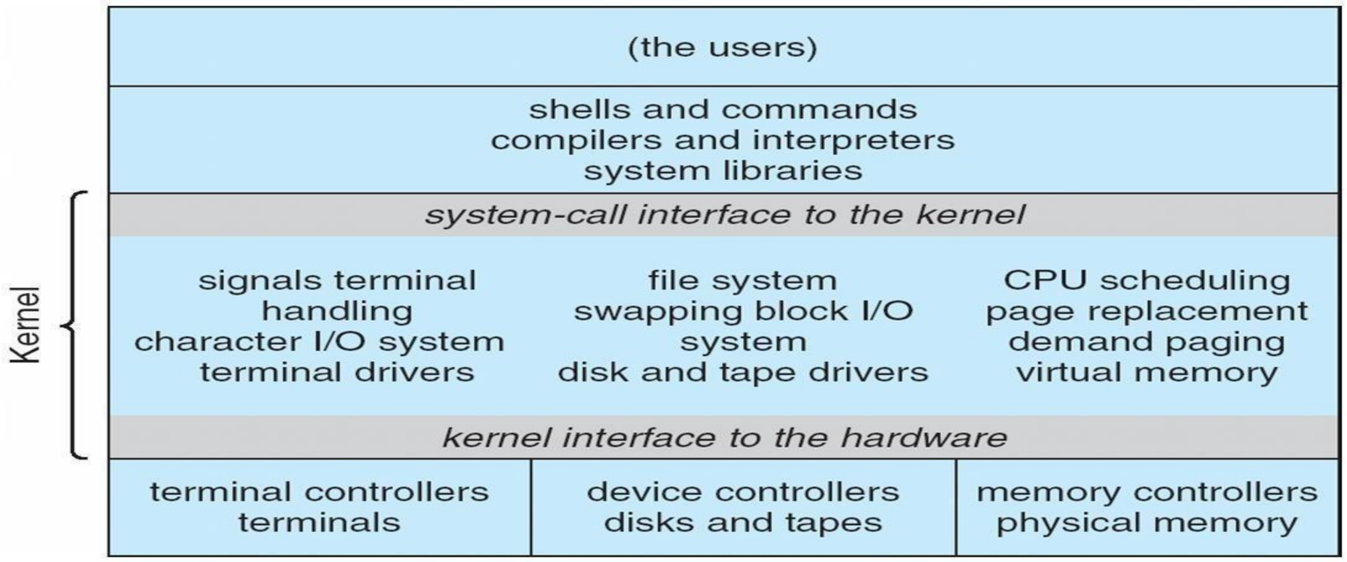


MS-DOS is all example for simple structure. It was designed to provide the most functionality in least space, so it was not divided into modules carefully.
In Ms-Dos interface and level of functionality are not well separated. Application programs are able to access the basic I/O routines to the display and disk driver. Due to this entire system may crash if it encounters the malicious program.
Another example of limited structuring is the original UNIX operating system.

It consist of two separable parts-

- The kernel
- The system programs

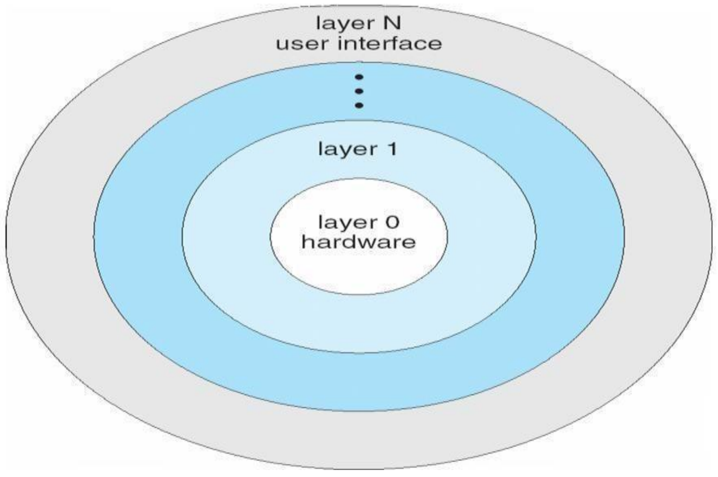
The kernel is separated into a series of interfaces and device drivers which have been added and expanded over the years as UNIX has evolved. This monolithic structure was difficult to implement and maintain.



A Traditional UNIX Operating System Model

Layered Approach

- ❖ In this operating system is broken into many layers.(levels)
- ❖ The bottom layer is the hardware (layer0)and the highest (layer N)is the user interface.



A Operating System Layered Approach

- Layer is an implementation of an abstract object.(Consist data and Operations to manipulate the data)
- The main advantage of layered approach is simplicity of construction and debugging
- Higher-level layers uses functions and services of only lower-level layers
- The major difficulty with layered approach is defining the various layers.

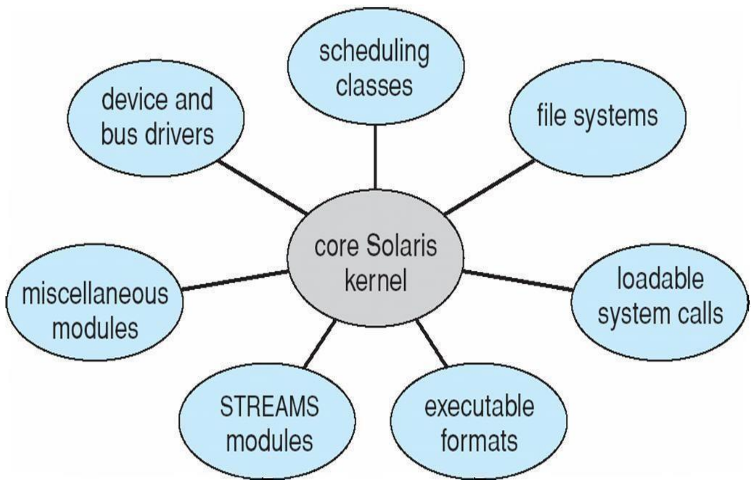
Micro kernels:

- Mach' Operating System modularized the kernel using the **Microkernel** approach.
- This method structure the operating system by removing all non-essential components from the kernel and *implementing* them as system and user-level programs. This results in a smaller kernel
- The main function of the microkernel is to provide a communication facility between the client program and the various services that are also running in user space.
- Benefit of microkernel approach is easy of extending the operating system. All new services are added to user space and do not require modification of the kernel.
- Microkernel can suffer from performance decreases due to increased system function overhead.

Modules

- In this object oriented programming technique is used to create a modular kernel.
- Here, kernel has a set of core components and dynamically links additional services either during boot time or during run time.

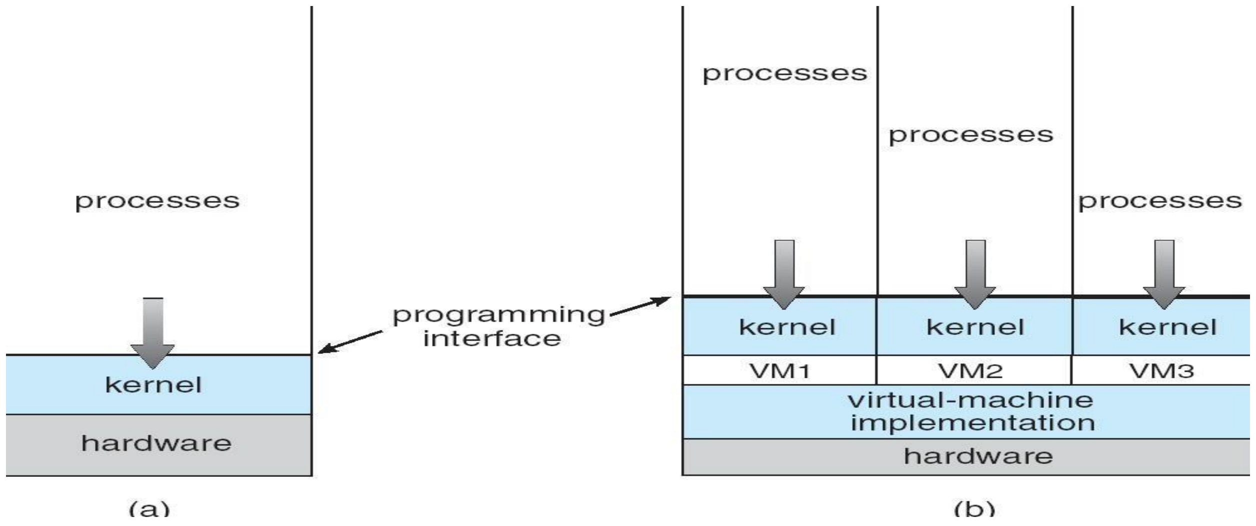
Ex: Solaris, Linux, Mac OS



- ❖ Solaris operating system structure is organized around a core kernel with seven types of loadable kernel modules.
- ❖ Such a design allow the kernel to provide core services yet also allows certain features to be implemented dynamically.
- ❖ It is more efficient than micro kernel approach

Virtual Machines

- ❖ Concept behind a virtual machine is to abstract the hardware of a single computer into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer
- ❖ Virtual machine approach provides are interface that is identical to the underlying hardware.
- ❖ With the help of virtual machines it is possible to share the same hardware among different operating system.
- ❖ Major difficulties with virtual m/cs are allocating disk system.



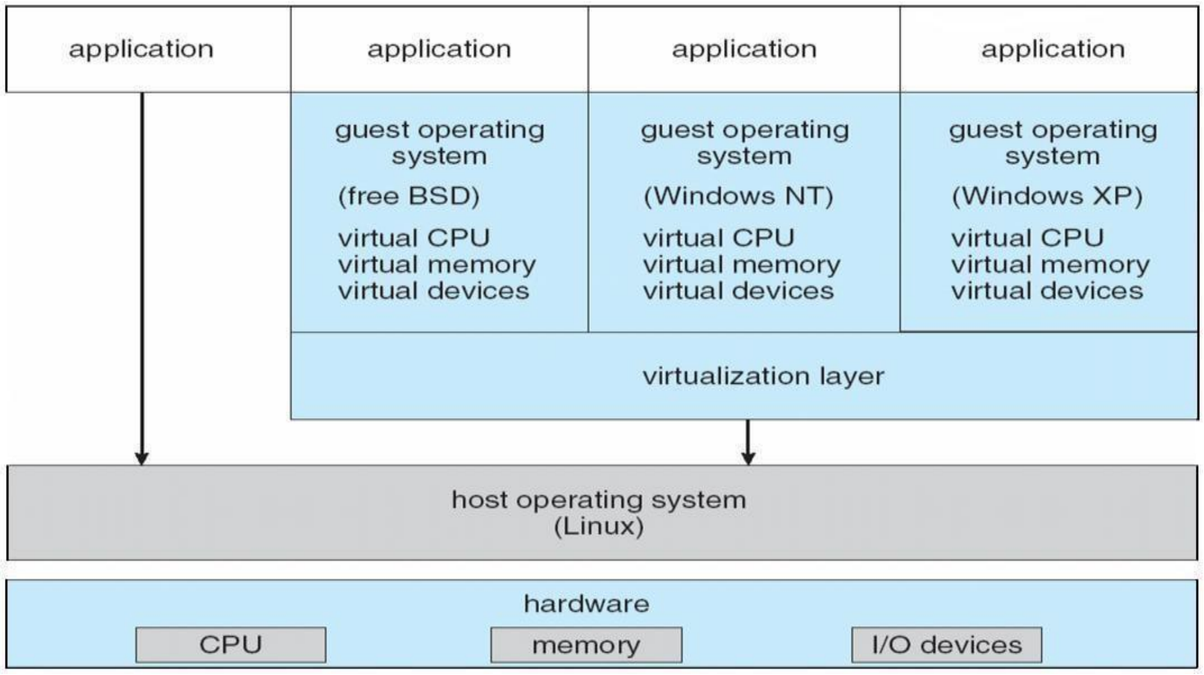
System Models- a) Non-virtual Machine b) Virtual Machine.

- ❖ Virtual machine concept has several advantages.
  - ✓ Each VM is completely isolated from all other VM's .So no protection problem
  - ✓ There is no direct sharing of resource.
  - ✓ Virtual machine system is a perfect vehicle for operating system research and development

Example: **VMware**

- ❖ VMwareabstractsIntel8086hardwareintoisolatedvirtualmachines
- ❖ VMware runs as an application on host operating system such as windows or Linux and allows this host system to concurrently run several different quest operating system as independent virtual machines.

- ❖ In the architecture diagram Linux is running as the host Operating system.
- ❖ Free BSD, Windows NT and Windows XP are running as quest operating



systems.

VMware Architecture

The JAVA Virtual Machine

Java is a popular object oriented programming language. For each Java class, the compiler produced an architecture neutral byte code output (.class) file that will run on any implementation of the JVM.

JVM is a specification for an abstract computer. It consists of a class loader and a Java interpreter that executes the architecture neutral byte codes.

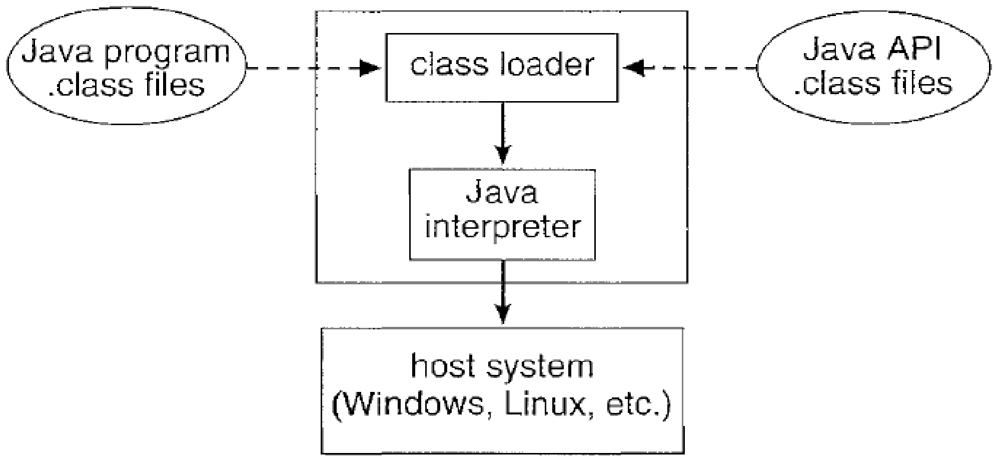


Figure 2.20 The Java virtual machine.

Operating System Generation

- ❖ Operating system is designed to run on class of machines at a variety of sites with a variety of peripheral configurations.
- ❖ Thesystemmustbeconfiguredorgeneratedforeachspecificcomputersites.
- ❖ This process is referred as system generation.(SYSGEN)
- ❖ This process requires the information like
    - o Type of CPU used
    - o Memory space available in the system
    - o Available device list
    - o Required Operating System Option

System Booting

The procedure of starting a computer by loading the kernel is known as booting the system. Bootstrap program or Bootstrap loader locates the kernel loads it into main memory and start its execution.

Bootstrap program is in the form of read only memory (ROM) because the RAM is in unknown state at a system Start up. All forms of ROM are knows as

FIRMWARE. For large OS like Windows, Mac OS, the Bootstrap loaders is stored in firmware and the OS is on disk. Bootstrap has a bit code to read a single block at a fixed location from disk into the memory and execute the code from that boot.