INDEX

UNIT-I

Operating System Introduction: Operating Systems Objectives and functions, Computer System Architecture, OS Structure, OS Operations, Evolution of Operating Systems - Simple Batch, Multi programmed, time shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, Special - Purpose Systems, Operating System services, user OS Interface, System Calls, Types of System Calls, System Programs, Operating System Design and Implementation, OS Structure, Virtual machines

A computer system is a collection of hardware and software components designed to provide an effective tool for computation.

Hardware generally refers to the electrical, mechanical and electronic parts that make up the computer(*i.e.*, Internal architecture of the computer (or) physical computing equipment).However the hardware is sophisticated, it cannot function properly without a proper driver which can drive it and bring it to the best advantage. For example, a car, even though sophisticated in its features, it cannot function independently without being properly driven by an efficient driver.

Similarly the hardware though technologically innovative, and which presents enhanced features, which needs set of programs to bring it to operation and to the best advantage. So, the driver that drives the hardware is software.

Software refers to the set of programs written to provide services to the system. It gives life and meaning to the hardware and bring it to the operational level, which otherwise is a useless piece of metal.

Software is basically of two types:

1. Application software
2. System software

Application Software: Set of programs written for a specific area of application. For example, word processors, spreadsheets and data base management systems, etc.

System Software: Set of programs written from the point of view of the machine *i.e.*, for the sake of the system. System software provides environment for execution of application software. One cannot aim to develop or write application software, without the presence and aid of system software.

NEED OF AN OPERATING SYSTEM

Operating system is an interface between user and hardware. OS creates user friendly environment.

Suppose when working with DOS-OS, if the user want to delete the program ,he has to type the command C:\DEL FILENAME and press the enter, then the program will be deleted. So ,the user delete the program very easily with the help of OS.

Suppose user want to delete the program without using OS, then he has to write a separate program for DEL command and perform the operation. Every time for doing any operation he has to write a separate program. So ,it is very difficult for the programmer, for that OS provides user friendly environment ,it is the main function of the OS. For example, MS-DOS provides different commands for performing different operations.

When the user sends a command, the OS must make sure that the command is executed or if it is not executed, must arrange for the user to get a message about explaining the error.

Another important function is resource management. The OS acts like a government, the government collects money from various resources and distribute to the different development activities. Similarly the OS collects all resources in the network environment and allocates the resources to requesting processes in an efficient manner. So, it is called as "*Resource Manager*".

The OS controls and co-ordinates the execution of the programs. So, it is sometimes called as Control program (It provides interface to various hardware components such as printer, monitor, keyboard, etc. So, it can able to control the execution of a program).

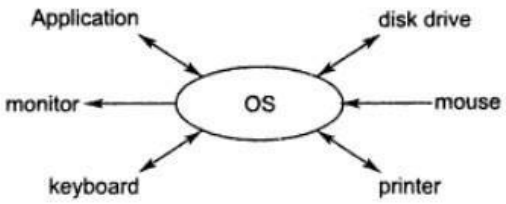
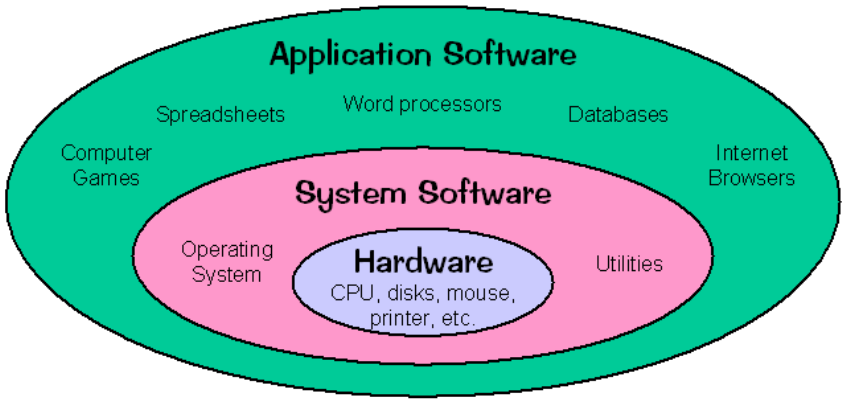


Fig. *OS Acts as Control Program*



OBJECTIVES OF O.S (GOALS)

The OS has 3 main objectives.

- **Convenience.** An OS makes a computer more convenient to the user for using. (Easy-to-use commands, graphical user interface(GUI))
- **Efficiency.** An OS allows the computer system resources to be used in an efficient manner, to ensure good resource utilization efficiency, and provide appropriate corrective actions when it becomes low.
- **Ability to evolve.** An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions without interfering with service.

Operating system performs the following functions:

1. Booting

Booting is a process of starting the computer operating system starts the computer to work. It checks the computer and makes it ready to work.

2. Memory Management

It is also an important function of operating system. The memory cannot be managed without operating system. Different programs and data execute in memory at one time. if there is no operating system, the programs may mix with each other. The system will not work properly.

3. Loading and Execution

A program is loaded in the memory before it can be executed. Operating system provides the facility to load programs in memory easily and then execute it.

4. Data security

Data is an important part of computer system. The operating system protects the data stored on the computer from illegal use, modification or deletion.

5. Disk Management

Operating system manages the disk space. It manages the stored files and folders in a proper way.

6. Process Management

CPU can perform one task at one time. if there are many tasks, operating system decides which task should get the CPU.

7. Device Controlling

operating system also controls all devices attached to computer. The hardware devices are controlled with the help of small software called device drivers..

8. Providing interface

It is used in order that user interface acts with a computer mutually. User interface controls how you input data and instruction and how information is displayed on screen. The operating system offers two types of the interface to the user:

1. Graphical-line interface: It interacts with of visual environment to communicate with the computer. It uses windows, icons, menus and other graphical objects to issues commands.
2. Command-line interface:it provides an interface to communicate with the computer by typing commands.

Computer System Architecture

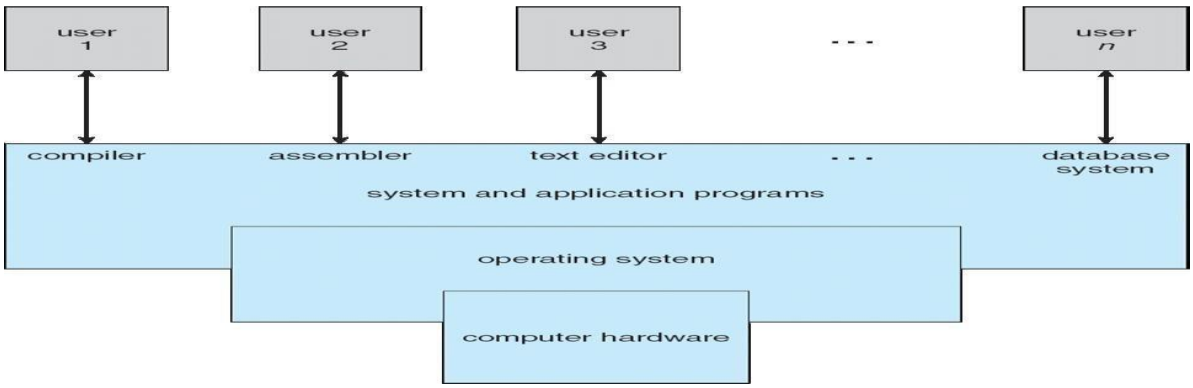
Computer system can be divided into four components Hardware – provides basic computing resources

- CPU, memory, I/O devices, Operating system

Controls and coordinates use of hardware among various applications and users

- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
- Word processors, compilers, web browsers, database systems, video games Users
- People, machines, other computers Four

Components of a Computer System



Computer architecture means construction/design of a computer. A computer system may be organized in different ways. Some computer systems have single processor and others have multiprocessors. So based on the processors used in computer systems, they are categorized into the following systems.

1. Single-processor system
2. Multiprocessor system
3. Clustered Systems:

1. Single-Processor Systems:

Some computers use only one processor such as microcomputers (or personal computers PCs). On a single-processor system, there is only one CPU that performs all the activities in the computer system. However, most of these systems have other special purpose processors, such as I/O processors that move data quickly among different components of the computers. These processors execute only a limited system programs and do not run the user program. Sometimes

they are managed by the operating system. Similarly, PCs contain a special purpose microprocessor in the keyboard, which converts the keystrokes into computer codes to be sent to the CPU. The use of special purpose microprocessors is common in microcomputer. But it does not mean that this system is multiprocessor. A system that has only one general-purpose CPU, is considered as single- processor system.

2. Multiprocessor Systems:

In multiprocessor system, two or more processors work together. In this system, multiple programs (more than one program) are executed on different processors at the same time. This type of processing is known as multiprocessing. Some operating systems have features of multiprocessing. UNIX is an example of multiprocessing operating system. Some versions of Microsoft Windows also support multiprocessing.

Multiprocessor system is also known as parallel system. Mostly the processors of multiprocessor system share the common system bus, clock, memory and peripheral devices. This system is very fast in data processing.

Types of Multiprocessor Systems:

The multiprocessor systems are further divided into two types; (i). Asymmetric multiprocessing system
(ii). Symmetric multiprocessing system

(i) Asymmetric Multiprocessing System(AMS):

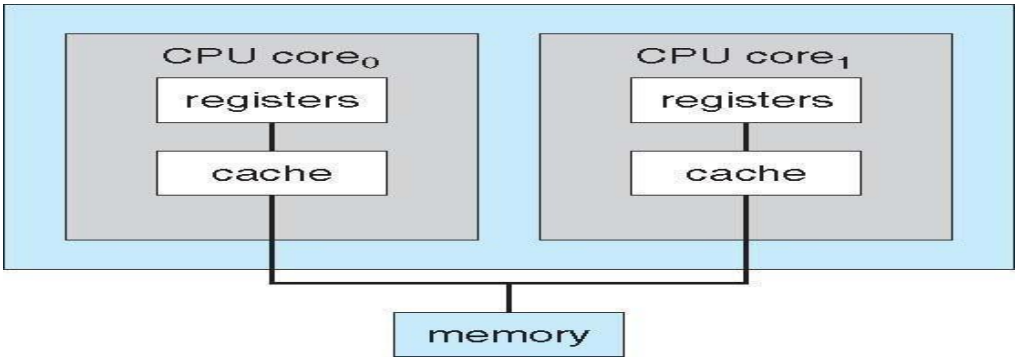
The multiprocessing system, in which each processor is assigned a specific task, is known as Asymmetric Multiprocessing System. For example, one processor is dedicated for handling user's requests, one processor is dedicated for running application program, and one processor is dedicated for running image processing and so on. In this system, one processor works as master processor, while other processors work as slave processors. The master processor controls the operations of system. It also schedules and distributes tasks among the slave processors. The slave processors perform the predefined tasks.

(ii) Symmetric Multiprocessing System(SMP):

The multiprocessing system, in which multiple processors work together on the same task, is known as Symmetric Multiprocessing System. In this system, each processor can perform all types of tasks. All processors are treated equally and no master-slave relationship exists between the processors.

For example, different processors in the system can communicate with each other. Similarly, an I/O can be processed on any processor. However, I/O must be controlled to ensure that the data reaches the appropriate processor. Because all the processors share the same memory, so the input data given to the processors and their results must be separately controlled. Today all modern operating systems including Windows and Linux provide support for SMP.

It must be noted that in the same computer system, the asymmetric multiprocessing and symmetric multiprocessing technique can be used through different operating systems.



A Dual-Core Design

3. Clustered Systems:

Clustered system is another form of multiprocessor system. This system also contains multiple processors but it differs from multiprocessor system. The clustered system consists of two or more individual systems that are coupled together. In clustered system, individual systems (or clustered computers) share the same storage and are linked together ,via Local Area Network (LAN).

A layer of cluster software runs on the cluster nodes. Each node can monitor one or more of the other nodes over the LAN. If the monitored machine fails due to some technical fault (or due to other reason), the monitoring machine can take ownership of its storage. The monitoring machine can also restart the applications that were running on the failed machine. The users of the applications see only an interruption of service.

Types of Clustered Systems:

Like multiprocessor systems, clustered system can also be of two types (i). Asymmetric Clustered System
(ii). Symmetric Clustered System

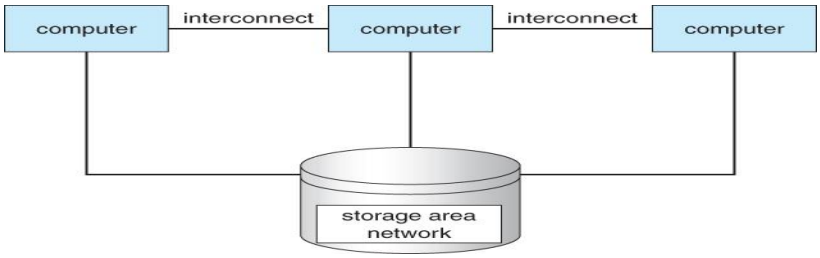
(i). Asymmetric Clustered System:

In asymmetric clustered system, one machine is in hot-standby mode while the other

machine is running the application. The hot-standby host machine does nothing. It only monitors the active server. If the server fails, the hot-standby machine becomes the active server.

(ii). Symmetric Clustered System:

In symmetric clustered system, multiple hosts (machines) run the applications. They also monitor each other. This mode is more efficient than asymmetric system, because it uses all the available hardware. This mode is used only if more than one application be available to run.



Operating System – Structure

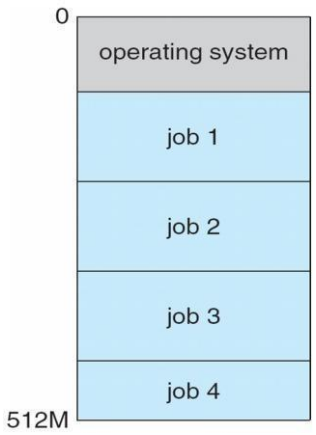
Operating System Structure

- **Multiprogramming** needed for efficiency
- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to
- Execute A subset of total jobs in system is kept in memory

Multiprogramming

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Following figure shows the memory layout for a multiprogramming system.



Operating system does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the job in the memory.
- Multiprogramming operating system monitors the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle unless there are no jobs

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

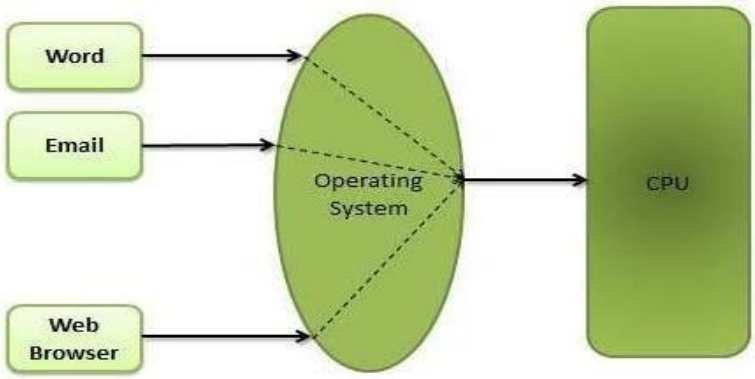
- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

2) Multitasking

Multitasking

Multitasking refers to term where multiple jobs are executed by the CPU simultaneously by switching between them.Switches occur so frequently that the users may interact with each program while it is running. Operating system does the following activities related to multitasking.

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- Operating System handles multitasking in the way that it can handle multiple operations / executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
- A time-shared operating system uses concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- Each user has at least one separate program in memory.



- A program that is loaded into memory and is executing is commonly referred to as a process.
- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.

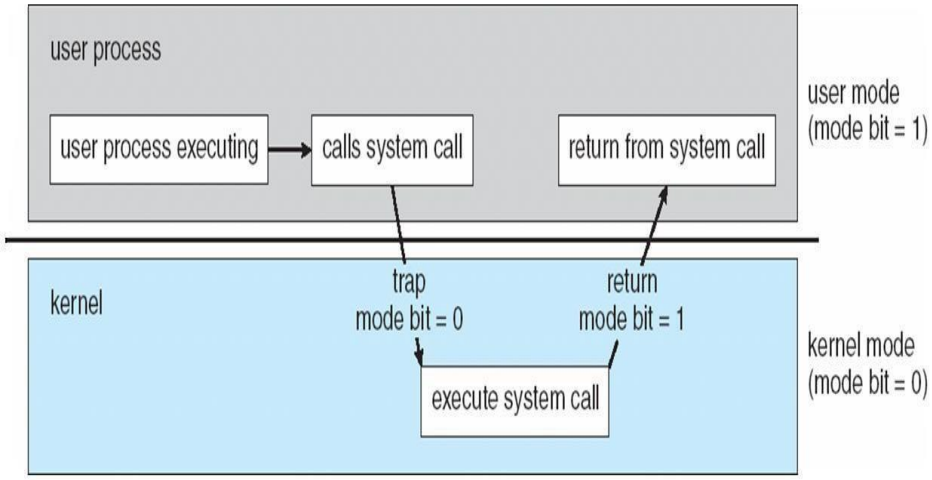
- Since interactive I/O typically runs at people speeds, it may take a long time to completed. During this time a CPU can be utilized by another process.
- Operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Operating-system Operations

1) Dual-Mode Operation·

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. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.

At the very least we need two separate modes of operation.user mode and kernel 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).with the mode bit we are able to distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user, 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), it 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). Thus, whenever the operating system gains control of the computer, it is in kernel mode. The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program.

The dual mode of operation provides us with the means for protecting the operating system from errant users-and errant users from one another. We accomplish this protection by designating some of the machine instructions that may cause harm as privileged instructions. 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/0 control timer management and interrupt management.

Timer

We must ensure that the operating system maintains control over the CPU. We must prevent a user program from getting stuck in an infinite loop or not calling system services and never returning control to the operating system. To accomplish this goal, we can use a **timer**. A timer can be set to interrupt the computer after a specified period.

Before turning over control to the user, the operating system ensures that the timer is set to interrupt. If the timer interrupts, control transfers automatically to the operating system, which may treat the interrupt as a fatal error or may give the program more time. Clearly, instructions that modify the content of the timer are privileged.

Thus, we can use the timer to prevent a user program from running too long.

EVOLUTION OF OPERATING SYSTEMS

Operating system and computer architecture have had a great deal of influence on each other. Operating systems were developed mainly to facilitate the use of the hardware and to bring it to the best advantage. Here we will briefly make a sketch of the evolutionary path of OS development.

Serial Processing

Before 1950's the programmers directly interact with computer hardware, there was no OS at that time. If the programmer want to execute the program on those days, he has to follow some serial steps:

- Type the program on punched card.
- Convert the punched card to card reader.
- Submit to the computing machine, if any error in the program, the error condition was indicated by lights.
- The programmer examine the registers and main memory to identify the cause of error.
- Take the output on the printers.
- Then the programmer is ready for the next program.

This type of processing is difficult for users, it takes much time and next program should wait for the completion of previous one. The programs are submitted to the machine one after the other. So, this method is called as "Serial processing".

Batch Processing

In olden days(before 1960's), it is difficult to execute a program using computer. Because the computer is located in different rooms, one room for card reader and one for executing the program and another room for printing the output. The user or machine operator, running between these three rooms to complete a job. This problem was solved by batch processing system.

In batch processing technique similar type of jobs batch together and execute at a time. The operator carries the group of jobs at a time from one room to another. Therefore the programmer need not run between these three rooms several times.

The batch processing had an advantage .In that for one batch, the compiler, assembler, the loader etc had to be loaded only once, thus reducing the setup time to some extent. For example, FORTRAN programs were grouped together as one batch say batch 1, the PASCAL programs into another batch say Batch 2, the COBOL programs into another batch say Batch 3, and so on. Now the operator can arrange for the execution of these source programs which has been batched together one by one. After the execution of batch1 was over, the operator would load the compiler, assembler and loader, etc for the batch 2 and so on.

Setup time for batch 1	Runtime for batch 1	Setup time for batch 2	Runtime for batch 2	...	...	...

Fig. *Batch Processing*

The main advantage of batch processing is setup time will be reduced to a large extent, but the disadvantage is that the CPU is idle for the time in between two batches.

If the programs were not batched up together, the set up time would be much more higher.

Setup time for program 1	Runtime for program 1	Setup time for program 2	Runtime for program 2	...	...	...

C

Multiprogramming

Multiprogramming is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor, there can be no true simultaneous execution of different programs. Instead the processor executes part of one program, then part of another, and so on. But to the user it appears that all programs are executing at the same time.

In multiprogramming, number of processes are reside in main memory at a time. The OS picks and begins to execute one of the jobs in the main memory. For example, consider the main memory consisting of 5 jobs at a time, the CPU executes one by one.

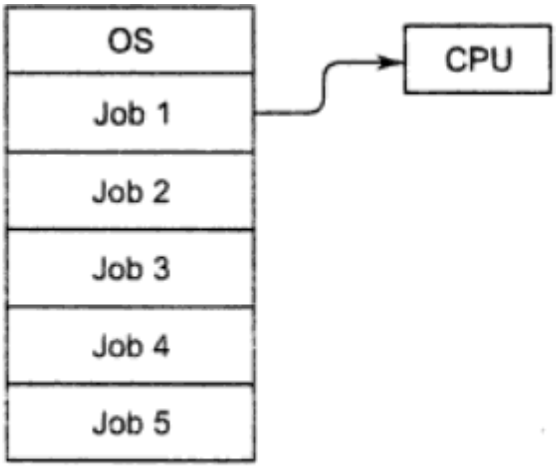


Fig. *Multiprogramming*

In non-multiprogramming system, the CPU can execute only one program at a time, if the running program waiting for any I/O device, the CPU becomes idle, so it will effect on the performance of the CPU.

But in multiprogramming environment, any I/O wait happened in a process, then the CPU switches from that job to another job in the job pool. If enough jobs could be held in main memory at once, the CPU is not idle at any time.

For Example: The idea is common in other life situations. The doctor does not have only one patient at a time, number of patients reside in the hospital under treatment. If the doctor has enough patients a doctor never needs to be idle.

Distributed Systems

A recent trend in computer system is to distribute computation among several processors. The processors in distribute system may vary in size and function, and referred by a number of different names such as sites, nodes, computers and so on depending on the context.

A distributed system is basically a collection of autonomous (independent by function) computer systems which co-operate with one another through their hardware and software interconnections.

In distributed systems, the processors cannot share memory or time, each processor has its own local memory. The processors communicate with one another through various communication lines such as high speed buses .These systems are also called as *"Loosely Coupled systems"*.

Distributed system = Network + Transparency(Invisible)

Advantages

1. **Resource sharing:** If a number of sites connected by a high speed communication lines, it is possible to share the resources from one site to another site.

   For example, S_1 and S_2 are two sites, these are connected by some communication lines, the site S_1 having the printer, but S_2 does not having the printer. Then the system can use the printer at S_1 without moving from S_2 to S_1.Therefore resource sharing is possible in distributed systems.
2. **Computation speedup:** A big computation is partitioned into number of partitions, these sub-partitions run concurrently in distributed systems.

   For example, site S_1 need to execute a big computation, this computation is divided into sub computations and these are executed by some other machines in different sites.
3. **Reliability:** If a resource or a system failed in one site due to technical problems. We can use other systems or other resources in some other sites.
4. **Communication:** Distributed systems provides communication which is not at all possible, that much in a centralized system. For Example, E-mail

Time Sharing Systems

Multiprogramming features were superimposed on batch processing to ensure good utilization of CPU but from the point of view of a user the service was poor as the response time, *i.e*, the time elapsed between submitting a job and getting the results was unacceptably high. Development of interactive terminals changed the scenario. Computation became an on-line activity. A user could provide inputs to a computation from a terminal and could also examine the output of the computation on the same terminal. Hence the response time needed to be drastically reduced. This was achieved by storing programs of several users in memory and providing each user a slice of time on CPU to process his/her program.

Time sharing or multitasking is a logical extension of multiprogramming. In time sharing environment, a number of jobs are loaded on to the memory and a number of users are communicating with the computer through different terminals. The OS allocates a fixed time interval (TIME SLICE) to each program in memory. Thus each program in memory is executed for a fixed interval of time.

As soon as the time allotted for a particular program is completed, the CPU starts executing the next program. This process is continued till all the programs in the memory are executed. A program may need number of time slices for its complete execution. Although the computer system is executing one job at a time, due to the speed of the CPU, every user on a terminal has the feeling that his program that is being executed continuously, because, after every time slice, the user gets a response from the computer. The user on the terminal is communicating with his running program, and is able to debug and experiment with his program.

Thus, the OS for a time sharing computer system has all the capabilities of a multiprogramming OS, but along with an additional capacity of allocating a fixed time slice of CPU to each program.

- Main advantage of time sharing system is efficient CPU utilization.
- The user can interact with the job while it is executing, but it is not possible in batch systems.

Personal-Computer Systems(PCs)

A personal computer (PC) is a small, relatively inexpensive computer designed for an individual user. In price, personal computers range anywhere from a few hundred dollars to thousands of dollars. All are based on the microprocessor technology that enables manufacturers to put an entire CPU on one chip.

At home, the most popular use for personal computers is for playing games. Businesses use personal computers for word processing, accounting, desktop publishing, and for running spreadsheet and database management applications.

Parallel Systems

Almost all the systems are uni-processor systems *i.e.*, they have only one CPU. Systems in which there are more than one CPU is called as Multi-processor systems. These systems have been developed to enhance the computing power of a computing system, and the features of this system is that, they share the memory, bus and the peripheral devices. These systems are referred as *"Tightly coupled systems"*. A system consisting of more than one processor and it is a tightly coupled, then the system is called "Parallel system".

In parallel systems number of processors executing their jobs in parallel (simultaneous process). Multi-processor systems are divided into following categories:

- Symmetric
- Asymmetric

In symmetric multi-processing, each processor runs a shared copy of operating system . The processors can communicate with each other and execute these copies concurrently. Thus, in a symmetric system, all the processors share an equal amount of load .Encore's version of UNIX for the Multimax computer is an example of symmetric multiprocessing. In this system various processors execute copies of UNIX operating system, thereby executing M processes if there are M processors.

Asymmetric multi-processing is based on the principle of master-slave relationship. In this system, one of the processors runs the operating system and that processor is called the master processor. Other processors run user processes and are known as slave processors. In other words, the master processor controls, schedules and allocates the task to the slave processors. Asymmetric multi-processing is more common in extremely large systems, where one of the time consuming tasks is processing I/O requests. In the asymmetric systems the processors do not share the equal load.

Advantages:

1. It results in saving money compared to the stand alone systems, since CPU'S can share memory, bus and peripherals.
2. Throughput can be increased
3. They increase the reliability.

Since there are more than one CPU, the failure of one or more of the CPU does not halt the entire system, but only slows down the work. For example, if there are five processors, all the five working together gives full efficiency. If two CPU's fail, then the system still works but only at 60% efficiency. This indicates increased aspect of reliability compared to stand alone systems.

Special purpose systems

a) Real-Time EmbeddedSystems

These devices are found everywhere, from car engines and manufacturing robots to DVDs and microwave ovens. They tend to have very specific tasks.

They have little or no user interface, preferring to spend their time monitoring and managing hardware devices, such as automobile engines and robotic arms.

b) Multimedia Systems

Most operating systems are designed to handle conventional data such as text files, programs, word-processing documents, and spreadsheets. However, a recent trend in technology is the incorporation of multimedia data into computer systems. Multimedia data consist of audio and video files as well as conventional files. These data differ from conventional data in that multimedia data-such as frames of video-must be delivered (streamed) according to certain time restrictions (for example, 30 frames per second). Multimedia describes a wide range of applications in popular use today. These include audio files such as MP3, DVD movies, video conferencing, and short video clips of movie previews or news stories downloaded over the Internet. Multimedia applications may also include live webcasts (broadcasting over the World Wide Web)

c) Hand held Systems

Handheld Systems include personal digital assistants (PDAs, cellular telephones. Developers of handheld systems and applications face many challenges, most of which are due to the limited size of such devices. For example, a PDA is typically about 5 inches in height and 3 inches in width, and it weighs less than one-half pound. Because of their size, most handheld devices have small amounts of memory, slow processors, and small display screens.

REAL-TIME OS

In a time shared computer system, generally the computer response time is of the order of 0.5 to 2 seconds, which means a user will get computers attention after this much of time. Longer response times may be irritating but not hazardous.

However a real-time OS is needed for the computer systems controlling a process or a real time situation, such as a machine or a satellite. In this case two important points to be noticed are:

- The OS should provide for interactive processing.
- The response time should be very small.

The sensors bring in the data from a device, the OS instructs the computer to analyze the data and send appropriate signals back to the device. Any delay on the part of the computer system or the OS can be catastrophic. Thus, the real-time OS have to work strict time limits and have to be quick. Apart from this, these systems must be highly reliable to avoid failure of the system being controlled.

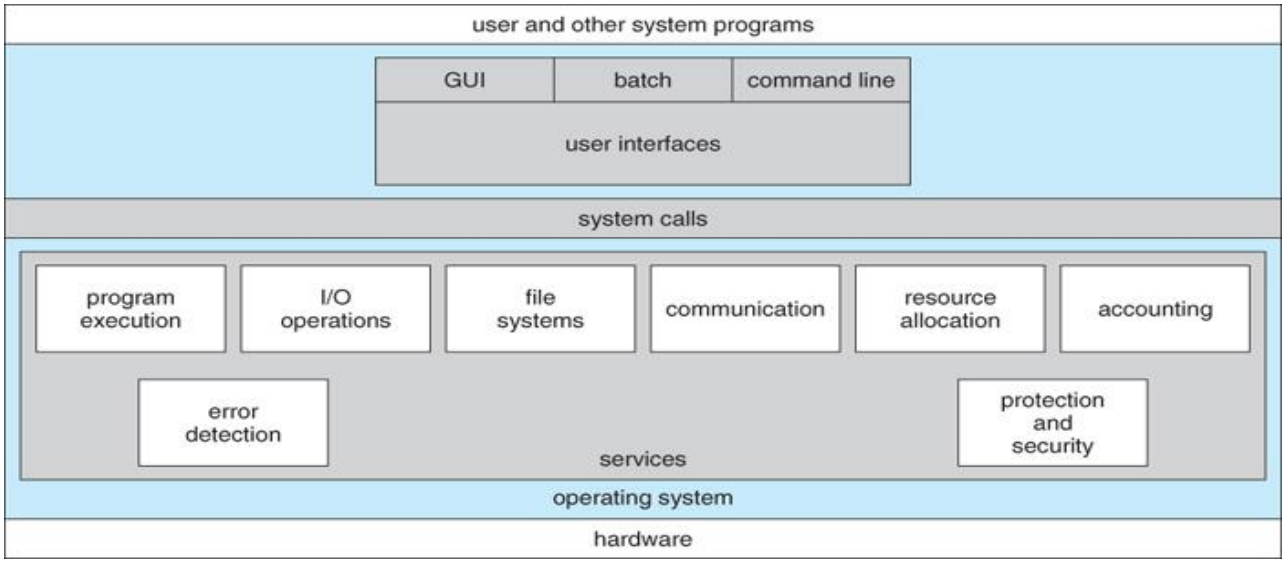
Here the main job of OS is instant handling of the signals or interrupts sent by the device which is being controlled by the computer system.

Real-time systems are systems that have in-built characteristics as supplying immediate response. A primary objective of the real-time system is to provide quick response time. User convenience and resource utilization are of secondary concern to real-time systems.

Real time System is of two types:

- **Hard real-time**
  - Guarantees that critical tasks complete within time.
  - All the delays in the system are bounded.
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- **Soft real-time**
  - Critical time tasks gets priority over other tasks, and retails that priority until it completes.
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Operating System Services



- One set of operating-system services provides functions that are helpful to the user

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

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

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

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

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

- **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

Protection involves ensuring that all access to system resources is controlled

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

User Operating System Interface - CLI

- Command Line Interface (CLI) or command interpreter allows direct command entry

Sometimes implemented in kernel, sometimes by systems program
sometimes multiple flavors implemented – shells
Primarily fetches a command from user and executes it

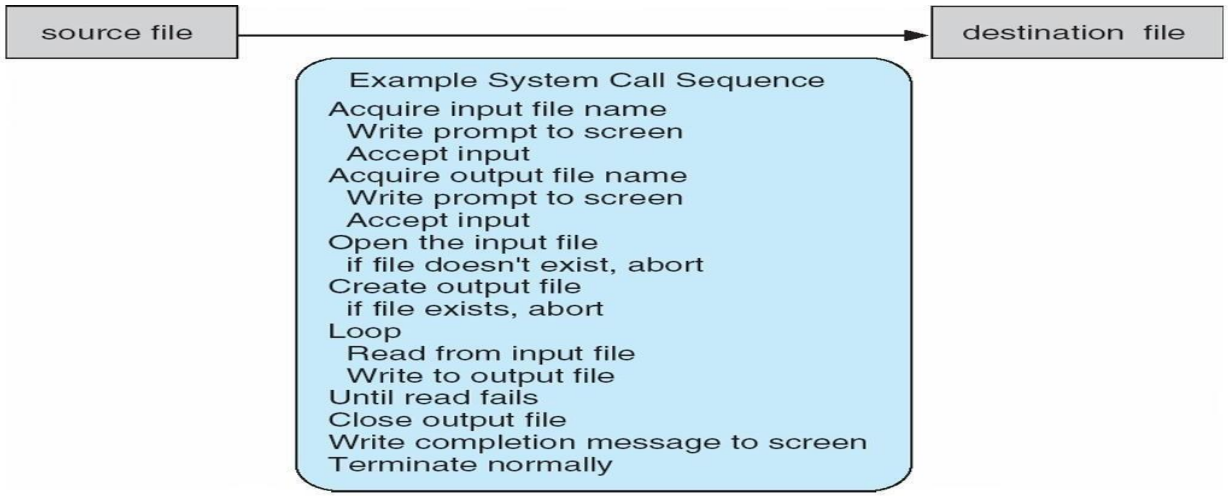
User Operating System Interface - GUI

- User-friendly desktop metaphor interface
- Usually mouse, keyboard, and monitor Icons
- represent files, programs, actions, etc
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
- Invented at Xerox PARC
- Many systems now include both CLI and GUI
- interfaces Microsoft Windows is GUI with CLI
- "command" shell
- Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call usenThree most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

Example of System Calls



Example of Standard API

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

A description of the parameters passed to ReadFile() HANDLE file—the file to be read

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

System Call Implementation

Typically, a number associated with each system call

System-call interface maintains a table indexed according to these Numbers

The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values

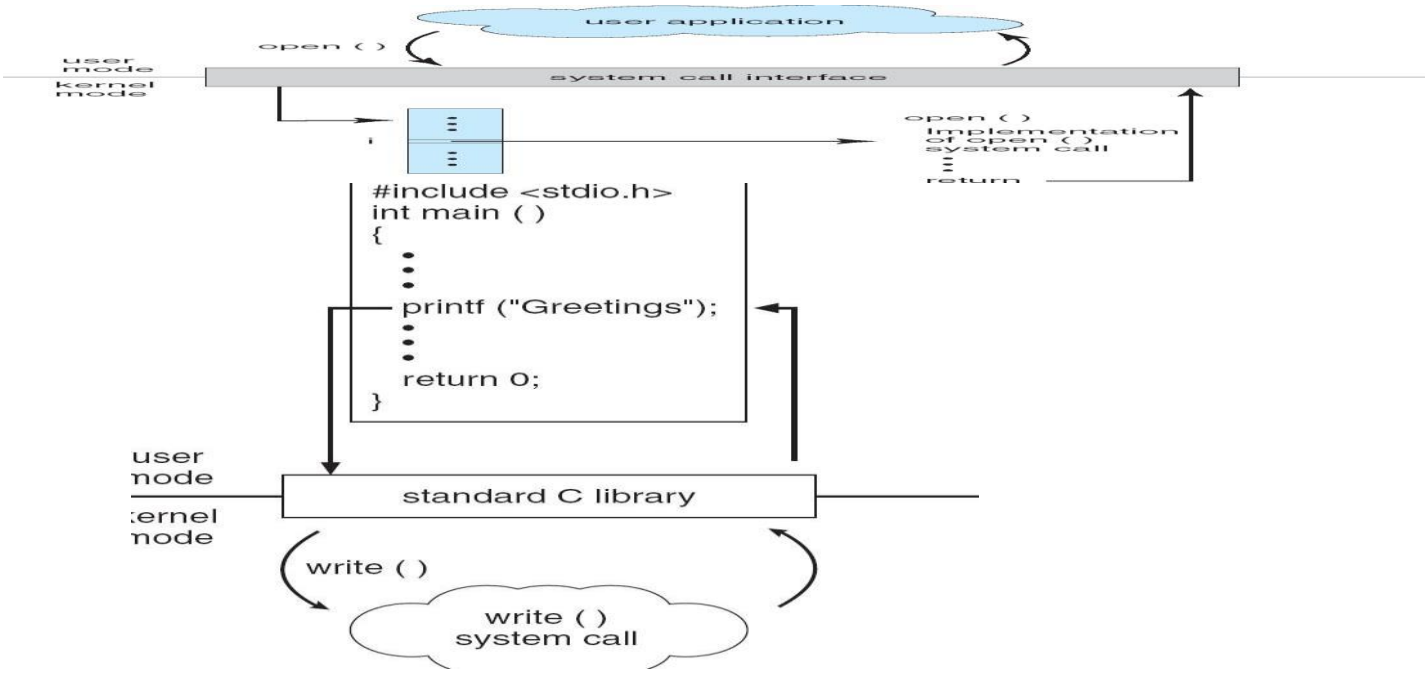
The caller need know nothing about how the system call is implemented Just needs to obey API and understand what OS will

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)

API – System Call – OS Relationship

Standard C Library Example



System Call Parameter Passing

- Often, more information is required than simply identity of desired system
- call Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the
- OS Simplest: pass the parameters in *registers*

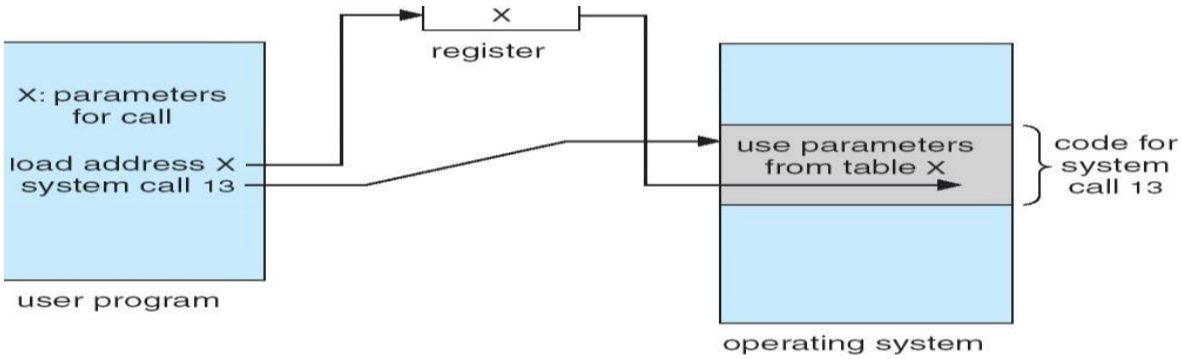
In some cases, may be more parameters than registers

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

 This approach taken by Linux and Solaris

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

Parameter Passing via Table



Types of System Calls

1. Process control
2. File management
3. Device management
4. Information maintenance
5. Communications

Process control

A running needs to halt its execution either normally or abnormally.

If a system call is made to terminate the running program, a dump of memory is sometimes taken and an error message generated which can be diagnosed by a debugger

- 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

OS provides an API to make these system calls for managing files

- create file, delete file
- open, close file
- read, write, reposition
- get and set file attributes

Device management

Process requires several resources to execute, if these resources are available, they will be granted and control retuned to user process. Some are physical such as video card and other such as file. User program request the device and release when finished

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

Information maintenance

System calls exist purely for transferring information between the user program and OS. It can return information about the system, such as the number of current users, the version number of the operating system, the amount of free memory or disk space and so on.

- get time or date, set time or date
- get system data, set system data
- get and set process, file, or device attributes

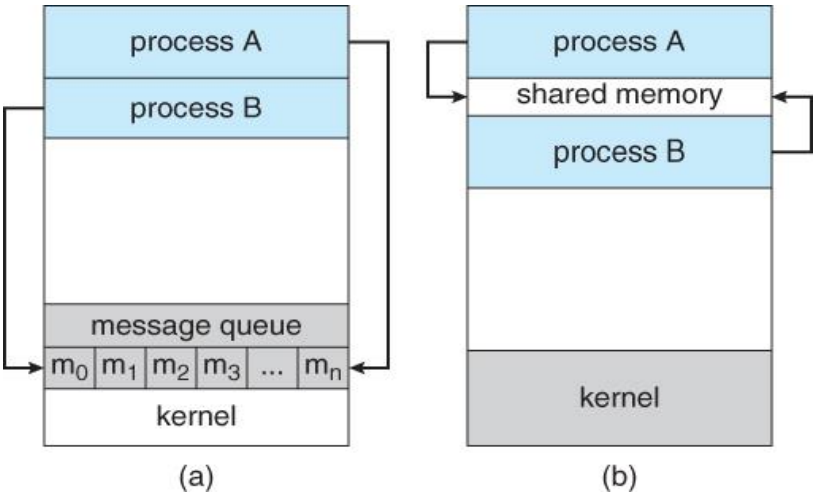
Communications

Two common models of communication

Message-passing model, information is exchanged through an inter process-communication facility provided by the OS.

Shared-memory model, processes use map memory system calls to gain access to regions of memory owned by other processes.

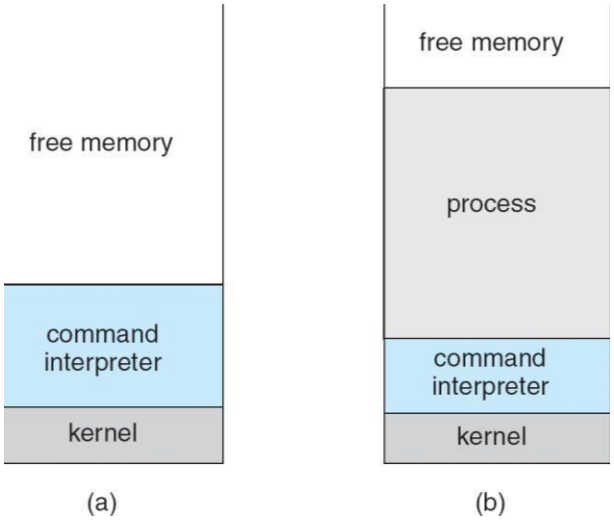
o create, delete communication connection
o send, receive messages
o transfer status information
o attach and detach remote devices



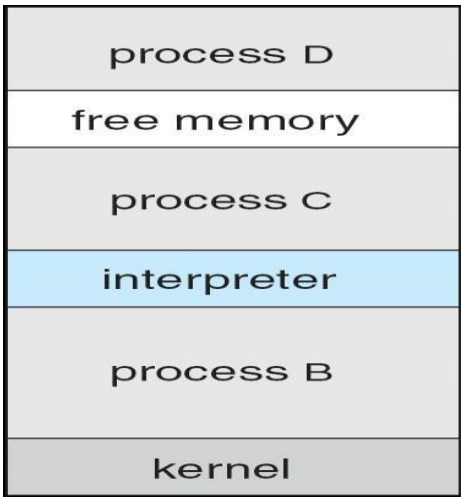
Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

MS-DOS execution



(a) At system startup (b) running a program FreeBSD Running Multiple Programs



System Programs

System programs provide a convenient environment for program development and execution. The can be divided into:

- File manipulation
- Status information
- File modification
- Programming language support
- Program loading and execution
- Communications
- Application programs

Most users' view of the operation system is defined by system programs, not the actual system calls provide a convenient environment for program development and execution

Some of them are simply user interfaces to system calls; others are considerably more complex

File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

- Status information

Some ask the system for info - date, time, amount of available memory, disk space, number of users

Others provide detailed performance, logging, and debugging information

Typically, these programs format and print the output to the terminal or other output devices

Some systems implement a registry - used to store and retrieve configuration information

- File modification

Text editors to create and modify files

Special commands to search contents of files or perform transformations of the text

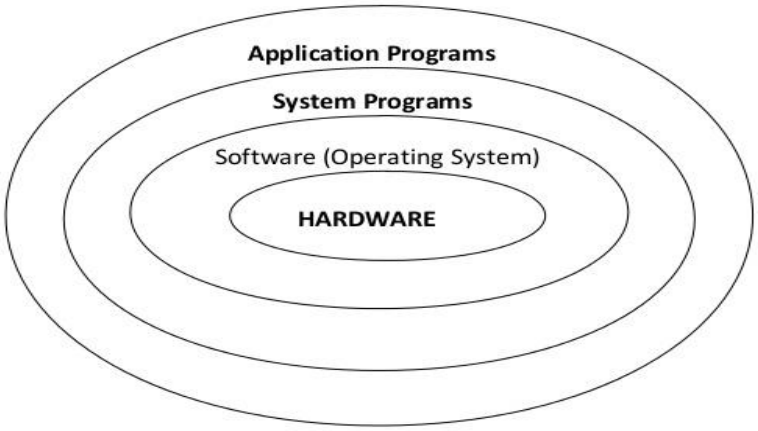
Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided

- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems

- Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

STRUCTURE OF OPERATING SYSTEM:



17

Operating System Design and Implementation

Design and Implementation of OS not "solvable", but some approaches have proven successful

Internal structure of different Operating Systems can vary widely

Start by defining goals and specifications Affected by

choice of hardware, type of system *User* goals and

System goals

User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast

System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

Important principle to separate

Policy: What will be done?

Mechanism: How to do it?

Mechanisms determine how to do something, policies decide what will be done

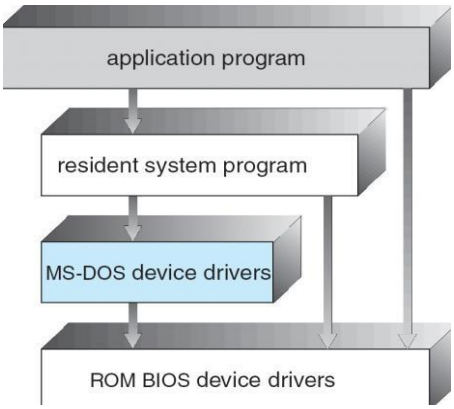
The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

Simple Structure

- MS-DOS – written to provide the most functionality in the least space Not divided into
- modules

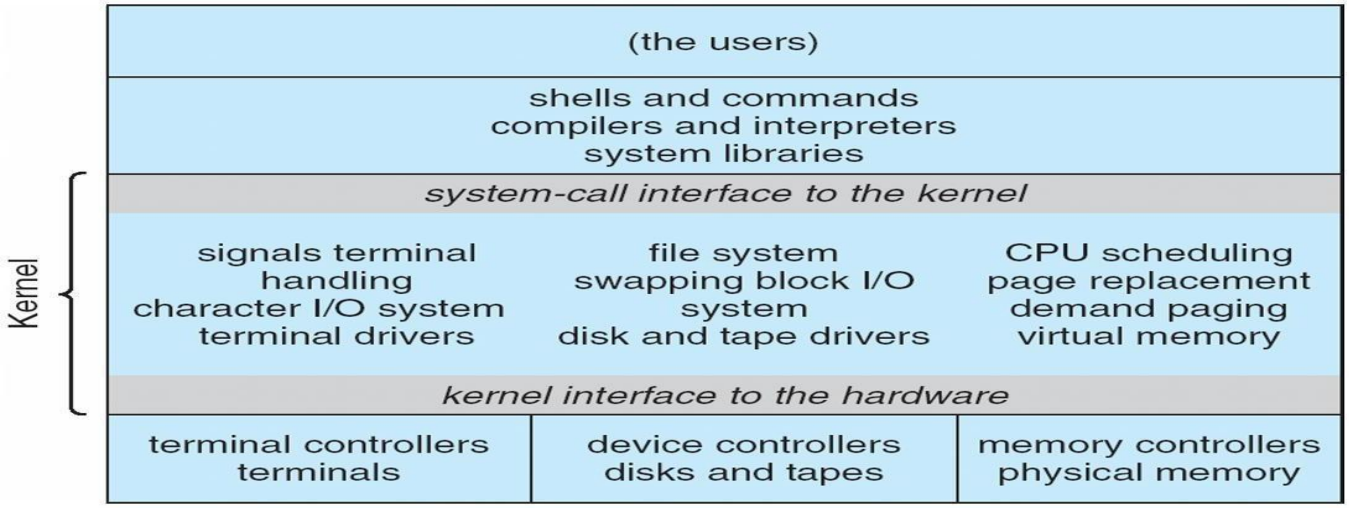
- Although MS-DOS has some structure, its interfaces and levels of Functionality are not well separated

MS-DOS Layer Structure



- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Traditional UNIX System Structure



UNIX

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.

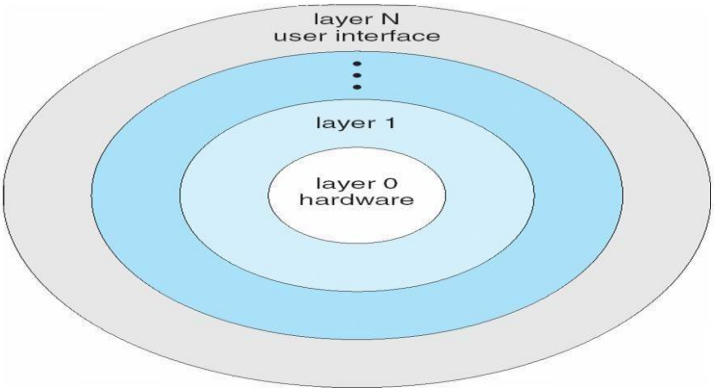
The UNIX OS consists of two separable parts

- 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

Layered Operating System



Micro kernel 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

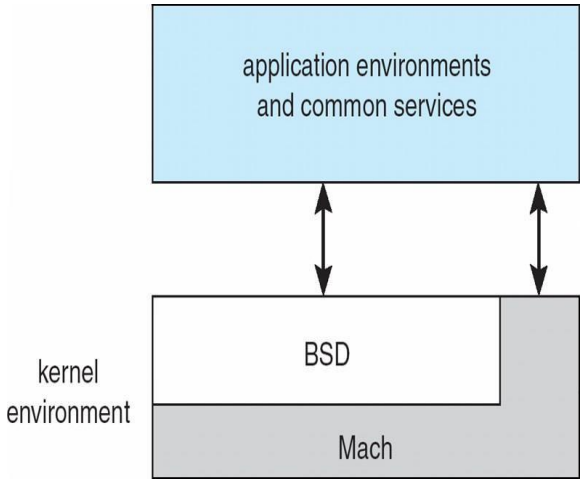
Easier to port the operating system to new architectures More reliable (less code is running in kernel mode)

More secure

Detriments:

Performance overhead of user space to kernel space communication

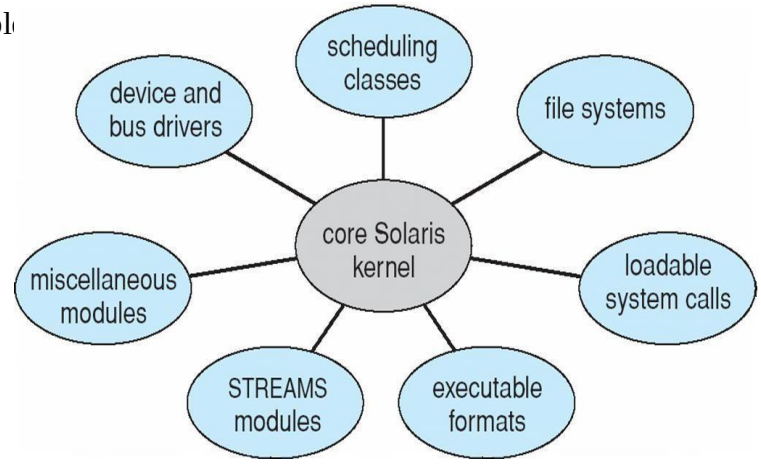
MacOS X Structure



Modules

Most modern operating systems implement kernel modules
Uses object-oriented approach
Each core component is separate
Each talks to the others over known interfaces
Each is loadable as needed within the kernel
Overall, similar to layers but with more flexibl

Solaris Modular Approach

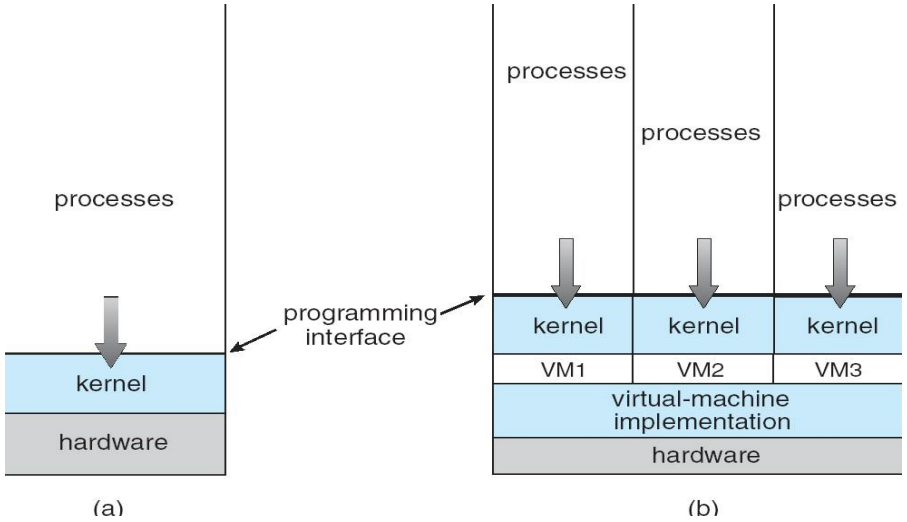


Virtual Machines

A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
A virtual machine provides an interface *identical* to the underlying bare hardware
The operating system host creates the illusion that a process has its own processor and (virtual memory)
Each guest provided with a (virtual) copy of underlying computer

Virtual Machines History and Benefits

First appeared commercially in IBM mainframes in 1972
Fundamentally, multiple execution environments (different operating systems) can share the same hardware
Protect from each other
Some sharing of file can be permitted, controlled
Commutate with each other, other physical systems via networking
Useful for development, testing
Consolidation of many low-resource use systems onto fewer busier systems
"Open Virtual Machine Format", standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms



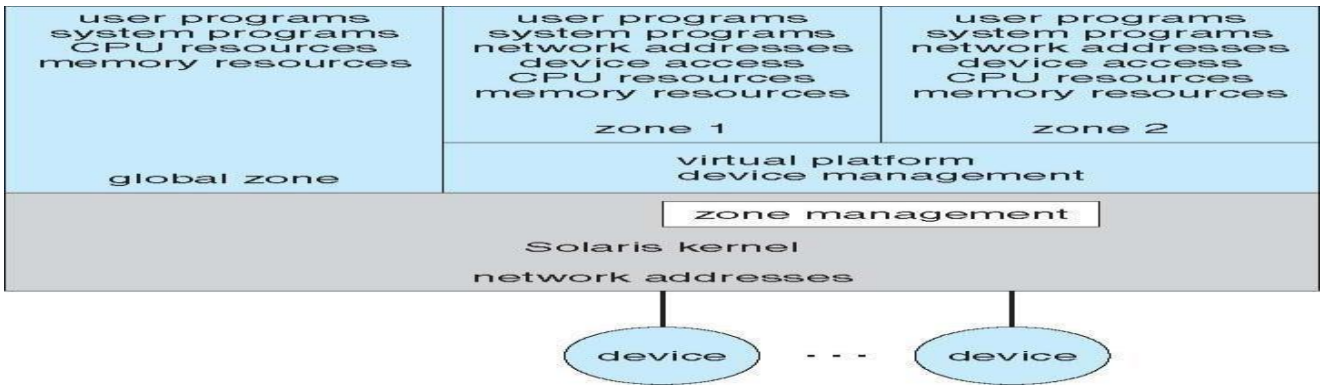
Para-virtualization

Presents guest with system similar but not identical to hardware

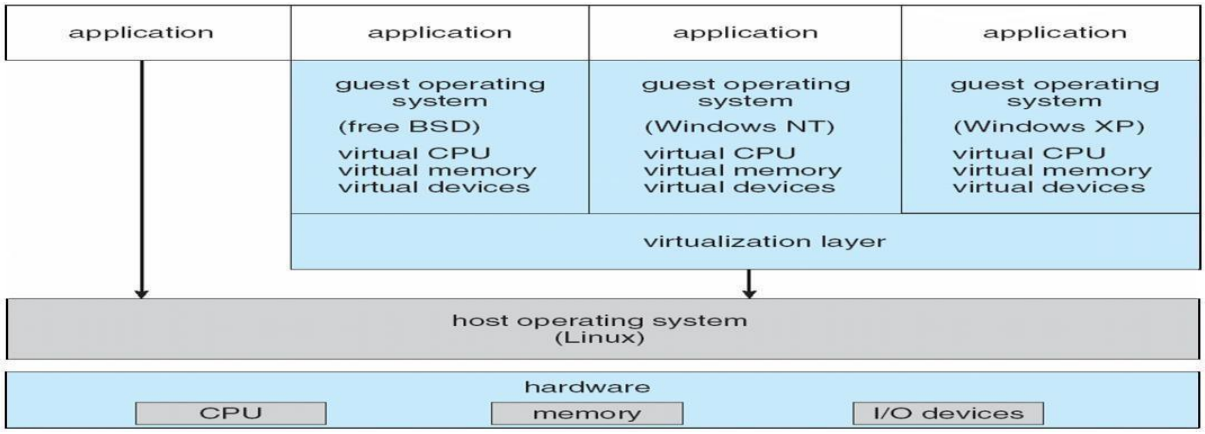
Guest must be modified to run on par virtualized hardware

Guest can be an OS, or in the case of Solaris 10 applications running in containers

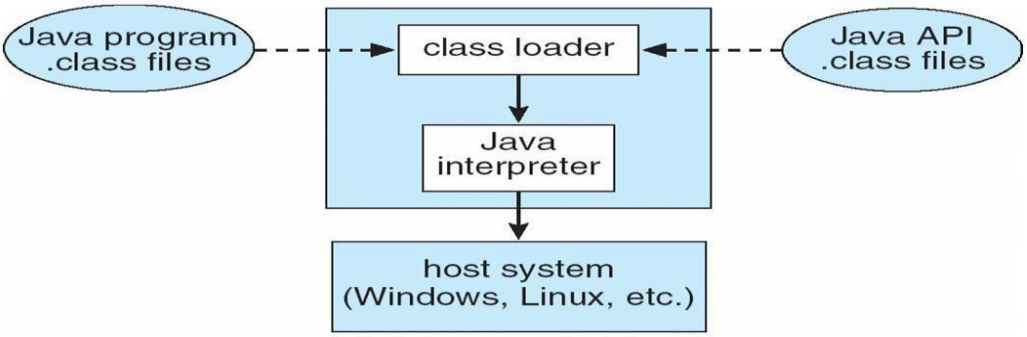
Solaris 10 with Two Containers



VMware Architecture



The Java Virtual Machine



Operating-System Debugging

Debugging is finding and fixing errors, or bugs
generate log files containing error information
Failure of an application can generate core dump file capturing memory of the process
Operating system failure can generate crash dump file containing kernel memory Beyond crashes, performance tuning can optimize system performance
Kernighan's Law: "Debugging is twice as hard as writing the code in the rst place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems
Probes fire when code is executed, capturing state data and sending it to consumers of those probes

Process

A process is a program at the time of execution.

Differences between Process and Program

Process	Program
Process is a dynamic object	Program is a static object
Process is sequence of instruction execution	Program is a sequence of instructions
Process loaded in to main memory	Program loaded into secondary storage devices
Time span of process is limited	Time span of program is unlimited
Process is a active entity	Program is a passive entity

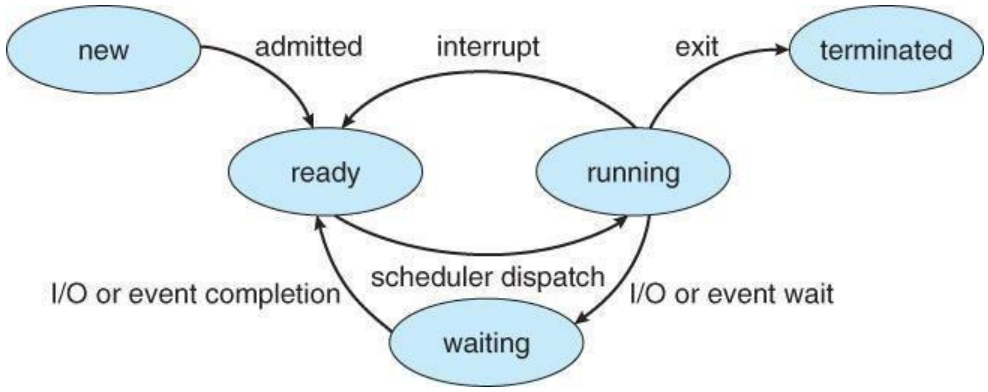
Process States

When a process executed, it changes the state, generally the state of process is determined by the current activity of the process. Each process may be in one of the following states:

1. New : The process is beingcreated.
2. Running : The process is beingexecuted.
3. Waiting : The process is waiting for some event tooccur.
4. Ready : The process is waiting to be assigned to a processor.
5. Terminated : The Process has finishedexecution.

Only one process can be running in any processor at any time, But many process may be in ready and waiting states. The ready processes are loaded into a “ready queue”.

Diagram of process state



a) **New ->Ready** : OS creates process and prepares the process to be executed,thenOSmoved the process into readyqueue.

b) **Ready->Running** : OS selects one of the Jobs from ready Queue and move themfrom ready to Running.

c) **Running->Terminated** : When the Execution of a process has Completed, OSterminatesthatprocess from running state. Sometimes OS terminates the process for someother reasons including Time exceeded, memory unavailable, access violation, protection Error, I/O failure and soon.

d) **Running->Ready** : When the time slot of the processor expired (or) If the processorreceivedanyinterrupt signal, the OS shifted Running -> ReadyState.

e) **Running -> Waiting** : A process is put into the waiting state, if the process need an event occur (or) an I/O Devicerequire.

f) **Waiting->Ready** : A process in the waiting state is moved to ready state whenthe eventforwhichit has beenCompleted.

Process Control Block:

Each process is represented in the operating System by a Process Control Block.

It is also called Task Control Block. It contains many pieces of information associated with a specific Process.

Process State
Program Counter
CPU Registers
CPU Scheduling Information
Memory – Management Information
Accounting Information
I/O Status Information

Process Control Block

1. **ProcessState** : The State may be new, ready, running, and waiting,Terminated…
2. **ProgramCounter** : indicates the Address of the next Instruction to be executed.
3. **CPUregisters** : registers include accumulators, stack pointers, General purpose Registers….

4. **CPU-SchedulingInfo** : includes a process pointer, pointers to schedulingQueues,other scheduling parametersetc.
5. **Memory management Info**: includes page tables, segmentation tables, value of base and limit registers.
6. **AccountingInformation:** includes amount of CPU used, time limits, Jobs(or)Process numbers.
7. **I/O StatusInformation**: Includes the list of I/O Devices Allocated to theprocesses, list of open files.

Threads:

A process is divide into number of light weight process, each light weight process is said to be a Thread. The Thread has a program counter (Keeps track of which instruction to execute next), registers (holds its current working variables), stack (execution History).

Thread States:

1. bornState : A thread is justcreated.
2. readystate : The thread is waiting forCPU.
3. running : System assigns the processor to thethread.
4. sleep : A sleeping thread becomes ready after the designated sleep timeexpires.
5. dead : The Execution of the threadfinished.

Eg: Word processor.

Typing, Formatting, Spell check, saving are threads.

Differences between Process and Thread

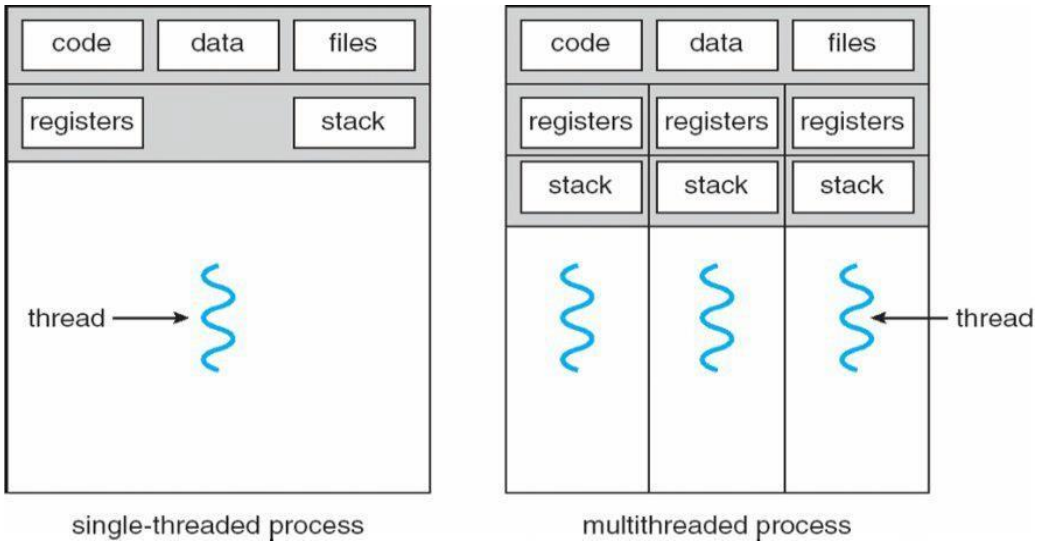
Process	Thread
Process takes more time to create.	Thread takes less time to create.
it takes more time to complete execution & terminate.	Less time to terminate.
Execution is very slow.	Execution is very fast.
It takes more time to switch b/w two processes.	It takes less time to switch b/w two threads.
Communication b/w two processes is difficult .	Communication b/w two threads is easy.
Process can't share the same memory area.	Threads can share same memory area.
System calls are requested to communicate each other.	System calls are not required.
Process is loosely coupled.	Threads are tightly coupled.
It requires more resources to execute.	Requires few resources to execute.

Multithreading

A process is divided into number of smaller tasks each task is called a Thread. Number of Threads with in a Process execute at a time is called Multithreading.
If a program, is multithreaded, even when some portion of it is blocked, the whole program is not blocked.The rest of the program continues working If multiple CPU's are available.
Multithreading gives best performance.If we have only a single thread, number of CPU's available, No performance benefits achieved.

- Process creation is heavy-weight while thread creation is light-weight

 Can simplify code, increase efficiency



 Kernels are generally multithreaded

CODE- Contains instruction
DATA- holds global variable **FILES-** opening and closing files
REGISTER- contain information about CPU state
STACK-parameters, local variables, functions
Types Of Threads:

1) User Threads : Thread creation, scheduling, management happen in user space by Thread Library. user threads are faster to create and manage. If a user thread performs a system call, which blocks it, all the other threads in that process one also automatically blocked, whole process is blocked.

Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls areblocking.
- Multithreaded application cannot take advantage ofmultiprocessing.

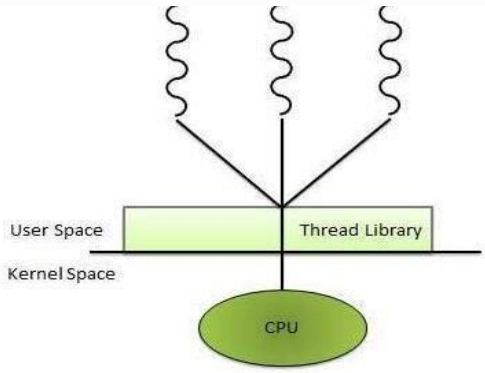
2) Kernel Threads: kernel creates, schedules, manages these threads .these threads are slower, manage. If one thread in a process blocked, over all process need not be blocked.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the userthreads.
- Transfer of control from one thread to another within same process requires a mode switch to the Kernel.



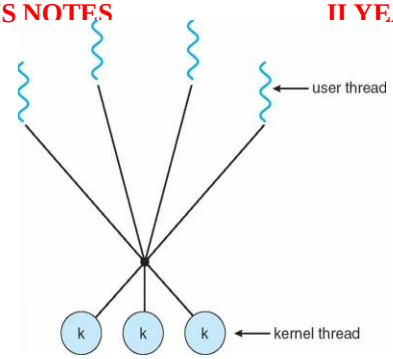
Multithreading Models

Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

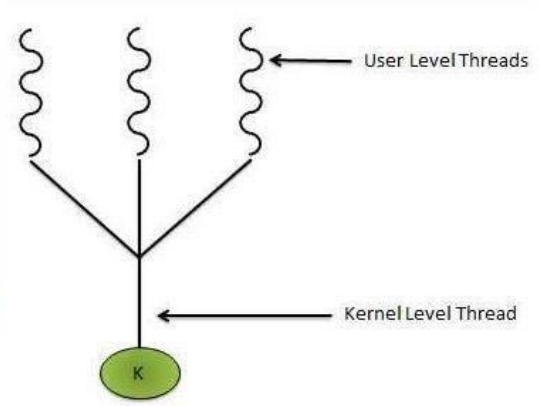
In this model, many user level threads multiplexes to the Kernel thread of smaller or equal numbers. The number of Kernel threads may be specific to either a particular application or a particular machine. Following diagram shows the many to many model. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallels on a multiprocessor.



Many to One Model

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in user space. When thread makes a blocking system call, the entire process will be blocks. Only one thread can access the Kernel at a time,so multiple threads are unable to run in parallel on multiprocessors.

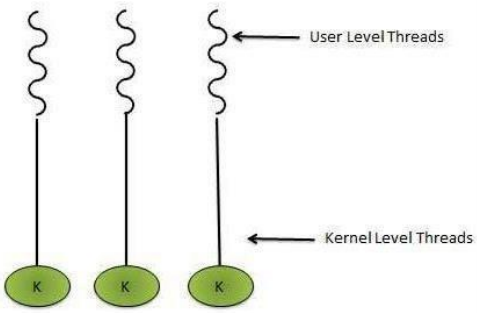
If the user level thread libraries are implemented in the operating system in such a way that system does not support them then Kernel threads use the many to one relationship modes.



One to One Model

There is one to one relationship of user level thread to the kernel level thread.This model provides more concurrency than the many to one model. It also another thread to run when a thread makes a blocking system call. It support multiple thread to execute in parallel on microprocessors.

Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship model.



Difference between User Level & Kernel Level Thread

S.N.	User Level Threads	Kernel Level Thread
1	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User level thread is generic and can run on any operating system.	Kernel level thread is specific to the operating system.
4	Multi-threaded application cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.