ILAHIA COLLEGE OF ENGINEERING AND TECHNOLOGY

OPERATING SYSTEM-S4 CSE

Module 1

Introduction: Functions of an operating system. Single processor, multiprocessor and clustered systems – overview. Kernel Data Structures – Operating Systems used in different computing environments.

Operating System Interfaces and implementation - User Interfaces, System Calls – examples. Operating System implementation approaches. Operating System Structure – Monolithic, Layered, Micro-kernel, Modular. System Boot process.

Introduction

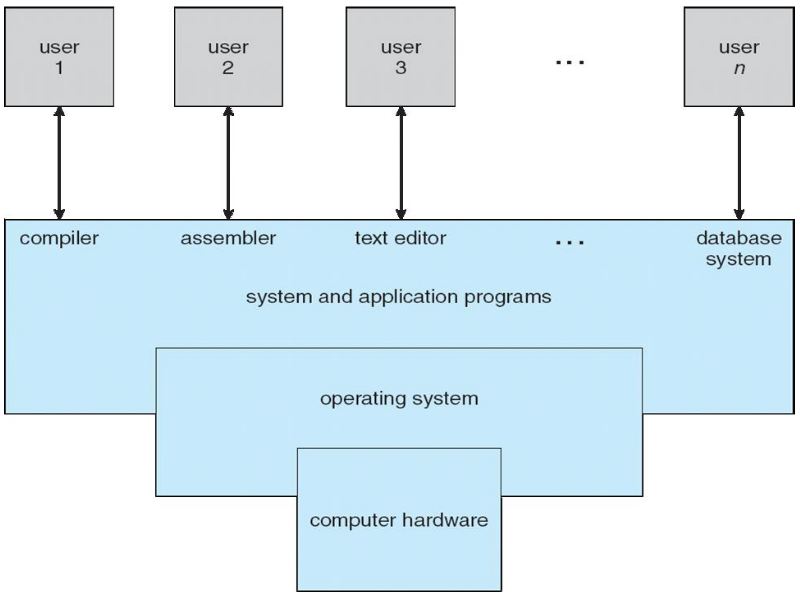
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Components of a Computer System

An operating system is an important part of almost every computer system. A computer system can be divided roughly into four components.

i. Hardware
ii. Operating system
iii. The application programs
iv. Users



- **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

What Operating Systems Do??

- **Role of OS depends on the point of view**

<u>User's view</u>

- In the case of Personal Computer, the operating system is designed mostly for ease of use, with some attention paid to performance and none paid to resource utilization—how various hardware and software resources are shared.
- In other cases, a user sits at a terminal connected to a mainframe or a minicomputer. Other users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization — to assure that all available CPU time, memory, and I/O are used efficiently
- In the case of hand held devices such as lap top or smart phones, operating system is designed mostly for ease of use and for resource utilization (battery life).

<u>System View</u>

- From system's view OS is the program most intimately involved with the hardware. So we can view OS as a resource allocator
- OS must decide and users so that it can operate the computer system efficient and fairly
- OS act as a control program that manages the execution of user programs and control various I/O devices
- OS is a **resource allocator**
    - Manages all resources
    - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
    - Controls execution of programs to prevent errors and improper use of the computer

<u>Functions of Operating Systems</u>

<u>Process Management</u>

A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.

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

- Process creation and deletion.
- Process suspension and resumption.

Provision of mechanisms for:

Process synchronization

Process communication

Main-Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.

Main memory is a volatile storage device. It loses its contents in the case of system failure.

The operating system is responsible for the following activities in connections with memory management:

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes to load when memory space becomes available.
- Allocate and deallocate memory space as needed.

File Management

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.

The operating system is responsible for the following activities in connections with file management:

- File creation and deletion.
- Directory creation and deletion.
- Support of primitives for manipulating files and directories.
- Mapping files onto secondary storage.
- File backup on stable (non volatile) storage media

I/O System Management

The I/O system consists of:

✦A buffer-caching system

✦A general device-driver interface

✦Drivers for specific hardware devices

Secondary-Storage Management

Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory.

Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.

The operating system is responsible for the following activities in connection with disk management:

✦Free space management

✦Storage allocation

✦Disk scheduling

Networking (Distributed Systems)

A distributed system is a collection processors that do not share memory or a clock. Each processor has its own local memory.

The processors in the system are connected through a communication network.

Communication takes place using a protocol.

A distributed system provides user access to various system resources.

Access to a shared resource allows:

✦Computation speed-up

✦Increased data availability

✦Enhanced reliability

Protection System

Protection
refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.

The protection mechanism must:

✦distinguish between authorized and unauthorized usage.

✦specify the controls to be imposed.

✦provide a means of enforcement.

Command-Interpreter System

Many commands are given to the operating system by control statements which deal with:

✦process creation and management

✦I/O handling

✦secondary-storage management

✦main-memory management

✦file-system access

✦protection

✦networking

Computer system architecture

Computer architecture means design or construction of a computer. A computer system may be organized in different ways. Some computer systems have single processor and other have multiprocessors. So computer systems categorized in these ways.

1. SingleProcessorSystems
2. MultiprocessorSystems
3. ClusteredSystems

SingleProcessor

Some computers use only one processor such as microcomputers.

On a single processor system, there is only one CPU that perform all the activities in the computer system, However, most of these systems have other special purpose processors, such as I/O Processor that move data rapidly among different components of the computer system.

MultiProcessor systems

Some systems have two or more processors. These systems are also known as parallel systems or tightly coupled systems.

Mostly the processors of these systems share the common system bus (electronic path) memory and peripheral (input/output) devices. These systems are fast in data processing and have capability to execute more than one program simultaneously on different processors. This type of processing is known as multiprogramming or multiprocessing.

Advantages:

I. **Increased Throughput** (means number of jobs completed by a CPU within a time period):
By increasing the number of processors, we expect to get more work done in less time

II. **Economical:** Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies.

III. **Increased Reliability**: If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

Multiple processors further divided in two types.

i. Asymmetric Multiprocessing Systems (AMS)
ii. Symmetric Multiprocessing Systems (SYS)

i) Asymmetric multiprocessing systems

The multiprocessing system, in which each processor is assigned a specific task, is known as Asymmetric Multiprocessing Systems.

In this system there exists **master slave relationship** like one processor defined as master and others are slave. The master processor controls the system and other processor executes predefined tasks. The master processor also defined the task of slave processors.

ii)Symmetric Multiprocessing System

The multiprocessing system in each processor performs all types of task within the operating system. All processors **are peers and no master slave relationship exits**.

In SMP systems many programs can run simultaneously. But I/O must control to ensure that data reach the appropriate processor because all the processor shares the same memory.

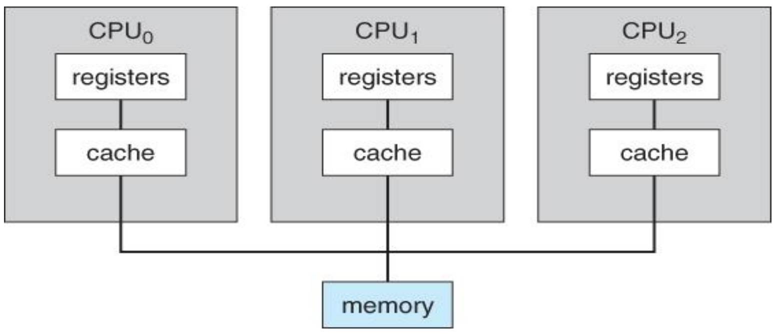
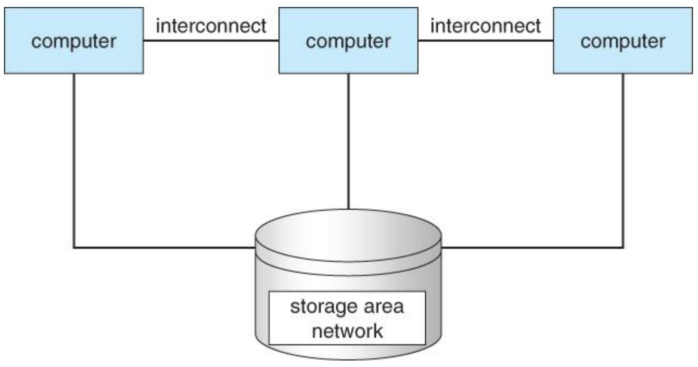


Fig:- **Symmetric Multiprocessing Architecture**

Clustered systems



Clustered systems are another form of multiprocessor system. This system also contains multiple processors but it differs from multiprocessor system.

The clustered system is composed of **multiple individual systems that connected together**. In clustered system, also individual systems or computers share the same storage and liked to gather via local area network.

A special type of software is known as cluster to control the node the systems.

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.

Kernel Data Structures

Several fundamental data structures are used extensively in operating systems.

1. **Lists, Stacks, and Queues**

An array is a simple data structure in which each element can be accessed directly. For example, main memory is constructed as an array.

After arrays, lists are the most fundamental data structures in computer science. Whereas each item in an array can be accessed directly, the items in a list must be accessed in a particular order. That is, a **list** represents a collection of data values as a sequence. The most common method for implementing this structure is a **linked list**, in which items are linked to one another. Linked lists are of several types:

- In a ***singly linked list,*** each item points to its successor
- In a ***doubly linked list,*** a given item can refer either to its predecessor or to its successor
- In a ***circularly linked list,*** the last element in the list refers to the first element, rather than to null

Linked lists accommodate items of varying sizes and allow easy insertion and deletion of items. The performance for retrieving a specified item in a list of size n is linear — $O(n)$

A **stack** is a sequentially ordered data structure that uses the last in, first out **(LIFO)** principle for adding and removing items, meaning that the last item placed onto a stack is the first item removed. The operations for inserting and removing items from a stack are known as ***push*** and ***pop***, respectively. An operating system often uses a stack when invoking function calls. Parameters, local variables, and the return address are pushed onto the stack when a function is called; returning from the function call pops those items off the stack.

A **queue**, in contrast, is a sequentially ordered data structure that uses the first in, first out **(FIFO)** principle: items are removed from a queue in the order in which they were inserted. Queues are also quite common in operating systems—jobs that are sent to a printer are typically printed in the order in which they were submitted. tasks that are waiting to be run on an available CPU are often organized in queues.

2. **Trees**

A **tree** is a data structure that can be used to represent data hierarchically. Data values in a tree structure are linked through parent–child relationships. In a **general tree**, a parent may have an unlimited number of children. In a **binary tree**, a parent may have at most two children, which we term the ***left child*** and the ***right child***. A **binary search tree** additionally requires an ordering between the parent's two children in which *le f t child <= right child*. Linux uses a balanced binary search tree as part its CPU-scheduling algorithm.

3. **Hash Functions and Maps**

A **hash function** takes data as its input, performs a numeric operation on this data, and returns a numeric value. This numeric value can then be used as an index into a table (typically an array) to quickly retrieve the data.

One use of a hash function is to implement a **hash map**, which associates (or ***maps***) [key:value] pairs using a hash function.

We can apply the hash function to the key to obtain the value from the hash map.

For Example: suppose that a user name is mapped to a password. Password authentication then proceeds as follows: a user enters his user name and password. The hash function is applied to the user name, which is then used to retrieve the password. The retrieved password is then compared with the password entered by the user for authentication.

Bitmaps

A **bitmap** is a string of *n* binary digits that can be used to represent the status of *n* items. For example, suppose we have several resources, and the availability of each resource is indicated by the value of a binary digit: 0 means that the resource is available, while 1 indicates that it is unavailable (or vice-versa). The value of the *i th* position in the bitmap is associated with the *i th* resource. As an example, consider the bitmap shown below:

0 0 1 0 1 1 1 0 1

Resources 2, 4, 5, 6, and 8 are unavailable; resources 0, 1, 3, and 7 are available.

Computing Environments

Operating systems are used in a variety of computing environments.

1. **Traditional Computing**

A few years ago, the "typical office environment" consisted of PCs connected to a network, with servers providing file and print services.

Remote access was awkward, and portability was achieved by use of laptop computers. Web technologies and increasing Companies establish portals, which provide web accessibility to their internal servers. Mobile computers can synchronize with PCs to allow very portable use of company information.

Mobile computers can also connect to wireless networks and cellular data networks to use the company's Web portal.

Fast data connections are allowing home computers to serve up Web pages and to run networks that include printers, client PCs, and servers. Many homes use firewalls to protect their networks from security breaches.

For a period of time, systems were either batch or interactive. Batch systems processed jobs in bulk, with predetermined input from files or other data sources.

Interactive systems waited for input from users. To optimize the use of the computing resources, multiple users shared time on these systems. Time-sharing systems used a timer and scheduling algorithms to cycle processes rapidly through the CPU, giving each user a share of the resources.

Traditional time-sharing systems are not common today. The same scheduling technique is still in use on desktop computers, laptops, servers, and even mobile computers, but frequently all the processes are owned by the same user. User processes, and system processes that provide services to the user, are managed so that each frequently gets a slice of computer time.

2. **Mobile Computing**

Mobile computing refers to computing on handheld smart phones and tablet computers. Features on mobile devices have become so rich that the distinction in functionality between a consumer laptop and a tablet computer may be difficult to discern.

Mobile systems are used not only for e-mail and web browsing but also for playing music and video, reading digital books, taking photos, and recording high-definition video.

Many developers are now designing applications that take advantage of the unique features of mobile devices, such as global positioning system (GPS) chips, accelerometers, and gyroscopes.

Two operating systems currently dominate mobile computing: Apple iOS and Google Android.

iOS was designed to run on Apple iPhone and iPad mobile devices. Android powers smart phones and tablet computers available from many manufacturers.

3. **Distributed Systems**

A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains.

A network, in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Networks vary by the protocols used, the distances between nodes, and the transport media

Networks are characterized based on the distances between their nodes.

- ✓ **A local-area network (LAN)** connects computers within a room, a building, or a campus.
- ✓ A **wide-area network (WAN)** usually links buildings, cities, or countries.
- ✓ A **metropolitan-area network (MAN)** could link buildings within a city.
- ✓ **BlueTooth and 802.11 devices use wireless technology** to communicate over a distance of several feet,

- ✓ A **personal-area network (PAN)** between a phone and a headset or a smartphone and a desktop computer.

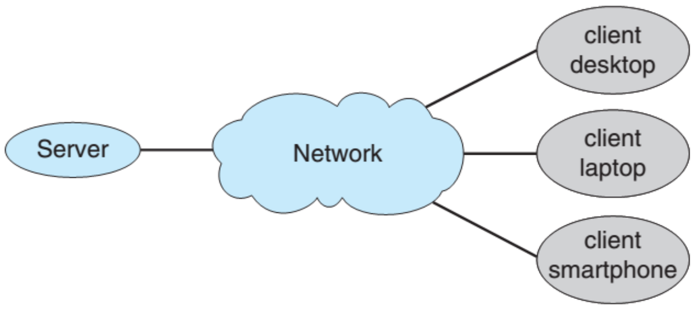
4. **Client-Server Computing**

This is a form of specialized distributed system in which server systems satisfy requests generated by client systems.

Server systems can be broadly categorized as **compute servers and file servers**:

The **compute-server system** provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client. A server running a database that responds to client requests for data is an example of such a system.

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.



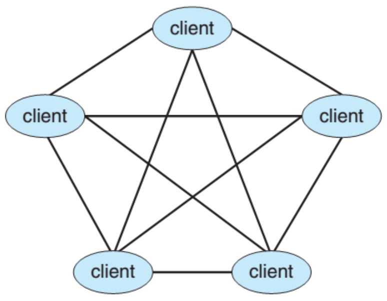
5. **Peer to Peer Computing**

In this model, clients and servers are not distinguished from one another. Instead, 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.

To participate in a peer-to-peer system, a node must first join the network of peers. Once a node has joined the network, it can begin providing services to—and requesting services from—other nodes in the network. Determining what services are available is accomplished in one of two general ways:

When a node joins a network, it registers its service with a centralized lookup service on the network. Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service.

A peer acting as a client must discover what node provides a desired service by broadcasting a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer making the request.



6. Virtualization

Virtualization is a technology that allows operating systems to run as applications within other operating systems. Operating system virtualization refers to the use of software to allow system hardware to run multiple instances of different operating systems concurrently, allowing you to run different applications requiring different operating systems on one computer system.

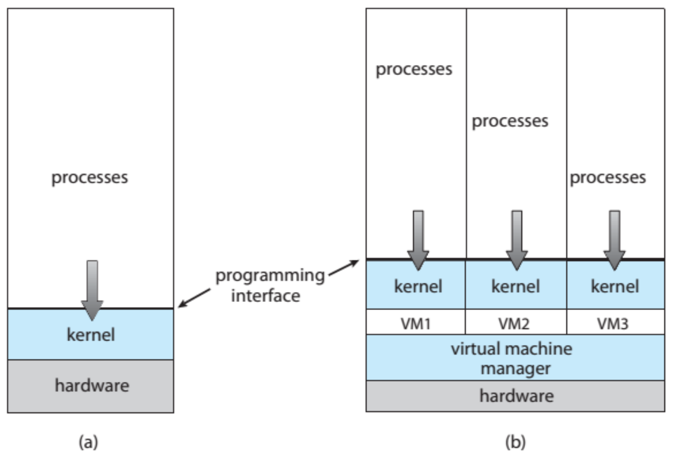


Figure (a) indicates non virtual machine and (b) indicates virtual machine

In this virtualization, a user installs the virtualization software in the operating system of his system like any other program and utilize this application to operate and generate various virtual machines(VM). Here, the virtualization software allows direct access to any of the created virtual machine to the user.
The operating systems do not interfere with each other or the various applications

Running multiple virtual machines allowed (and still allows) many users to run tasks on a system designed for a single user.

7. Cloud Computing

Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network. For example, the Amazon Elastic Compute Cloud (EC2) facility has thousands of servers, millions of virtual machines, and peta bytes of storage available for use by anyone on the Internet. Users pay per month based on how much of those resources they use
types of cloud computing:-

- ✓ **Public cloud** —a cloud available via the Internet to anyone willing to pay for the services
- ✓ **Private cloud** —a cloud run by a company for that company's own use
- ✓ **Hybrid cloud** —a cloud that includes both public and private cloud components
- ✓ Software as a service (SaaS) —one or more applications (such as word processors or spreadsheets) available via the Internet
- ✓ Platform as a service (PaaS) —a software stack ready for application use via the Internet (for example, a database server)
- ✓ Infrastructure as a service (IaaS) —servers or storage available over the Internet (for example, storage available for making backup copies of production data)

8. Real-Time Embedded Systems

Embedded devices are found everywhere, from car engines and manufacturing robots to DVDs and microwave ovens. They tend to have very specific tasks. So the operating systems provide limited features. Usually, they have little or no user interface.

May involve specialized chips, or generic CPUs applied to a particular task. Process control devices require real-time (interrupt driven) OSes. Response time can be critical for many such devices.

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 or the flow of data; thus, it is often used as a control device in a dedicated application. Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are realtime systems. A real-time system has well-defined, fixed time constraints. Processing must be done within the defined constraints, or the system will fail.

Operating System Interfaces and implementation

User Interface

`Almost all operating systems have a User Interface (UI). This interface can take several forms. One is a Command Line Interface (CLI) or command Interpreter and another is a Graphical User Interface (GUI).

1. A command-line interface, or **command interpreter**, that allows users to directly enter commands to be performed by the operating system.
2. a graphical user interface, or GUI, that allows users to interface with the operating system via i/o devices and icons

1. Command Interpreters

Some operating systems include the command interpreter in the kernel. These interpreters are known as **shells**.

Eg: ***Bourne shell***, ***C shell***, ***Bourne-Again shell***, ***Korn shell***

The main function of the command interpreter is to get and execute the next user-specified command. Many of the commands given at this level manipulate files: create, delete, list, print, copy, execute, and so on. . These commands can be implemented in two general ways.

1. The command interpreter itself contains the code to execute the command. For example, a command to delete a file may cause the command interpreter to jump to a section of its code that sets up the parameters and makes the appropriate system call.
2. Implements most commands through system programs. In this case, the command interpreter does not understand the command in any way; it merely uses the command to identify a file to be loaded into memory and executed.

2. Graphical User Interface

A second strategy for interfacing with the operating system is through a user friendly graphical user interface, or GUI.

Users employ a mouse-based window and-menu system characterized by a **desktop** metaphor. The user moves the mouse to position its pointer on images, or **icons**, on the screen (the desktop) that represent programs, files, directories, and system functions. Depending on the mouse pointer’s location, clicking a button on the mouse can invoke a program, select a file or directory—known as a **folder**—or pull down a menu that contains commands.

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

System calls

System call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.

-Provide an interface to the services that are available to O.S

- Written in high level language(C,C++)

- To understand system calls, first one needs to understand the difference between kernel mode and user mode of a CPU.

Program execution done in two modes

1. User mode
2. Kernel mode

User mode- does not have direct access to memory, hardware and resources. If a program crashes , it does not affect the system.

Kernel mode- direct access to memory, hardware and resources. In this mode, if a program crashes entire system crashes

When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a system call.

When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a **context switch**.

Generally, system calls are made by the user level programs in the following situations:

A. Creating, opening, closing and deleting files in the file system.

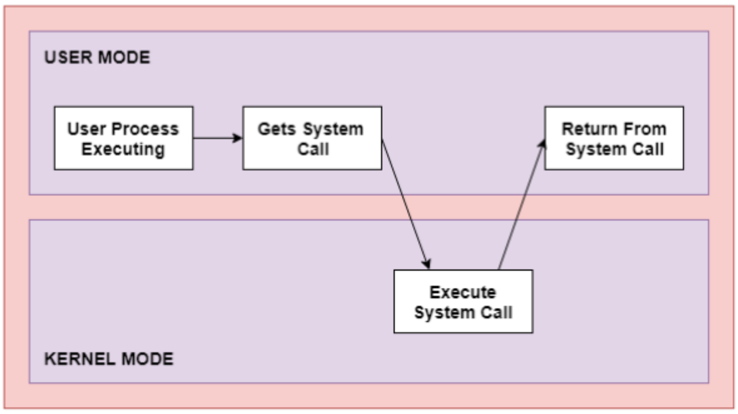
B. Creating and managing new processes.

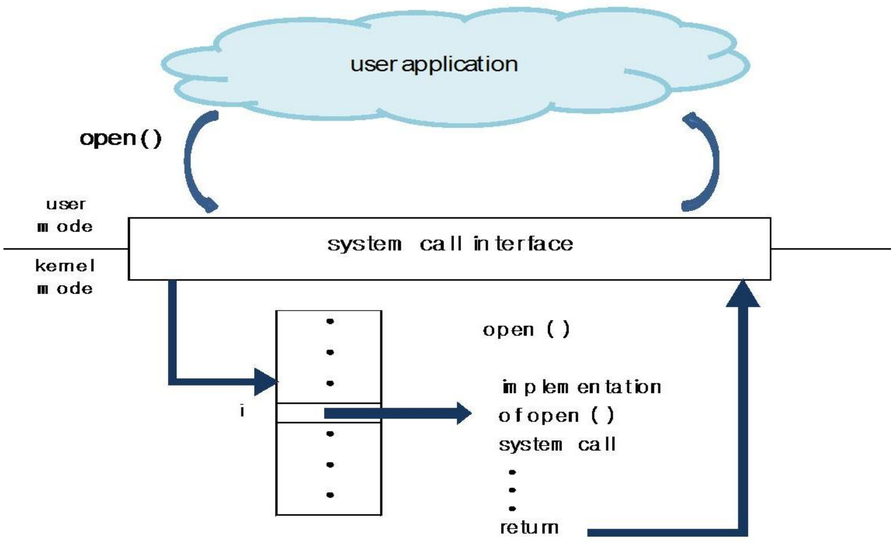
C. Creating a connection in the network, sending and receiving packets.

D. Requesting access to a hardware device, like a mouse or a printer.

E. system supports these two modes.

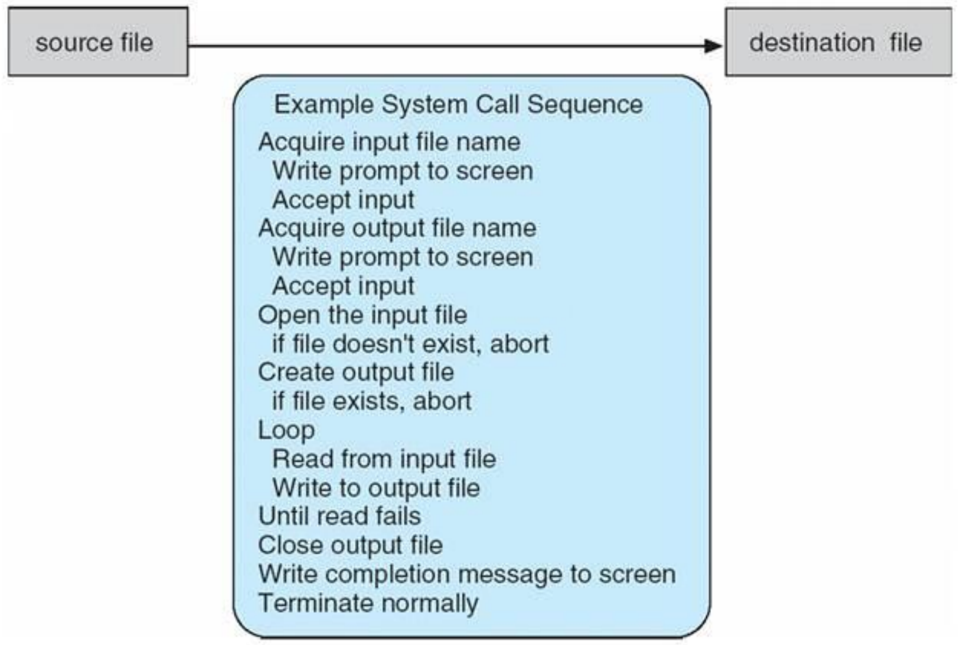
System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.





System call – Example

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

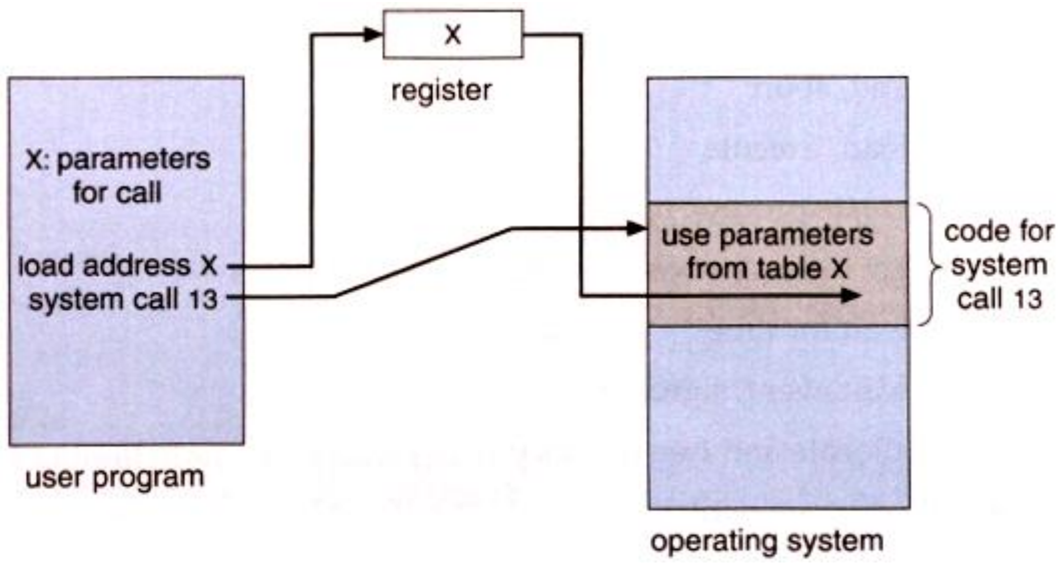


Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file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

System call Parameters

Three general methods are used to pass parameters between running program and the operating system

1. Parameters can be passed in registers.
2. When there are more parameters than registers, parameters can be stored in a block and the block address can be passed as a parameter to a register.
3. Parameters can also be pushed on or popped off the stack by the operating system.



Types of System Calls: There are 5 different categories of system calls –

1. **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
2. **File management:**
   create file, delete file
   open, close
   read, write, reposition
   get file attributes, set file attributes
3. **Device management**
   request device, release device
   read, write, reposition
   get device attributes, set device attributes
   logically attach or detach devices
4. **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

5. Communication

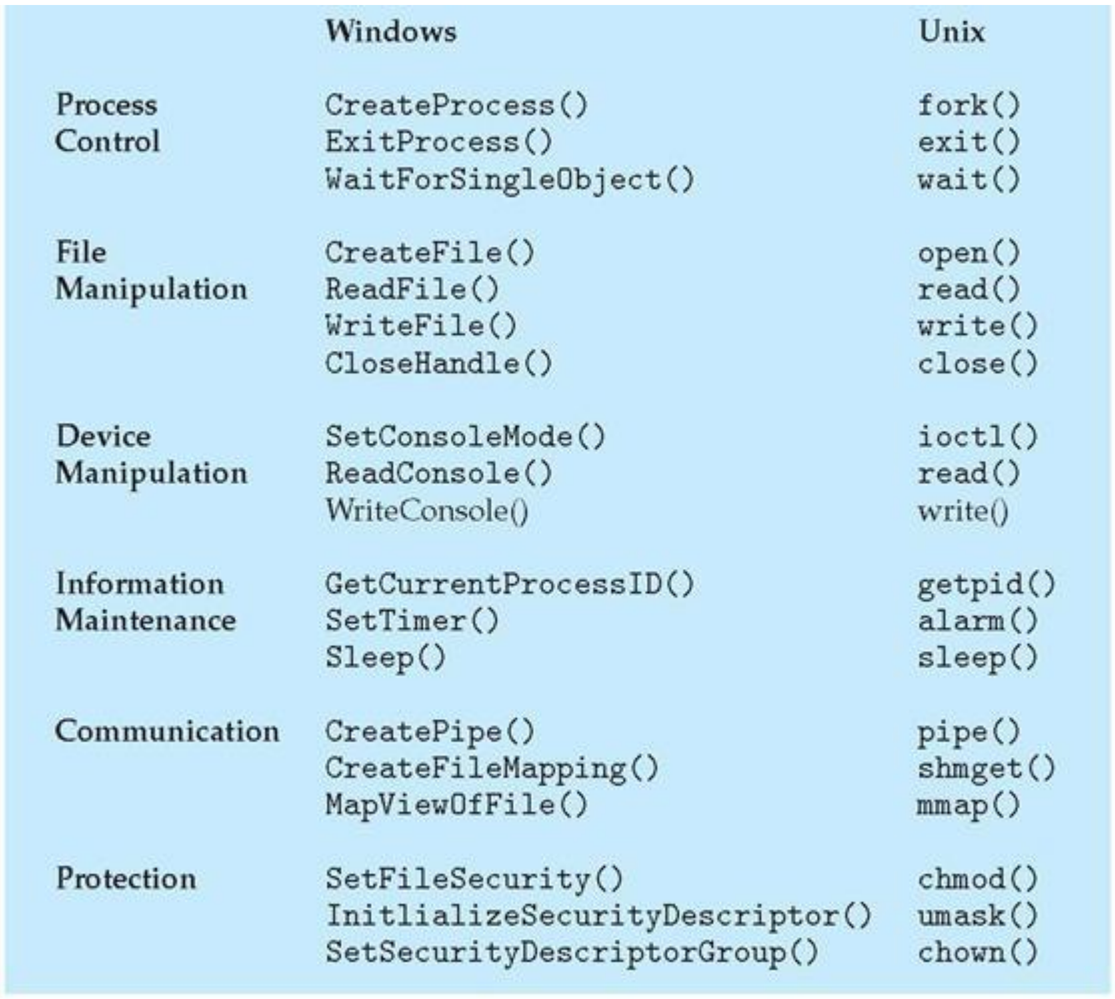
create, delete communication connection

send, receive messages

transfer status information

attach or 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()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()

<u>Operating System Implementation</u>

Different problems we face in designing and implementing an OS

1. <u>Design Goals</u>

The first problem in designing a system is to define goals and specifications. the design of the system will be affected by the choice of hardware and the type of system: batch, time sharing, single user, multiuser, distributed, real time, or general purpose. The requirements can be divided into two basic groups: **user goals** and **system goals**.

- Users want the system should be convenient to use, easy to learn and to use, reliable, safe, and fast.
- The designers want the system should be easy to design, implement, and maintain; and it should be flexible reliable, error free, and efficient

In short, no unique solution to the problem of defining the requirements for an operating system. Different requirements can result in a large variety of solutions for different environments.

2. Mechanisms and Policies

Mechanisms determine ***how*** to do something; policies determine ***what*** will be done.

Policies are likely to change across places or over time.

A change in policy would then require redefinition of only certain parameters of the system. In the worst case, each change in policy would require a change in the underlying mechanism.

Policy decisions are important for all resource allocation. Whenever it is necessary to decide whether or not to allocate a resource, a policy decision must be made.

3. Implementation

Once an operating system is designed, it must be implemented. Early operating systems were written in assembly language. Now most are written in a higher-level language such as C or an even higher-level language such as C++. An operating system can be written in more than one language. The lowest levels of the kernel might be assembly language. Higher-level routines might be in C, and system programs might be in C or C++

The *advantages* of using a higher-level language are the code can be written faster, is more compact, and is easier to understand and debug and easier to **port**—to move to some other hardware.

The *disadvantages* of implementing an operating system in a higher-level language are reduced speed and increased storage requirements.

Major *performance improvements* in operating systems are more likely to be the result of better data structures and algorithms than of excellent assembly-language code.

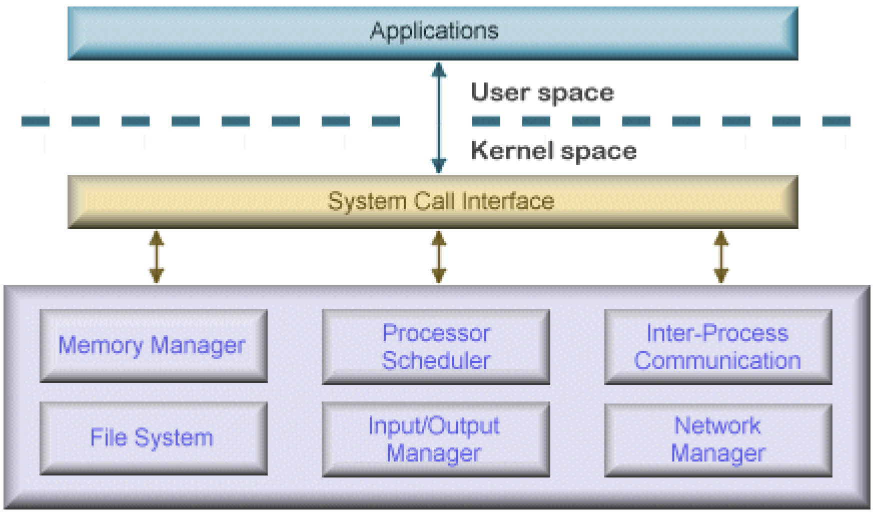
Operating System Structure

Different operating system structures are

1. **Monolithic systems**
2. **Layered Architecture**
3. **Micro Kernel Architecture**
4. **Modular Systems**

1. Monolithic Systems

- Most common Architecture
- Entire operating system is working in Kernel mode
- Every component in OS is contained in Kernel and can directly communicate with any other.



2. **Layered Architecture**

The operating system is divided into a number of layers (levels), each built on top of lower layers.

-Grouping components that perform similar functions into 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

 The main *advantage* is simplicity of construction and debugging.

 The main *difficulty* is defining the various layers.

 The main *disadvantage* is that the OS tends to be less efficient than other implementations.

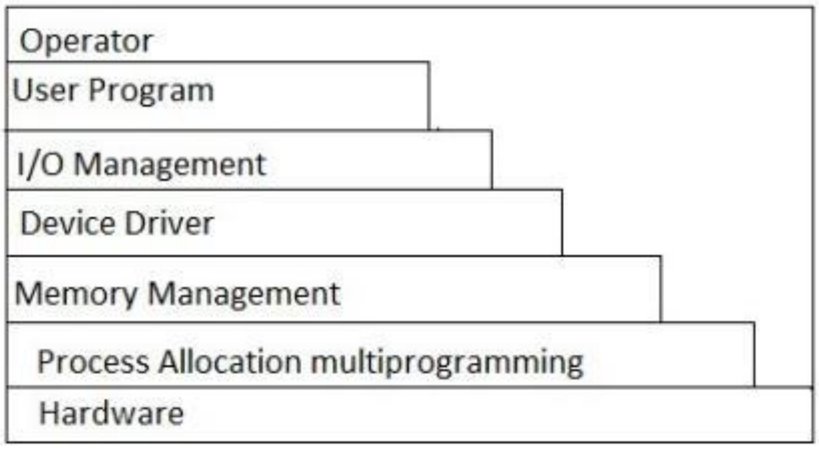
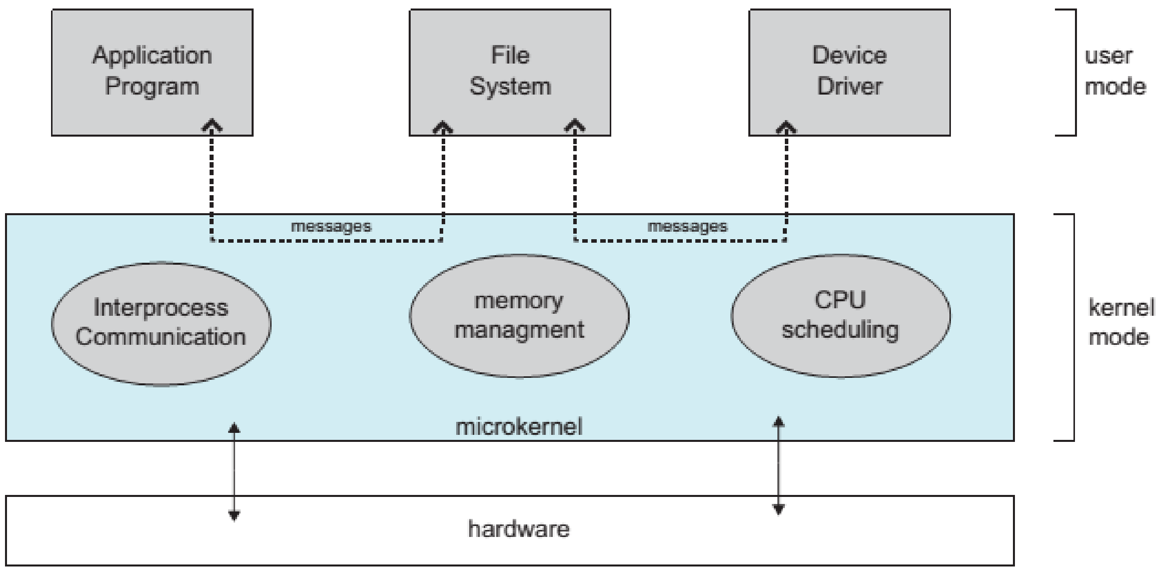


fig:- layered Architecture

3. **Micro Kernel Architecture**

This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel.

Micro kernels provide minimal process and memory management, in addition to a communication facility.



The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through **message passing.**

If the client program wishes to access a file, it must interact with the file server. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel.

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

Disadvantages

Performance overhead of user space to kernel space

4. 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 flexible

The best current methodology for operating-system design involves using **loadable kernel modules**.

Here, the kernel has a set of core components and links in additional services via modules.

The idea of the design is for the kernel to provide core services while other services are implemented dynamically, as the kernel is running.

Linking services dynamically is preferable to adding new features directly to the kernel, which would require recompiling the kernel every time a change was made.

This system is more flexible than a layered system, because any module can call any other module. The approach is also similar to the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules, but it is more efficient, because modules do not need to invoke message passing in order to communicate.

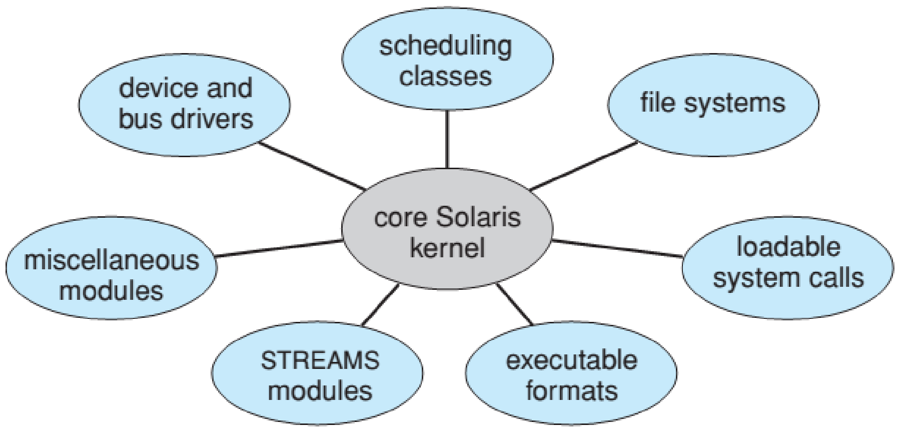


Figure 2.15 Solaris loadable modules.

The Solaris operating system structure, shown in Figure 2.15, is organized around a core kernel with seven types of loadable kernel modules:

„

System Boot Process

The procedure of starting a computer by loading the kernel is known as **booting** the system. On most computer systems, a small piece of code known as the **bootstrap program** or **bootstrap loader** locates the kernel, loads it into main memory, and starts its execution.

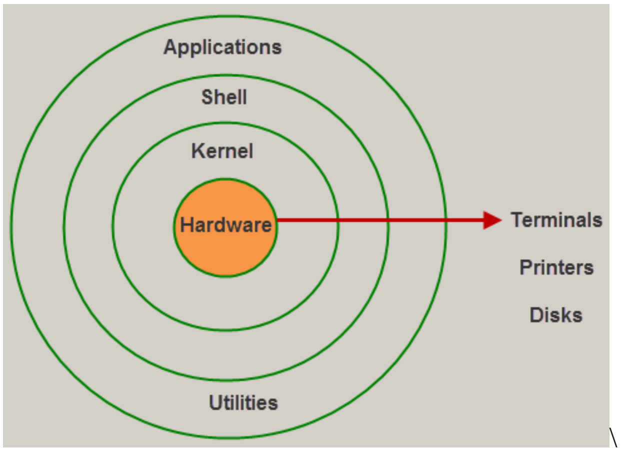
Booting Steps

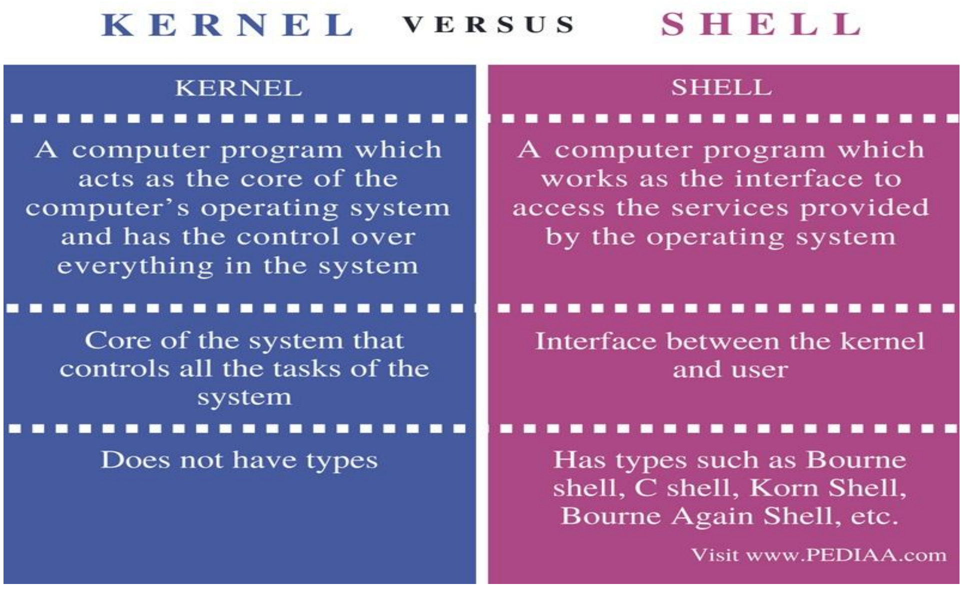
1. When the computer's power is first turned on, the CPU initializes itself, which is triggered by a series of clock ticks genearted by the system clock. Part of the CPU's initialization is to look to the system's ROM,**BIOS** for its first instruction in the startup program.
2. The **ROM BIOS** stores the first instruction, which is the instruction to run the power-on self test(POST), in a predetermined memory address. POST begins by checking the BIOS chip and then tests CMOS RAM. If the POST does not detect a battery failure, it then continues to initialize the CPU, checking the hardware devices (such as the video card), secondary storage devices, such as hard drives and floppy drives,ports and other hardware devices, such as the keyboard and mouse, to ensure they are functioning properly.
3. Once the POST has determined that all components are functioning properly and the CPU has successfully initialized, the BIOS looks on Drive A, where the OS boot files are located.
4. BIOS next look at the first sector and copies information from it into specific locations in RAM. This information is known as **boot record**
5. The boot record loads the initial system file into RAM
6. The initial file then loads the rest of the OS into RAM.

Components of Operating System

Components of OS are

1. Kernel
2. Shell
3. File System
4. System Libraries
5. System Utilities





KERNEL VERSUS SHELL

KERNEL	SHELL
A computer program which acts as the core of the computer's operating system and has the control over everything in the system	A computer program which works as the interface to access the services provided by the operating system
Core of the system that controls all the tasks of the system	Interface between the kernel and user
Does not have types	Has types such as Bourne shell, C shell, Korn Shell, Bourne Again Shell, etc.

Visit www.PEDIAA.com

File System

Functions of file system are

- Handles Directory or folder
- Position of a file in directory hierarchy
- Any access to a file by other software
- Information needed to access an open file
- Handles user created group of files

System Libraries

It contains standard set of function through which application can interact with the kernel

System Utilities

These are programs that perform individual, specialized management tasks