ST.MARY'S GROUP OF INSTITUTIONS,HYDERABAD

UNIT I - OPERATING SYSTEMS

Mr T.NAGARJUN
M.Tech
Asst. Prof
Data science

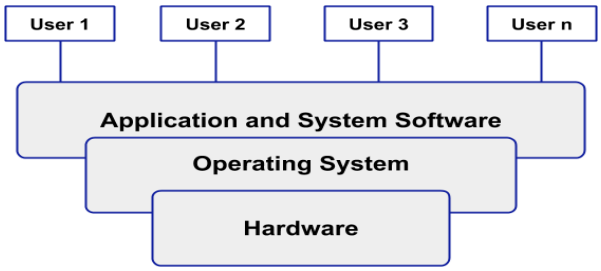
UNIT-I

Operating System - Introduction, Structures - Simple Batch, Multiprogrammed, Time-shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, System components, Operating System services, System Calls

---

1. INTRODUCTION TO OPERATING SYSTEMS

Operating Systems is the interface between the user and the computer hardware.



Operating system goals:

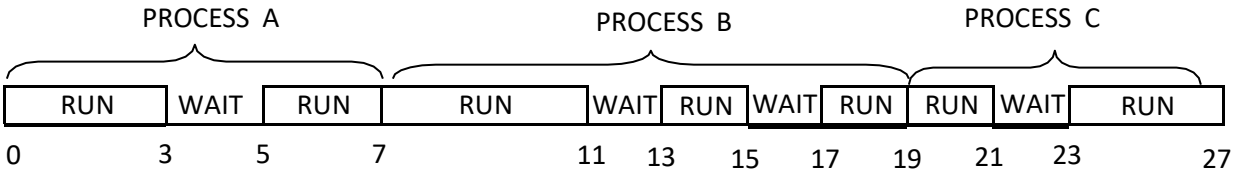
- **Convenience:** An OS makes a computer more convenient to use.
- **Efficiency:** An OS allows the computer resources to be used in an efficient manner.
- **Ability to evolve:** An OS is designed to permit new system functions.

2. TYPES OF OPERATING SYSTEMS

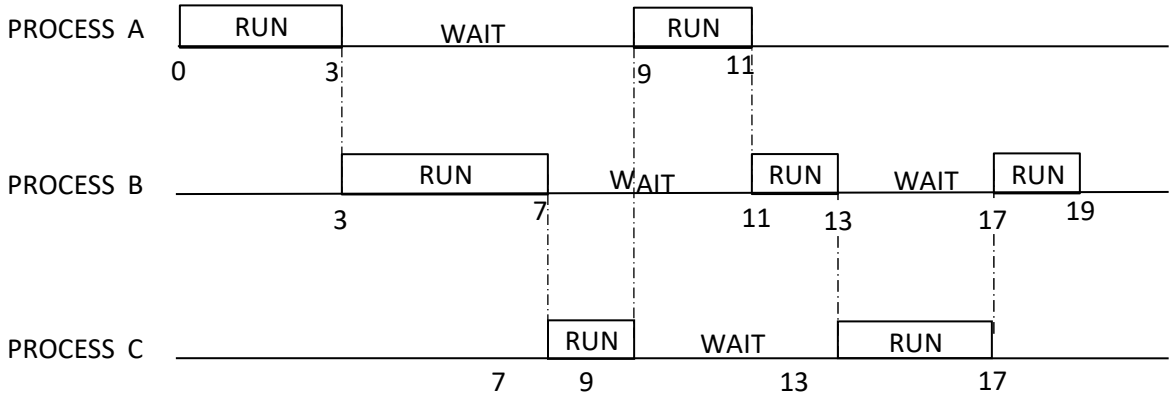
Following are some of the most widely used types of Operating system.

- Simple Batch System
- Multiprogramming System
- Time Sharing Systems
- Multiprocessor System
- Distributed Operating System
- Parallel Processing System
- Real-time Operating System
- Personal Computer OS
- Handheld System

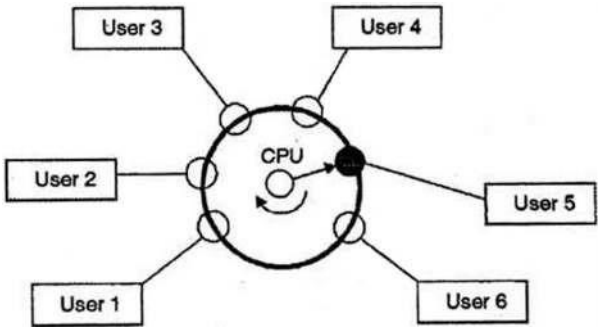
i. **Batch Operating System:** In this type of system, there is no direct interaction between user and the computer. The user has to submit a job (written on cards or tape) to a computer operator. Then computer operator group similar jobs into batches and gives to an input device. Then a special program called the monitor, manages the execution of each program in the batch. It follows serial processing. When the current job execution completes, then only next job is assigned to the CPU for execution. After executing all the jobs in a batch one after the other, then select other batch of jobs for execution.



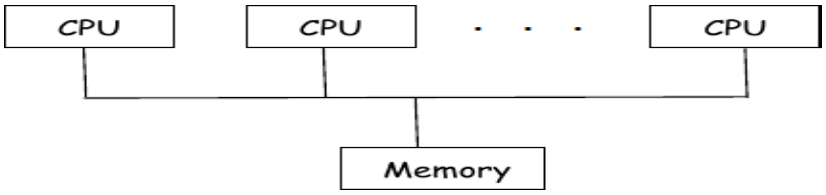
ii. **Multiprogramming OS:** The processes ready for execution are loaded into the main memory. The OS selects one of these processes and assigns it to the CPU. Whenever the executing process needs to wait for any operation like I/O, the OS selects another process from the job queue and assigns it to the CPU. Latter, when a partially executed process gets a chance to execute, it resumes the execution from where it stopped previously. This is shown in the below diagram. This way, the CPU is kept busy and the throughput increases.



iii. **Time-Sharing Operating Systems:** This was introduced in the 1960s when computer was too expensive. The solution was to allow many users to use one computer by providing each user one time-slot. Each user sit in front of one terminal and all the terminals are connected to one computer. The time sharing operating system executes each user task in a given fixed time slot and then switch to next user task execution. This process is repeated until all tasks are executed.



iv. **Multiprocessor System:** A Multiprocessor system consists of several processors that share a common physical memory. In this system all processors operate under single operating system and provide higher computing power. Each processor can execute one process simultaneously. If possible, system divides a single task into many subtasks and then these subtasks can be executed in parallel on different processors.

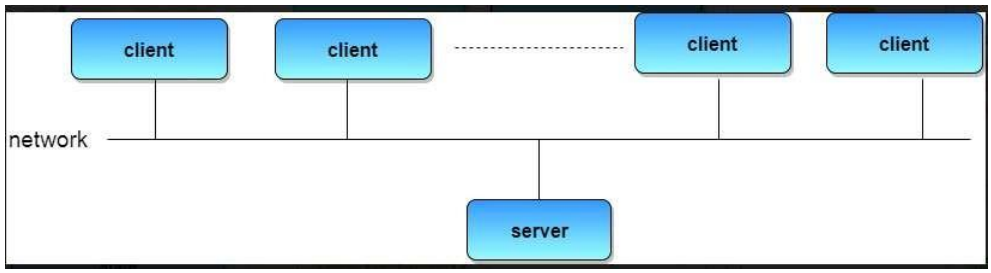


v. **Distributed Operating System:** A collection of physically separated independent computers linked together using the network and works as a single system is called distributed system. All the nodes (computers) in this system communicate with each other and controlled by the distributed operating system software.

There are two types of distributed operating systems:

a) Client-Server Systems
b) Peer-to-Peer Systems

Client-Server Systems: Centralized system act as server which takes requests from the clients and sends back the response to the client systems. The general structure of a client-server system is depicted in the figure below:



Peer-to-Peer Systems: In this model, clients and servers are not distinguished from one another; instead, all nodes (computers) within the system are considered peers. Each node may act as either a client or a server, depending on whether it is requesting or providing a service. Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck; but in a peer-to-peer system, all the nodes can acts as a server. Therefore the load of the server is distributed among all the nodes. This type of system is also known as loosely coupled system.

vi. Parallel Processing Systems: Parallel processing systems are designed to speed up the execution of programs by dividing the program into multiple parts and processing these parts simultaneously. Parallel systems can be designed in three ways:

1. Single computer with multiple processors (also known as Multiprocessor system).
2. A set of computers connected by a network to form a parallel processing cluster (also known as Distributed system).
3. A combination of both (1) and (2).

Parallel systems are more difficult to program than computers with a single processor because the processes of multiple CPUs must be coordinated and synchronized. Several models for connecting processors and memory modules exist, and each topology requires a different programming model. Flynn has classified the computer systems based on parallelism in the instructions and in the data streams into four types. These are:

1. Single instruction stream, single data stream (SISD).
2. Single instruction stream, multiple data stream (SIMD).
3. Multiple instruction streams, single data stream (MISD).
4. Multiple instruction stream, multiple data stream (MIMD).

The above classification of parallel computing system is focused in terms of two independent factors: the number of data streams that can be simultaneously processed, and the number of instruction streams that can be simultaneously processed. Out of the four types of Flynn classifications, only two of them are relevant to parallel computers. These are SIMD and MIMD computers. A parallel computer may be a supercomputer with hundreds or thousands of processors or may be a network of workstations.

vii. Real Time Operating System: A real-time operating system (RTOS) is an OS that guarantees applications to be executed within a specified time bound. There are two types of Real-Time Operating System. They are:

1. **Hard Real-Time Operating System**: A process should be executed in given deadline. The deadline should not be crossed. In this type of OS, maximum processor time is allocated to critical operations and completes them on time. Preemption time for Hard Real-Time Operating System is almost less than few microseconds.

   Examples are Airbag control in cars, anti-lock brake, engine control system, etc.

2. **Soft Real-Time Operating System**: A process might not be executed in the given deadline. The execution may be completed even after the deadline without harming the system. In this type of RTOS, the critical task will get priority over other tasks, but no guarantee of completing it in a defined time.

   Examples are a digital camera, automatic washing machines, etc.

viii. **Personal computer operating system:** Personal computer operating system is made only for personal. Personal computers are intended to be operated directly by an end user, rather than by a computer expert or technician. Only single user can use the computer at any time. The laptops, computer systems, tablets etc. are personal computers and the operating system such as windows, Linux, android, etc. are personal computer operating system. Personal computer operating system can be used for your personal purposes, for example, to chat with friends, reading some articles from internet, designing your website, programming something, watching some videos and movies, listening to some songs and many more.

ix. **Handheld Systems:** Handheld systems include Personal Digital Assistants (PDAs), such as tablets, palmtops or Cellular mobiles with Internet connection. Most of the handheld devices have a small amount of memory, slow processors, small display screens and battery. Faster processors require more power. To include a faster processor in a handheld device would require a larger battery that would have to be replaced more frequently. Some handheld devices may use wireless technology such as BlueTooth or wify allowing remote access to e-mail and web browsing.

3. OPERATING SYSTEM FUNCTIONS / SERVICES / TASKS

The services provided by operating system software are:

i. **Process management:** Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

ii. **Memory management:** Memory management module performs the task of allocation and de-allocation of memory space to programs.

iii. **File management**: It manages all the file-related activities such as creating, deleting, accessing, sharing, and protection of files.

iv. **Device Management**: Device management keeps tracks of all input/output devices. It also performs the task of allocation and de-allocation of the devices.

v. **Security**: Security module protects the files of a computer system against malware threat and unauthorized access.

vi. **User Interface:** It provides a user interface. It may be command line or graphical user interface (GUI). A user can very easily use computer system with the help of the GUI.

vii. **Program development**: The OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.

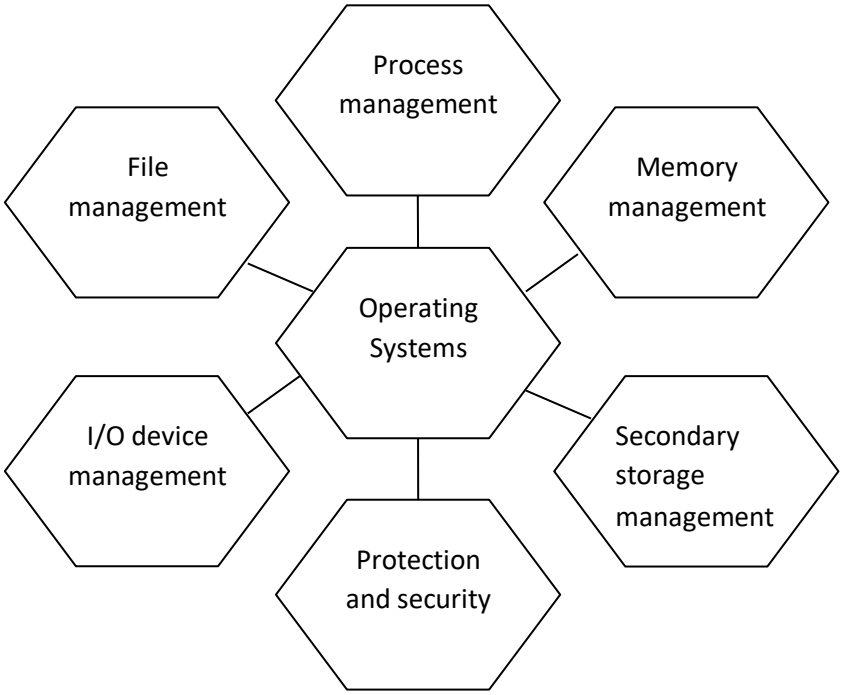
viii. **Program execution:** A number of steps need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialized, and other resources must be prepared. The OS handles the CPU scheduling duties for the user.

ix. **Error detection and response:** A variety of errors can occur while a computer system is running. These include internal and external hardware errors, such as a memory error, or a device failure or malfunction; and various software errors, such as division by zero, attempt to access forbidden memory location, and inability of the OS to grant the request of an application. In each case, the OS must provide a response that clears the error condition with the least impact on running applications. The response may range from ending the program that caused the error, to retrying the operation, to simply reporting the error to the application.

x. **Accounting:** A good OS will collect usage statistics for various resources and monitor performance parameters such as response time. On any system, this information is useful in anticipating the need for future enhancements and in tuning the system to improve performance. On a multiuser system, the information can be used for billing purposes.

4. SYSTEM COMPONENTS

The operating system comprises a set of software components that can be used to manage interactions with the hardware. They are:



Process management: The process management component manages all the processes that are running simultaneously on the operating system. When any software application program executes, one or more processes associated with it runs. For example, when you use a browser like Google Chrome, there is a process running for that browser program. Similarly, the OS runs many processes to perform various functions. All these processes should be managed by process management. The following are the main functions of process management.

- Process creation and deletion.
- Process suspension and resumption.
- Process synchronization
- Communication among process

Main Memory Management: It deals with main memory also called as RAM. Any program execution starts only after loading it in to RAM. The following are the main functions of main 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.

Secondary storage management: Since main memory (RAM) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory. The secondary storage management component manages the storage devices, like a hard drive, USB flash drive (pen drive), DVD drive, or floppy disk drive. The following are the main functions of secondary storage management.

- Free space management
- Storage allocation
- Disk scheduling

File management: It deals with management of files in a computer system. It helps with creating and manipulating files. The following are the main functions of file management.

- File and directory creation and deletion.
- Manipulating files and directories.
- Mapping files onto secondary storage.
- Backup files on stable storage media.

I/O Device Management: It deals with managing the I/O devices connected to computer system. It helps to hide the variations of specific hardware devices from the user. The following are the main functions of I/O device management.

- It offers buffer caching system.
- It provides device drivers to run the basic I/O devices such as keyboard and mouse.
- It provides interface to connect and run any I/O device.

Protection and Security Management: The various processes in an operating system need to be secured from each other's activities. For that purpose, various mechanisms can be used to ensure that those processes which want to operate files, memory CPU, and other hardware resources should have proper authorization from the operating system. The following are the main functions of protection and security management.

- It protects the files from unauthorized usage.
- It provide security mechanisms
- It provide control access mechanism

5. SYSTEM CALLS

A system call is a programmatic interface between user process and OS kernel. It is a mechanism in which user program make a request to get service form kernel. System call provides the services to the user programs via Application Program Interface (API). The common APIs available for most operating systems or virtual machines are:

API	Operating System
Win32 API	Windows
POSIX API	Unix, Linux, Mac OS X
Java API	Java Virtual Machine (JVM)

To understand system calls, first we need to understand the difference between **kernel mode** and **user mode** of a CPU. All modern operating systems support kernel mode and user mode.

Kernel Mode:

- All OS related functions get executed in kernel mode. The mode bit = 0.
- When CPU is in **kernel mode**, the code being executed can access any memory address and any hardware resource.
- Hence kernel mode is a very privileged and powerful mode.
- If a program crashes in kernel mode, the entire system will be halted.

User Mode

- All user applications get executed in user mode only. The mode bit = 1.
- When CPU is in **user mode**, the programs don't have direct access to memory and hardware resources.
- In user mode, if any program crashes, only that particular program is halted.
- That means the system will be in a safe state even if a program in user mode crashes.

Generally user application executes in user mode and when it needs support from OS, a system call is generated. When an application makes a system call, it passes the required parameters to the kernel. The three general methods for passing parameters to the OS kernel are:

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

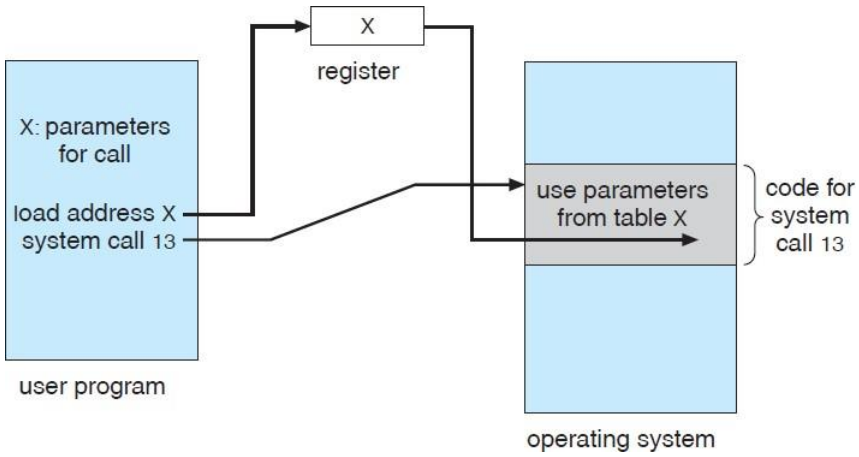
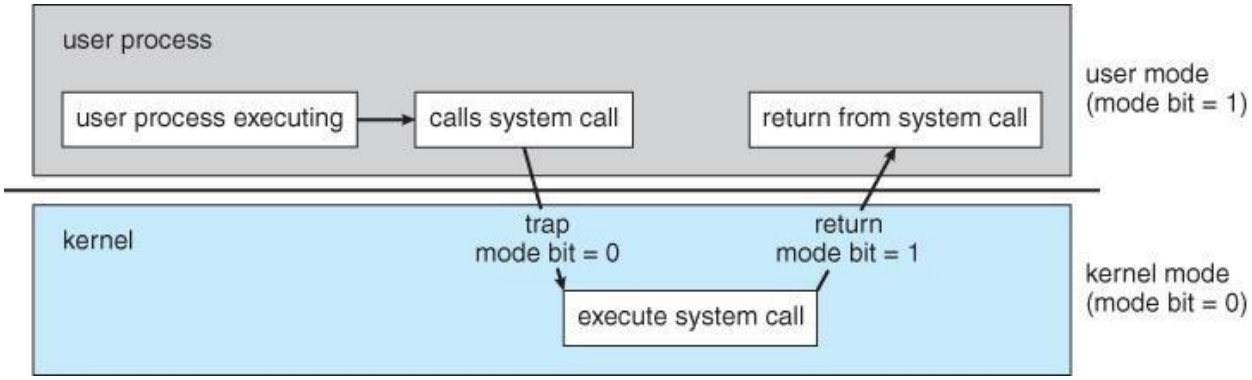
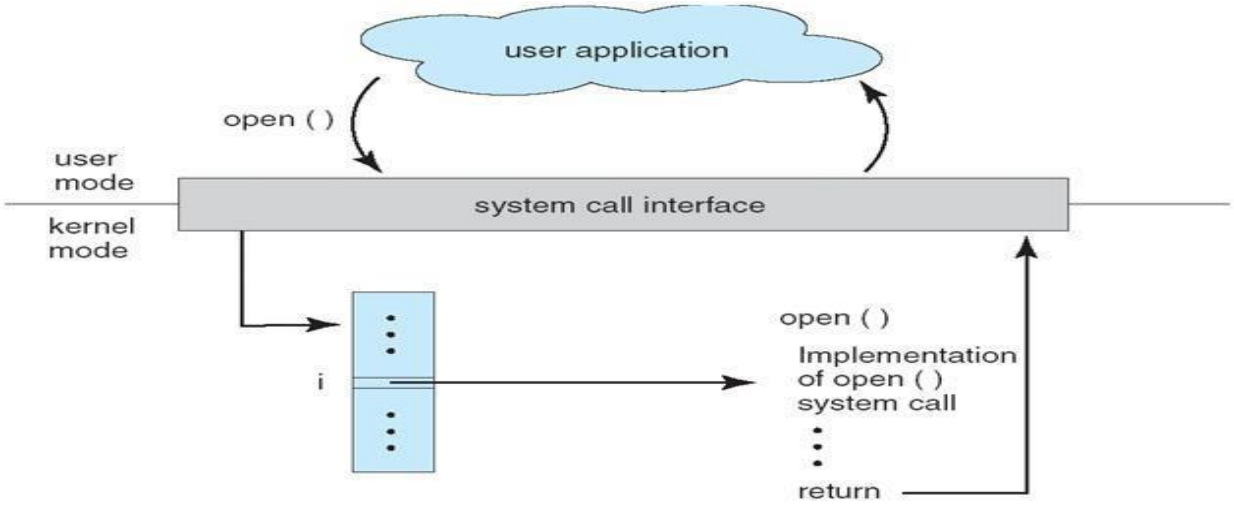


Figure: Passing parameters as a table

Before the system call enters into kernel mode, the user application pauses and the control is transferred to the kernel by setting mode bit = 0. If the request is allowed, the kernel processes the request, for example creating or deleting a file. When the operation is complete, the kernel passes the output back to the application and set the mode bit =1. The application receives the output from the kernel as parameter value. Once the input is received, the application resumes the process. The interaction between user mode and kernel mode is shown in the below figure.



For example, when a user application needs a file, a system call called open( ) is generated. The user application passes the parameters such as file name, path name to the kernel by changing the mode bit to 0. If the requested file exists (request is valid), the file is opened by the kernel and changes the mode bit to 1. Later, the address of the opened file is passed back as a return value to the user application. This is shown in the below figure.



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

- Creating, opening, closing and deleting files in the file system.
- Creating and managing new processes.
- Creating a connection in the network, sending and receiving packets.
- Requesting access to a hardware device, like a mouse or a printer.

The system calls can be categorized broadly into five types. They are:

i. **Process Control:** These system calls deal with process management and control. Some of the important system call under this category are:

CreateProcess( ): It used to create a new process.

ExitProcess( ): It is used to stop executing a process.

WaitForSingleObject( ): It is used to block the current executing process.

ii. **File Management:** These system calls deals with file manipulation and management. Some of the important system call under this category are:

CreateFile( ): It is used to create a new file.

ReadFile( ): It is used to open an existing file to read.

WriteFile( ): It is used to open an existing file or create a file to write.

CloseHandle( ): It is used to closes the *file* currently in use.

iii. **Device Management:** These system calls deals with device manipulation and management. Some of the important system call under this category are:

SetConsoleMode( ): It is used to Sets the input mode of a console's input buffer or the output mode of a console screen buffer.

ReadConsole( ): It is used to reads input character from the input console buffer.

WriteConsole( ): It is used to write character into the output console buffer.

iv. **Information Maintenance:** These system calls handle information and its transfer between the operating system and the user program. Some of the important system call under this category are:

GetCurrentProcessID( ): It is used to get the current executing process ID.

SetTimer( ): It is used to creates a timer with the specified time-out value.

Sleep( ): It is used to make a procee to sleep, which places it into an inactive state.

v. **Communication:** These system calls are useful for inter-process communication. Some of the important system call under this category are:

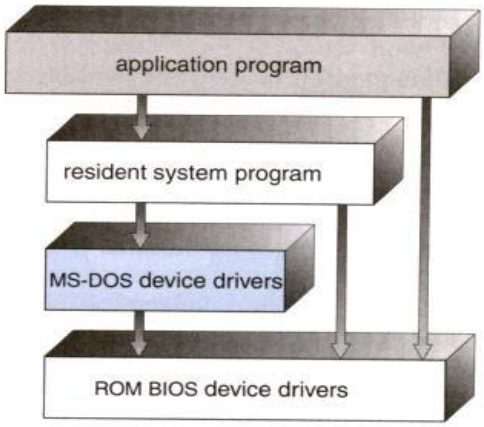
CreatePipe( ): It creates an anonymous pipe to perform read and write operations.

CreateFileMapping( ):

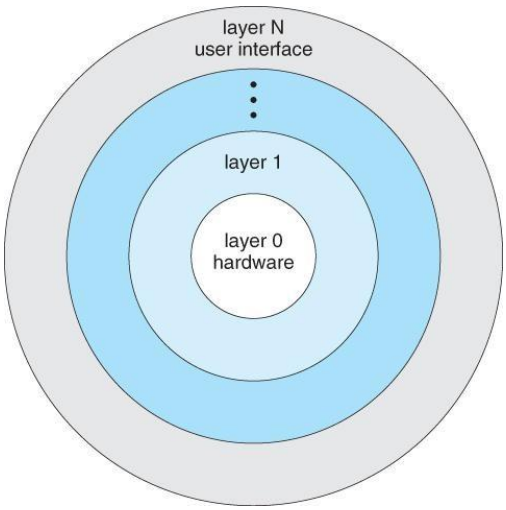
MapViewOfFile( ): It maps a file or device to a memory location.

6. OPERATING SYSTEM STRUCTURE

i. Simple Structure: Operating systems such as MS-DOS and the original UNIX did not have well-defined structures. Frequently, such operating systems started as small, simple, and limited systems and then grew beyond their original scope. MS-DOS is an example of such a system. It was originally designed and implemented by a few people who had no idea that it would become so popular. It was written to provide the most functionality in the least space, so it was not designed carefully. In MS-DOS, the interfaces and levels of functionality are not well separated. So, errors in applications could cause the whole system to crash.

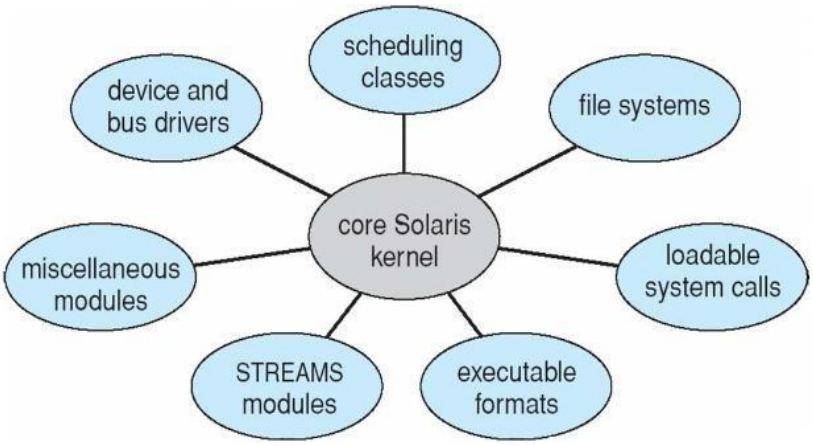


ii. Layered Approach: The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface. The lower layer provides service to its next higher layer. The main advantage of the layered approach is simplicity of construction and debugging. It is easier to create, maintain and update the operating system. Change in one layer does not affect the rest of the layers.



iii. Microkernel: This method structures the operating system by removing all non-essential components from the kernel and implementing them as system or user-level programs. The result is a smaller kernel. Kernel is the core part of an OS which manages system resources. It also acts like a bridge between application and hardware of the computer. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel. All new services are added to user space and as a result there is no need to modify the kernel. As most of the services are running as user processes rather than kernel processes, the microkernel provides more security and reliability.

iv. Modules: The best methodology for operating-system design involves using modular kernel. In this, the kernel has a set of core components and additional services are dynamically linked either during boot time or during run time. It is used in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X. For example, the Solaris operating system structure is organized around a core kernel with seven types of loadable kernel modules:



Even though this structure resembles a layered system, it is more flexible than a layered system because any module can call any other module. Furthermore, this approach is like the microkernel approach but it is more efficient, because modules do not need to invoke message passing in order to communicat