Operating Systems

UNIT - I

Syllabus: Operating Systems Overview: Introduction, Operating system functions, Operating systems operations, Computing environments, Free and Open-Source Operating Systems System Structures: Operating System Services, User and Operating-System Interface, system calls, Types of System Calls, system programs, Operating system Design and Implementation, Operating system structure, Building and Booting an Operating System, Operating system debugging.

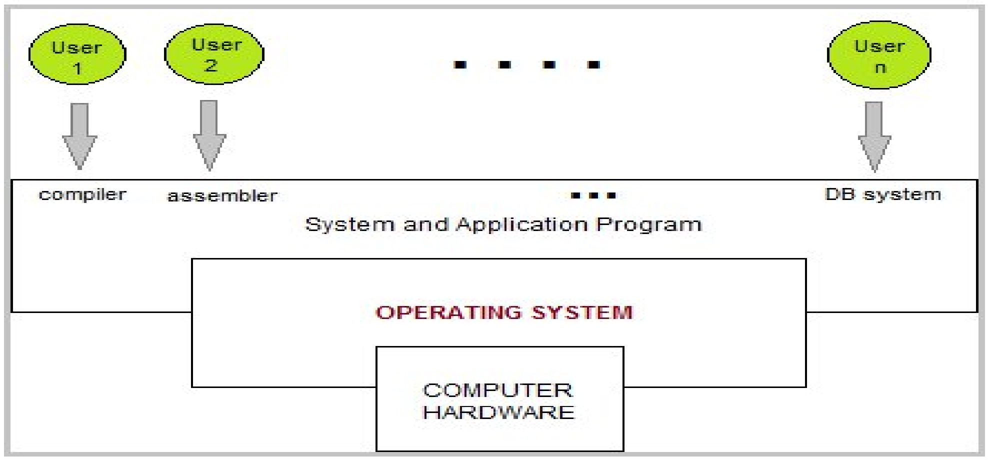
Introduction:

An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

Definition: An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Four Components of a Computer System



Two Views of Operating System

1. User's View
2. System View

1. User View

The user view of the computer refers to the interface being used. Such systems are designed for one user to monopolize its resources, to maximize the work that the user is performing. In these cases, the operating system is designed mostly for ease of use, with some attention paid to performance, and none paid to resource utilization.

2. System View

Operating system can be viewed as a resource allocator also. A computer system consists of many resources like - hardware and software - that must be managed efficiently. The operating system acts as the manager of the resources, decides between conflicting requests, controls execution of programs etc.

Operating system Definition:

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Operating system functions:

Following are some of important **functions** of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management: Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management −

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management: In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management −

- Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management: An Operating System manages device communication via their respective drivers. It does the following activities for device management −

- Keeps tracks of all devices. Program responsible for this task is known as the **I/O controller**.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management: A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management −

- Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities: Following are some of the important activities that an Operating System performs −

- **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** − Recording delays between request for a service and response from the system.
- **Job accounting** − Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** − Production of dumps, traces, error messages, and other debugging and error detecting aids.
- **Coordination between other software's and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Operating systems operations:

Following are some of important **operations** of an operating System.

- Interrupt Handling
- Multitasking
- Dual-Mode Operation
- Virtualization

1. Interrupt Handling

Interrupts are essential for efficient OS operation, allowing it to respond promptly to critical events.

Definition

An **interrupt** is a signal sent to the CPU by hardware or software indicating that an event requires immediate attention.

Types of Interrupts

- Hardware Interrupts:
  - Generated by hardware devices like keyboards, mice, or disk controllers.
  - Examples: A keystroke, a mouse click, or a completed disk read.
- Software Interrupts:

  - Triggered by programs when they need OS services, often via system calls.
  - Example: A request to read a file.

2. Multitasking

Multitasking allows the CPU to manage and execute multiple tasks (processes) simultaneously.

Definition

Multitasking is the ability of the OS to share CPU time among multiple processes, giving the illusion of parallel execution.

Mechanism

- Time Sharing:
  - The CPU divides its time into small intervals, called time slices.
  - Each process gets a time slice in a cyclic order.
- Context Switching:
  - The CPU saves the state of the current process and loads the state of the next process.
  - This involves saving registers, program counters, and memory maps.

Types of Multitasking

- Preemptive Multitasking:
  - The OS can interrupt a process to allocate the CPU to another process.
  - Example: Most modern operating systems like Windows and Linux.
- Cooperative Multitasking:
  - Processes voluntarily yield control of the CPU.
  - Example: Older operating systems like Windows 3.x.

3. Dual-Mode Operation

Dual-mode operation ensures system security and stability by separating user activities from core OS tasks.

Definition

Dual-mode operation involves two modes:

- User Mode:
  - For executing user applications with restricted access to hardware and critical resources.

- Kernel Mode:
  - For executing OS-level tasks with full access to hardware.

Mechanism

- Mode Bit:
  - A hardware-provided mode bit determines the current mode:
    - 0 for kernel mode.
    - 1 for user mode.

- Switching Modes:
    - The OS transitions between user and kernel modes when:
        - A system call is made (user → kernel).
        - The user program resumes execution (kernel → user).

Example:

A file read request involves:

1. A system call from user mode.
2. Switching to kernel mode to access the disk.
3. Returning the result to user mode.

4. Virtualization

Virtualization enables the OS to simulate hardware or create isolated environments for applications.

Definition

Virtualization is the creation of a virtual version of hardware or software resources, allowing multiple operating systems or applications to run on a single physical machine.

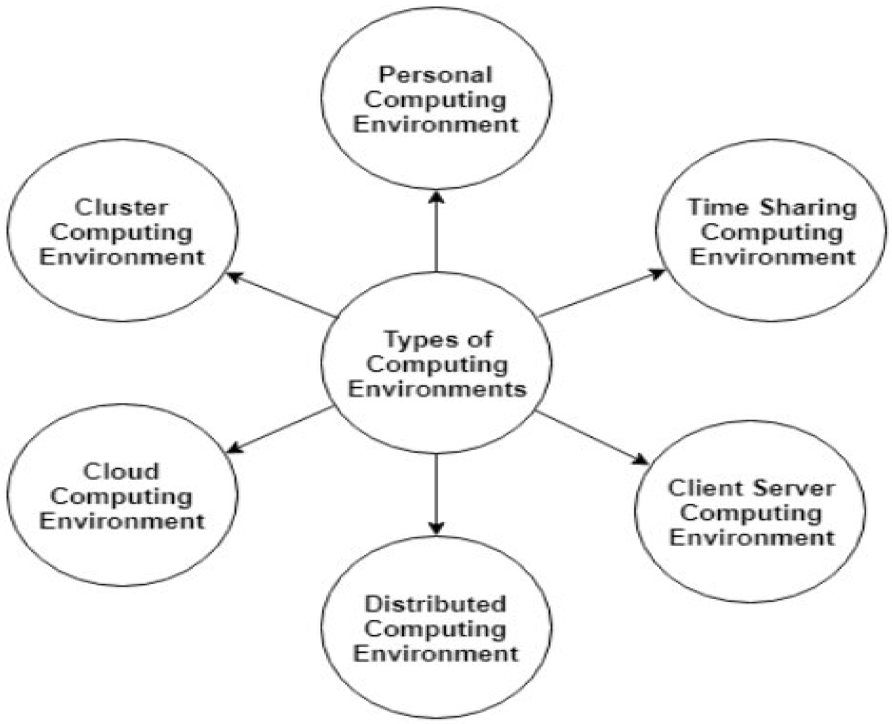
Types of Virtualization

- Hardware Virtualization:
    - Simulates physical hardware to create multiple virtual machines (VMs).
    - Example: VMware, Virtual Box.
- Application Virtualization:
    - Runs applications in isolated environments.
    - Example: Containers like Docker.
- Network Virtualization:
    - Creates virtual network resources, such as virtual switches and routers.
- Desktop Virtualization:
    - Provides virtual desktops to users over a network.

Computing environments:

A computer system uses many devices, arranged in different ways to solve many problems. This constitutes a computing environment where many computers are used to process and exchange information to handle multiple issues.

The different types of Computing Environments are −


Personal Computing Environment
Cluster Computing Environment
Time Sharing Computing Environment
Types of Computing Environments
Cloud Computing Environment
Client Server Computing Environment
Distributed Computing Environment

1.Personal Computing Environment

In the personal computing environment, there is a single computer system. All the system processes are available on the computer and executed there. The different devices that constitute a personal computing environment are laptops, mobiles, printers, computer systems, scanners etc.

2.Time Sharing Computing Environment

The time sharing computing environment allows multiple users to share the system simultaneously. Each user is provided a time slice and the processor switches rapidly among the users according to it. Because of this, each user believes that they are the only ones using the system.

3.Client Server Computing Environment

In client server computing, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network but sometimes they may reside in the same system.

4.Distributed Computing Environment

A distributed computing environment contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.

5.Cloud Computing Environment

The computing is moved away from individual computer systems to a cloud of computers in cloud computing environment. The cloud users only see the service being provided and not the internal details of how the service is provided. This is done by pooling all the computer resources and then managing them using a software.

6.Cluster Computing Environment

The clustered computing environment is similar to parallel computing environment as they both have multiple CPUs. However a major difference is that clustered systems are created by two or more individual computer systems merged together which then work parallel to each other.

Free and Open-Source Operating Systems:

Free and Open-Source Operating Systems (FOSS OS) play a significant role in modern computing due to their flexibility, transparency, and cost-effectiveness.

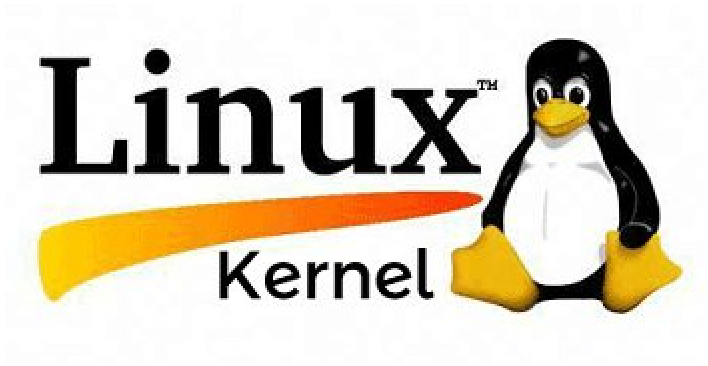
Definition

A **Free and Open-Source Operating System** is software distributed under an open-source license that grants users the freedom to:

- **View** and study the source code.
- **Modify** the software to suit their needs.
- **Distribute** copies of the original or modified software.

The open-source operating systems are Linux Kernel, Linux Lite, Linux mint, Fedora, Solus**,** Chrome OS etc.

Linux Kernel



Linux kernel was developed by Linus Torvalds. It offers the essential functions required for an operating system, such as data cancellation, memory processing, and interactions with computer hardware. It is open-source software, and many developers researched the source code and produced a plethora of helpful plug-ins and operating systems to meet their requirements.

Linux Lite

Linux Lite is a lightweight open-source operating system designed for beginners with less knowledge about Linux-based operating systems and can be installed on any old PC and run smoothly.

Linux mint

Linux Mint is a powerful Linux-based operating system that exudes modernity and power. It is simple to use and includes complete multimedia capabilities, making it a user-friendly open-source operating system. It is an Ubuntu-based distribution that is popular among both beginners and experts. It is built on the Debian platform and includes one of the most powerful software managers. It is more stable and has better visual aesthetics than Ubuntu.

Fedora

Fedora is another popular Linux-based operating system, and it is widely considered the best open-source operating system after Ubuntu. It is an RPM-based general-purpose operating system that is supported by Red Hat and built by the Fedora Project community.

React OS

ReactOS is another free and open-source operating system that has nearly 1 million downloads in over **100** countries. This community-based OS may run Windows apps, making it an excellent alternative to the Windows operating system.

Solus

Solus is a free and open-source operating system for your desktop computer. It's a new operating system from the Linux family, released in **2012**. More than **6000** registered users are currently using the software. VLC, XChat, Transmission, Thunderbird, OpenShot Video Editor, Firefox, Budgie desktop environment, and LibreOffice Suite are all included with Solus.

Chrome OS

Chrome OS is a partly open-source operating system with various attractive features. It's a part of the Chromium and Linux families, with features including better security, compatibility for supported Android and Chrome apps, Aura windows manager, Google cloud print, integrated media player, virtual desktop access, and cloud-based management. The only issue with the operating system is that it only supports Nexus devices or its hardware.

Comparison:

Aspect	Free and Open-Source OS	Proprietary OS
Cost	Free	Requires licensing fees (e.g., Windows, macOS).
Customization	Fully customizable	Limited customization, controlled by the vendor.
Transparency	Open source; anyone can inspect the code	Closed source; code is hidden from users.
Support	Community-driven, with optional paid support	Vendor-provided support with costs.
Ease of Use	Can have a steep learning curve	User-friendly and designed for non-technical users.
Security	Open review reduces vulnerabilities	Security depends on vendor updates and patches.

Operating System Services:

- **User-Oriented Services:**
  - Program execution, I/O operations, file system manipulation.
- **System-Oriented Services:**
  - Resource allocation, accounting, error detection, protection and security.

User-Oriented Services:

- **Program execution:** The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).
- **I/O operations:** A running program may require I/O device. For specific devices, special functions may be desired (such as recording to a CD or DVD drive). For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.
- **File-system manipulation:** Operating System is used to control operations on files like creating, opening, reading, and writing.

System-Oriented Services:

- **Resource allocation.** When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.
- **Error detection.** The operating system needs to be constantly aware of possible errors. Errors may occur in the CPU and memory, in I/O devices, and in the user program. For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.
- **Accounting:** Accounting keep track of which users use how much and what kinds of computer resources. This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics.
- **Protection and security.** The owners of information stored in a multiuser or networked computer system may want to control use of that information. When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the operating system itself. Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources.

User and Operating-System Interface:

The **User and Operating-System Interface** connects users with the operating system, enabling them to interact and execute tasks such as file management and running applications. The three main types of interfaces are the Command-Line Interface (CLI), Graphical User Interface (GUI), and Touch-Based Interfaces. Each serves different purposes depending on the user's needs and the system's capabilities.

1. Command-Line Interface (CLI)

The CLI allows users to type commands into a terminal to directly interact with the operating system. It is efficient, resource-light, and suitable for advanced users who need precision and control. However, it has a steep learning curve and requires familiarity with command syntax. CLI is commonly used by system administrators and developers for automation and advanced tasks.

2. Graphical User Interface (GUI)

The GUI provides a user-friendly environment with visual elements like windows, icons, and menus. Users can interact using a mouse, trackpad, or touchscreen. It is intuitive and ideal for everyday tasks, making it popular in operating systems like Windows and macOS. While easier to use, GUI systems require more system resources compared to CLI.

3. Touch-Based Interfaces

Touch-based interfaces are designed for devices with touchscreens, allowing users to interact through gestures like tapping and swiping. They are common in mobile operating systems like Android and iOS, providing a natural and intuitive experience. While user-friendly, they may lack precision and are better suited for simple tasks rather than complex workflows.

4. Comparison of Interfaces:

Feature	CLI	GUI	Touch-Based Interface
Ease of Use	Difficult for beginners	Easy and intuitive	Extremely user-friendly
Resource Usage	Minimal	High	Moderate to high
Task Complexity	Ideal for advanced tasks	Suitable for general tasks	Limited to basic and moderate tasks
Learning Curve	Steep	Low	Very low
Devices	Servers, professional systems	Desktops, laptops	Mobile devices

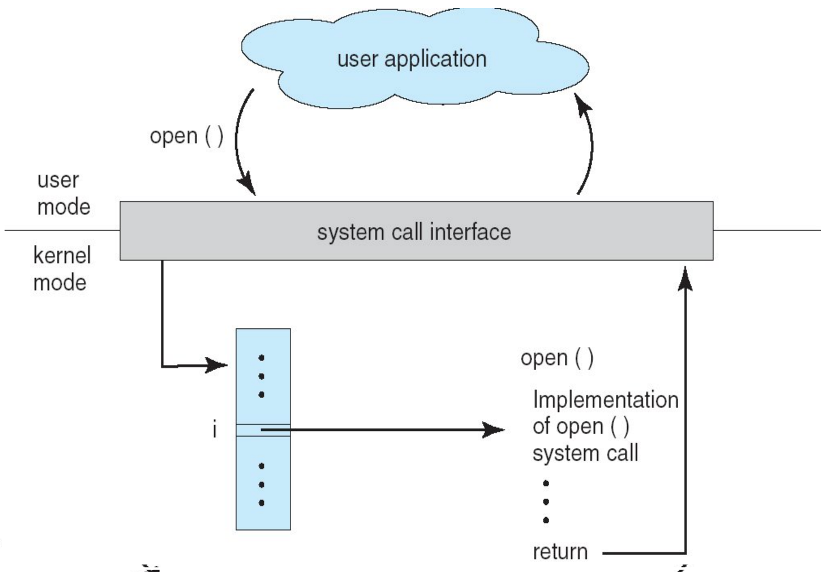
System Calls:

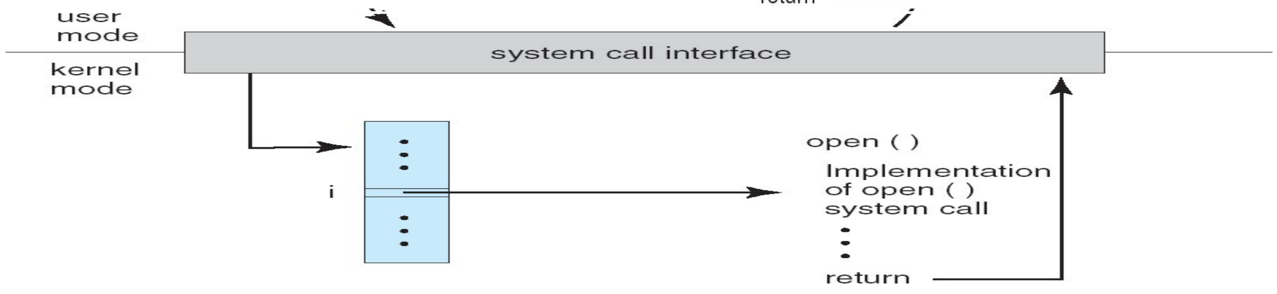
In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls:

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection

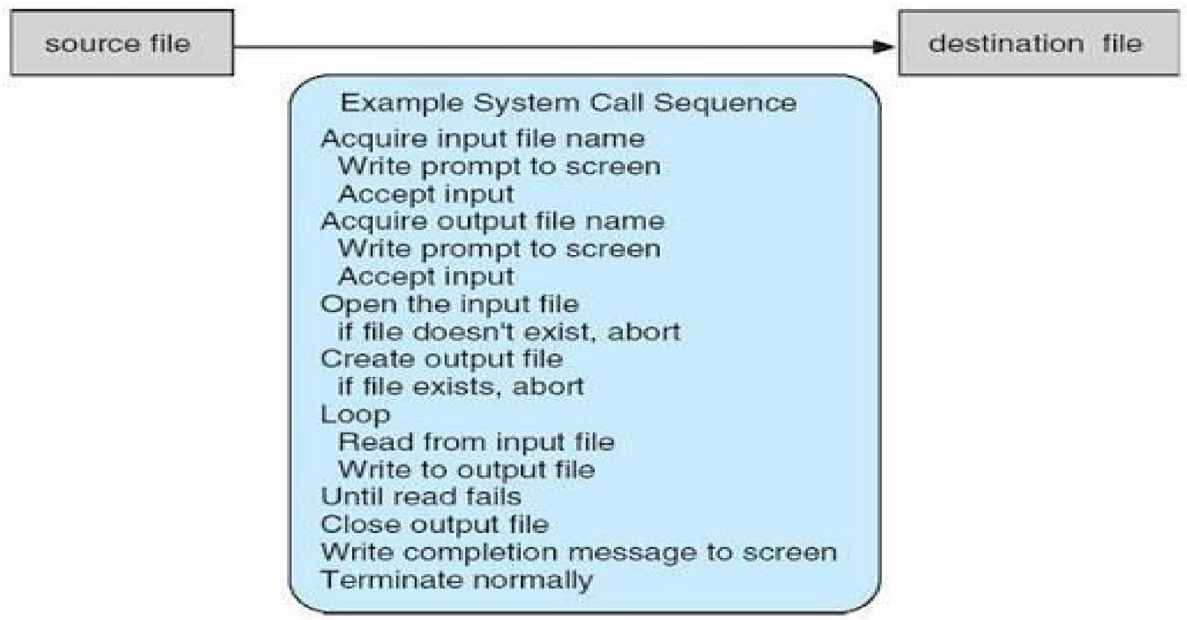
Networking, etc





Example of System Call:

- For example if we need to write a program code to read data from one file, copy that data into another file. The first information that the program requires is the name of the two files, the input and output files.
- First call is to write a prompting message on the screen
- Second, to read from the keyboard, the characters which define the two files.



```
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
```

How System Call works:

Step 1) The processes executed in the user mode till the time a system call interrupts it.
Step 2) After that, the system call is executed in the kernel-mode on a priority basis.
Step 3) Once system call execution is over, control returns to the user mode.,
Step 4) The execution of user processes resumed in Kernel mode.

Why do you need System Call in OS:

- Reading and writing from files demand system calls.
- If a file system wants to create or delete files, system calls are required.

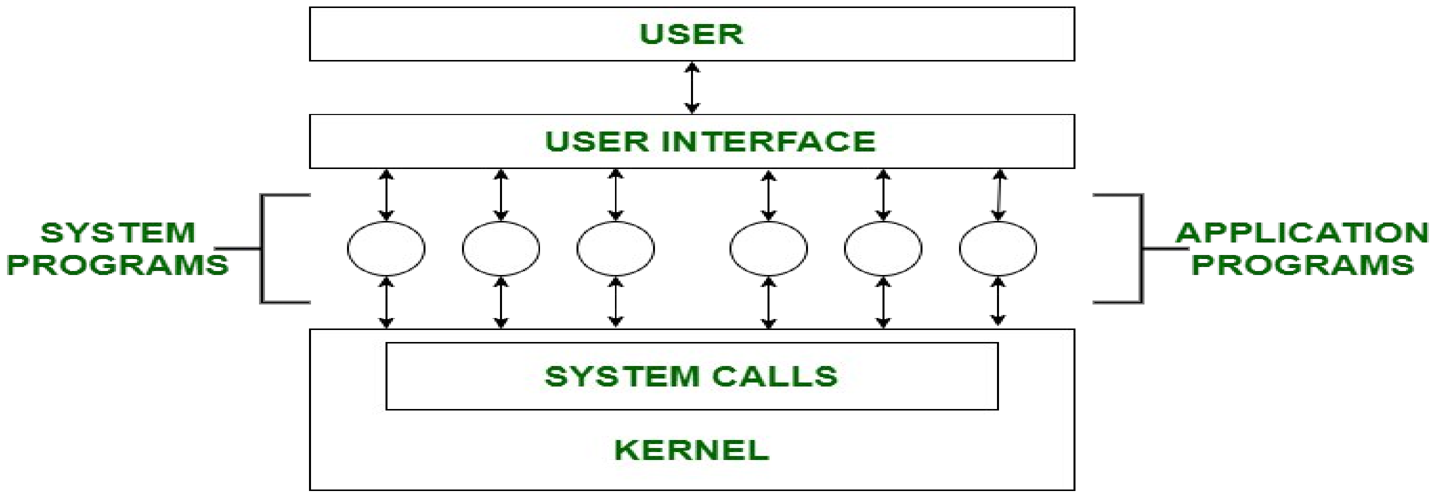
- System calls are used for the creation and management of new processes.
- Network connections need system calls for sending and receiving packets.
- Access to hardware devices like scanner, printer, need a system call.

Types of System Calls: There are 5 different types of system calls.

1. Process Control:
   This system calls perform the task of process creation, process termination, etc.
   - fork ( ) : This system call is used to create a new process, which is called as child process.
   - wait ( ): A call to wait() blocks the calling process until one of its child processes exits or a signal is received.
   - exit ( ): This system call terminates a process normally.
2. File Manipulation:
   File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.
   - open ( ): This is used for creating and opening a file.
   - read ( ): This is used to read content from specified file descriptor.
   - write ( ): This is used to wrote content to specified file descriptor.
   - close ( ): This is used to close opened file descriptors.
3. Device Manipulation:
   Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.
   - iocntl ( ): This system call is used to control specified I/O device.
   - read ( ): This system call is used to read data from specified device.
   - write ( ): This system call is used to write data to specified device.
4. Communications:
   These types of system calls are specially used for interprocess communications.
   - pipe ( ) : This system call sends output of one command as input of another command.
   - shmget ( ): This system call is used to create a shared memory.
   - shmat ( ): This system call is used to access shared memory.
5. Information Maintenance:
   It handles information and its transfer between the OS and the user program
   - getpid ( ): This system call is used to get the process id of the current process.
   - alarm ( ): This system call only be **used** by a process to send a specific signal to itself.
   - sleep ( ): This **system call is used** when user wants to halt the current process for sometime. It keep the locks hold on resources till the **sleep** time is over and again starts the execution of the process. Here process has control throughout the execution.

System programs:

System programs in an operating system are software tools that help users manage files, run programs, and control system resources. They include file managers, program loaders, compilers, and system utilities, making it easier to operate the computer efficiently and keep it running smoothly.



Different types of system programs are:

1. **File Management**: A file is a collection of specific information stored in the memory of a computer system. File management is defined as the process of manipulating files in the computer system, its management includes the process of creating, modifying and deleting files.
2. **Command Line Interface (CLI's) :** CLIs is the essential tool for user . It provide user facility to write commands directly to the system for performing any operation. It is a text-based way to interact with operating system. CLIs can perform many tasks like file manipulation, system configuration and etc.
3. **Device drivers:** Device drivers work as a simple translator for OS and devices . Basically it act as an intermediatory between the OS and devices and provide facility to both OS and devices to understand each other's language so that they can work together efficiently without interrupt.
4. **Status Information**: Information like date, time amount of available memory, or disk space is asked by some users. Others providing detailed performance, logging, and debugging information which is more complex. All this information is formatted and displayed on output devices or printed. Terminal or other output devices or files or a window of GUI is used for showing the output of programs.
5. **File Modification:** This is used for modifying the content of files. Files stored on disks or other storage devices, we use different types of editors. For searching contents of files or perform transformations of files we use special commands.
6. **Programming-Language support:** For common programming languages, we use Compilers, Assemblers, Debuggers, and interpreters which are already provided to users. It provides all support to users. We can run any programming language. All-important languages are provided.
7. **Program Loading and Execution:** When the program is ready after Assembling and compilation, it must be loaded into memory for execution. A loader is part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages for starting a program. Loaders, relocatable loaders, linkage editors, and Overlay loaders are provided by the system.
8. **Communications:** Connections among processes, users, and computer systems are provided by programs. Users can send messages to another user on their screen, User can send e-mail, browsing on web pages, remote login, the transformation of files from one user to another.

Operating system Design and Implementation:

An operating system is a construct that allows the user application programs to interact with the system hardware. Operating system by itself does not provide any function but it provides an atmosphere in which different applications and programs can do useful work.

There are many problems that can occur while designing and implementing an operating system.

Operating System Design Goals

It is quite complicated to define all the goals and specifications of the operating system while designing it.The design changes depending on the type of the operating system i.e if it is batch system, time shared system, single user system, multi user system, distributed system etc.

There are basically two types of goals while designing an operating system. These are −

User Goals

The operating system should be convenient, easy to use, reliable, safe and fast according to the users. However, these specifications are not very useful as there is no set method to achieve these goals.

System Goals

The operating system should be easy to design, implement and maintain. These are specifications required by those who create, maintain and operate the operating system. But there is not specific method to achieve these goals as well.

Operating System Mechanisms and Policies

There is no specific way to design an operating system as it is a highly creative task. However, there are general software principles that are applicable to all operating systems.

A subtle difference between mechanism and policy is that mechanism shows how to do something and policy shows what to do. Policies may change over time and this would lead to changes in mechanism. So, it is better to have a general mechanism that would require few changes even when a policy change occurs.

For example - If the mechanism and policy are independent, then few changes are required in mechanism if policy changes. If a policy favours I/O intensive processes over CPU intensive processes, then a policy change to preference of CPU intensive processes will not change the mechanism.

Operating System Implementation

The operating system needs to be implemented after it is designed. Earlier they were written in assembly language but now higher level languages are used. The first system not written in assembly language was the Master Control Program (MCP) for Burroughs Computers.

Operating system structure:

A system structure for an operating system is like the blueprint of how an OS is organized and how its different parts interact with each other.

the following structures in the operating system:

- Simple Structure
- Monolithic Structure
- Micro-Kernel Structure
- Layered Structure
- Modular Structure

1. Simple Structure

Simple structure operating systems do not have well-defined structures and are small, simple, and limited. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such an operating system. In MS-DOS, application programs are able to access the basic I/O routines. These types of operating systems cause the entire system to crash if one of the user programs fails.

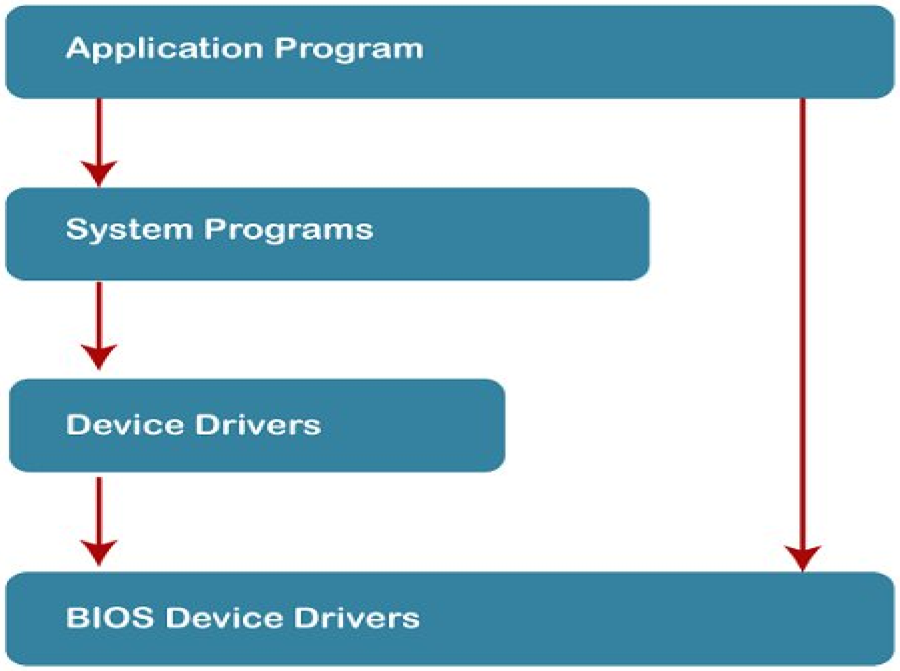


Fig: MS-DOS Layer structure

2. Monolithic Structure:

A monolithic structure is a type of operating system architecture where the entire operating system is implemented as a single large process in kernel mode. Essential operating system services, such as process management, memory management, file systems, and device drivers, are combined into a single code block.

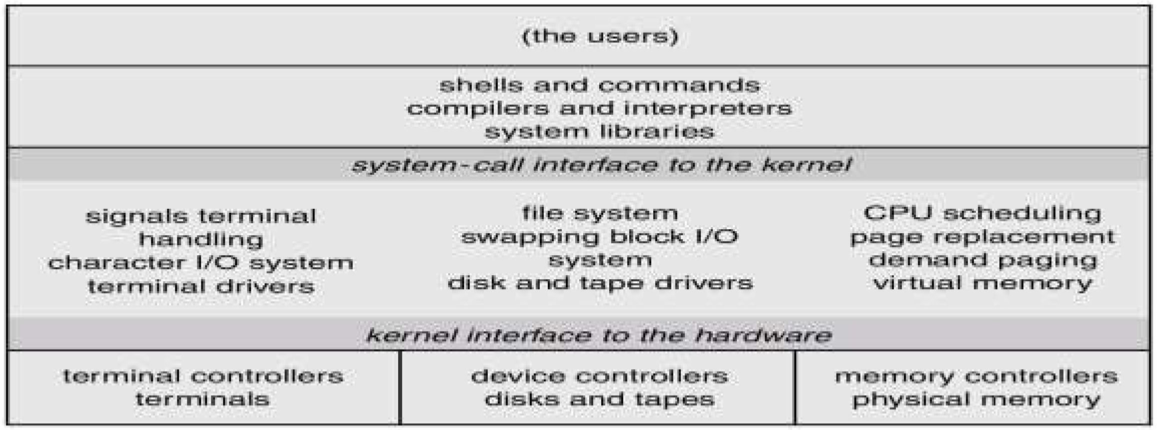


Fig: Monolithic Structure

3. Micro-Kernel Structure:

Micro-Kernel structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This results in a smaller kernel called the micro-kernel. Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails, then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

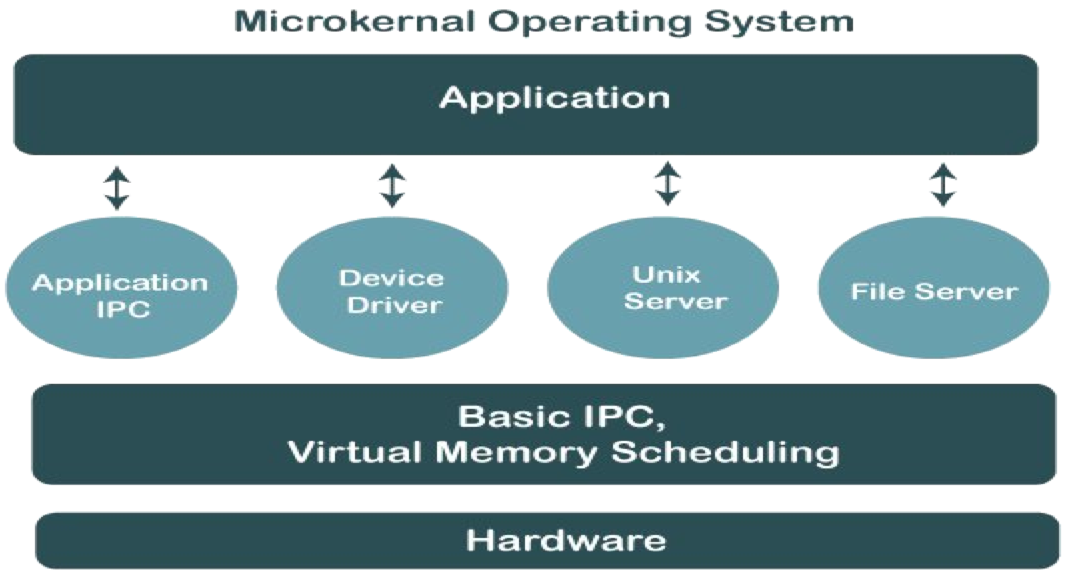


Fig: Micro-Kernel Structure

4. Layered Structure:

An OS can be broken into pieces and retain much more control over the system. In this structure, the OS is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware, and the topmost layer (layer

N) is the user interface. These layers are so designed that each layer uses the functions of the lower-level layers. This simplifies the debugging process, if lower-level layers are debugged and an error occurs during debugging, then the error must be on that layer only, as the lower-level layers have already been debugged. The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover, careful planning of the layers is necessary, as a layer can use only lower-level layers. UNIX is an example of this structure.

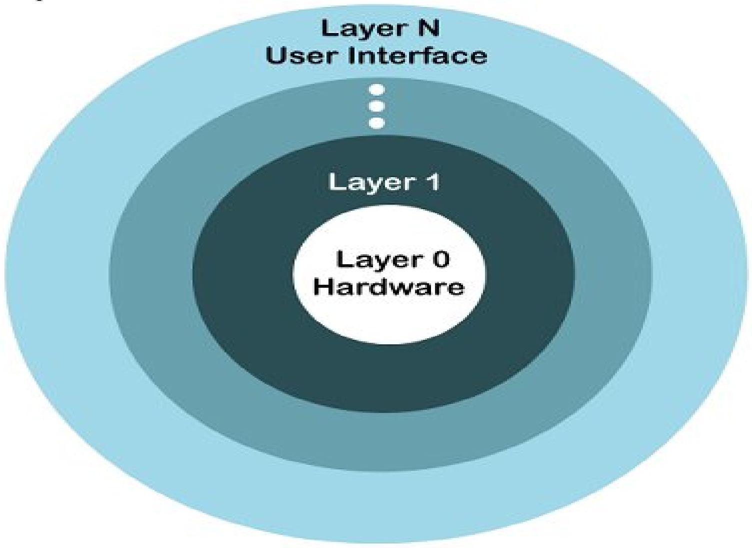


Fig: Layered Structure

5. Modular Structure:

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only a set of core components and other services are added as dynamically loadable modules to the kernel either during runtime or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces, but it is more flexible than a layered structure as a module can call any other module. For example Solaris OS is organized as shown in the figure.

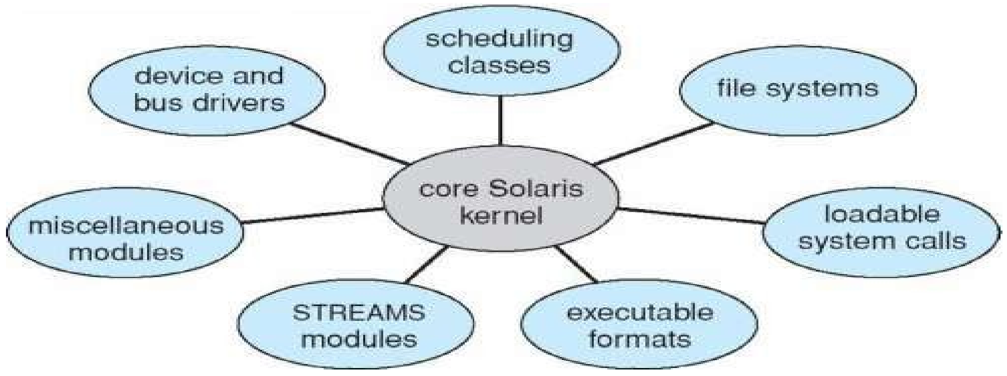


Fig: Modular Structure

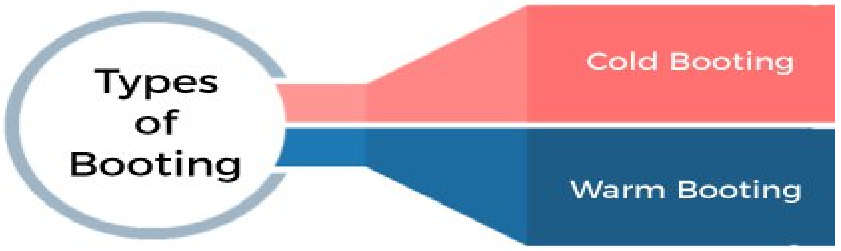
Building and Booting an Operating System:

Booting:

When a computer system is started, there is a mechanism in the system that loads the operating system from the secondary storage into the main memory, or RAM, of the system. This is called the **booting process** of the system.

Types of Booting

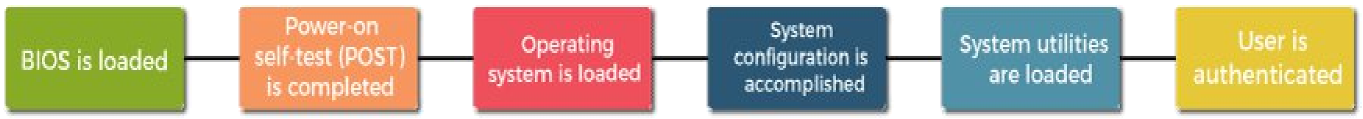
There are two types of booting in an operating system.



1. **Cold Booting:** When the computer starts for the first time or is in a shut-down state and switch on the power button to start the system, this type of process to start the computer is called cold booting. During cold booting, the system will read all the instructions from the ROM (BIOS) and the Operating System will be automatically get loaded into the system. This booting takes more time than Hot or Warm Booting.
2. **Warm Booting:** Warm or Hot Booting process is when computer systems come to no response or hang state, and then the system is allowed to restart during on condition. It is also referred to as rebooting. There are many reasons for this state, and the only solution is to reboot the computer. Rebooting may be required when we install new software or hardware. The system requires a reboot to set software or hardware configuration changes, or sometimes systems may behave abnormally or may not respond properly. In such a case, the system has to be a force restart. Most commonly ***Ctrl+Alt+Del*** button is used to reboot the system. Else, in some systems, the external reset button may be available to reboot the system.

Booting Process in Operating System

When our computer is switched on, it can be started by hardware such as a button press, or by software command, a computer's central processing unit (CPU) has no software in its main memory, there is some process which must load software into main memory before it can be executed. Below are the six steps to describe the boot process in the operating system, such as:


BIOS is loaded
Power-on self-test (POST) is completed
Operating system is loaded
System configuration is accomplished
System utilities are loaded
User is authenticated

Step 1: Once the computer system is turned on, ***BIOS*** (Basic Input /Output System) performs a series of activities or functionality tests on programs stored in ROM, called on ***POST*** (Power-on Self-Test) that checks to see whether peripherals in the system are in perfect order or not.

Step 2: After the BIOS is done with pre-boot activities or functionality test, it read bootable sequence from ***CMOS*** (Common Metal Oxide Semiconductor) and looks for master boot record in the first physical sector of the bootable disk as per boot device sequence specified in ***CMOS***. For example, if the boot device sequence is:

- Floppy Disk
- Hard Disk
- CDROM

Step 3: After this, the master boot record will search first in a floppy disk drive. If not found, then the hard disk drive will search for the master boot record. But if the master boot record is not even present on the hard disk, then the CDROM drive will search. If the system cannot read the master boot record from any of these sources, ROM displays **"*No Boot device found*"** and halted the system. On finding the master boot record from a particular bootable disk drive, the operating system loader, also called Bootstrap loader, is loaded from the boot sector of that bootable drive· into memory. A bootstrap loader is a special program that is present in the boot sector of a bootable drive.

Step 4: The bootstrap loader first loads the ***IO.SYS*** file. After this, ***MSDOS.SYS*** file is loaded, which is the core file of the DOS operating system.

Step 5: After this, **MSDOS.SYS** file searches to find Command Interpreter in **CONFIG.SYS** file, and when it finds, it loads into memory. If no Command Interpreter is specified in the **CONFIG.SYS** file, the **COMMAND.COM** file is loaded as the default Command Interpreter of the DOS operating system.

Step 6: The last file is to be loaded and executed is the **AUTOEXEC.BAT** file that contains a sequence of DOS commands. After this, the prompt is displayed. We can see the drive letter of bootable drive displayed on the computer system, which indicates that the operating system has been successfully on the system from that drive.

Dual Booting
When two operating systems are installed on the computer system, then it is called dual booting. Multiple operating systems can be installed on such a system. But to know which operating system is to boot, a boot loader that understands multiple file systems and multiple operating systems can occupy the boot space.

Comparison between Booting and Dual Booting

Parameter	**Booting**	**Dual Booting**
Definition	The process of starting up a computer	The process of installing and running multiple operating systems on a single computer
Purpose	Loads the operating system into memory and initializes the computer	Allows users to choose between different operating systems at start-up
Single OS	Only one operating system is	Multiple operating systems are installed on

Parameter	Booting	Dual Booting
	installed and runs on the computer	different partitions or drives
Configuration	The computer is configured to boot directly into the installed operating system	The computer is configured with a boot loader to choose between different operating systems
Setup Complexity	Relatively simpler, as there is only one operating system to configure	Requires additional setup and configuration to manage multiple operating systems
Resource Utilization	Utilizes the full resources of the computer for a single operating system	Resources are divided among the installed operating systems, potentially affecting performance

Operating system debugging:

Debugging is the activity of finding and fixing errors in a system, both in hardware and in software. Debugging can also include performance tuning, which seeks to improve performance by removing processing bottlenecks (points in a system or process where the flow or speed of movement is limited or constrained).

Process Failure Analysis

If a process fails, operating systems write the error information to a log file to alert system operators or users that the problem occurred. The operating system can also take a capture of the memory of the process and store it in a file for later analysis.

A failure in the kernel is called a crash. When a crash occurs, error information is saved to a log file, and the memory state is saved to a crash dump. If in the file-system code kernel failure occurs then the kernel should try to save the state of the file on the file system before rebooting.

Performance Tuning

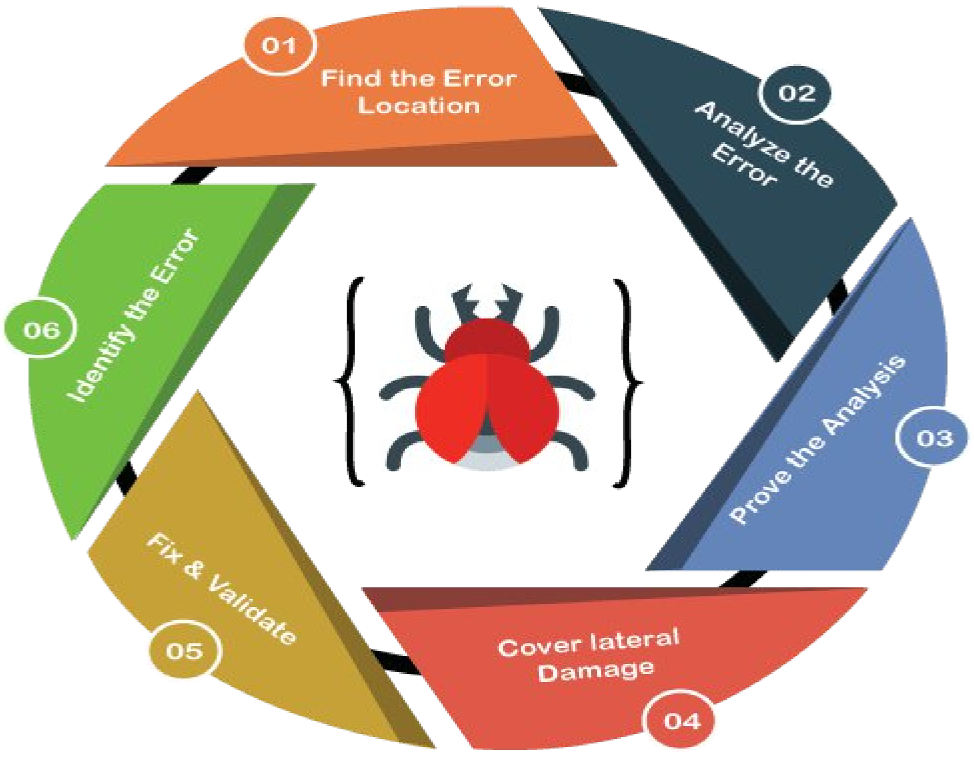
Performance tuning seeks to improve performance by removing processing bottlenecks. To identify bottlenecks, we monitor system performance.

Approach for Performance Tuning

The operating system has means of computing and displaying measures of system behavior. In a number of systems, the operating system produces trace listings of system behaviour. All interesting events are logged with their time and important parameters and are written to a file. Later, an analysis program can process the log file to determine system performance and identify bottlenecks and inefficiencies. These traces can be run as input for a simulation of a suggested improved system. Traces also can help people **operating system**find errors in operating-system behaviour.

Different Steps involved in Debugging:

Following are the different steps that are involved in debugging:



1. **Identify the Error:** Identifying an error in a wrong may result in the wastage of time. It is very obvious that the production errors reported by users are hard to interpret, and sometimes the information we receive is misleading. Thus, it is mandatory to identify the actual error.
2. **Find the Error Location:** Once the error is correctly discovered, you will be required to thoroughly review the code repeatedly to locate the position of the error. In general, this step focuses on finding the error rather than perceiving it.
3. **Analyse the Error:** The third step comprises error analysis, a bottom-up approach that starts from the location of the error followed by analyzing the code. This step makes it easier to comprehend the errors. Mainly error analysis has two significant goals, i.e., evaluation of errors all over again to find existing bugs and postulating the uncertainty of incoming collateral damage in a fix.
4. **Prove the Analysis:** After analyzing the primary bugs, it is necessary to look for some extra errors that may show up on the application. By incorporating the test framework, the fourth step is used to write automated tests for such areas.
5. **Cover Lateral Damage:** The fifth phase is about accumulating all of the unit tests for the code that requires modification. As when you run these unit tests, they must pass.
6. **Fix & Validate:** The last stage is the fix and validation that emphasizes fixing the bugs followed by running all the test scripts to check whether they pass.