Operating System - Services

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

1. Program execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

2. I/O Operation

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

3. File system manipulation

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

4. Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

5. Error handling

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

6. Resource Management

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

7. **Protection**

Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

What is the user interface and operating system interface?

The user and operating system are connected with each other with the help of interface, so interface is used to connect the user and OS.

In computers there are different types of interface that can be used for connection with computers to users and their connection is responsible for data transfer.

Also, in computers there are different interfaces. These interfaces are not necessarily used but can be used in computers whenever it is needed. So, different types of tasks can be performed by the help of different interfaces.

Command line interface

The command-line interface is an interface whenever the user needs to have different commands regarding the input and output and then a task is performed so this is called the command-line argument and it is used to execute the output and create, delete, print, copy, paste, etc.

All these operations are performed with the help of the command-line interface.

The interface is always connected to the OS so that the command given by the user directly works by the OS and a number of operations can be performed with the help of the command line interface because multiple commands can be interrupted at same time and execute only one.

The command line interface is necessary because all the basic operations in the computer are performed with the help of the OS and it is responsible for memory management. By using this we can divide the memory and we can use the memory.

Command Line Interface advantages −

- Controls OS or application

- faster management
- ability to store scripts which helps in automating regular tasks.
- Troubleshoot network connection issues.

Command Line Interface disadvantages –

- The steeper learning curve is associated with memorizing commands and a complex syntax.
- Different commands are used in different shells.

Graphical user interface

The graphical user interface is used for playing games, watching videos, etc. these are done with the help of GUI because all these applications require graphics.

The GUI is one of the necessary interfaces because only by using the user can clearly see the picture, play videos.

So we need GUI for computers and this can be done only with the help of an operating system. When a task is performed in the computer then the OS checks the task and defines the interface which is necessary for the task. So, we need GUI in the OS.

The basic components of GUIs are –

- Start menu with program groups
- Taskbar which showing running programs
- Desktop screen
- Different icons and shortcuts.

Choice of interface

The interface that is used with the help of OS for a particular task and that task can be performed with minimum possible time and the output is shown on the screen in that case we use the choice of interface.

The choice of interface means the OS checks the task and finds out which interface can be suitable for a particular task. So that type of interface is called the choice of interface and this can be done with the help of an OS.

System Calls in Operating System (OS)

A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request. A system call can be written in assembly language or a high-level language like C or Pascal. System calls are predefined functions that the operating system may directly invoke if a high-level language is used.

In this article, you will learn about the system calls in the operating system and discuss their types and many other things.

What is a System Call?

A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.

The Application Program Interface (API) connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.

How are system calls made?

When a computer software needs to access the operating system's kernel, it makes a system call. The system call uses an API to expose the operating system's services to user programs. It is the only method to access the kernel system. All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.

Below are some examples of how a system call varies from a user function.

1. A system call function may create and use kernel processes to execute the asynchronous processing.
2. A system call has greater authority than a standard subroutine. A system call with kernel-mode privilege executes in the kernel protection domain.
3. System calls are not permitted to use shared libraries or any symbols that are not present in the kernel protection domain.
4. The code and data for system calls are stored in global kernel memory.

Why do you need system calls in Operating System?

There are various situations where you must require system calls in the operating system. Following of the situations are as follows:

1. It is must require when a file system wants to create or delete a file.
2. Network connections require the system calls to sending and receiving data packets.
3. If you want to read or write a file, you need to system calls.
4. If you want to access hardware devices, including a printer, scanner, you need a system call.
5. System calls are used to create and manage new processes.

How System Calls Work

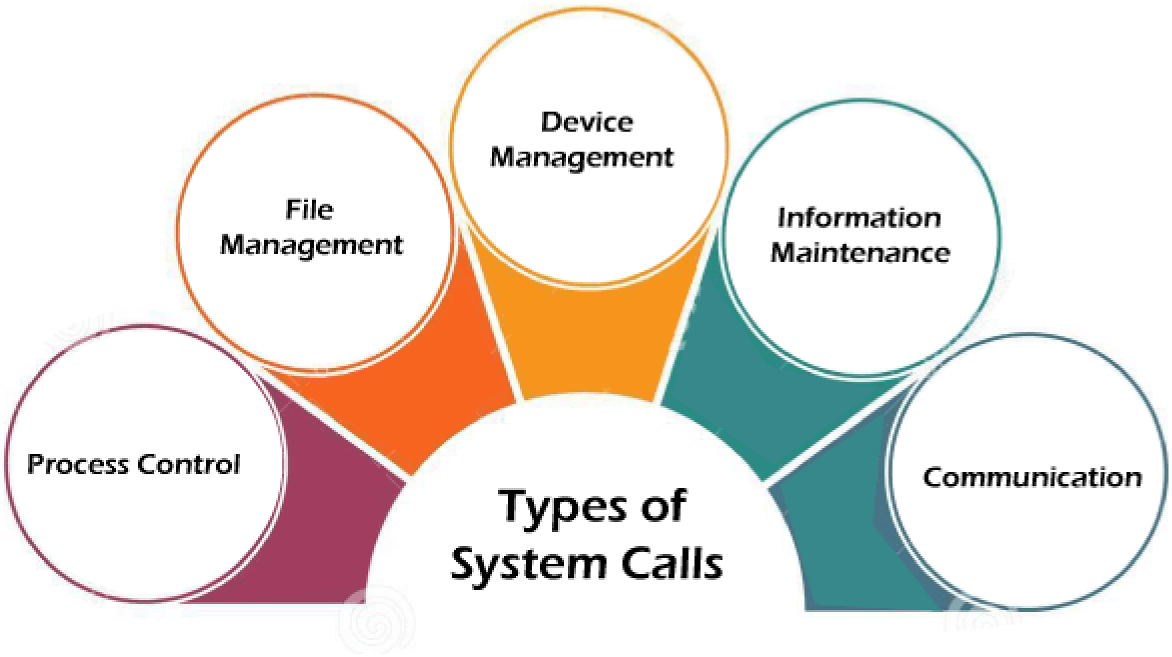
The Applications run in an area of memory known as user space. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel. It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.

If the request is permitted, the kernel performs the requested action, like creating or deleting a file. As input, the application receives the kernel's output. The application resumes the procedure after the input is received. When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.

A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

Types of System Calls

There are commonly five types of system calls. These are as follows:



1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communication

Now, you will learn about all the different types of system calls one-by-one.

Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Examples of Windows and Unix system calls

There are various examples of Windows and Unix system calls. These are as listed below in the table:

Process	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		Fork()
		Exit()
		Wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		Open()
		Read()
		Write()
		Close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		Ioctl()
		Read()
		Write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		Getpid()
		Alarm()
		Sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		Pipe()
		Shmget()
		Mmap()

Protection	SetFileSecurity()
	InitializeSecurityDescriptor()
	SetSecurityDescriptorgroup()
		Chmod()
		Umask()
		Chown()

Here, you will learn about some methods briefly:

open()

The open() system call allows you to access a file on a file system. It allocates resources to the file and provides a handle that the process may refer to. Many processes can open a file at once or by a single process only. It's all based on the file system and structure.

read()

It is used to obtain data from a file on the file system. It accepts three arguments in general:

- A file descriptor.
- A buffer to store read data.
- The number of bytes to read from the file.

The file descriptor of the file to be read could be used to identify it and open it using open() before reading.

wait()

In some systems, a process may have to wait for another process to complete its execution before proceeding. When a parent process makes a child process, the parent process execution is suspended until the child process is finished. The wait() system call is used to suspend the parent process. Once the child process has completed its execution, control is returned to the parent process.

write()

It is used to write data from a user buffer to a device like a file. This system call is one way for a program to generate data. It takes three arguments in general:

- A file descriptor.
- A pointer to the buffer in which data is saved.
- The number of bytes to be written from the buffer.

fork()

Processes generate clones of themselves using the fork() system call. It is one of the most common ways to create processes in operating systems. When a parent process spawns a child process, execution of the parent process is interrupted until the child process completes. Once the child process has completed its execution, control is returned to the parent process.

close()

It is used to end file system access. When this system call is invoked, it signifies that the program no longer requires the file, and the buffers are flushed, the file information is altered, and the file resources are de-allocated as a result.

exec()

When an executable file replaces an earlier executable file in an already executing process, this system function is invoked. As a new process is not built, the old process identification stays, but the new process replaces data, stack, data, head, etc.

exit()

The exit() is a system call that is used to end program execution. This call indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the exit() system function.

Differentiate between Application Programming Interfaces (APIs) and system calls.

Application programming interface

We know that multiple devices and applications share data between them. Some devices include online reservations and some in booking systems.

API (Application Programming Interface) is used to establish connectivity among devices and applications. However, it is an interface which takes the requests from the user and informs the system about what has to be done and returns the response back to the user.

Example

Consider an online travel agency having information about multiple airlines. The travel agency interacts with the airline's API.

The Application interface takes the requests from the customer to book seats and also select meals from the travel service to the airline system. Then, the API delivers the airline's responses back to the online travel agency and they display the details to the users.

This is an example of real-world application for an API.

System call

System call provides an interface between user program and operating system.

When the user wants to give an instruction to the OS then it will do it through system calls. Or a user program can access the kernel which is a part of the OS through system calls.

It is a programmatic way in which a computer program requests a service from the kernel of the operating system.

Program executes in two modes, which are as follows −

- **User mode** − Cannot access any hardware resources, which perform only the user operations.
- **Kernel mode** − Can access hardware resources like RAM, Printer.

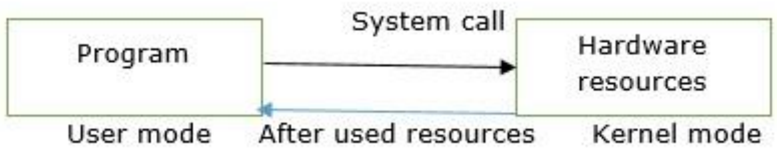
The processor in a computer switches between the two modes depending upon what types of code are running on the processor. A process running in the user mode cannot access the virtual addresses that are reserved for the operating system.

The system is in user mode when the operating system is running a user application such as handling a text editor. The transition from the user mode to kernel mode occurs, when the application requests the help of the operating system or an interrupt or a system call occurs.

The mode bit is set to 1 in the user mode. When a program needs any hardware resources, it needs to make a call to the kernel.

Through system call, the program will switch to the kernel. It will happen with the hardware resources in the kernel mode. After compilation of the work of hardware resources, it will again come back to user mode. When it will require hardware then only it will come to kernel mode.

Given below is the figure representing the structure of a system call in an operating system −



Due to security reasons, user applications are not given access to hardware resources, when they need to do any I/O or require some memory, it requests OS one of all these. This request is made through system calls.

Differences

The major differences between Application Programming Interface (API) and system call are as follows −

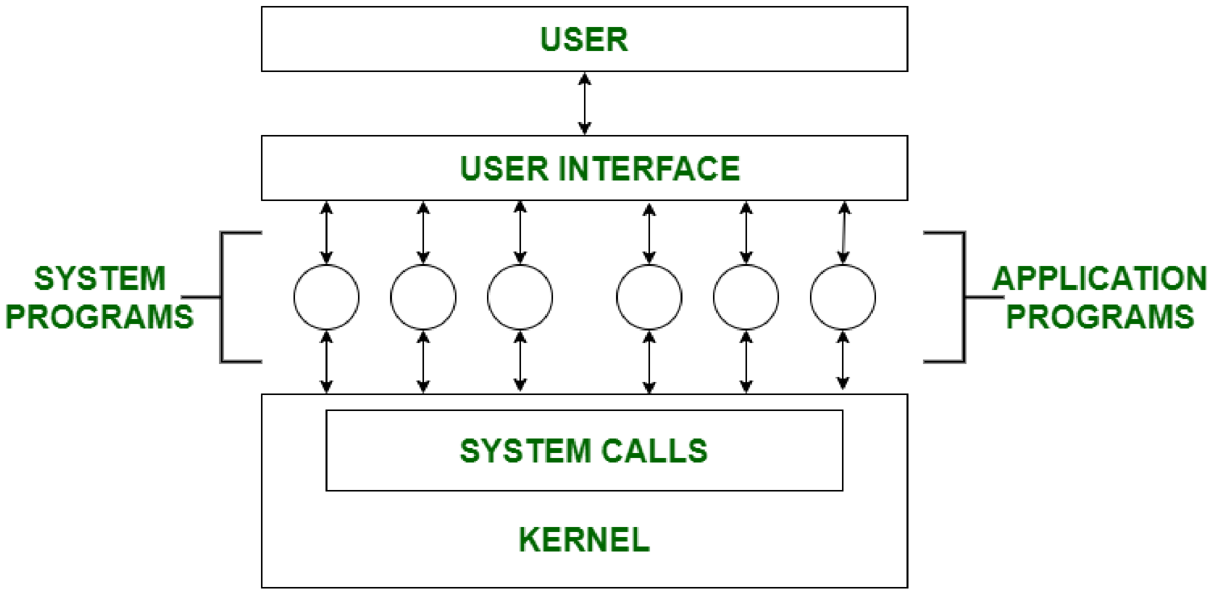
Application Program Interface	**System Call**
API is a set of protocols, routines, and functions which allows the exchange data among various applications and devices.	System call allows a program to request services from the kernel.
The protocols and functions in API that define the methods of communication among various components.	It is a method which allows a program to request services from the operating system's kernel.
It can be a web-based system, operating system, database or software library.	It provides an interface between user programs and operating systems.

System Programs in Operating System

System Programming can be defined as the act of building Systems Software using System Programming Languages. According to Computer Hierarchy, one which comes at last is Hardware. Then it is Operating System, System Programs, and finally Application Programs.

Program Development and Execution can be done conveniently in System Programs. Some of the System Programs are simply user interfaces, others are complex. It traditionally lies between the user interface and system calls.

In the context of an operating system, system programs are nothing but a special software which give us facility to manage and control the computer’s hardware and resources. As we have mentioned earlier these programs are more closely with the operating system so it executes the operation fast and helpful in performing essential opeartion which can’t be handled by application software.



Note: The user can only view up-to-the System Programs he can&#x2019t see System Calls.

Here are the examples of System Programs:

1. File Management &#x2013 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 intermediatry 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 &#x2013 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 &#x2013 For modifying the contents of files we use this. For Files stored on disks or other storage devices, we used different types of editors. For searching contents of files or perform transformations of files we use special commands.
6. Programming-Language support &#x2013 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 languages of importance are already provided.
7. Program Loading and Execution &#x2013 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 &#x2013al 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.