GURU99

System Call in OS (Operating System): What is, Types and Examples

By : Lawrence Williams

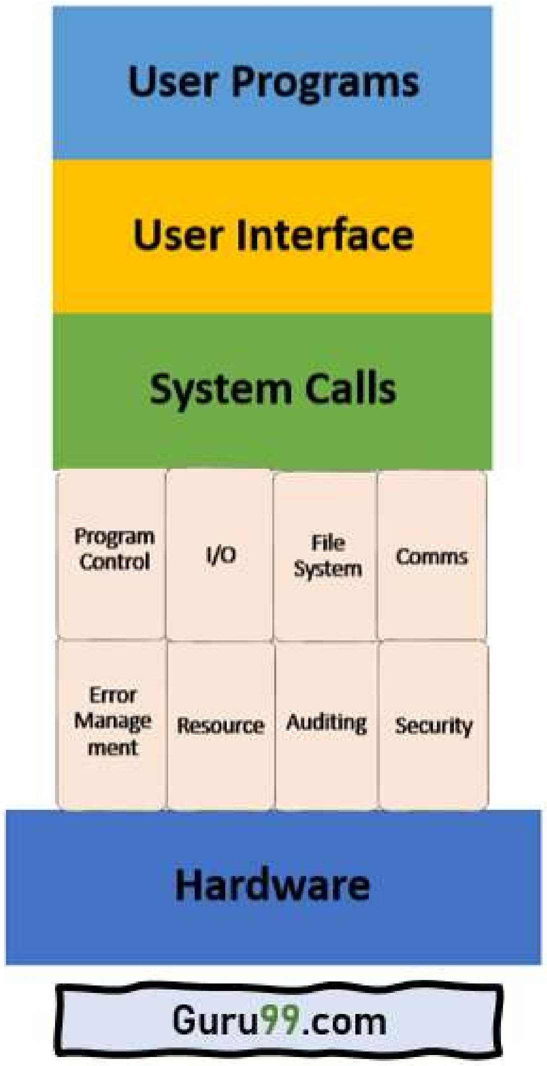
Updated July 24, 2023

What is System Call in Operating System?

A **system call** is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS.

System call offers the services of the operating system to the user programs via API (Application Programming Interface). System calls are the only entry points for the kernel system.

Table of Content:



System Calls in Operating System

Example of System Call

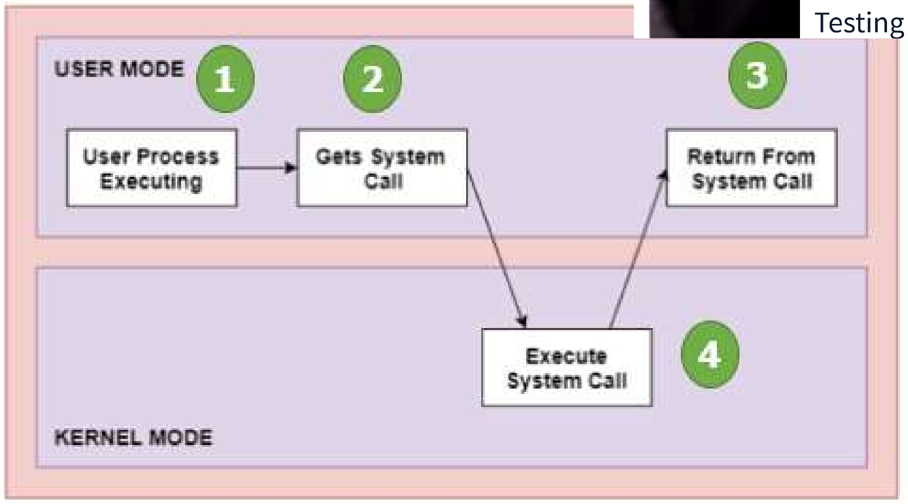
For example if we need to write a program code to read data from one file, copy that data into another file. The first information that the program requires is the name of the two files, the input and output files.

In an interactive system, this type of program execution requires some system calls by OS.

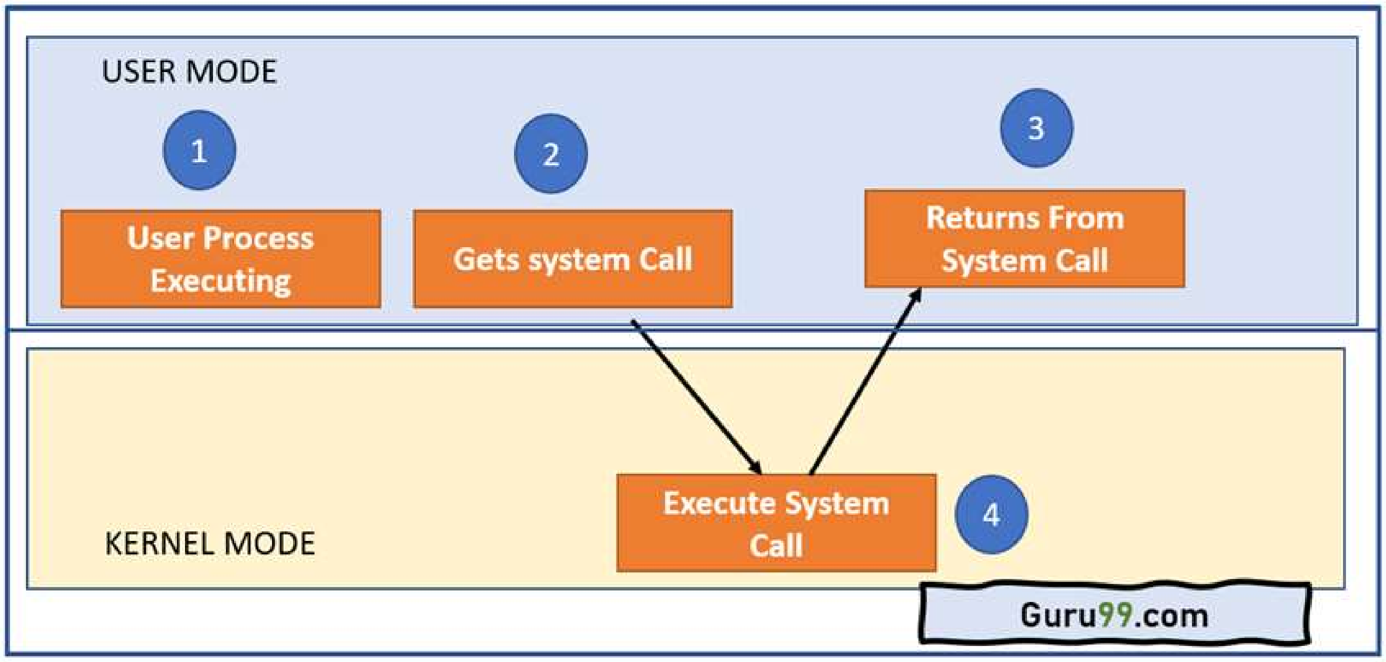
- First call is to write a prompting message on the screen
- Second, to read from the keyboard, the characters which define the two files.

How System Call Works?

Here are the steps for System Call in OS:



Guru99.com



Architecture of the System Call

As you can see in the above-given System Call example diagram.

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 Calls in OS?

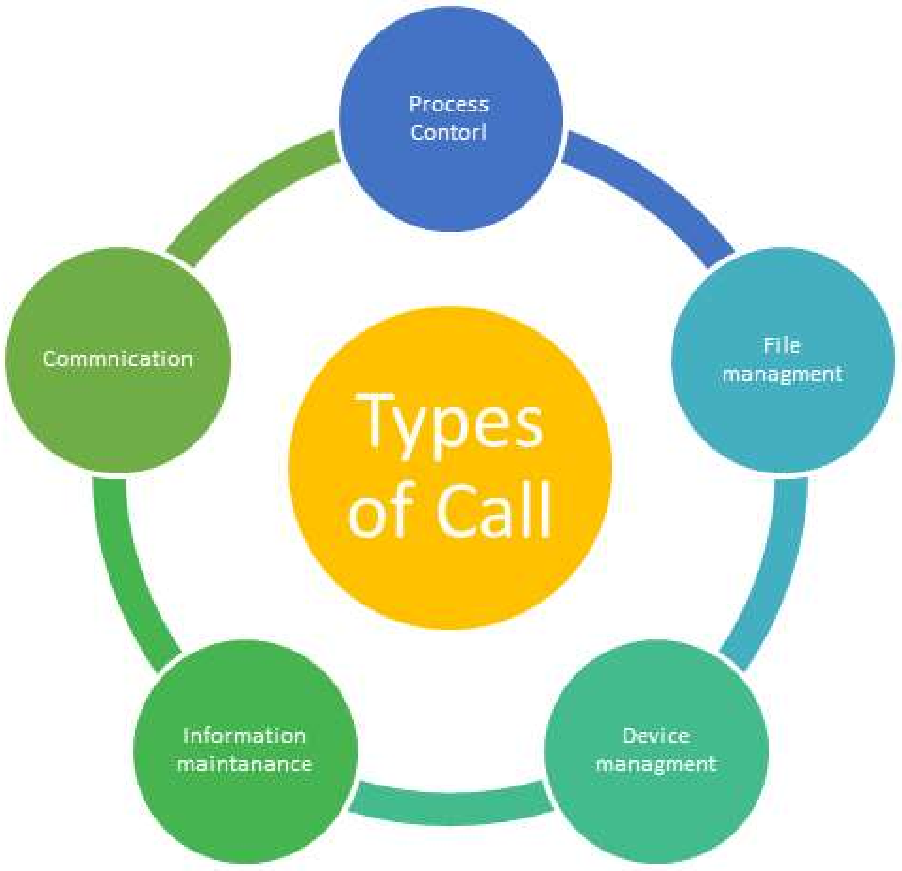
Following are situations which need system calls in OS:

- 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

Here are the five types of System Calls in OS:

- Process Control
- File Management
- Device Management
- Information Maintenance
- Communications



Types of System calls in OS

Process Control

This system calls perform the task of process creation, process termination, etc.

Functions:

- End and Abort
- Load and Execute
- Create Process and Terminate Process
- Wait and Signal Event
- Allocate and free memory

File Management

File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.

Functions:

- Create a file
- Delete file
- Open and close file
- Read, write, and reposition
- Get and set file attributes

Device Management

Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.

Functions:

- Request and release device
- Logically attach/ detach devices
- Get and Set device attributes

Information Maintenance

It handles information and its transfer between the OS and the user program.

Functions:

- Get or set time and date
- Get process and device attributes

Communication

These types of system calls are specially used for interprocess communications.

Functions:

- Create, delete communications connections
- Send, receive message
- Help OS to transfer status information
- Attach or detach remote devices

Rules for passing Parameters for System Call

Here are general common rules for passing parameters to the System Call:

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

Important System Calls Used in OS

wait()

In some systems, a process needs to wait for another process to complete its execution. This type of situation occurs when a parent process creates a child process, and the execution of the parent process remains suspended until its child process executes.

The suspension of the parent process automatically occurs with a wait() system call. When the child process ends execution, the control moves back to the parent process.

fork()

Processes use this system call to create processes that are a copy of themselves. With the help of this system Call parent process creates a child process, and the execution of the parent process will be suspended till the child process executes.

exec()

This system call runs when an executable file in the context of an already running process that replaces the older executable file. However, the original process identifier remains as a new process is not built, but stack, data, head, data, etc. are replaced by the new process.

kill()

The kill() system call is used by OS to send a termination signal to a process that urges the process to exit. However, a kill system call does not necessarily mean killing the process and can have various meanings.

exit()

The exit() system call is used to terminate program execution. Specially in the multi-threaded environment, this call defines that the thread execution is complete. The OS reclaims resources that were used by the process after the use of exit() system call.

Summary

Categories	Windows	Unix

Process control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

Device manipulation	SetConsoleMode()
	ReadConsole()
		Ioctl()
		read()

Categories	Windows	Unix
	WriteConsole()	write()

File manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		Open()
		Read()
		write()
		close!)

Information maintanence	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		Pipe()
		shm_open()
		mmap()

Protection	SetFileSecurity()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup ()
		Chmod()
		Umask()
		Chown()

You Might Like:

- Round Robin Scheduling Algorithm with Example
- Process Synchronization: Critical Section Problem in OS
- Process Scheduling in OS: Long, Medium, Short Term Scheduler
- Priority Scheduling Algorithm: Preemptive, Non-Preemptive EXAMPLE
- SSD vs HDD: What is the Difference Between SSD and HDD

⟵ Prev Report a Bug Next ⟶