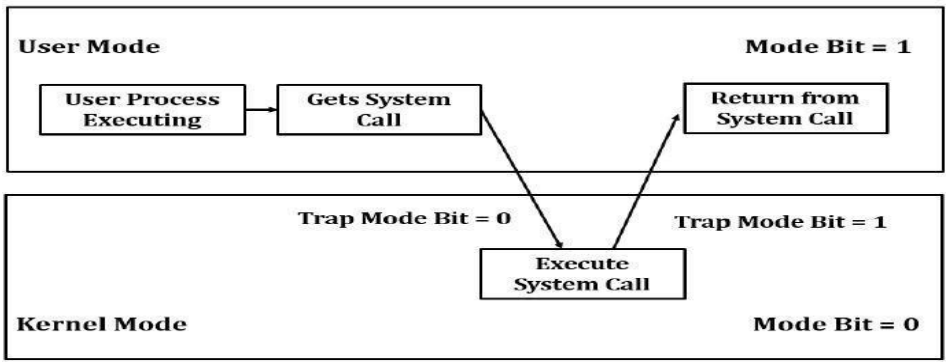
1.3 SYSTEM CALL

System calls are the fundamental interface through which user programs request services from the operating system kernel. They act as a bridge, allowing applications to access hardware resources and perform operations that are otherwise restricted to the kernel's privileged environment.

- System calls are a set of functions that a program can invoke to request specific services from the operating system.
- They provide a controlled and secure way for user programs to interact with the kernel, which manages the system's resources.
- System calls are essential for tasks like file I/O, process management, memory allocation, and device interaction.

Working of System Call



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.

Need of System Call

- 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.

Example of how system calls are used:

Writing a simple program to read data from one file and copy them to another file.

The first input that the program will need is the names of the two files: the input file and the output file. These names can be specified in many ways, depending on the operating-system design.

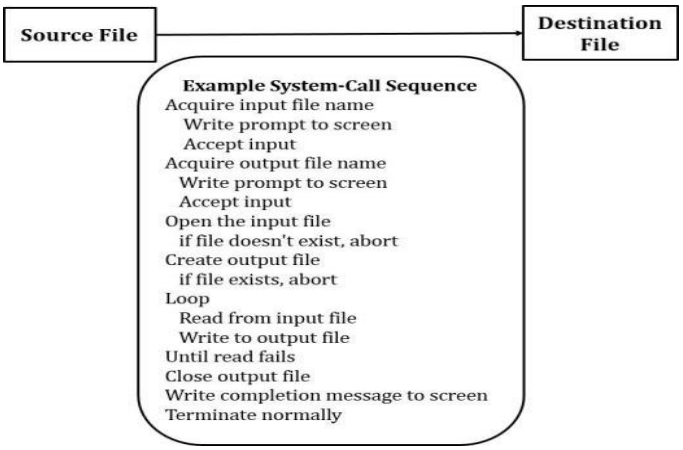
One approach is to pass the names of the two files as part of the command For

example, the UNIX cp command:

cp in.txt out.txt

This command copies the input file in.txt to the output file out.txt.

A second approach is for the program to ask the user for the names. In an interactive system, this approach will require a sequence of system calls, first to write a prompting message on the screen and then to read from the keyboard the characters that define the two files. Second approach is descripted in the below diagram.

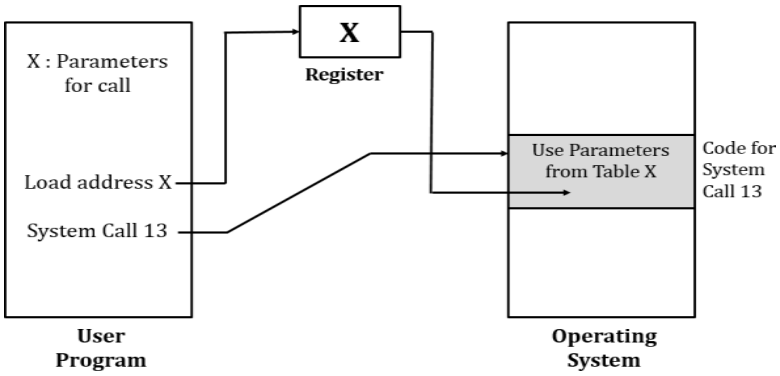


Passing of Parameters as a table

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. For five or fewer parameters, registers are used.
- More than five parameters, the block method is used. The parameters are generally stored in a block, or table, in memory, and the address of the block is passed as a parameter in a register



1.3.1Types of System calls

Here are the five types of System Calls in OS:

1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communications
6. Protection

1.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**

Example: fork( ), exit ( ), wait ( ), end() or abort(),exec().

The Linux System calls under this are **fork()** , **exit()** , **exec().**

a. **fork()**
   - This system call is used by the Processes to create a copy of the same process(themselves).
   - Parent process creates a child process, and the execution of the parent process will be suspended till the child process executes.

b. **exit()**
   - It is used to terminate execution of program.
   - Mainly used in the multithreaded environment
   - Which means execution of thread/process is completed.
   - The OS reclaims the allocated resources that were used by the process.

c. **exec()**
   - This system call replaces the current running process with a new process.
   - Here a new process is allowed to run/execute by replacing the current process.

2.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.**

Example: open( ), read ( ), write( ), close ( ), move( ), copy ( ).

open():

- It is the system call to open a file.
- This system call just opens the file, to perform operations such as read and write, we need to execute different system call to perform the operations.

read():

- This system call opens the file in reading mode
- We cannot edit the files with this system call.
- Multiple processes can execute the read() system call on the same file simultaneously.

write():

- This system call opens the file in writing mode
- We can edit the files with this system call.
- Multiple processes cannot execute the write() system call on the same file simultaneously.

close():

- This system call closes the opened file.

3.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.**
- **Example:** ioctl(), **read ( ), write( ), release( )**

ioctl():

- ioctl() is referred to as Input and Output Control.
- ioctl is a system call for device-specific input/output operations and other operations which cannot be expressed by regular system calls.

4.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 id**.

The System calls under this are **getpid(), alarm(), sleep().**

- **getpid():**
  - getpid stands for Get the Process ID.
  - The getpid() function shall return the process ID of the calling process.
- **alarm():**
  - This system call sets an alarm clock for the delivery of a signal that when it has to be reached.
  - It arranges for a signal to be delivered to the calling process.
- **sleep():**
  - This System call suspends the execution of the currently running process for some interval of time.
  - Meanwhile, during this interval, another process is given chance to execute.

Example: time( ) and date(),getpid(),alarm(),sleep()**.**

5. Interprocess Communication:

These types of system calls are specially used for interprocess communications through message passing and the shared-memory.

Functions:

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

Example: **pipe(), shmget(),mmap(),open_connection() and close_connection(), accept_connection()**

These types of system calls are specially used for inter-process communications.

Two models are used for inter-process communication

1. Message Passing(processes exchange messages with one another)
2. Shared memory(processes share memory region to communicate)

The system calls under this are **pipe() , shmget() ,mmap().**

- **pipe():**
    - The pipe() system call is used to communicate between different Linux processes.
    - It is mainly used for inter-process communication.
    - The pipe() system function is **used to open file descriptors**.
- **shmget():**
    - shmget stands for shared memory segment.
    - It is mainly used for Shared memory communication.
    - This system call is **used to access the shared memory**.
    - Used to access the messages in order to communicate with the process.
- **mmap():**
    - The mmap() system call is responsible for mapping the content of the file to the virtual memory space of the process.
    - System call that **maps files or devices into memory**, allowing processes to access them as if they were in the process's address space.

6. **Protection**

Protection provides a mechanism for controlling access to the resources provided by a computer system.

Functions:

- **Get or set permission to access resources such as files and disks**
- **Allow or deny user to access certain resources.**

Example: set_permission( ) and get_permission( )

allow_user( ) and deny_user( )

Some important System Calls Used in OS

wait()

- Used when a process needs to wait for another process to complete its execution.
- Used by the parent process which wait for its child to complete its execution.

- Until the child process complete its execution the parents process remains idle or suspended using the system call wait().
- After finishing the execution of the child process, the control is transfer back to the parent process.

kill():

- It is used by OS to terminate the process in case of any emergency or error.
- This system call does not necessarily mean killing the process and can have various meanings.

exit():

- It is used to terminate execution of program.
- Mainly used in the multithreaded environment
- Which means execution of thread/process is completed.
- The OS reclaims the allocated resources that were used by the process.

Examples of System Calls: (hints)

- **Process Management:** fork(), exit(), wait(),exec()
- **File Management:** open(), read(), write(), close()
- **Memory Management:** malloc(), free(), brk(), sbrk()
- **Device Management:** ioctl(), read(), write() (for devices)
- **Information Maintenance:** getpid(), alarm(), sleep()
- **Interproces Communication:** pipe(), shm_open(), mmap()
- **Protection:** set_permission( ) and get_permission( ) , allow_user( ) and deny_user( )