# 

# **Operating System Lab Assignment IV**

### **4ITRC2**

*Purvanshi patidar*

*BE II Year*

*Information technology – ‘A’*

# Aim

To study and learn about various system calls.

# To perform

Comprehensive study of different categories of Linux system calls, categorized as

1. Process Management System calls

fork(), exec(), wait(), exit().

1. File Management System calls

open(), read(), write(), close().

1. Device Management System calls

read(), write(), ioctl(), select ().

1. Network Management System calls

socket(), connect(), send(), recv().

1. System Information Management System calls

getpid(), getuid(), gethostname(), sysinfo().

# To Submit

Write up for the exhaustive study of the above mentioned system call categories with their examples.

## Introduction

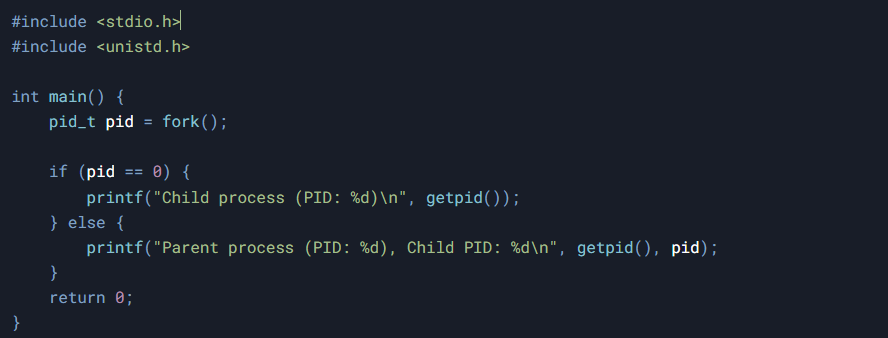
System calls are the primary interface between a process and the operating system. They allow user-level processes to request services from the kernel, such as file manipulation, process control, device management, and network communication. In this, we explore different categories of Linux system calls through examples and descriptions.

## Categories of Linux System Calls

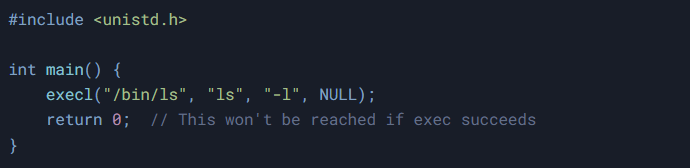
### Process Management System Calls

Process management system calls deal with the creation, execution, synchronization, and termination of processes. These are the building blocks of multitasking and concurrent execution in Linux.

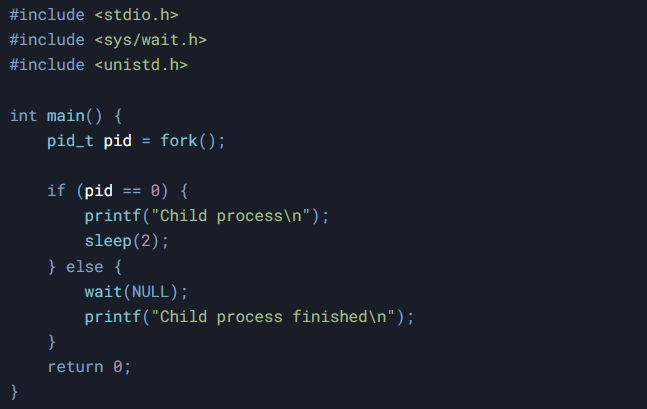
* **fork()**
  + - * Creates a new process by duplicating the current process. The new process (child) is an exact copy of the parent process.
      * Returns 0 in child process, positive PID of child in parent, or -1 on failure.
* **Example**:



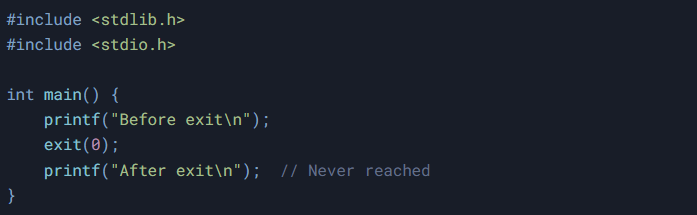
* **exec()**
* Replaces the current process image with a new process image.
* **Variants**: execl(), execp(), execv() etc.
* **Example**:

****

* **wait()**
* Suspends execution of the calling process until one of its children terminates.
* **Example:**

****

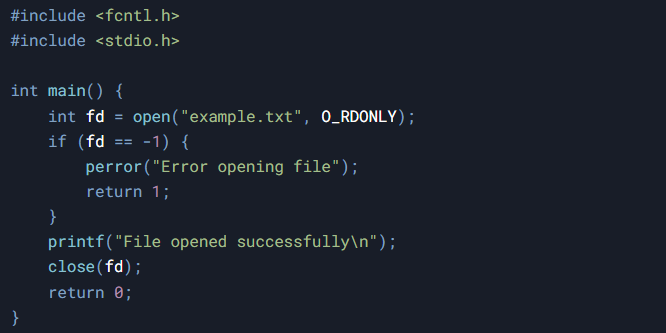
* **exit()**
* Terminates the calling process immediately.
* **Example:**

****

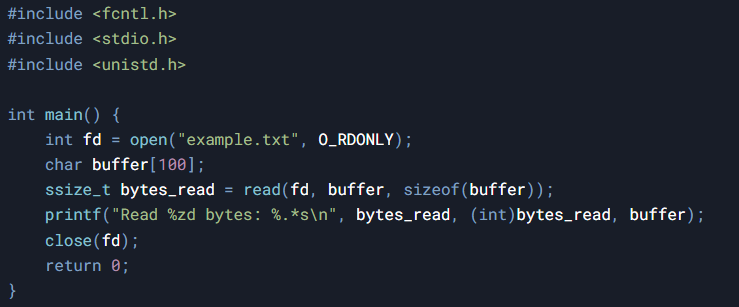
### File Management System Calls

File management system calls are used to perform file operations such as creating, opening, reading, writing, and closing files. They form the foundation of data storage and retrieval in Linux.

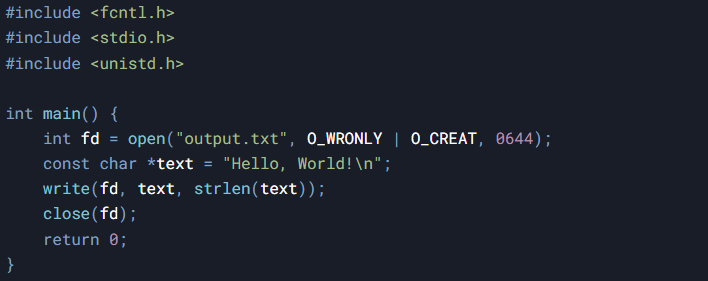
* **open()**
* Opens a file and returns a file descriptor.
* **Example:**



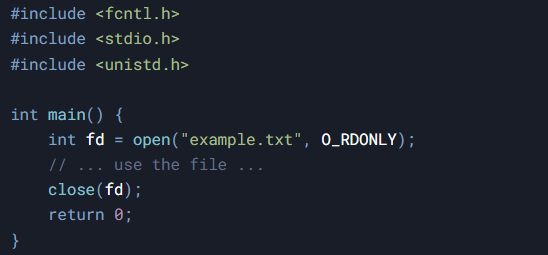
* **read()**
* Reads data from a file descriptor.
* **Example:**



* **write()**
* Writes data to a file descriptor.
* **Example:**



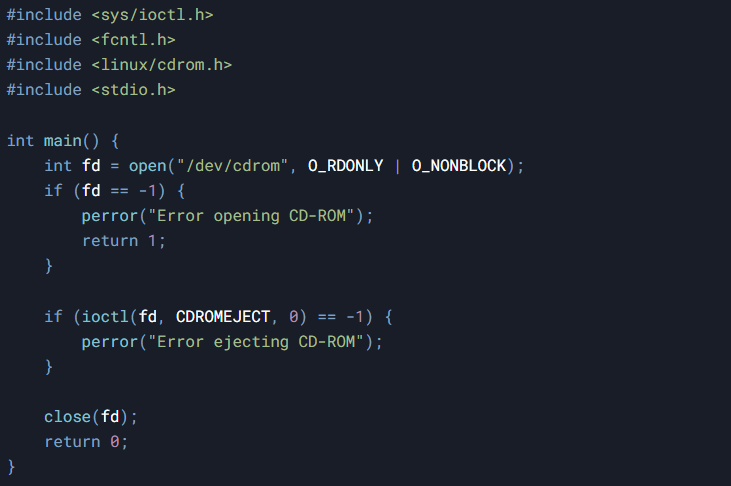
* **close()**
* Closes a file descriptor.
* **Example:**



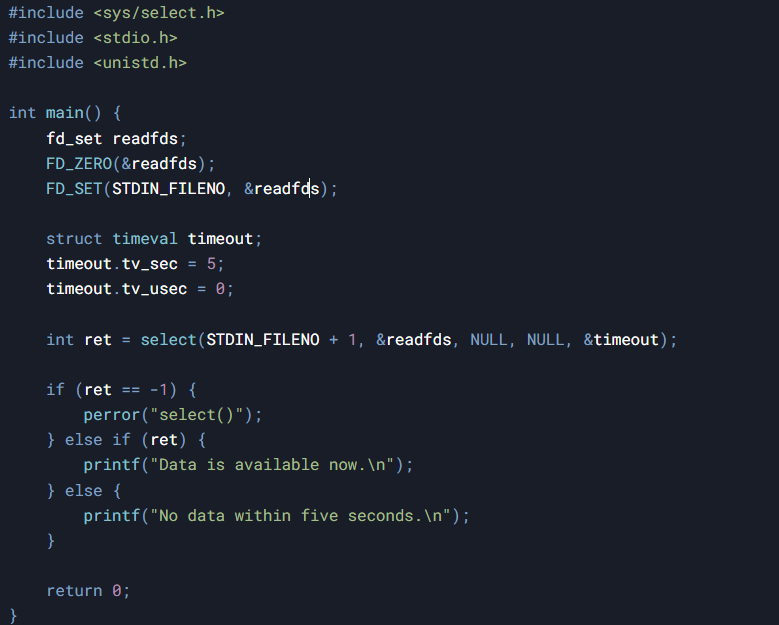
### Device Management System Calls

These system calls allow interaction with hardware devices such as keyboards, USBs, and disk drives. Devices are usually represented as files under the /dev directory in Linux.

* **read()/write()**
* Same as file read/write but used on device file descriptors.
* **ioctl()**
* Used for device-specific operations that don't fit the standard read/write model.
* **Example:**



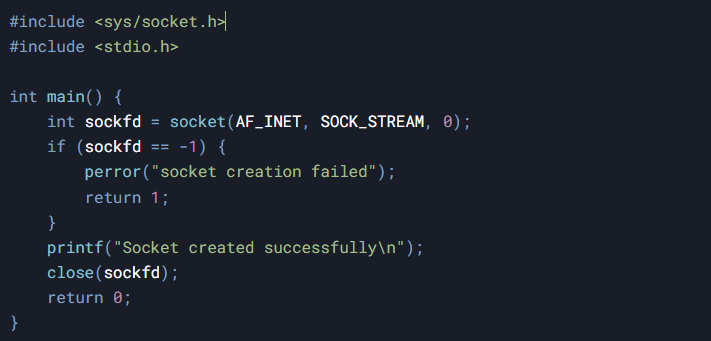
* **Select()**
* Monitors multiple file descriptors for readiness for I/O operations.
* **Example:**



### Network Management System Calls

Network system calls are used for data communication between systems over a network. They form the basis of networking applications such as web browsers, FTP clients, and chat applications.

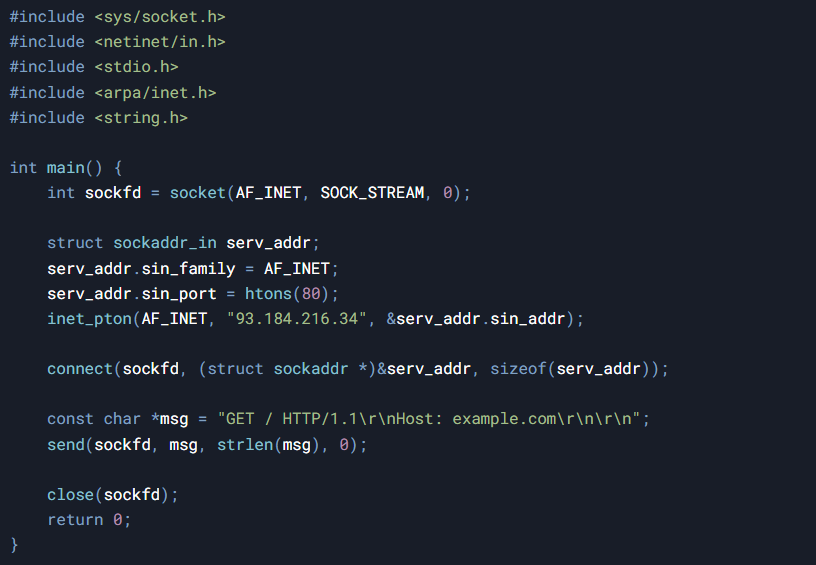
* **socket()**
* Creates an endpoint for communication.
* **Example:**



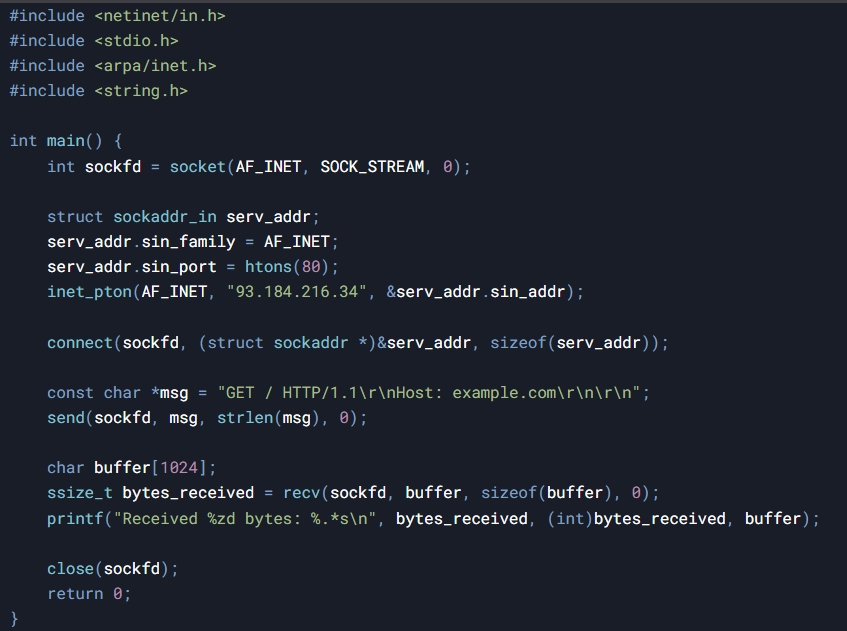
* **connect()**
* Initiates a connection on a socket.
* **Example:**

****

* **send()**
* Sends a message on a socket.
* **Example:**



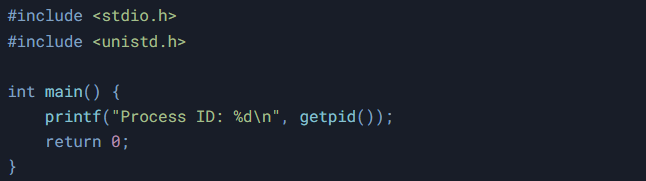
* **recv()**
* Receives a message from a socket.
* **Example:**



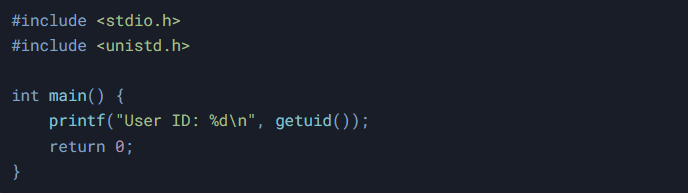
### System Information Management System Calls

These system calls are used to obtain information about the current process, user, or system state (e.g., process ID, system uptime, hostname, etc.).

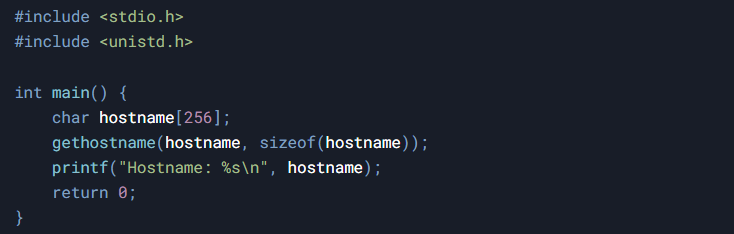
* **getpid()**
* Returns the process ID of the calling process.
* **Example:**



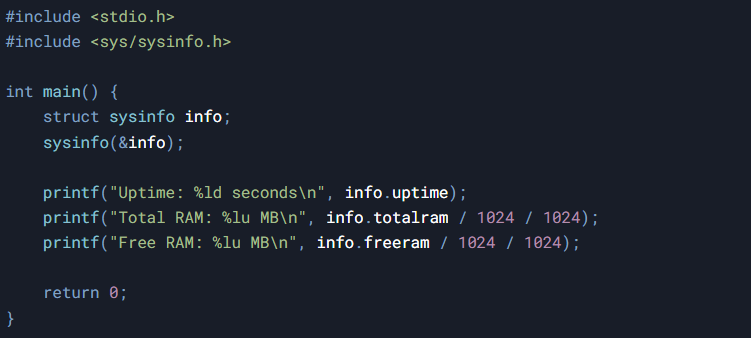
* **getuid()**
* Returns the real user ID of the calling process.
* **Example:**



* **gethostname()**
* Returns the standard host name for the current machine.
* **Example:**



* **sysinfo()**
* Returns system information such as uptime, load averages, and memory usage.
* **Example:**



## Conclusion

Linux system calls provide the essential interface between user space and kernel space, enabling applications to perform fundamental operations. This writeup has covered five major categories of system calls with practical examples for each. Understanding these system calls is crucial for system programming and developing efficient Linux applications. Each category serves specific purposes in process control, file operations, device management, networking, and system information retrieval, forming the foundation of Linux system programming.