Introduction to System Calls in Operating Systems

January 20, 2025

1 Introduction

In an operating system, system calls provide an interface between a process and the operating system kernel. These calls are used to perform tasks that are critical for the execution of processes, such as file handling, process control, memory management, and inter-process communication.

2 Types of System Calls

System calls can be classified into the following categories:

2.1 Process Control System Calls

These system calls are used to control processes, including creation, termination, and synchronization. Some common process control system calls are:

- `fork()` - Creates a new process by duplicating the calling process.
- `exec()` - Replaces the current process with a new process.
- `exit()` - Terminates the calling process.
- `wait()` - Suspends the execution of the calling process until a child process exits.

2.2 File Management System Calls

File management system calls allow a process to interact with the file system. These include:

- `open()` - Opens a file.
- `read()` - Reads data from a file.
- `write()` - Writes data to a file.

- `close()` - Closes a file.
- `unlink()` - Deletes a file.

2.3 Memory Management System Calls

Memory management system calls handle the allocation and deallocation of memory. These include:

- `malloc()` - Allocates a block of memory.
- `free()` - Frees the allocated memory.
- `brk()` - Sets the end of the data segment for a process.

2.4 Device Management System Calls

These system calls are used to interact with devices like printers, disks, and terminals. Common device management system calls are:

- `ioctl()` - Controls device-specific operations.
- `read()` - Reads data from a device.
- `write()` - Writes data to a device.

2.5 Information Management System Calls

Information management system calls provide system-related information. These include:

- `getpid()` - Returns the process ID of the calling process.
- `gettimeofday()` - Returns the current time.
- `uname()` - Returns system information.

2.6 Communication System Calls

Communication system calls enable communication between processes. Some examples are:

- `pipe()` - Creates a unidirectional data channel.
- `msgget()` - Creates a message queue.
- `semget()` - Creates a semaphore set.

3 Lab Exercises

Exercise 1: Process Creation and Termination

1. Write a C program that creates a child process using `fork()` and prints the process ID of both parent and child processes.
2. Implement the `exit()` system call to terminate the child process and display an appropriate message.

Exercise 2: File Handling

1. Write a program to create a file using `open()`, write some text into the file using `write()`, and then close the file using `close()`.
2. Implement file reading functionality using `read()` and display the content of the file.

Exercise 3: Memory Management

1. Implement a program that allocates a block of memory using `malloc()` and initializes the memory with values. Then, free the memory using `free()`.
2. Use `brk()` to increase or decrease the data segment size.

4 Conclusion

System calls are crucial for the execution of processes and facilitate interaction between user-level programs and the kernel. Understanding these calls is essential for developing low-level applications and interacting with the operating system.