Experiment 09

Aim: To study and implement the read(), write(), and fork() system calls in Unix/Linux operating systems.

System Calls in Unix/Linux

A system call is a direct interface between a user program and the operating system kernel. It allows programs to request services such as file I/O, process control, and inter-process communication.

In this experiment, we focus on the following three fundamental system calls:

1. **write() – For low-level output operations.**
2. **read() – For low-level input operations.**
3. **fork() – For process creation.**

write() System Call

The write() system call is used to **output data** to a file descriptor, such as the standard output (screen).

```
localhost:~# vi writesc.c
#include <stdio.h>
#include <unistd.h>

int main() {
    int count;
    count = write(1, "hello\n", 6);
    printf("Total bytes written: %d\n", count);
    return 0;
}
localhost:~# gcc writesc.c -o writesc
localhost:~# ./writesc
hello
Total bytes written: 6
```

read() System Call

The read() system call is used to **read data** from an input file descriptor, such as the keyboard (standard input).

```
localhost:~# vi readsc.c
#include <stdio.h>
#include <unistd.h>

int main() {
    int nread;
    char buff[20];

    // Read 10 bytes from standard input
    nread = read(0, buff, 10);

    // Write the read bytes to standard output
    write(1, buff, 10);

    return 0;
}
```

```
localhost:~# gcc readsc.c -o readsc
localhost:~# ./readsc
123456789
123456789
```

fork() System Call

The fork() system call is used to **create a new child process** from the parent process.

```
localhost:~# vi forksc.c
```

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t p;

    printf("Before fork\n");

    p = fork(); // Create child process

    if (p == 0) {
        printf("I am child having id %d\n", getpid());
        printf("My parent id is %d\n", getppid());
    }
    else {
        printf("My child id is %d\n", p);
        printf("I am parent having id %d\n", getpid());
    }

    printf("Common statement\n"); // Executes in both processes
    return 0;
}
```

```
localhost:~# gcc forksc.c -o forksc
localhost:~# ./forksc
Before fork
My child id is 89
I am parent having id 88
Common statement
I am child having id 89
My parent id is 88
Common statement
```