

**Ex. No.: 5****System Calls Programming****Date:18.02.25****Aim:**

To experiment system calls using fork(), execlp() and pid() functions.

**Algorithm:****1. Start**

o Include the required header files (stdio.h and stdlib.h).

**2. Variable Declaration**

o Declare an integer variable pid to hold the process ID.

**3. Create a Process**

o Call the fork() function to create a new process. Store the return value in the pid variable:

- ☐ If fork() returns:
- ☐ -1: Forking failed (child process not created).
- ☐ 0: Process is the child process.
- ☐ Positive integer: Process is the parent process.

**4. Print Statement Executed Twice**

o Print the statement:

scss

Copy code

THIS LINE EXECUTED TWICE

(This line is executed by both parent and child processes after fork()).

**5. Check for Process Creation Failure**

o If pid == -1:

☐ Print:

Copy code

CHILD PROCESS NOT CREATED

☐ Exit the program using exit(0).

## 6. Child Process Execution

o If pid == 0 (child process):

- ☐ Print:
- ☐ Process ID of the child process using getpid().
- ☐ Parent process ID of the child process using getppid().

## 7. Parent Process Execution

o If pid > 0 (parent process):

- ☐ Print:
- ☐ Process ID of the parent process using getpid().
- ☐ Parent's parent process ID using getppid().

## 8. Final Print Statement

o Print the statement:

objectivec

Copy code

IT CAN BE EXECUTED TWICE

(This line is executed by both parent and child processes).

## 9. End

### Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int pid = fork();

    if (pid < 0) {
```

```
    printf("CHILD PROCESS NOT CREATED\n");
    exit(0);
}
else if (pid == 0) {
    printf("\n--- Child Process ---\n");
    printf("Child PID: %d\n", getpid());
    printf("Parent PID: %d\n", getppid());

    printf("\nExecuting 'ls -l' command in Child Process:\n");
    execlp("ls", "ls", "-l", NULL);

    printf("execlp() failed!\n");
    exit(1);
}
else {
    wait(NULL);
    printf("\n--- Parent Process ---\n");
    printf("Parent PID: %d\n", getpid());
    printf("Parent's Parent PID: %d\n", getppid());

    printf("\nBoth processes executed successfully.\n");
}
return 0;
}
```

**Output:**

```
File Actions Edit View Help
(kali@kali)-[~]
$ gcc syscall.c -o system_calls

(kali@kali)-[~]
$ ./system_calls

--- Child Process ---
Child PID: 6046
Parent PID: 6045

Executing 'ls -l' command in Child Process:
total 112
-rw-rw-r-- 1 kali kali 45 Jan 24 09:40 college
-rw-rw-r-- 1 kali kali 121 Jan 24 11:43 cse
-rw-rw-r-- 1 kali kali 4 Jan 24 11:33 cse.txt
drwxr-xr-x 3 kali kali 4096 Jan 21 06:01 Desktop
drwxr-xr-x 2 kali kali 4096 Jan 21 06:00 Documents
drwxr-xr-x 3 kali kali 4096 Feb 16 07:50 Downloads
-rw-rw-r-- 1 kali kali 7 Jan 24 11:38 eee
-rw-rw-r-- 1 kali kali 568 Feb 4 11:17 emp.awk
-rw-rw-r-- 1 kali kali 55 Feb 4 11:09 emp.dat
-rw-rw-r-- 1 kali kali 131 Jan 28 11:24 fibo.sh
-rw-rw-r-- 1 kali kali 0 Jan 28 11:06 leap
-rw-rw-r-- 1 kali kali 278 Jan 28 11:06 leap.sh
-rw-rw-r-- 1 kali kali 448 Feb 4 11:16 marks.awk
-rw-rw-r-- 1 kali kali 88 Feb 4 11:16 marks.dat
drwxr-xr-x 2 kali kali 4096 Jan 21 06:00 Music
drwxr-xr-x 2 kali kali 4096 Feb 16 07:11 Pictures
drwxr-xr-x 2 kali kali 4096 Jan 21 06:00 Public
drwxrwxr-x 2 kali kali 4096 Jan 24 11:31 receee
-rw-rw-r-- 1 kali kali 170 Jan 28 11:11 reverse.sh
-rw-rw-r-- 1 kali kali 4 Jan 24 09:06 rit
-rw-rw-r-- 1 kali kali 44 Jan 24 09:29 sample
-rw-rw-r-- 1 kali kali 27 Jan 24 09:46 student
-rw-rw-r-- 1 kali kali 813 Feb 18 11:47 syscall.c
-rwxrwxr-x 1 kali kali 16304 Feb 18 11:50 system_calls
drwxr-xr-x 2 kali kali 4096 Jan 21 06:00 Templates
drwxr-xr-x 2 kali kali 4096 Jan 21 06:00 Videos

--- Parent Process ---
Parent PID: 6045
Parent's Parent PID: 1414

Both processes executed successfully.
```

## Result:

Thus, the program using system calls was successfully executed.