Ex. No.: 5 **231901039**

Date: 15/02/2025

System Calls Programming

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

Algorithm:

1. **Start** ○ Include the required header files (stdio.h and stdlib.h).
2. **Variable Declaration**
   ○ Declare an integer variable pid to hold the process ID.
3. **Create a Process**
   ○ 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** ○ 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**
   ○ If pid == -1:

- Print:

  Copy code
  CHILD PROCESS NOT CREATED

- Exit the program using exit(0).

6. **Child Process Execution** ○ 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** ○ 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** ○ 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>

int main() {
    int pid;

    // Step 3: Create a process
    pid = fork();

    // Step 4: Print statement executed by
both parent and child
    printf("THIS LINE EXECUTED TWICE\n");

    // Step 5: Check if fork failed
    if (pid == -1) {
        printf("CHILD PROCESS NOT
CREATED\n");
        exit(0);
    }

    // Step 6: Child process
    else if (pid == 0) {
        printf("CHILD PROCESS\n");
        printf("Process ID: %d\n", getpid());
```

```
        printf("Parent Process ID: %d\n",
getppid());
        // Optional: replace the child
process with another program using execlp
        // Example: run "ls -l"
        printf("Executing 'ls -l' using
execlp:\n");
        execlp("ls", "ls", "-l", NULL);
        // If execlp fails
        printf("execlp failed\n");
        exit(1);
    }
    // Step 7: Parent process
    else {
        printf("PARENT PROCESS\n");
        printf("Process ID: %d\n", getpid());
        printf("Parent's Parent Process ID:
%d\n", getppid());
    }
    // Step 8: Final print statement
    printf("IT CAN BE EXECUTED TWICE\n");

    return 0;
}
```

Output:

THIS LINE EXECUTED TWICE

PARENT PROCESS

Process ID: 7082

Parent's Parent Process ID: 7073

IT CAN BE EXECUTED TWICE

THIS LINE EXECUTED TWICE

CHILD PROCESS

Process ID: 7083

Parent Process ID: 7082

Executing 'ls -l' using execlp:

Result:

system calls using fork(), execlp() and pid() functions where experimented.