# Ex. No.: 5 Roll no:231901002

**Date: *18.02.2025***

# System Calls Programming

**Aim:**

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

# Algorithm:

1. **Start**
   * Include the required header files: stdio.h, stdlib.h, and unistd.h.

# Variable Declaration

* + Declare an integer variable pid to hold the process ID.

# Create a Process

* + Call the fork() function and store the return value in pid.
    - If fork() returns:
    - -1: Forking failed.
    - 0: This is the child process.
    - Positive value: This is the parent process.

1. **Print Statement Executed Twice** o Print: o THIS LINE EXECUTED TWICE
2. **Check for Process Creation Failure** o If pid == -1, print:
   * CHILD PROCESS NOT CREATED
     + Exit the program.

# Child Process Execution

* + If pid == 0, print:
    - The process ID of the child using getpid().
    - The parent process ID of the child using getppid().

# Parent Process Execution

* + If pid > 0, print:
    - The process ID of the parent using getpid().
    - The parent’s parent process ID using getppid().

1. **Final Print Statement** o Print: o IT CAN BE EXECUTED TWICE

# End

**Program Code:**

// filename: systemcall.c #include <stdio.h> #include <stdlib.h> #include <unistd.h>

int main() { int pid;

pid = fork(); // Create new process

printf("THIS LINE EXECUTED TWICE\n");

if (pid == -1) {

printf("CHILD PROCESS NOT CREATED\n");

exit(0);

}

if (pid == 0) {

printf("Child Process ID: %d\n", getpid()); printf("Parent Process ID of Child: %d\n", getppid());

} else {

printf("Parent Process ID: %d\n", getpid()); printf("Parent's Parent Process ID: %d\n", getppid());

}

printf("IT CAN BE EXECUTED TWICE\n");

return 0;

}

# Sample Output:

THIS LINE EXECUTED TWICE

Parent Process ID: 12345 Parent's Parent Process ID: 1000 IT CAN BE EXECUTED TWICE THIS LINE EXECUTED TWICE

Child Process ID: 12346

Parent Process ID of Child: 12345 IT CAN BE EXECUTED TWICE

# Result:

The program was successfully executed. It demonstrated the use of system calls fork(), getpid(), and getppid() to manage parent and child processes.