Ex. No. 5

SYSTEM CALLS PROGRAMMING

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

33

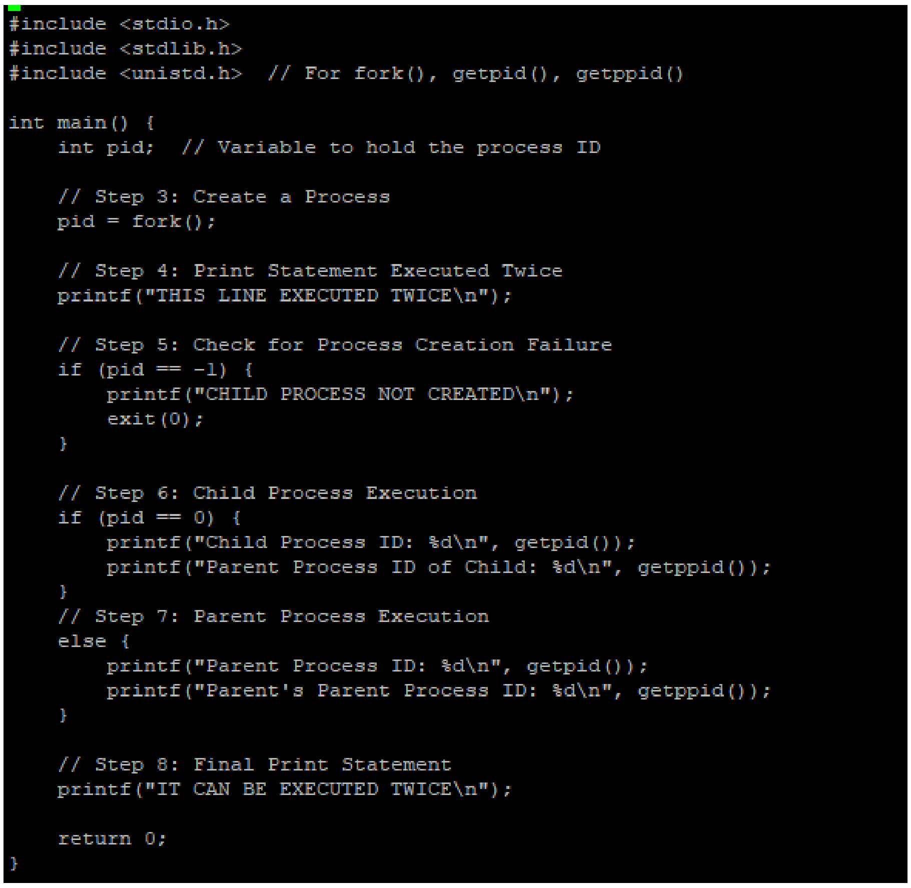
Copy code

IT CAN BE EXECUTED TWICE

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

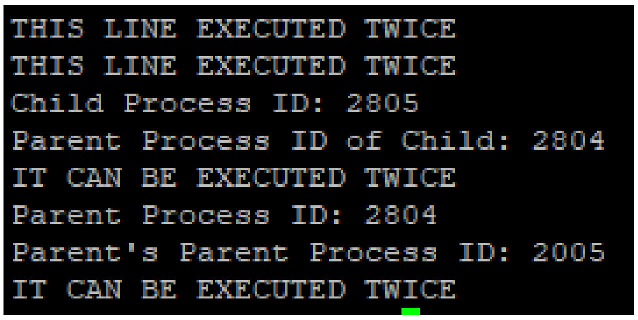
9. End

Program:



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>  // For fork(), getpid(), getppid()

int main() {
    int pid;  // Variable to hold the process ID

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

    // Step 4: Print Statement Executed Twice
    printf("THIS LINE EXECUTED TWICE\n");

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

    // Step 6: Child Process Execution
    if (pid == 0) {
        printf("Child Process ID: %d\n", getpid());
        printf("Parent Process ID of Child: %d\n", getppid());
    }
    // Step 7: Parent Process Execution
    else {
        printf("Parent 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
THIS LINE EXECUTED TWICE
Child Process ID: 2805
Parent Process ID of Child: 2804
IT CAN BE EXECUTED TWICE
Parent Process ID: 2804
Parent's Parent Process ID: 2005
IT CAN BE EXECUTED TWICE
```