Operating System – CS23431

Ex 5	**System Calls Programming**
Name: B M Madhumitha	
Reg No: 230701168	

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

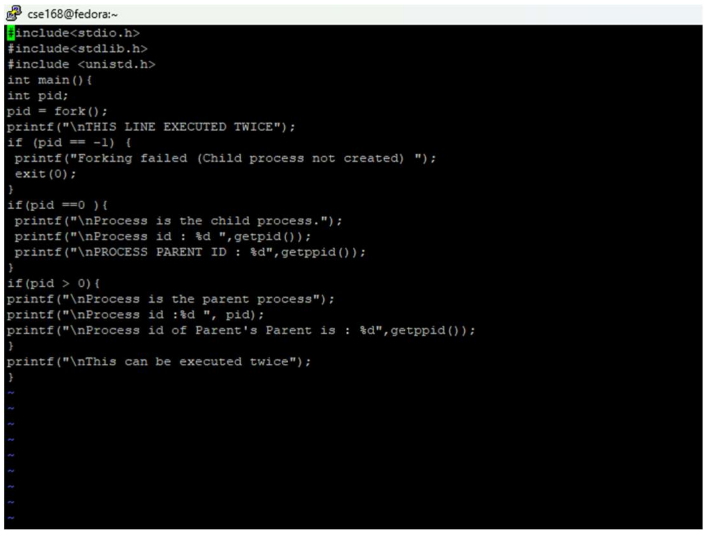
(This line is executed by both parent and child processes).

9. **End**

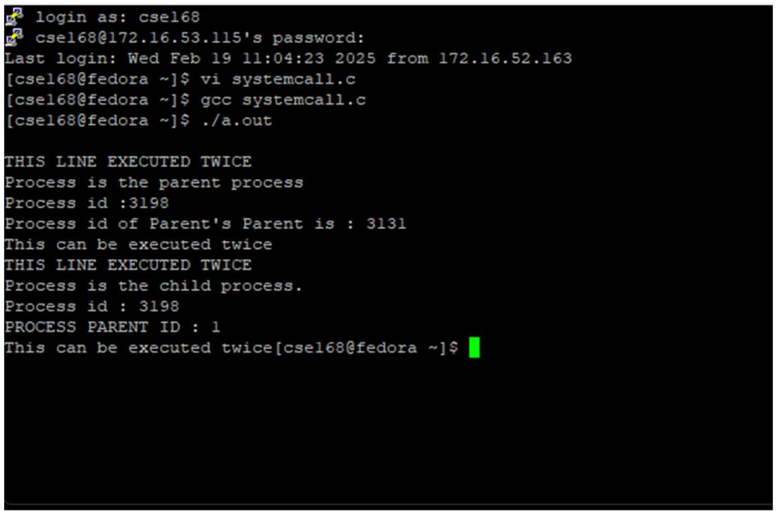
Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
   int pid;
   pid = fork();
   printf("THIS LINE EXECUTED TWICE\n");

   if (pid == -1) {
      printf("\nForking failed (child process not created)");
   }

   if (pid == 0) {
      printf("\nProcess is the child process");
      printf("\nProcess ID: %d", getpid());
      printf("\nProcess PARENT ID: %d", getppid());
   }

   if (pid > 0) {
      printf("\nProcess is the parent process");
      printf("\nProcess ID: %d", getpid());
      printf("\nProcess ID of Parent's parent is: %d", getppid());
   }

   printf("\nThis can be executed twice\n");

   return 0;
}
```

Console:



```
cse168@fedora:~
#include<stdio.h>
#include<stdlib.h>
#include <unistd.h>
int main(){
int pid;
pid = fork();
printf("\nTHIS LINE EXECUTED TWICE");
if (pid == -1) {
 printf("Forking failed (Child process not created) ");
 exit(0);
}
if(pid ==0 ){
 printf("\nProcess is the child process.");
 printf("\nProcess id : %d ",getpid());
 printf("\nPROCESS PARENT ID : %d",getppid());
}
if(pid > 0){
printf("\nProcess is the parent process");
printf("\nProcess id :%d ", pid);
printf("\nProcess id of Parent's Parent is : %d",getppid());
}
printf("\nThis can be executed twice");
}
~
~
~
~
~
~
~
~
~
```



```
login as: cse168
cse168@172.16.53.115's password:
Last login: Wed Feb 19 11:04:23 2025 from 172.16.52.163
[cse168@fedora ~]$ vi systemcall.c
[cse168@fedora ~]$ gcc systemcall.c
[cse168@fedora ~]$ ./a.out

THIS LINE EXECUTED TWICE
Process is the parent process
Process id :3198
Process id of Parent's Parent is : 3131
This can be executed twice
THIS LINE EXECUTED TWICE
Process is the child process.
Process id : 3198
PROCESS PARENT ID : 1
This can be executed twice[cse168@fedora ~]$
```

Result:

Thus, the program was successfully executed.