**EX. NO:5**

**DATE: 12.02.2025**

**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.

1. 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()).

1. Check for Process Creation Failure o If pid == -1:

Print:

Copy code

CHILD PROCESS NOT CREATED

Exit the program using exit(0).

1. 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().

1. 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().

1. Final Print Statement o Print the statement:

objectivec

33

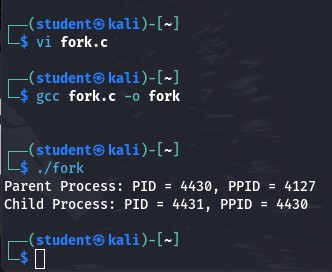
Copy code

IT CAN BE EXECUTED TWICE

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

1. End

**Program: fork()**



#include <stdio.h>

#include <stdlib.h> #include <unistd.h> int main() {

int pid = fork(); if (pid == -1) { printf("CHILD PROCESS NOT CREATED\n"); exit(0);

}

if (pid == 0) {

printf("Child Process: PID = %d, PPID = %d\n", getpid(), getppid());

} else {

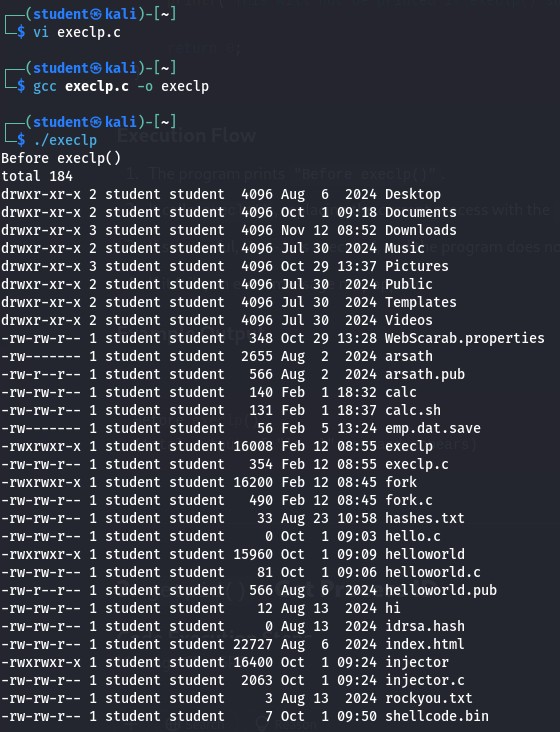
printf("Parent Process: PID = %d, PPID = %d\n", getpid(), getppid());

}

return 0;

}

**execlp()**



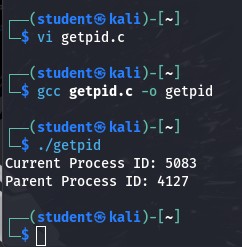
#include <stdio.h>

#include <stdlib.h> #include <unistd.h> int main() { printf("Before execlp()\n"); // Step 1: Print initial message execlp("ls", "ls", "-l", NULL); // Step 2: Execute "ls -l" command printf("This will not be printed if execlp() succeeds.\n"); // Step 3: This line is never executed if execlp() works

return 0;

}

**getpid()**



#include <stdio.h> #include <unistd.h> int main() { printf("Current Process ID: %d\n", getpid()); // Step 1: Get current PID printf("Parent Process ID: %d\n", getppid()); // Step 2: Get parent process ID return 0;

}

**opendir() and readdir()**



#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

int main() { struct dirent \*de;

DIR \*dr = opendir("."); // Step 1: Open current directory if (dr == NULL) { // Step 2: Check for failure printf("Could not open current directory\n"); return 0;

}

while ((de = readdir(dr)) != NULL) // Step 3: Read directory entries printf("%s\n", de->d\_name); closedir(dr); // Step 4: Close directory

return 0;

}

**Result:**

Hence, system calls are executed successfully.