Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

AIM: Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

ALGORITHM:

1. Include necessary headers: Include the necessary header files like and for using system calls.

2. Declare variables: Declare variables to hold the process ID (pid\_t pid) and the parent process ID (pid\_t ppid).

3. Get the current process ID: Use the getpid() system call to retrieve the process ID of the current process.

4. Get the parent process ID: Use the getppid() system call to retrieve the parent process ID.

5. Display the process IDs: Print the process IDs to the console.

6 Create a new process: Use the fork() system call to create a new process. Check the return value of fork() to determine whether the code is running in the parent or child process.

7. Display process IDs in child and parent processes: Depending on whether the process is the parent or the child, display the respective process IDs.

PROGRAM:

#include<stdio.h>

#include<unistd.h>

int

main()

{

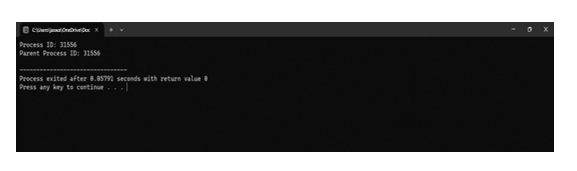
}

printf("Process ID: %d\n", getpid() );

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

return 0;

OUTPUT:



25) Construct a C program to implement the I/O system calls of UNIX (fcntl, seek,

stat, opendir, readdir)

PROGRAM:

#include<stdio.h>

#include<fcntl.h>

#include<errno.h>

extern int errno; int

main()

{

}

int fd = open("foo.txt", O\_RDONLY | O\_CREAT);

printf("fd = %d\n", fd);

if (fd ==-1)

{

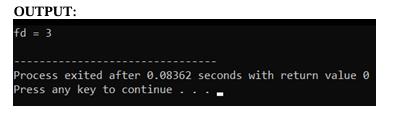
printf("Error Number % d\n", errno);

perror("Program");

}

return 0;

OUTPUT:



26) Construct a C program to implement the file management operations.

PROGRAM:

#include

<stdio.h>

#include <stdlib.h> int

main() {

FILE \*file;

file = fopen("example.txt", "w"); if

(file == NULL) {

printf("Error opening the file for writing.\n"); return

1;

}

fprintf(file, "Hello, World!\n");

fprintf(file, "This is a C file management example.\n");

fclose(file);

file = fopen("example.txt", "r"); if

(file == NULL) {

printf("Error opening the file for reading.\n"); return

1;

}

char buffer[100];

while (fgets(buffer, sizeof(buffer), file) != NULL) {

printf("%s", buffer);

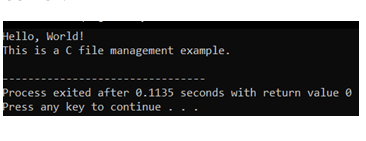
}

fclose(file);

return 0;

}

OUTPUT:



27) Develop a C program for simulating the function of ls UNIX Command.

PROGRAM:

#include<stdio.h>

#include<dirent.h>

int main()

{

char fn[10], pat[10], temp[200]; FILE

\*fp;

printf("\n Enter file name : ");

scanf("%s", fn);

printf("Enter the pattern: ");

scanf("%s", pat);

fp = fopen(fn, "r");

while (!feof(fp)) { fgets(temp,

sizeof(fp), fp); if (strcmp(temp,

pat))

printf("%s", temp);

}

fclose(fp);

return 1;

}

