**Lab 3: System Calls for Basic Process Management**

**Objectives:**

The objectives of the C program are to demonstrate key concepts in process management. It aims to show how the fork () function creates a child process from a parent process, enabling concurrent execution. The program also illustrates the use of the execve() function to replace the current process with a new program, initializing a fresh execution context. By printing process IDs, it provides insights into parent-child process relationships. Additionally, the program prints a constant value, showcasing basic output operations in C. Overall, the objective is to provide a clear understanding of process creation, execution, and replacement within a C program.

1. **Printing a constant value:**

ls

mkdir process

cd process

nano sulav.c

Write following text in nano window.

#include <stdio.h>

#include <time.h>

#include <unistd.h>

int main(){

float pi;c

pi=3.1415;

printf("\n The value of pi is %.4f\n",pi);

}

return 0;

Press ctrl+x to save file.

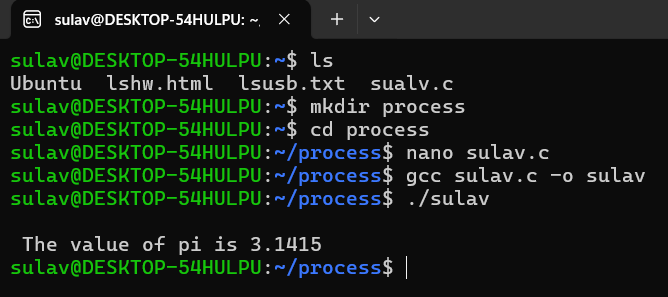
gcc sulav.c -o sulav

./sulav

**Interpretation:**

A simple C program for printing a constant value.

**Output:**

****

1. **Creating child process from a parent process using function fork ():**

nano fork.c

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

// make two process which run same

// program after this instruction

if(fork()==0){

printf("I am a child.\n");

}

else {

printf("I am the parent.\n");

}

printf("Hello!\n");

return 0;

}

Press ctrl+x to save file.

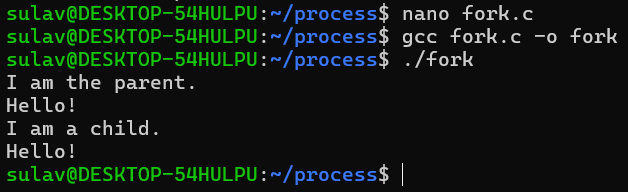
gcc -o fork fork.c

./fork

**Interpretation:**

A program for creating child process from a parent process using function fork ().

**Output:**

****

1. **Demonstration of fork () function which is used to create a child process from parent process:**

nano pid.c

#include <stdio.h>

#include <unistd.h> //contains fork prototype

int main()

{

int pid,pid2;

pid = fork();

if (pid == 0)

{

printf("\nChild Process id : %d",getpid());

printf("\nChild Process with parent id : %d \n",getppid());

}

pid2=fork();

if(pid2==0){

printf("\nChild Process id : %d",getpid());

printf("\nChild Process with parent id : %d \n",getppid());

}

return 0;

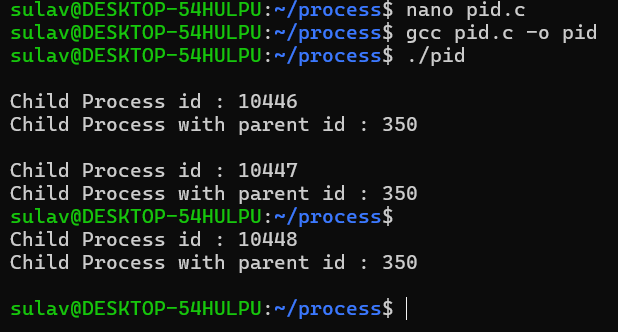
}

**Interpretation:**

A simple program for demontration of fork() function which is used to create

a child process from parent process.

**Output:**

****



1. **for demonstrating execve():**

nano execute.c

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main() {

char \*temp[] = {NULL,"hello","world",NULL};

temp[0]="basic";

if(fork()==0){

printf("I am a child process.\n");

printf("My id is: %d\n",getpid());

printf("My parent is: %d\n",getppid());

printf("I am going to execute a c program\n");

execve("basic",temp,NULL);

}

else{

printf("The id of current process is %d\n",getpid());

}

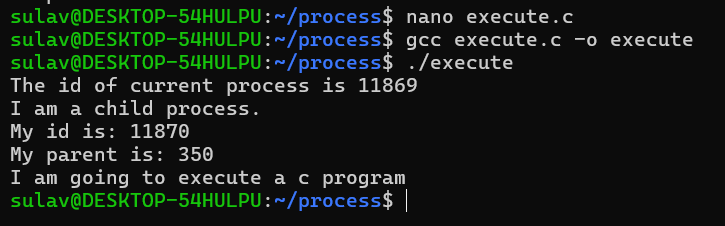
return 0;

}

**Interpretation:**

A program for demonstrating execve() function which causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.

**Output:**

****

**Conclusion:**

In conclusion, the C program effectively demonstrates fundamental concepts of process management through the use of fork () and execve(). It highlights the ability to create a child process from a parent process, allowing parallel execution and showcasing the relationship between parent and child processes via their IDs. The program also demonstrates how a process can be completely replaced with a new program using execve(), emphasizing the initialization of new execution contexts. By including basic output operations, the program provides a comprehensive example of process creation, replacement, and communication. The use of fork () enhances multitasking capabilities, while execve() underscores the importance of careful process management due to the complete replacement of the original process. Overall, the program serves as a practical guide to understanding process management in C, illustrating both its advantages and limitations.