**Task No. 1:** Write what you have learned in few lines on each of the three programs that were using the fork() system call.

**Solution:**

**PROGRAM 1**

#include<stdio.h>

#include <sys/types.h>

#include<unistd.h>

int main() {

printf(“before forking \n”);

fork();

printf(“after forking \n”);

return 0;

}

**PROGRAM 2**

#include<stdio.h>

#include <sys/types.h>

#include<unistd.h>

int main ()

{

fork();

fork();

printf("hello world \n”);

return 0;

}

**PROGRAM 3**

#include<stdio.h>

#include <sys/types.h>

#include<unistd.h>

int i = 5;

void parent\_process();

void child\_process();

int main() {

pid\_t pid;

pid = fork();

if(pid == 0) {

i += 10;

child\_process();

}

else {

parent\_process();

}

return 0;

}

void parent\_process() {

printf("I am a parent process and my value of 'i' is %d \n",i);

}

void child\_process() {

printf("I am a child process and my value of 'i' is %d \n",i);

}

**Explanation: -**

**Program 1:** This program demonstrates the basic use of the fork() system call. The fork() function creates a new process by duplicating the calling process. After the fork() function is called, both the parent and child processes will continue executing from the point where fork() was called. In this program, the printf() function is called before and after the fork() function to show that both processes continue executing after the fork() function is called.

**Program 2:** This program demonstrates how the parent and child processes can have different behavior after the fork() function is called. In this program, a global variable i is declared and initialized to 5. After the fork() function is called, the child process increments the value of i by 10 and calls a function named child\_process(), while the parent process calls a function named parent\_process(). These two functions print the value of i along with a message indicating whether they are being executed by the parent or child process.

**Program 3:** This program demonstrates how multiple child processes can be created by calling the fork() function multiple times. In this program, the fork() function is called twice in succession. As a result, four processes are created in total: one parent process and three child processes. Each of these processes will print the message “hello world” to the standard output.

**Task No. 2:** Write a C program that uses fork() system call to print a single line eight times without using for loop and repeated printf command.

**Solution:**

#include<stdio.h>

#include <sys/types.h>

#include<unistd.h>

void main (){

printf("---------------Forking Example----------------\n");

fork();

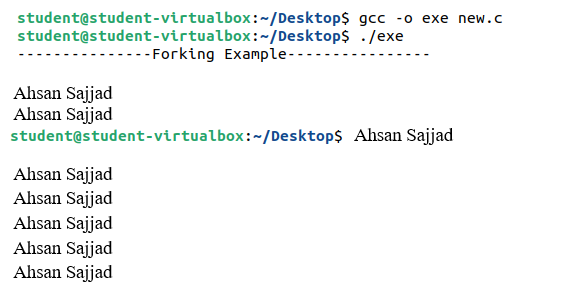
fork();

fork();

printf("Ahsan Sajjad\n");

}

**Output:**



**Task No. 3:** Code the C program given below and explain what it does along with providing a snapshot of the output. Investigate and write about the usage of execlp() system call.

**Solution:**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

int pid;

pid = fork();

if (pid < 0)

{

fprintf(stderr, "Fork failed!\n");

exit(-1);

}

else if (pid == 0)

{

printf("I am the child, return from fork=%d\n", pid);

execlp("/bin/ls", "ls", NULL);

}

else

{

printf("I am the parent, return from fork, child pid=%d\n", pid);

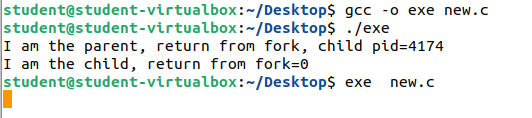
}

}

Explanation:-

In this program,We are creating two processes:- parent and child.We are checking if it is a parent process,Then,We are printing i am parent statement.Else,We are printing i am child statement and displaying all the folders and files present in the directory.

**Output:**



**Task No. 4:** Write a program to find sum of even numbers in parent process and sum of odd numbers in child process.

**Solution:**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

int n = 10;

int pid = fork();

if (pid == 0) {

// Child process

int sum = 0;

for (int i = 1; i <= n; i++) {

if (i % 2 == 1) {

printf("%d ", i);

sum += i;

}

}

printf("\nSum of odd numbers: %d\n", sum);

} else {

// Parent process

wait(NULL);

int sum = 0;

for (int i = 1; i <= n; i++) {

if (i % 2 == 0) {

printf("%d ", i);

sum += i;

}

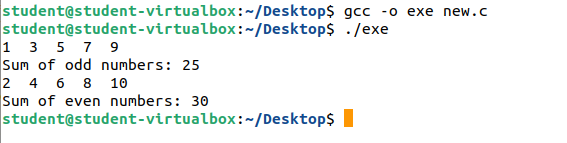
}

printf("\nSum of even numbers: %d\n", sum);

}

return 0;

}

**Output:**