Session: 2021 – 2025

Submitted by:

Abdul Mateen 2021-CS-190

Supervised to:

Dr. Abqa Javed

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

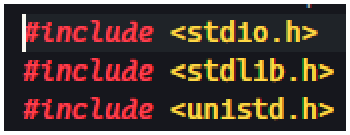
Description:

The Given programs calculates the Factorial using IPC (PIPE Implementation). The program is written in C Language.

Explanation

Headers

The header used are

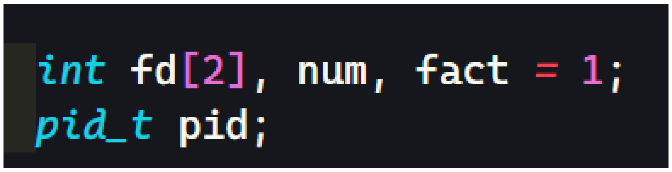


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

- The **stdio.h** header file allows us to perform input and output operations in C.
- The **stdlib.h** header file allows us to declares various utility functions for type conversions, memory allocation and algorithms.
- The **unistd.h** provides access to the POSIX operating system API.

Variables

After that we declared some variables

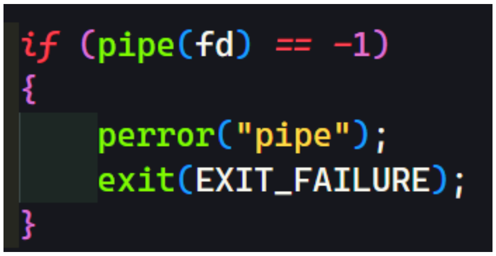


```c
int fd[2], num, fact = 1;
pid_t pid;
```

- **Fd** is an array that will hold the file descriptors for the read and write ends of the pipe.
- **Num** will hold the number entered by the user.
- **Fact** will hold the factorial of that number.
- **pid** is a variable of type **pid_t**, which will hold the process ID of the child process created by fork().

Pipe Creation

Next, a pipe is created

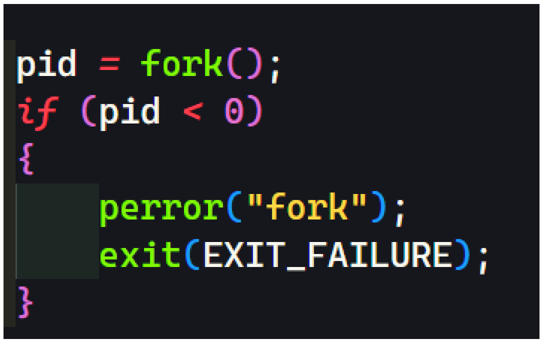


```c
if (pipe(fd) == -1)
{
    perror("pipe");
    exit(EXIT_FAILURE);
}
```

pipe() creates a pipe and returns two file descriptors: **fd[0]** is the read end of the pipe, and fd[1] is the write end of the pipe. If **pipe()** returns -1, an error has occurred and the program will exit.

Child Process

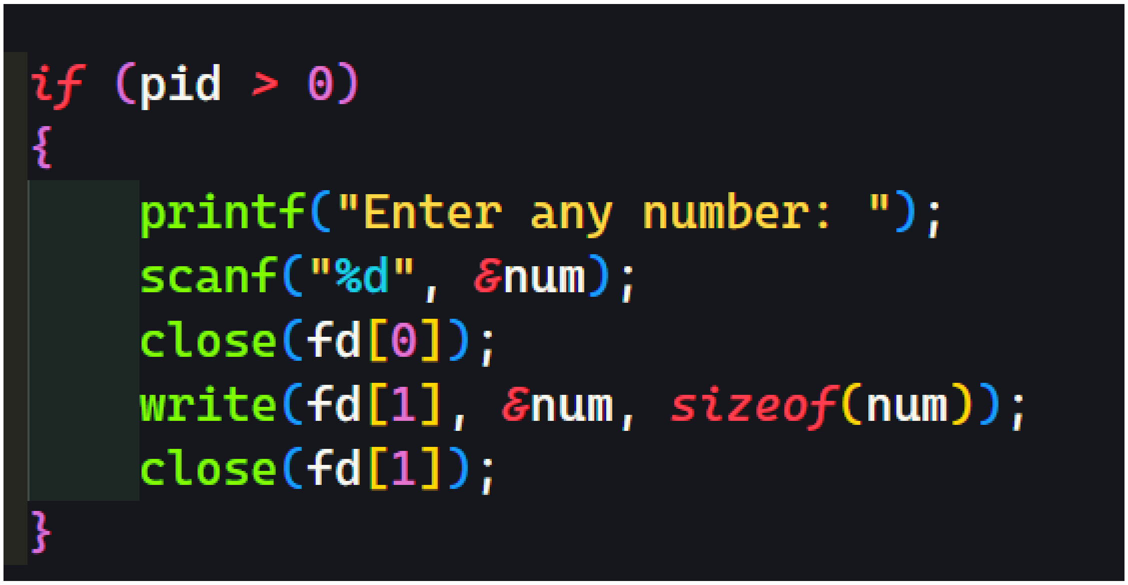
Next a child process is created



```
pid = fork();
if (pid < 0)
{
	perror("fork");
	exit(EXIT_FAILURE);
}
```

fork() creates a new child process by duplicating the calling process. If **fork()** returns -1, an error has occurred and the program will exit.

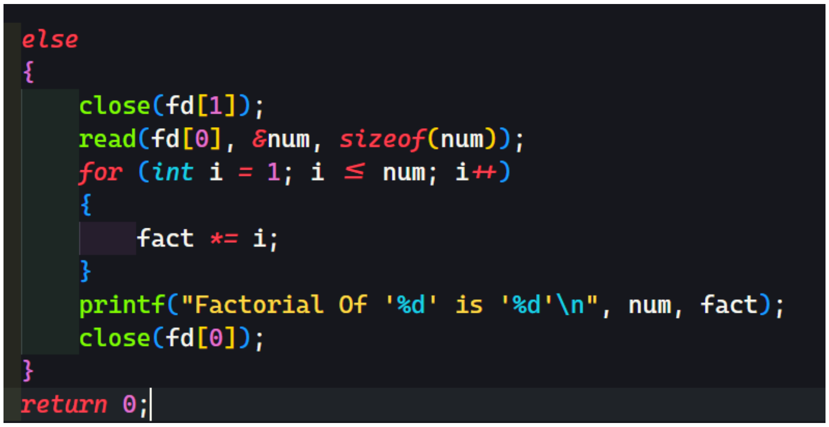
If fork() returns a positive value, the parent process is executing. The parent process prompts the user to enter a number, reads the number, and writes it to the pipe:



```
if (pid > 0)
{
	printf("Enter any number: ");
	scanf("%d", &num);
	close(fd[0]);
	write(fd[1], &num, sizeof(num));
	close(fd[1]);
}
```

close(fd[0]) closes the read end of the pipe, since the parent process will be writing to the pipe.
write(fd[1], &num, sizeof(num)) writes the number entered by the user to the write end of the pipe.
close(fd[1]) closes the write end of the pipe, since the parent process has finished writing to the pipe.

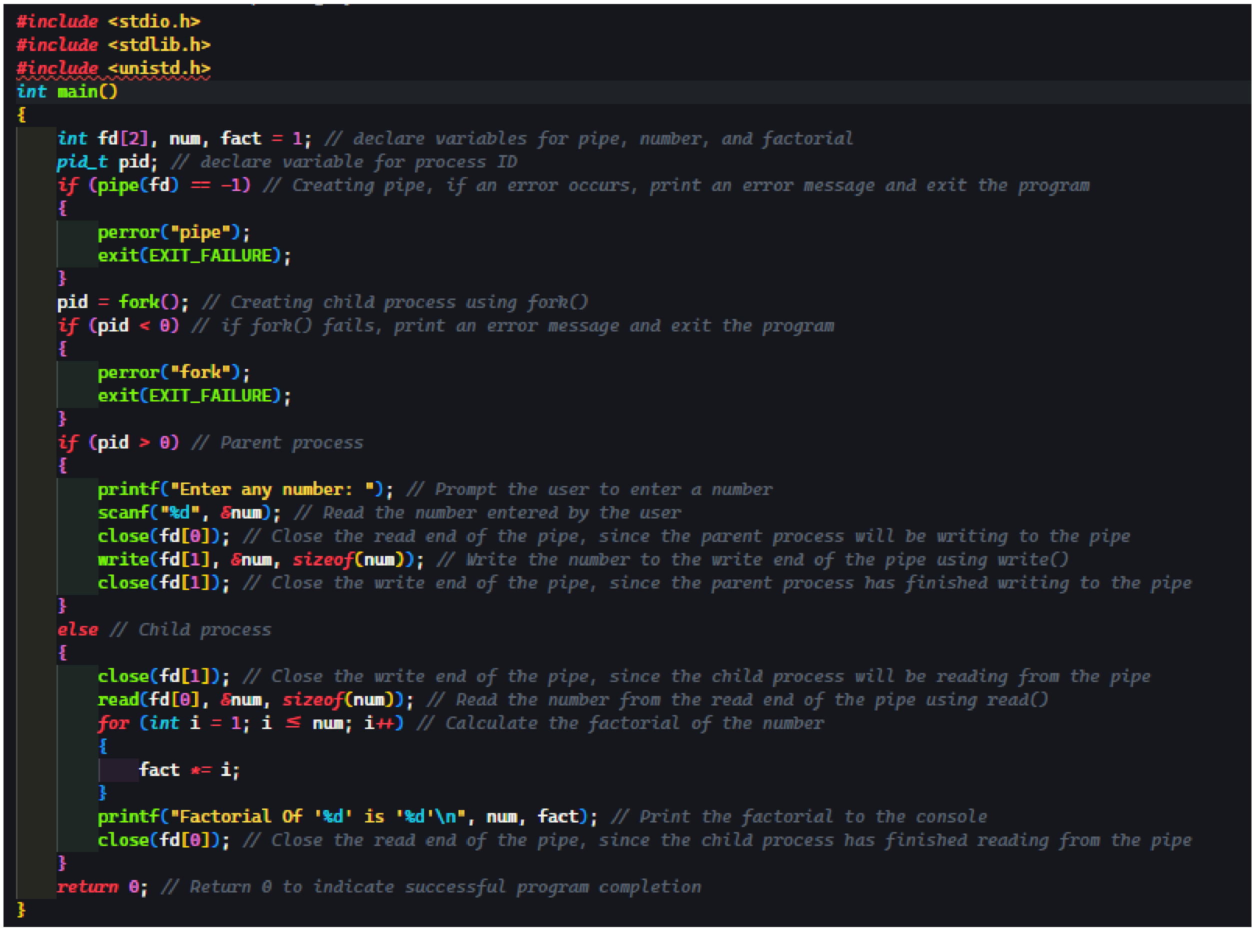
If **fork()** returns 0, the child process is executing. The child process reads the number from the pipe, calculates its factorial, and prints it to the console:



```
else
{
    close(fd[1]);
    read(fd[0], &num, sizeof(num));
    for (int i = 1; i <= num; i++)
    {
        fact *= i;
    }
    printf("Factorial Of '%d' is '%d'\n", num, fact);
    close(fd[0]);
}
return 0;
```

Code

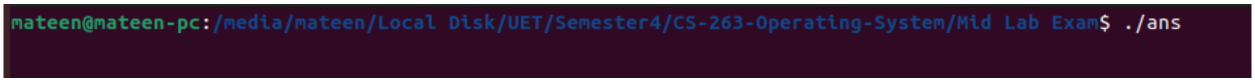
This is entire code of the Solution



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main()
{
    int fd[2], num, fact = 1; // declare variables for pipe, number, and factorial
    pid_t pid; // declare variable for process ID
    if (pipe(fd) == -1) // Creating pipe, if an error occurs, print an error message and exit the program
    {
        perror("pipe");
        exit(EXIT_FAILURE);
    }
    pid = fork(); // Creating child process using fork()
    if (pid < 0) // if fork() fails, print an error message and exit the program
    {
        perror("fork");
        exit(EXIT_FAILURE);
    }
    if (pid > 0) // Parent process
    {
        printf("Enter any number: "); // Prompt the user to enter a number
        scanf("%d", &num); // Read the number entered by the user
        close(fd[0]); // Close the read end of the pipe, since the parent process will be writing to the pipe
        write(fd[1], &num, sizeof(num)); // Write the number to the write end of the pipe using write()
        close(fd[1]); // Close the write end of the pipe, since the parent process has finished writing to the pipe
    }
    else // Child process
    {
        close(fd[1]); // Close the write end of the pipe, since the child process will be reading from the pipe
        read(fd[0], &num, sizeof(num)); // Read the number from the read end of the pipe using read()
        for (int i = 1; i <= num; i++) // Calculate the factorial of the number
        {
            fact *= i;
        }
        printf("Factorial Of '%d' is '%d'\n", num, fact); // Print the factorial to the console
        close(fd[0]); // Close the read end of the pipe, since the child process has finished reading from the pipe
    }
    return 0; // Return 0 to indicate successful program completion
}
```

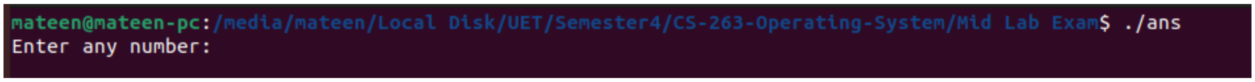
Output

This code takes input from user as it starts



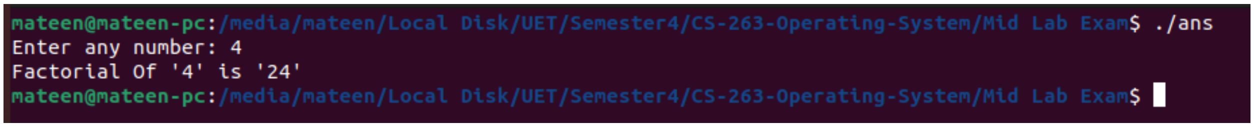
```
mateen@mateen-pc:/media/mateen/Local Disk/UET/Semester4/CS-263-Operating-System/Mid Lab Exam$ ./ans
```

Takes input



```
mateen@mateen-pc:/media/mateen/Local Disk/UET/Semester4/CS-263-Operating-System/Mid Lab Exam$ ./ans
Enter any number:
```

Perform calculations



```
mateen@mateen-pc:/media/mateen/Local Disk/UET/Semester4/CS-263-Operating-System/Mid Lab Exam$ ./ans
Enter any number: 4
Factorial Of '4' is '24'
mateen@mateen-pc:/media/mateen/Local Disk/UET/Semester4/CS-263-Operating-System/Mid Lab Exam$
```

And gives output on terminal.