FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

Student:

FIRST NAME: **Nouredeen Ahmed Mahmoud Ali**

LAST NAME: **Hammad**

NO: **2121221362**

DEPARTMENT: **BLM**

The project:

SUBJECT: **OpSys Project 1**

The lesson:

NAME: **Operating Systems**

INSTRUCTOR: **Prof. Dr. Ali Yılmaz ÇAMURCU / Arş. Gör. Samet KAYA**

CONTENT

1- Abstract

In this project I created a calculator that works on Linux's terminal. Each operation in the calculator runs as a separate sub-program and communicates with the main program using named pipes (FIFOs).
I learned how to use forks to run multiple subprograms, how to use inter-process communication, and how to send exit signals to the subprograms.

I also focused on following industry naming conventions for C language, ensuring memory safety, preventing any code repetition and writing easily readable and well commented code.

2- Project Topic

It is a calculator that runs on Linux's terminal that consists of 5 programs:

1. trmn.c
2. adder.c
3. subtractor.c
4. multiplier.c
5. divider.c

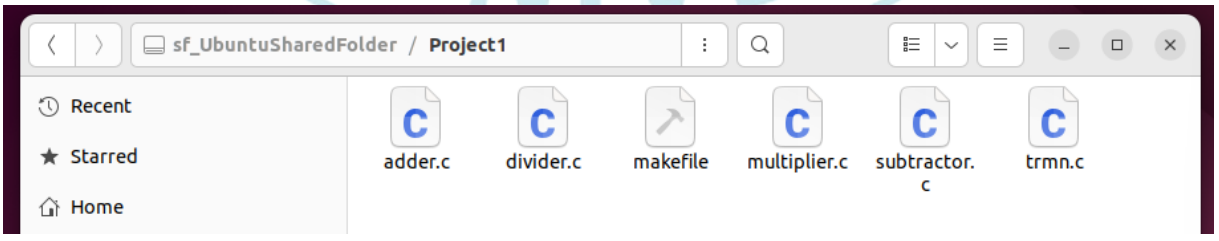
Trmn.c is the main program that launches the other 4 programs and maintains communications with them, it provides the user with a menu to choose which operation they want to perform, asks for the user's input, performs the operation and displays the result.

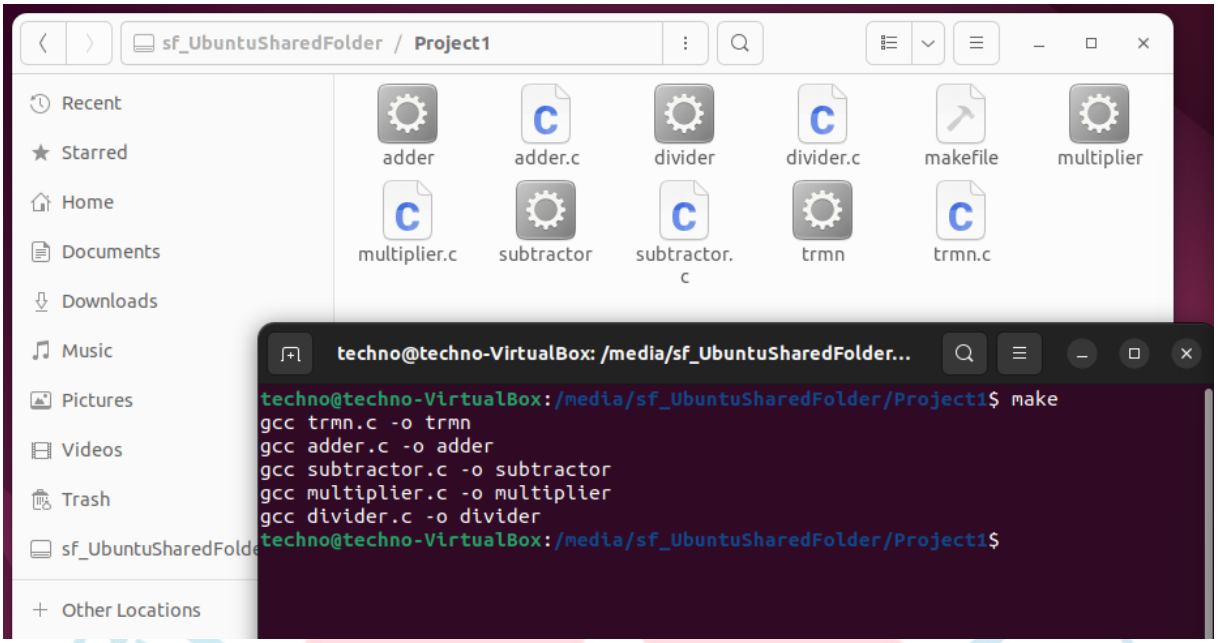
The trmn.c program displays the menu again after every operation, so the user can use it continually until exit (5) is entered.

The rest of the programs run on an infinite while loop waiting for an input to read from the named pipe then performs the operation and writes the result into the named pipe. The program terminates when receiving the exit signal.

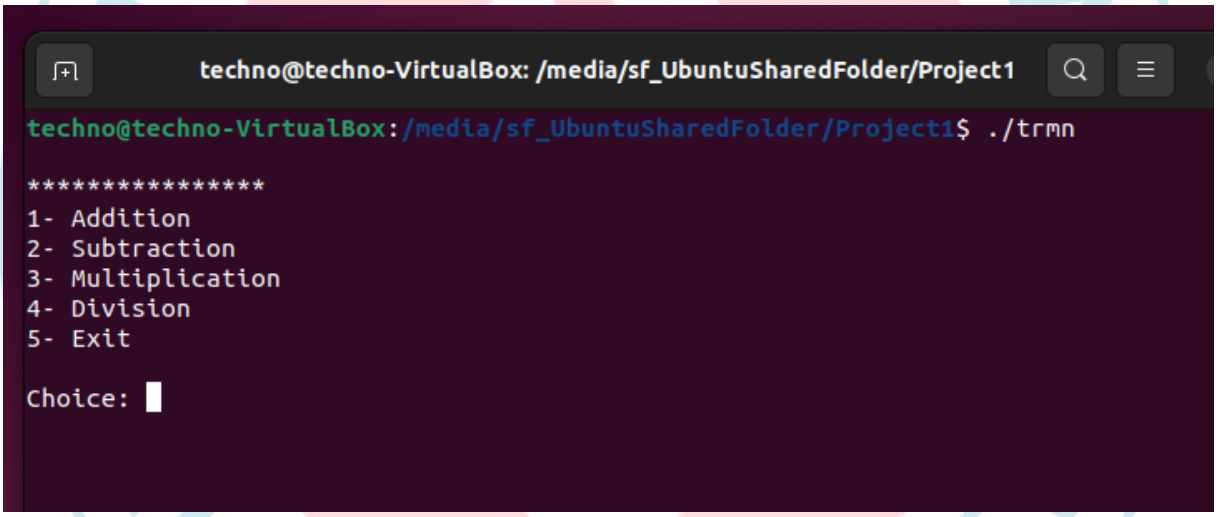
3- Project Outputs and Success Criteria

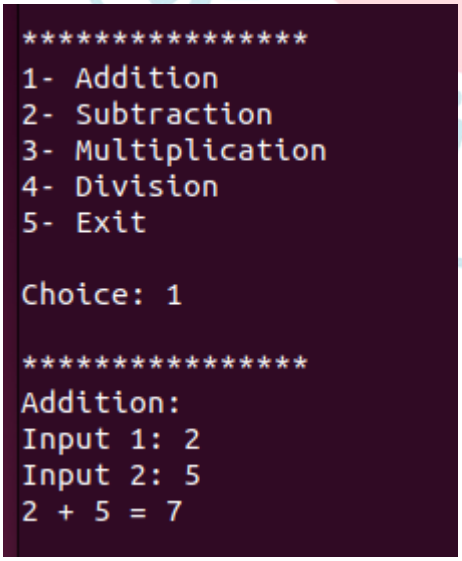
1) Makefile:

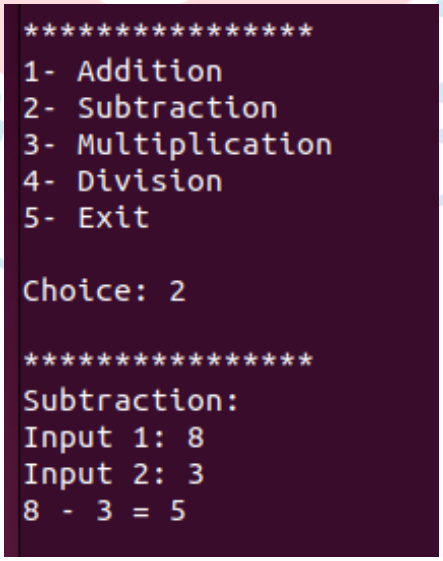




2) All operations:







```
****************
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Exit

Choice: 3

****************
Multiplication:
Input 1: 5
Input 2: 2
5 * 2 = 10
```

```
****************
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Exit

Choice: 4

****************
Division:
Input 1: 20
Input 2: 2
20 / 2 = 10
```

3) Exit:

```
****************
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Exit

Choice: 5
techno@techno-VirtualBox:/media/sf_UbuntuSharedFolder/Project1$
```

4- Things Done During the Project

I started the project by comparing the different ways of inter-process communication and I decided that named pipes (FIFOs) were the most suitable option.

I started the project by experimenting with named pipes and researching them to understand how they work exactly before I started working on the project's code.

I did all the testing and compilation on an Ubuntu virtual machine.

Then I developed only the adder program and some of the trmn program to check if things work correctly, because if these two work then the other operation will work if done in the same way.

When I was developing, from the start I was thinking and researching about how the structure of the code can be optimized and readable. I didn't want to create each sub-program and manage their communication separately.

I structured the code so that there is a **setup** function to initialize all the sub-process by forking them and initializing their pipes by calling **initPipe** and **forkProcess** in a for loop. The **terminate** function sends the exit signal to all sub-programs, unlinks all the named pipes and waits for termination.

I implemented functions such as **performOperation**, **readFromPipe**, **writeToPipe**, **getUserInputArr**, **get1UserInput** to prevent any code repetition.

After that, I downloaded **CppCheck** which is a linting program. I scanned my code to look for improvements. It suggested 4-5 things and I applied them. Finally, I created the makefile and started writing this report.

5- Additional explanations

Exit signal:

The exit signal in this project is a 2 element array with {INT_MIN, INT_MIN} for 3 reasons.

1. The trmn program sends data to the named pipes in the form of a 2 element int array.
2. The program can make calculations on negative numbers, so -1 would not be a valid value for an exit signal.
3. There's a very low chance that a user will want to perform any operation on 2 INT_MIN values.

Global Variables:

g_fifos is an array of strings that stores the path of each of the named pipes (FIFOs).

g_prog_names is an array of strings that stores the file names of each sub-program.

g_pids is an array that stores the PIDs of each forked process.

Note: both are indexed in the same order, so **g_fifos[0]** is the path for the adder sub-program's FIFO and **g_prog_names[0]** is also the name of the adder sub-program.

Input control:

The program checks user input at every step and makes sure it is valid, it will continue in a while loop until a valid input is entered.

In the menu there are 2 conditions:

1. The input must be an integer.
2. The input must be a number between 1 and 5.

```
****************
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Exit

Choice: -2
Invalid input! Please enter a number between 1 and 5.
```

```
****************
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Exit

Choice: abc
Invalid input! Please enter an integer value.
```

For operations there is 1 condition:

1. The input must be an integer.

For the division operation there is an extra condition that prevents division by zero.

```
****************
Addition:
Input 1: abc
Invalid input! Please enter an integer value.

Input 1: 5
Input 2: fsm
Invalid input! Please enter an integer value.

Input 2: 3
5 + 3 = 8
```

```
****************
Division:
Input 1: 10
Input 2: 0
You cannot divide by zero! Please enter a non-zero value.

Input 2: 5
10 / 5 = 2
```

6- References

I used some Youtube videos explaining pipes and named pipes for research.

I followed this makefile tutorial: [Makefile Tutorial By Example].

I used Bing AI to ask for general guidance, make it explain concepts with examples, help me debug and ask for industry conventions.