



FATİH SULTAN MEHMET

VAKIF ÜNİVERSİTESİ

Student:

FIRST NAME: **Nouredeen Ahmed Mahmoud Ali**

LAST NAME: **Hammad**

NO: **2121221362**

DEPARTMENT: **BLM**

The project:

SUBJECT: **OpSys Project 2**

The lesson:

NAME: **Operating Systems**

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

CONTENT

1-	Abstract.....	3
2-	Project Topic.....	3
3-	Project Outputs and Success Criteria.....	3
4-	Things Done During the Project.....	5
5-	Additional explanations.....	6
6-	References.....	7

1- Abstract

This project involves the development of a worker monitoring system designed to run in a Linux terminal. The system consists of a main program (Worker Monitor) and four worker programs (Adder, Subtractor, Multiplier, Divider). The communication between the main program and worker programs is achieved through named pipes (FIFOs). The project focuses on the use of threads, locks, and inter-process communication to ensure proper synchronization and functionality.

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.

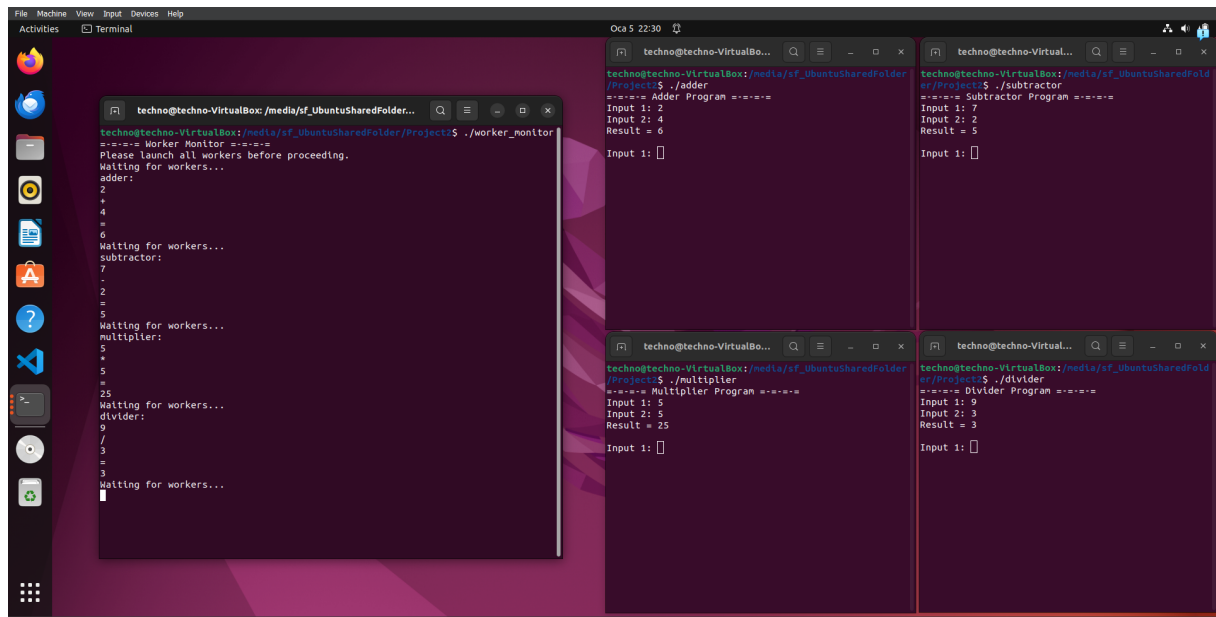
2- Project Topic

The primary goal is to create a worker monitor that operates in a terminal environment, so that operations on workers can be seen on the monitor. The system comprises the following programs:

1. **Worker Monitor (trmn.c):**
 - Launches and coordinates communication with worker programs.
 - Handles inputs through named pipes.
 - Creates a thread for each worker and constantly waits for input from the pipe.
2. **Worker Programs (adder.c, subtractor.c, multiplier.c, divider.c):**
 - Perform specific arithmetic operations (addition, subtraction, multiplication, division).
 - Communicate with the Worker Monitor through named pipes.
 - Run in an infinite loop.

3- Project Outputs and Success Criteria

1) Basic Operations:



```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./worker_monitor
===== Worker Monitor =====
Please launch all workers before proceeding.
Waiting for workers...
adder:
2
+
4
=
6
Waiting for workers...
subtractor:
7
-
2
=
5
Waiting for workers...
multiplier:
5
*
5
=
25
Waiting for workers...
divider:
9
/
3
=
3
Waiting for workers...
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./adder
===== Adder Program =====
Input 1: 2
Input 2: 4
Result = 6
Input 1: 
```

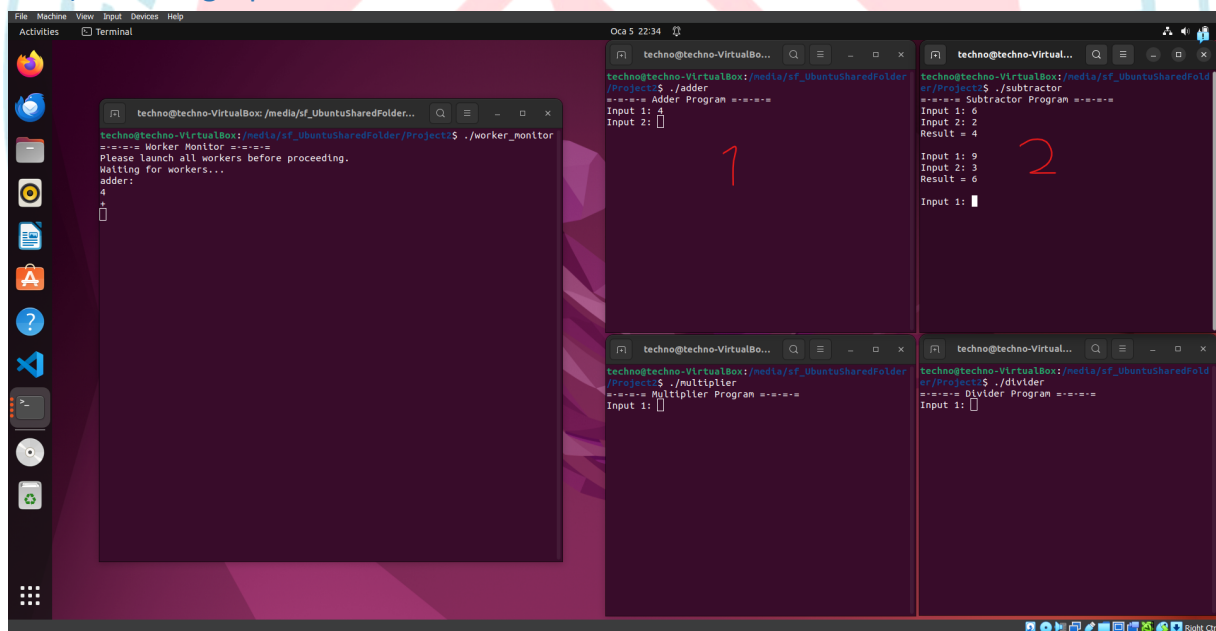
```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./subtractor
===== Subtractor Program =====
Input 1: 7
Input 2: 2
Result = 5
Input 1: 
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./multiplier
===== Multiplier Program =====
Input 1: 5
Input 2: 5
Result = 25
Input 1: 
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./divider
===== Divider Program =====
Input 1: 9
Input 2: 3
Result = 3
Input 1: 
```

The program works well when each worker performs its calculation separately.

2) Blocking Operation:



```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./worker_monitor
===== Worker Monitor =====
Please launch all workers before proceeding.
Waiting for workers...
adder:
4
+
0
=
4
Waiting for workers...
```

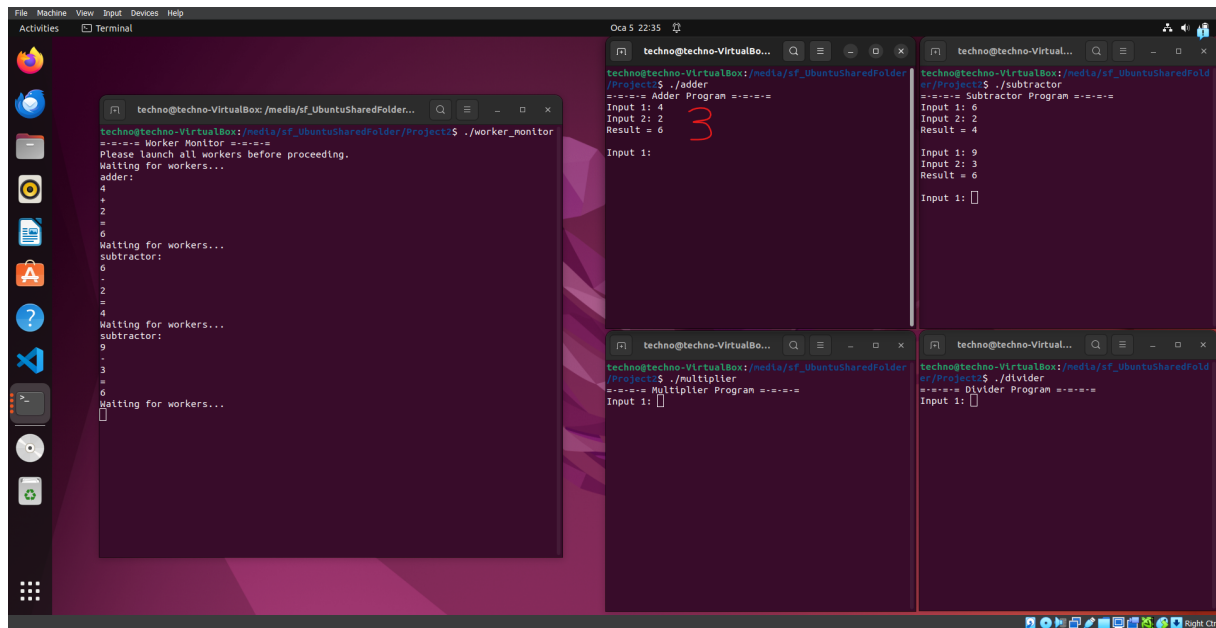
```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./adder
===== Adder Program =====
Input 1: 4
Input 2: 
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./subtractor
===== Subtractor Program =====
Input 1: 6
Input 2: 2
Result = 4
Input 1: 
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./multiplier
===== Multiplier Program =====
Input 1: 
```

```
techno@techno-VirtualBox: /media/sf_ubuntuSharedFolder/Project2$ ./divider
===== Divider Program =====
Input 1: 
```

If a worker starts an operation but doesn't finish it, the monitor will not display other workers' operations.



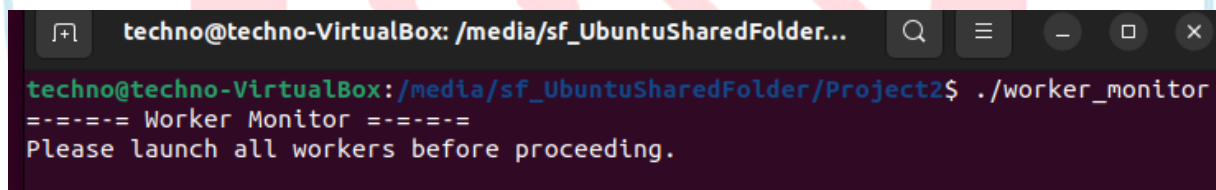
```
techno@techno-VirtualBox: /media/sf_UbuntuSharedFolder/Project2$ ./worker_monitor
===== Worker Monitor =====
Please launch all workers before proceeding.
Waiting for workers...
adder:
4
+
2
=
6
Waiting for workers...
subtractor:
6
-
2
=
4
Waiting for workers...
multiplier:
9
*
3
=
27
Waiting for workers...
divider:
9
/
3
=
3
```

When the worker does finish its operation, the monitor prints it and then prints all other operations in the queue that were done while the monitor was waiting.

3) FIFO Safety Measures:

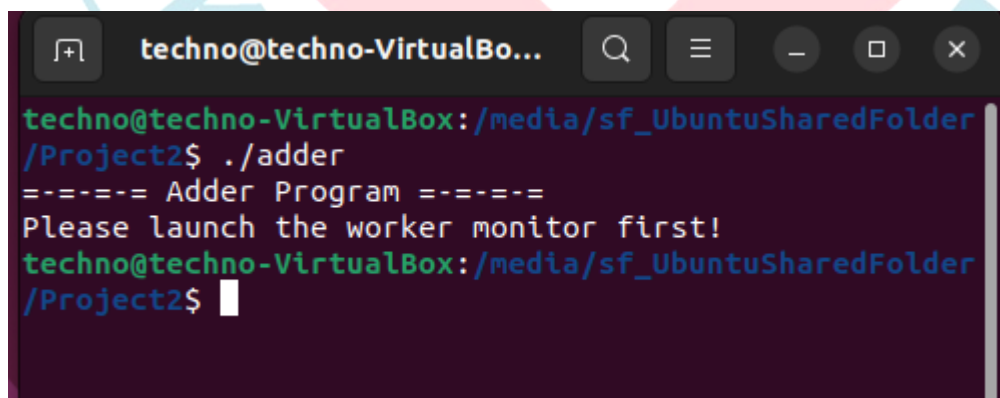
The named pipes (FIFOs) are created by the monitor.

The monitor waits for all workers to be launched first.



```
techno@techno-VirtualBox: /media/sf_UbuntuSharedFolder/Project2$ ./worker_monitor
===== Worker Monitor =====
Please launch all workers before proceeding.
```

If a worker runs before the monitor is run, the worker will terminate.



```
techno@techno-VirtualBox: /media/sf_UbuntuSharedFolder/Project2$ ./adder
===== Adder Program =====
Please launch the worker monitor first!
techno@techno-VirtualBox: /media/sf_UbuntuSharedFolder/Project2$
```

4- Things Done During the Project

Decided that named pipes (FIFOs) were the most suitable option.

Started with the Adder program, gradually expanding to other operations.

Ensured consistent code structure and naming conventions.

Implemented a setup function for initialization and a termination function for graceful exit.

Created utility functions to enhance code modularity.

Conducted extensive testing on the Adder program and key components of the Worker Monitor.

Debugged and refined code to ensure correct functionality.

Employed a setup function to initialize all sub-processes, reducing redundancy.

Developed a Makefile for easy compilation.

5- Additional explanations

Input control:

In all workers, 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.

6- References

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