FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

Name & Surname: Metin Kağıt

Student No: 1821221033

Department: BLM

Lecture: Operating Systems - Project 2

CONTENT

1- Abstract

The aim of this project is to investigate the use of threads, thread synchronization, named pipes, and threads with pipelines in the C programming language. To achieve this, the program runs each requested task in a separate thread. The first step is to create the thread and named channels in the 'monitor.c' class. Then, a task (function) is assigned to the threads. Threads created in this way will start listening to pipe channels. If the threads receives a message from any of the other classes ('adder.c', 'multiplier.c', 'divider.c', and 'subtractor.c'), it will be displayed on the screen.

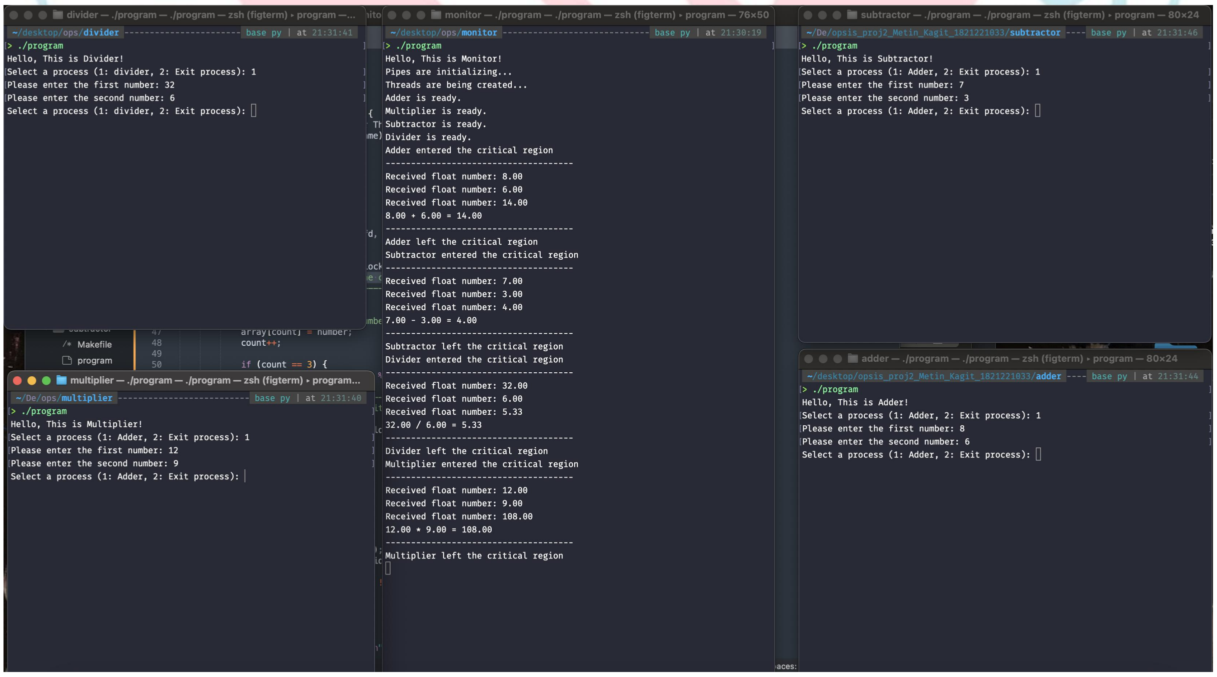
2- Project Topic

The topic of the project is implementing one of the fundamental operating system concept which is threads by using the C programming language. The program create 4 thread and provide communication between the 'monitor.c' and the other classes through pipe channels.

3- Project Outputs and Success Criteria

The target outputs of this project is to comprehend the working principles of named pipes and threads. Additionally, one of the project's goals is to learn how to handle deadlock, starvation, and critical section problems.

The project's success criterion is to complete the tasks specified in the document. This involves creating multiple threads and channels, and assigning a task thread to listen to the channels and print received messages. When a thread enters the critical section to print the message, it prevents other threads from entering the critical section due to the lock mechanism.


divider — ./program — ./program — zsh (figterm) ▸ program —...
monitor — ./program — ./program — zsh (figterm) ▸ program — 76×50
subtractor — ./program — ./program — zsh (figterm) ▸ program — 80×24
multiplier — ./program — ./program — zsh (figterm) ▸ program...
adder — ./program — ./program — zsh (figterm) ▸ program — 80×24

```
~/desktop/ops/divider ---------------------- base py | at 21:31:41
> ./program
Hello, This is Divider!
Select a process (1: divider, 2: Exit process): 1
Please enter the first number: 32
Please enter the second number: 6
Select a process (1: divider, 2: Exit process):
```

```
~/desktop/ops/monitor ---------------------------- base py | at 21:30:19
> ./program
Hello, This is Monitor!
Pipes are initializing...
Threads are being created...
Adder is ready.
Multiplier is ready.
Subtractor is ready.
Divider is ready.
Adder entered the critical region
-------------------------------------
Received float number: 8.00
Received float number: 6.00
Received float number: 14.00
8.00 + 6.00 = 14.00
-------------------------------------
Adder left the critical region
Subtractor entered the critical region
-------------------------------------
Received float number: 7.00
Received float number: 3.00
Received float number: 4.00
7.00 - 3.00 = 4.00
-------------------------------------
Subtractor left the critical region
Divider entered the critical region
-------------------------------------
Received float number: 32.00
Received float number: 6.00
Received float number: 5.33
32.00 / 6.00 = 5.33
-------------------------------------
Divider left the critical region
Multiplier entered the critical region
-------------------------------------
Received float number: 12.00
Received float number: 9.00
Received float number: 108.00
12.00 * 9.00 = 108.00
-------------------------------------
Multiplier left the critical region
```

```
~/De/opsis_proj2_Metin_Kagit_1821221033/subtractor ---- base py | at 21:31:46
> ./program
Hello, This is Subtractor!
Select a process (1: Adder, 2: Exit process): 1
Please enter the first number: 7
Please enter the second number: 3
Select a process (1: Adder, 2: Exit process):
```

```
~/De/ops/multiplier ------------------------- base py | at 21:31:40
> ./program
Hello, This is Multiplier!
Select a process (1: Adder, 2: Exit process): 1
Please enter the first number: 12
Please enter the second number: 9
Select a process (1: Adder, 2: Exit process):
```

```
~/desktop/opsis_proj2_Metin_Kagit_1821221033/adder ---- base py | at 21:31:44
> ./program
Hello, This is Adder!
Select a process (1: Adder, 2: Exit process): 1
Please enter the first number: 8
Please enter the second number: 6
Select a process (1: Adder, 2: Exit process):
```

Görsel 1.1

4- Things Done During the Project

The project starts with the creation of the required classes and makefiles. Following this step, four threads and pipes are created and the necessary variables are initialised. Tasks are then assigned to the threads. In addition, a function is written to print messages, which can also be called 'critical section'.

After the 'monitor' programme was created, programmes for 'adder.c', 'multiplier.c', 'divider.c' and 'subtractor.c' were created. Within these programmes, basic constructs were written to receive input from the user and write the numbers and results to the pipe writing end. After the first messages were written, the thread read them from 'monitor.c' and entered the critical section. In the critical section, it locked the area to prevent other threads from entering. After completing its task, it unlocked the area and left the section so that the next thread in the queue could enter.

In summary, these five programs run independently on different terminals and work together through pipes and threads. It is important to avoid deadlocks and starvation, which is why we have created a structure that uses flags to prevent a single thread from repeatedly entering a critical section.

5- Additional explanations

After opening five terminals, access the directory of each program.

- To compile, enter the "**make**" command for each ones.
- After compiling, enter the command "**./program**" for each ones to run the program.

Note: To view higher resolution images of the outputs, please refer to the 'screenshots' folder. Additional screenshots showing the lock mechanism can also be found there. They were not included here due to page size constraints.

6- References

- Operating System Lecture Labrotory Examples.

- https://stackoverflow.com/

- https://www.tutorialspoint.com/multithreading-in-c

- https://www.scaler.com/topics/multithreading-in-c/