**GIT Department of Computer Engineering**
**CSE 344 - Spring 2020**
# Homework 3 Report

**Esra EMİRLİ**
**151044069**

## 1 Problem Solution Approach

Firstly, two structs have been created to ensure that values are carried within the pipe. One of them is struct, which will be sent to children from P1 named Struct. It carries the corners of A and B to be used, the value of n, and which corner to calculate. Another is the struct that will be passed from P1 named resultStruct to children. It carries C inside which corner it calculates and holds its values. Since the bytes that can be carried in a pipe are limited, two separate structs were used. For the same reason, the number of elements of 2D arrays has been limited. In addition, a dynamic array of *dummy_array* is created for the function to calculate the singular values. Since this array can contain a maximum of $2^4$, n value is restricted. Later getopt() library were used as method for parsing command line arguments. If n value is negative as requested in homework, it gives error. If it is greater than 4, it returns an error for the reasons mentioned above.

### 1.1 P1 writes to pipe

Pipe and then 4 children were created. The pid value of each child was kept. Input files are read in P1. If there are not sufficient characters in the files then it will print an error message and exit.

For the corners to be calculated, which A and B corners should be sent, they are assigned to the struct. The information of all corners is written on the pipe, respectively. When the writing is finished, SIGUSR2 signal is sent to all children whose pids are already registered. Then P1 prevents all children from being executed until they complete their calculations. Wait was used to perform this operation. Each child automatically sends a SIGCHLD signal when it is done. wait catches this signal and realizes that all children are finished.

### 1.2 Child (P2, P3, P4, P5) Processes

While writing on P1 to the pipe continues in P1 process, the children are suspended and waiting for the SIGUSR2 signal. In the handling process of the SIGUSR2 signal, a *wakeUp* variable 1 is made to wake the children. After the children wake up, they read a structure object in the pipe to the Struct structure. The corner specified in the object is calculated with the given A and B arrays. This result is assigned to the C array in the resultStruct object that is planned to be sent in the pipe. Then this object is written on the pipe. All these calculations are made asynchronously among children. The process of the child writing on the pipe ends. In this case it triggers the SIGCHLD signal.

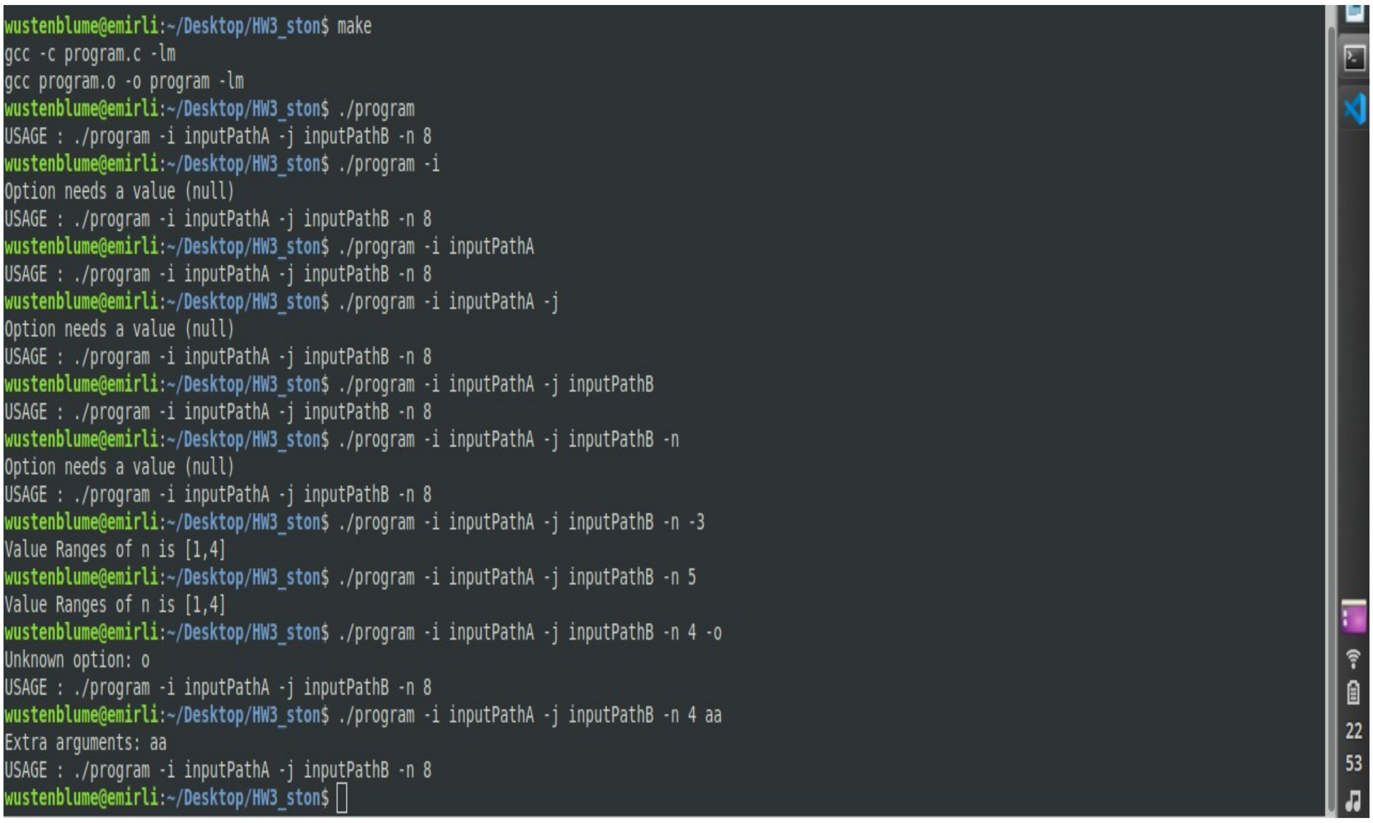
If CTRL-C is sent while these calculations are done, it is captured and zombies are blocked by sending the SIGTERM signal to the parent process.

### 1.3 P1 reads from pipe

P1 starts reading the pipe after all the child processes are finished. These objects are kept in an array named corners. The pipe is then closed. These collected partial outputs are combined and then the individual values of the product matrix are calculated. Later, the information contained in this http://www.mymathlib.com/c_source/matrices/linearsystems/singular_value_decomposition.c has been saved to the singular_value_decomposition.c file and this file has been added to the assignment. Then the singular values of the product matrix has been calculated with using Singular_Value_Decomposition() function in singular_value_decomposition.c file. Finally, singular values are printed on the terminal. At the end of all these processes, the P1 process ends.

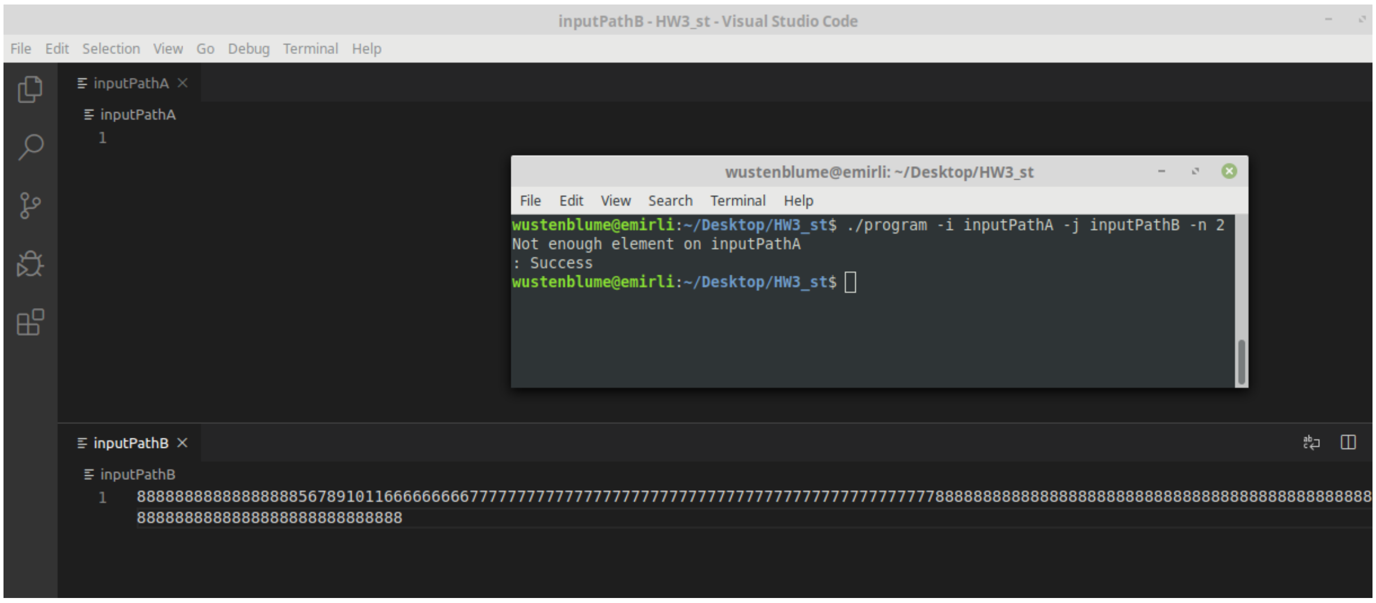
## 2 Running And Results

- Argument controls



```
wustenblume@emirli:~/Desktop/HW3_ston$ make
gcc -c program.c -lm
gcc program.o -o program -lm
wustenblume@emirli:~/Desktop/HW3_ston$ ./program
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i
Option needs a value (null)
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j
Option needs a value (null)
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB -n
Option needs a value (null)
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB -n -3
Value Ranges of n is [1,4]
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB -n 5
Value Ranges of n is [1,4]
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB -n 4 -o
Unknown option: o
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathA -j inputPathB -n 4 aa
Extra arguments: aa
USAGE : ./program -i inputPathA -j inputPathB -n 8
wustenblume@emirli:~/Desktop/HW3_ston$
```

- Not enough characters in inputPathA

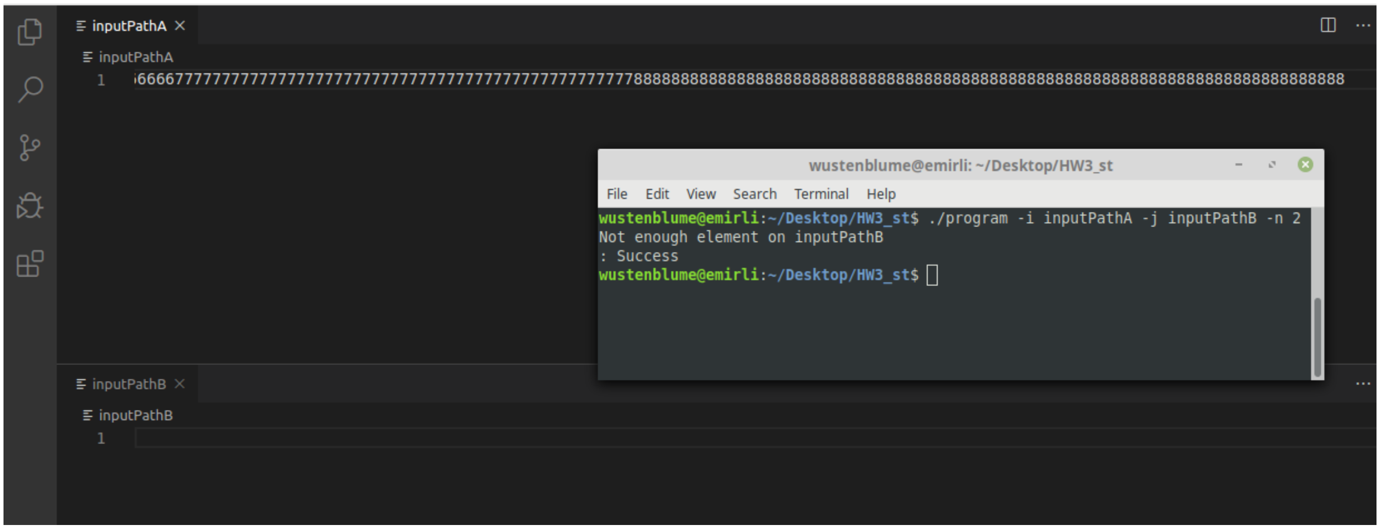

inputPathB - HW3_st - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
inputPathA
inputPathA
inputPathB
inputPathB

```
1
```

```
wustenblume@emirli:~/Desktop/HW3_st$ ./program -i inputPathA -j inputPathB -n 2
Not enough element on inputPathA
: Success
wustenblume@emirli:~/Desktop/HW3_st$
```

```
1   8888888888888888885678910116666666666777777777777777777777777777777777777777777777777777777777788888888888888888888888888888888888888888888888888
    8888888888888888888888888888888
```

- Not enough characters in inputPathB

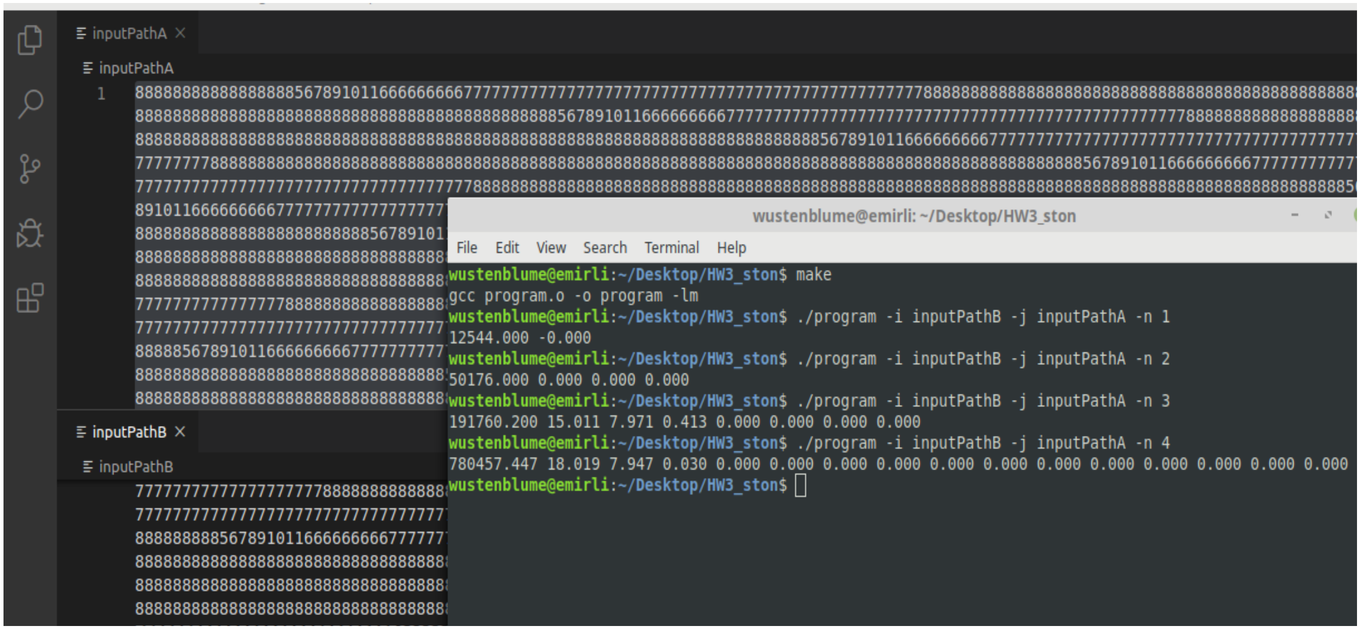


```
1   i6666777777777777777777777777777777777777777777777777777778888888888888888888888888888888888888888888888888888888888888888888888888888888888888
```

```
wustenblume@emirli:~/Desktop/HW3_st$ ./program -i inputPathA -j inputPathB -n 2
Not enough element on inputPathB
: Success
wustenblume@emirli:~/Desktop/HW3_st$
```

```
1
```

- Enough charachers both of inputPath



```
1   888888888888888888567891011666666666677777777777777777777777777777777777777777777777777777777777888888888888888888888888888888888888888888888888888
    888888888888888888888888888888888888888888888888888856789101166666666667777777777777777777777777777777777777777777777777777777778888888888888888888
    8888888888888888888888888888888888888888888888888888888888888888888888888888888888856789101166666666667777777777777777777777777777777777777777777
    777777778888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888567891011666666666677777777777
    7777777777777777777777777777777777777888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888885
    89101166666666667777777777777777
    8888888888888888888888888888856789101
    88888888888888888888888888888888888
    88888888888888888888888888888888888
    7777777777777777788888888888888888
    777777777777777777777777777777777777
    888885678910116666666666777777777
    88888888888888888888888888888888888
    88888888888888888888888888888888888
```

```
wustenblume@emirli:~/Desktop/HW3_ston$ make
gcc program.o -o program -lm
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathB -j inputPathA -n 1
12544.000 -0.000
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathB -j inputPathA -n 2
50176.000 0.000 0.000 0.000
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathB -j inputPathA -n 3
191760.200 15.011 7.971 0.413 0.000 0.000 0.000 0.000
wustenblume@emirli:~/Desktop/HW3_ston$ ./program -i inputPathB -j inputPathA -n 4
780457.447 18.019 7.947 0.030 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
wustenblume@emirli:~/Desktop/HW3_ston$
```

```
    77777777777777777788888888888888
    7777777777777777777777777777777777
    8888888885678910116666666666777777
    88888888888888888888888888888888888
    88888888888888888888888888888888888
    88888888888888888888888888888888888
```