

Operating Systems

Lab #02 Report

Name: Yousef Mohamed Zook

ID: 86

Code organization:

The code is split into 9 functions make 3 main jobs:

- 1- reading file – which is responsible for reading arrays from files
 - 2- write into file – which is responsible for writing an array to a file
 - 3- compute matrices multiplication – Make the actual multiplication with different methods
-

Main functions:

1- reading file functions:

- * *void setRowAndColInt(FILE* file);*
- * *int fromStrToDigit(char* str);*
- * *void readArr(FILE* file, int arr[rows_int][cols_int]);*

2- write into file:

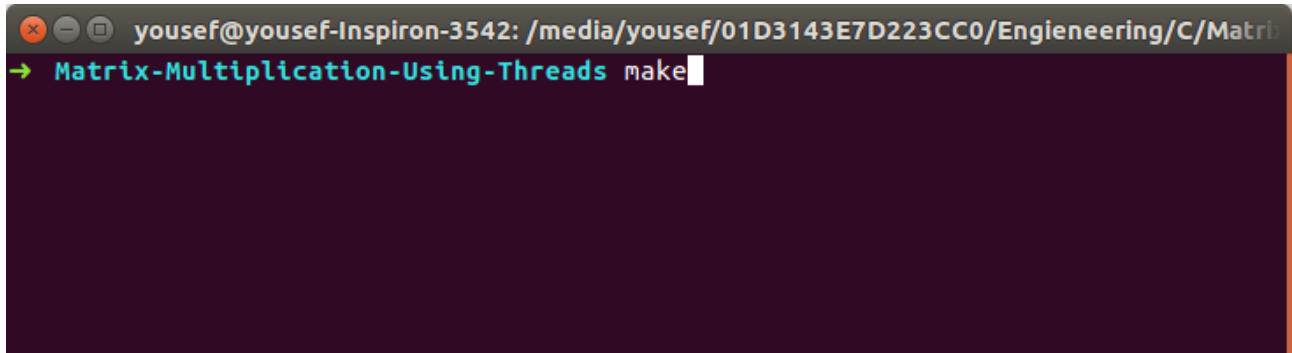
- * *void writeToFile(char* file_name, int rows, int cols, long matrix[rows][cols]);*

3- compute matrices multiplication

- * *void serialMultiplication(arrA, arrB, arrC);*
- * *void case1Multiplication(arrA, arrB, arrC);*
- * *void case2Multiplication(arrA, arrB, arrC);*

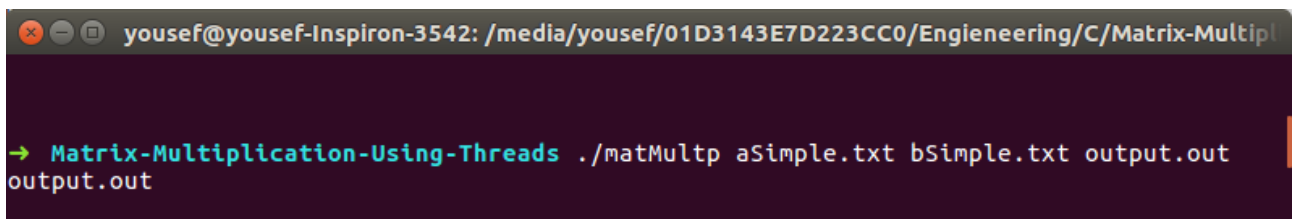
how to compile and run the code :

There exist a file called **Makefile** contains the compiling to ease the running for you, You have to just write “*make*” in the command prompt in the directory of the project as this:



```
yousef@yousef-Inspiron-3542: /media/yousef/01D3143E7D223CC0/Engieneering/C/Matrix-Multiplication-Using-Threads$ make
```

then you have to write “*./matMult [first Array file name] [Second Array file name] [Output file name]*” , where first, second file and output file names is optional but if there are not written the program will try the default “a.txt , b.txt and c.out”



```
yousef@yousef-Inspiron-3542: /media/yousef/01D3143E7D223CC0/Engieneering/C/Matrix-Multiplication-Using-Threads$ ./matMult aSimple.txt bSimple.txt output.out
```

sample run:

after running the program you will see in the terminal the time of each of serial multiplication , row thread multiplication and cell thread multiplication take to finish

```
yousef@yousef-Inspiron-3542: /media/yousef/01D3143E7D223CC0/Engieneering/C/Matrix-Multipl
→ Matrix-Multiplication-Using-Threads ./matMultp aSimple.txt bSimple.txt output.out
Normal case - Microseconds taken: 2
Row case - Number of threads : 2
Row case - Microseconds taken: 1076
Cell case - Number of threads : 8
Cell case - Microseconds taken: 201
→ Matrix-Multiplication-Using-Threads
```

also you will find an output file with the name you gave to it or `c.out` if default is used

simple input

```
row=2 col=3
1 2 3
4 5 6
```

```
row=3 col=4
1 2 3 4
5 6 7 8
9 10 11 12
```

simple output

```
38 44 50 56
83 98 113 128
```

```
method 1 using row threading
38 44 50 56
83 98 113 128
```

```
method 2 using cell threading
38 44 50 56
83 98 113 128
```

*sample run 2 : arrays of 500*500*

```
Normal case - Microseconds taken: 1844674
Row case - Number of threads : 500
Row case - Microseconds taken: 294316
Cell case - Number of threads : 250000
Cell case - Microseconds taken: 140170
```

comparison btetween mthods:

For big inputs it seems that method of cells threads take less time than the other but in very small inputs it showed the inverse as the time taken to create threads is more than the actual time of computations.

At sample run 2:

- Row method take about *0.3 Second*
- Cell method take about *0.15 second* - (About half of the time)
- Normal Serial method take about *1.9 Second*