Matrix Multiplication Using Threads

NAME: Islam Mostafa Abdelaziz Aboulkhair Gaber

ID: 13

EMAIL ADDRESS: Islammostafagaber@gmail.com

Educational ADDRESS: es-Islamabdelaziz2022@alexu.edu.eg

Code organization and main functions:

The code is split into 9 functions making 3 jobs:

- Reading file: which is responsible for reading arrays from files.
  - Void setRowandColInt(FILE* file);
  - int fromStrToDigit(char* str);
  - void readArr(FILE* file, int arr[rows_int][cols_int]);

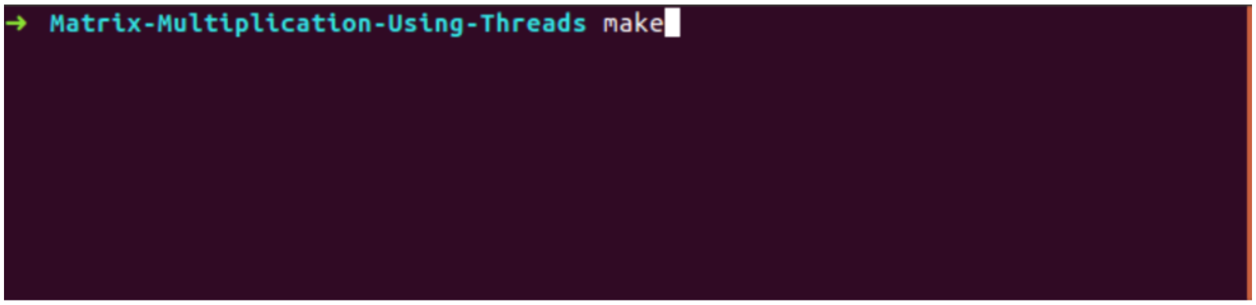
- Writing into file: which is responsible for writing an array to file.
  - Void writeToFile(char* file_name, int rows, int cols, long matrix[rows][cols]);

- Compute matrices multiplicaton: which make the actual multiplication with different methods.
  - Void normalMultiplication(int ra, int ca, int arrA[ra][ca], int rb, int cb, int arrB[rb][cb] , long arrC[ra][cb]);
  - void case1Multiplication(int ra, int ca, int arrA[ra][ca], int rb, int cb, int arrB[rb][cb] , long arrC[ra][cb]);
  - void * rowMultiplyer(void * args);
  - void case2Multiplication(int ra, int ca, int arrA[ra][ca], int rb, int cb, int arrB[rb][cb] , long arrC[ra][cb]);
  - void * cellMultiplyer(void * args);

How to compile and run the code:

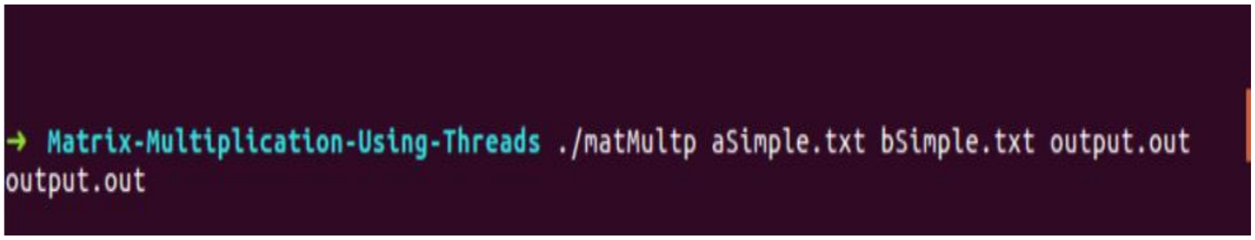
- there exists a file called "Makefile" contains the compiling to ease the running.

- You have to just write "make" in the command prompt in the directory of the project as this:



```
→ Matrix-Multiplication-Using-Threads make
```

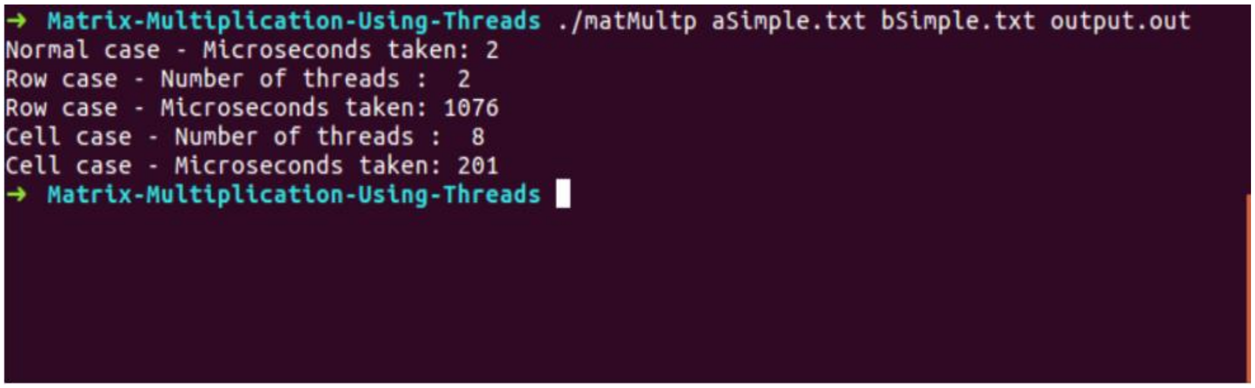
- Then you have to write "./matMult [first Array file name] [second Array file name] [output file name]" , where first, second and output file names is optional but if there are not written the program will try the default "a.txt , b.txt , c.out".



```
→ Matrix-Multiplication-Using-Threads ./matMultp aSimple.txt bSimple.txt output.out
output.out
```

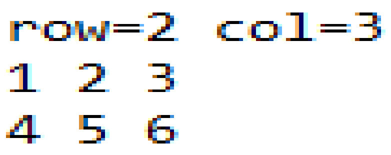
Sample runs:

- After running the program, you will see in the terminal the time of each of normal multiplication, row thread multiplication and cell thread multiplication take to finish.

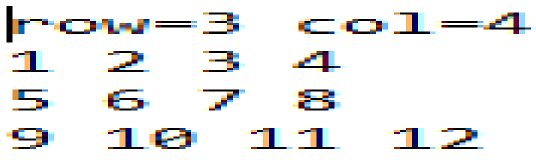


```
→ Matrix-Multiplication-Using-Threads ./matMultp aSimple.txt bSimple.txt output.out
Normal case - Microseconds taken: 2
Row case - Number of threads :  2
Row case - Microseconds taken: 1076
Cell case - Number of threads :  8
Cell case - Microseconds taken: 201
→ Matrix-Multiplication-Using-Threads
```

- Also you will find an output file with the name you gave to it or c.out if default is used.
    - ➢Simple input:

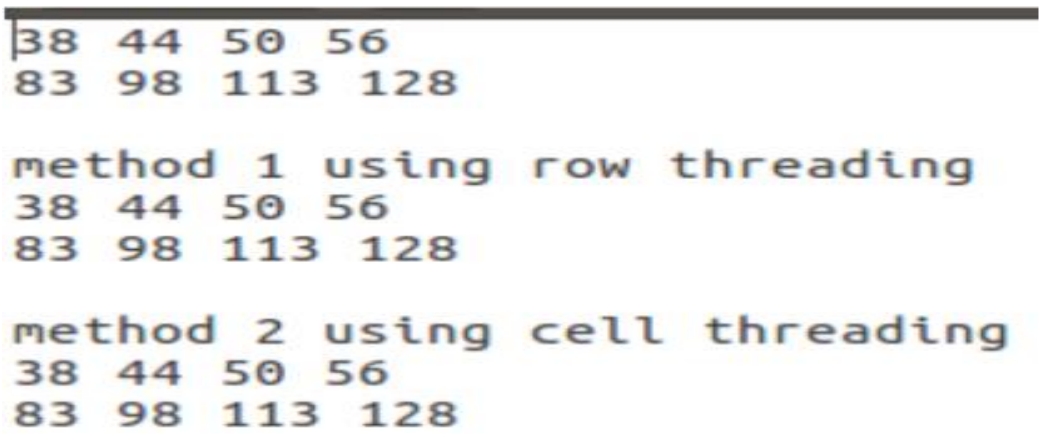


```
row=2 col=3
1 2 3
4 5 6
```



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

    - ➢Simple output:

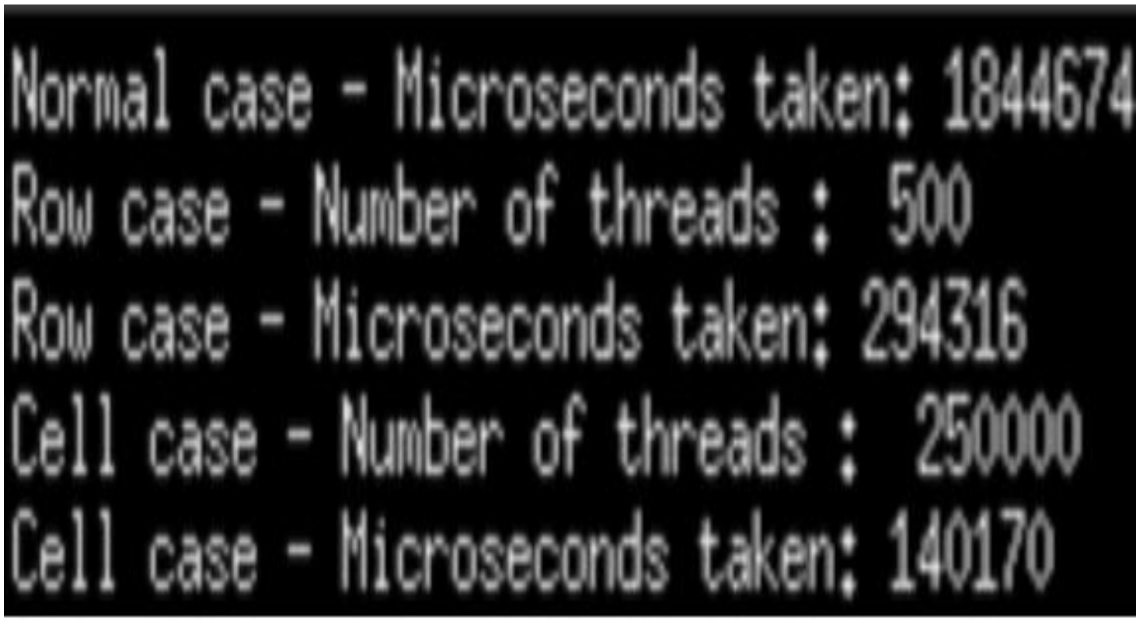


```
38 44 50 56
83 98 113 128

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 between methods:

- For big inputs it seems that method of cells threads takes 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 takes about 0.3 second.
  - Cell method takes about 0.15 second (about half of the time).
  - Normal method takes about 1.9 second.