Lab#2: Matrix Multiplication (Multi-Threading)

Student Name: Mohamed Ayman Saaed Mahmoud

ID: 19016250

- File Organization:

I’ve one folder has files:

1- Source code “matMultp.c”
2- Bash script to build source code “build.sh”
3- Five text files for input and output default “a.txt, b.txt, c_per_matrix.txt, c_per_row.txt, c_per_element.txt”

- Main Functions

- thread_per_matrix():
    - It doesn’t take arguments
    - It performs multiplication of matrix A * B
    - It put the result in matrix CPerMatrix

- thread_per_row():
    - It takes row number as argument
    - Perform multiplication of row of A * all columns of matrix B
    - It put the result in matrix CPerRow

- thread_per_element()
    - It takes struct has row number and column number as argument
    - Perform multiplication of row of A * a column of matrix B
    - It put the result in matrix CPerElement

- parse_argv():
    - It parse arguments that passes to main function
    - Set files variables name

- read_from(matrix, file):
    - It takes matrix and file name and
    - It serialize input from file to matrix

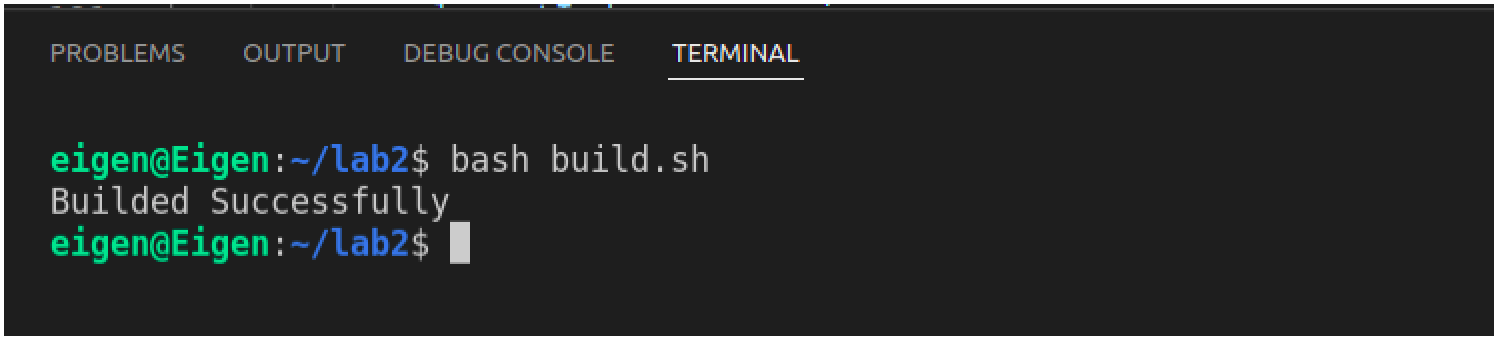
- write_file(matrix, file, det):
    - It takes matrix, file name and det variable
    - Det variable determine which file postfix
    - It write element of matrix to file name

- Construct():
    - It takes matrix reference, row and columns size
    - It allocate matrix in heap

- Code Organization
    - I’ve divided my code into function each does separate take
    - When you run the argument you entered taken by function parse_argv and it assign files name variables
    - Then I declare thread and path threads functions to each thread
    - Matrices allocated in Matrix struct has row and columns size and a reference to matrix elements in heap

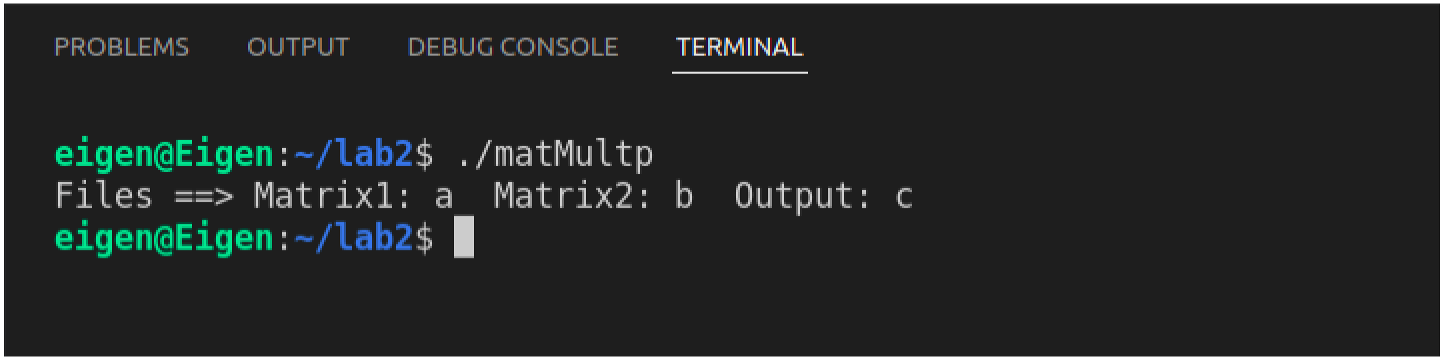
- How to compile and run

1- Open terminal in the folder that has source code “matMultp.c”
2- You will find file “build.sh” in that folder
3- Run the following command “bash build.sh”

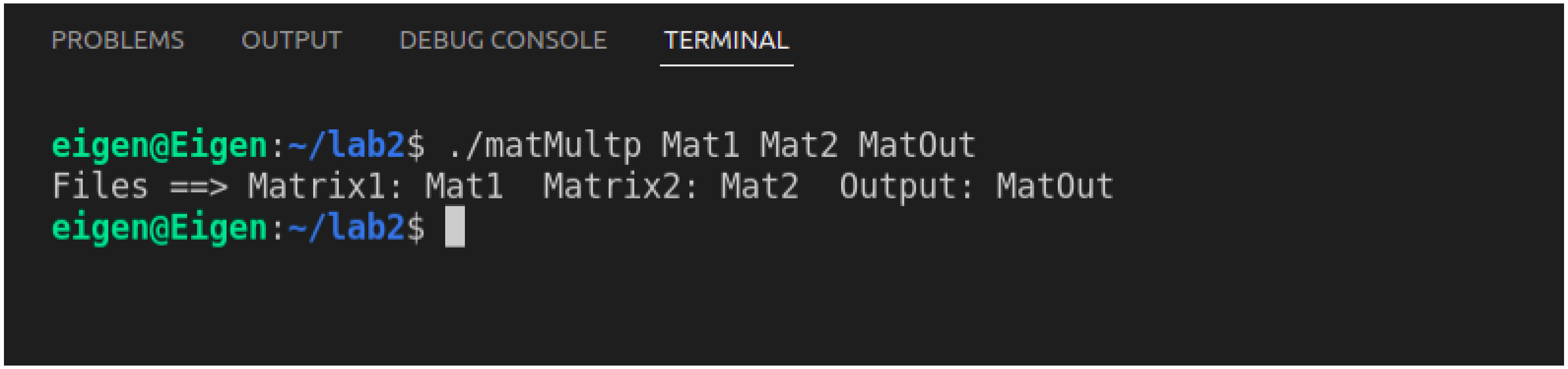


```
eigen@Eigen:~/lab2$ bash build.sh
Builded Successfully
eigen@Eigen:~/lab2$ 
```

4- You will notice that file “matMultp” created.
5- Execute command “./matMultp ‘arg1’ ‘arg2’ ‘arg3’ to run the file



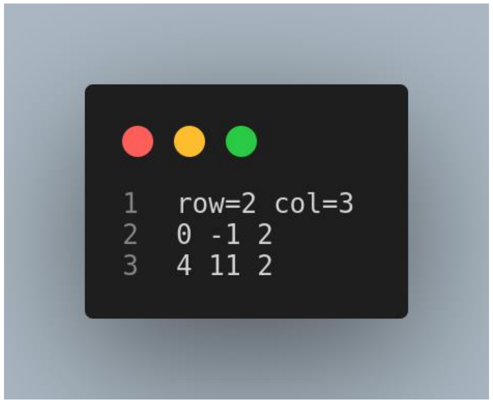
```
eigen@Eigen:~/lab2$ ./matMultp
Files ==> Matrix1: a  Matrix2: b  Output: c
eigen@Eigen:~/lab2$ 
```



```
eigen@Eigen:~/lab2$ ./matMultp Mat1 Mat2 MatOut
Files ==> Matrix1: Mat1  Matrix2: Mat2  Output: MatOut
eigen@Eigen:~/lab2$ 
```

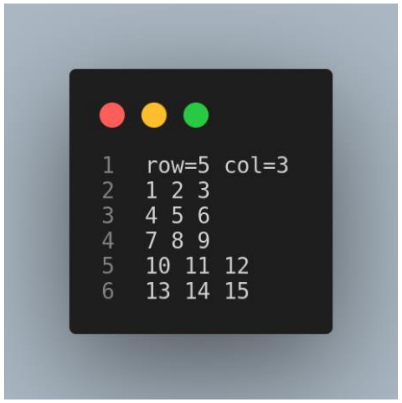
- Sample Runs

$$\mathbf{C} = \mathbf{A}\cdot\mathbf{B} = \begin{pmatrix} 0 & -1 & 2 \\ 4 & 11 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 & -1 \\ 1 & 2 \\ 6 & 1 \end{pmatrix} = \begin{pmatrix} 11 & 0 \\ 35 & 20 \end{pmatrix}$$



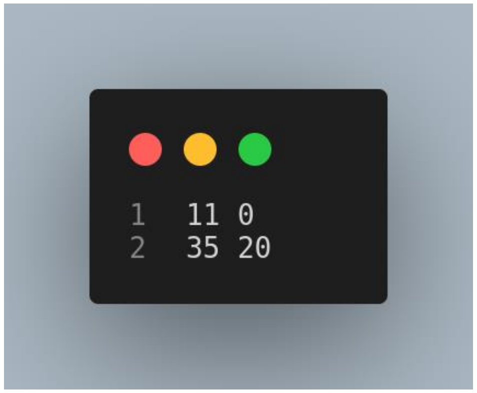
```
row=2 col=3
0 -1 2
4 11 2
```

a.txt



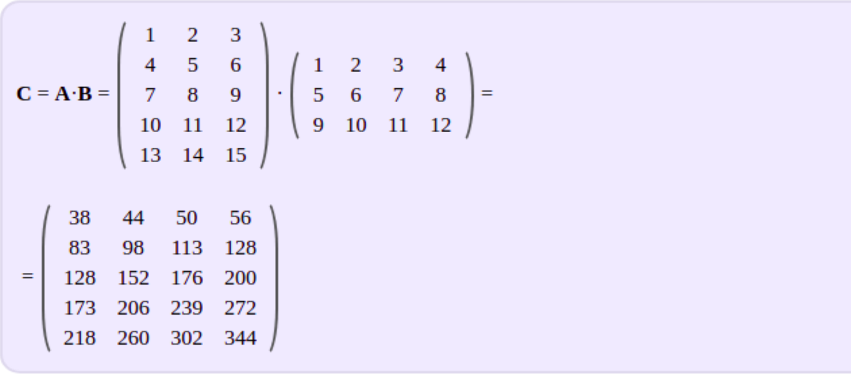
```
row=5 col=3
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
```

b.txt



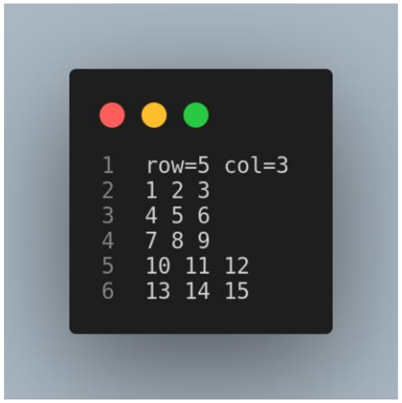
```
11 0
35 20
```

output



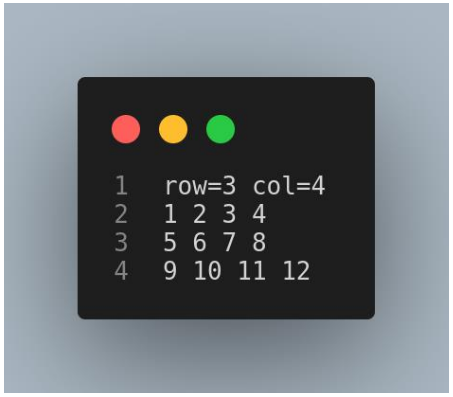
$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} =$$

$$= \begin{pmatrix} 38 & 44 & 50 & 56 \\ 83 & 98 & 113 & 128 \\ 128 & 152 & 176 & 200 \\ 173 & 206 & 239 & 272 \\ 218 & 260 & 302 & 344 \end{pmatrix}$$



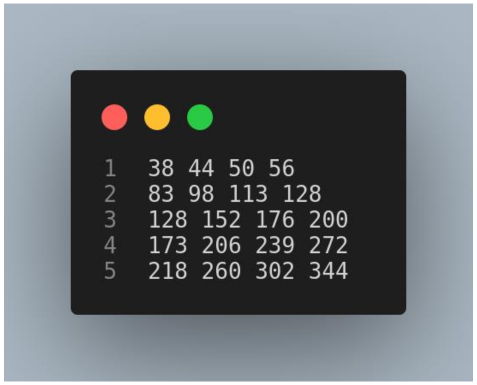
```
row=5 col=3
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
```

a.txt



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

b.txt

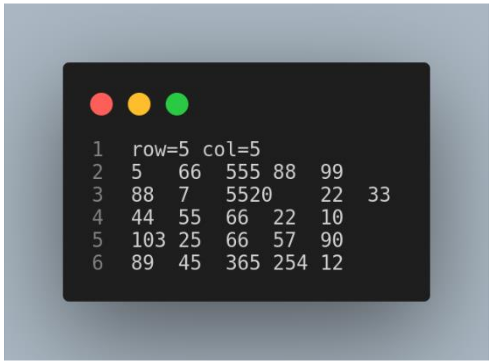


```
38 44 50 56
83 98 113 128
128 152 176 200
173 206 239 272
218 260 302 344
```

output

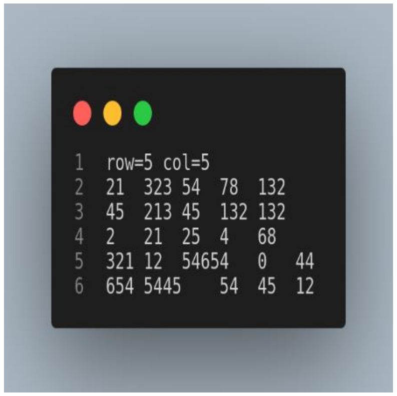
$$\mathbf{C} = \mathbf{A}\cdot\mathbf{B} = \begin{pmatrix} 5 & 66 & 555 & 88 & 99 \\ 88 & 7 & 5520 & 22 & 33 \\ 44 & 55 & 66 & 22 & 10 \\ 103 & 25 & 66 & 57 & 90 \\ 89 & 45 & 365 & 254 & 12 \end{pmatrix} \cdot \begin{pmatrix} 21 & 323 & 54 & 78 & 132 \\ 45 & 213 & 45 & 132 & 132 \\ 2 & 21 & 25 & 4 & 68 \\ 321 & 12 & 54654 & 0 & 44 \\ 654 & 5445 & 54 & 45 & 12 \end{pmatrix} =$$

$$= \begin{pmatrix} 97179 & 567439 & 4832013 & 15777 & 52172 \\ 41847 & 325784 & 1347237 & 31353 & 389264 \\ 17133 & 82027 & 1209429 & 11406 & 18644 \\ 80577 & 530714 & 3128475 & 15648 & 24972 \\ 94006 & 114385 & 13898720 & 14882 & 53828 \end{pmatrix}$$



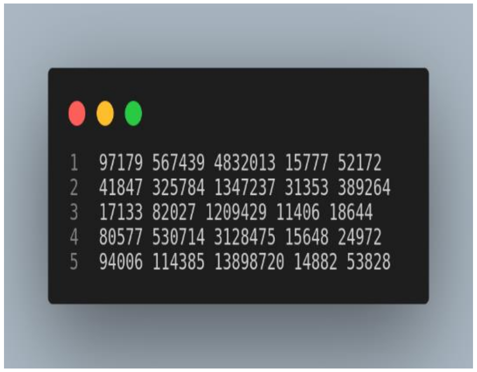
```
row=5 col=5
5   66  555 88  99
88  7   5520    22  33
44  55  66  22  10
103 25  66  57  90
89  45  365 254 12
```

a.txt



```
row=5 col=5
21  323 54  78  132
45  213 45  132 132
2   21  25  4   68
321 12  54654   0   44
654 5445    54  45  12
```

b.txt



```
97179 567439 4832013 15777 52172
41847 325784 1347237 31353 389264
17133 82027 1209429 11406 18644
80577 530714 3128475 15648 24972
94006 114385 13898720 14882 53828
```

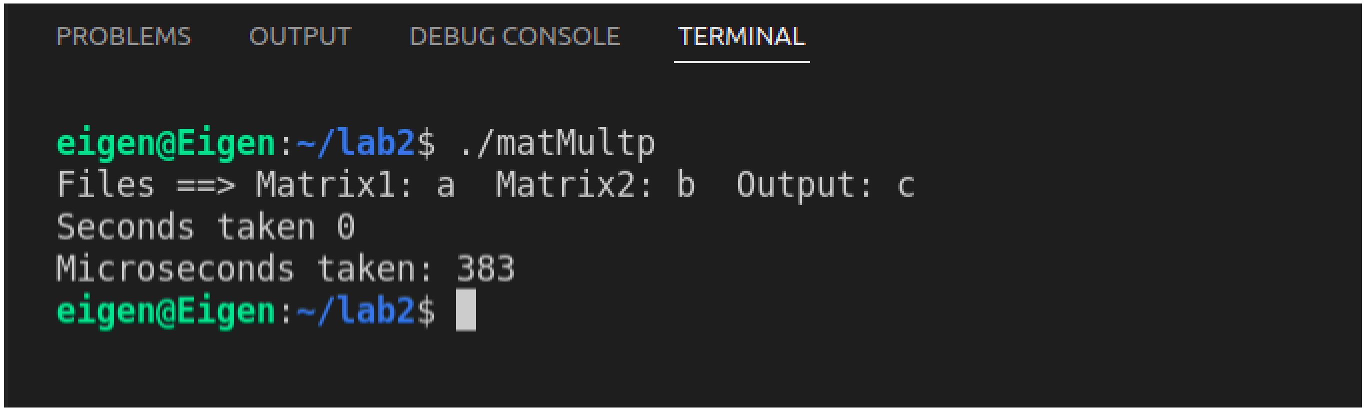
output

- Comparison between methods

1- Thread per matrix

  a. It needs one thread.

  b. Time it takes.

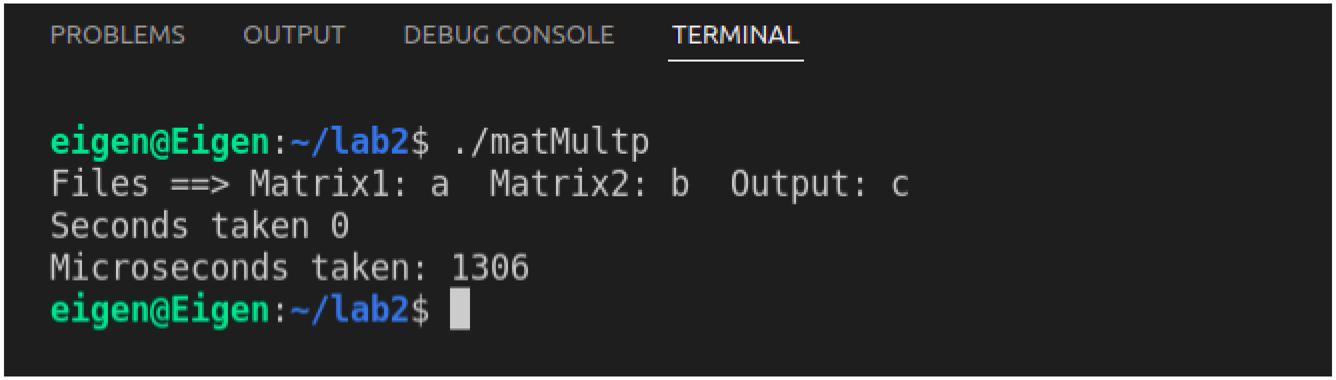


```
eigen@Eigen:~/lab2$ ./matMultp
Files ==> Matrix1: a  Matrix2: b  Output: c
Seconds taken 0
Microseconds taken: 383
eigen@Eigen:~/lab2$ 
```

2- Thread per row

  a. It needs threads equal to number of rows of matrix A.

  b. Time it takes.

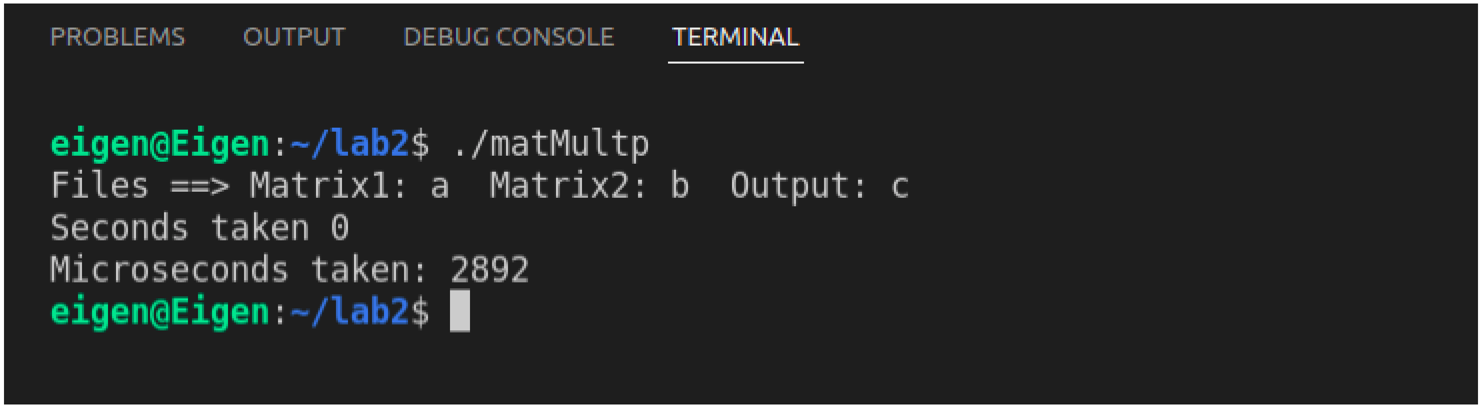


```
eigen@Eigen:~/lab2$ ./matMultp
Files ==> Matrix1: a  Matrix2: b  Output: c
Seconds taken 0
Microseconds taken: 1306
eigen@Eigen:~/lab2$ 
```

3- Thread per element

a. It needs threads equal to the dimension of the result array.

b. Time it takes.



```
eigen@Eigen:~/lab2$ ./matMultp
Files ==> Matrix1: a  Matrix2: b  Output: c
Seconds taken 0
Microseconds taken: 2892
eigen@Eigen:~/lab2$ 
```