Faculty of Engineering, Alexandria University

Computer and Systems Engineering Department

CS 333 | Operating Systems | Fall 2017

Name: Aya Aly Saad Aly Abouzeid

SID: 2

A) Code organization:

The code is divided into six .c files,

1. **main.c:**

   -Determine which files the program will work with.

   -call the multiplying functions.

   -Compute execution time.

   -Call the printing functions.

2. **read_from_files.c:**

   -Read the given matrices files.

   -Fill the matrices with the given input.

3. **matMult.c:**

   -Compute the matrices multiplication using no threads.

4. **Method1.c:**

   -Compute the multiplication using threads for each row in the output matrix.

5. **Method2.c:**

   -Compute the multiplication using threads for each element in the output matrix.

6. **output_to_file.c:**

-Prints the output of multiplication to the given output files.

B) Main functions:

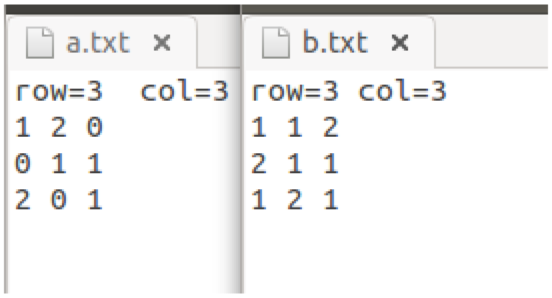
- **readMatrices()**
  - reads marices files.
- **multiplyMatrices()**
  - calls the three multipling functions.
- **getDimentions()**
  - reads the matrices dimentions in the first line.
- **MultiplyNoThreads()**
  - multiply the matrices using no threads.
- **useMethod1()**
  - creates threads for computing each row.
- **Multiply1()**
  - multiplies the 2 matrices using a thread for each row.
- **useMethod2()**
  - creates threads for computing each element.
- **Multiply2()**

  - multiplies the 2 matrices using a thread for each element.

- **writeOutput1() , writeOutput2()**
  - prints the output to external files.

C) Sample runs:

1-

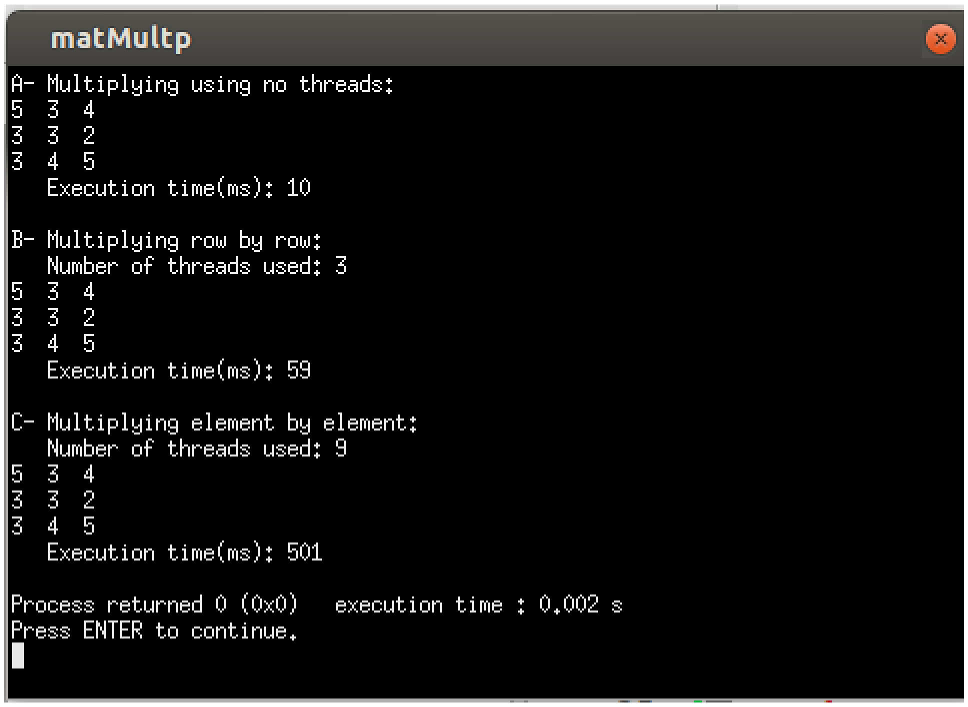


a.txt

```
row=3  col=3
1 2 0
0 1 1
2 0 1
```

b.txt

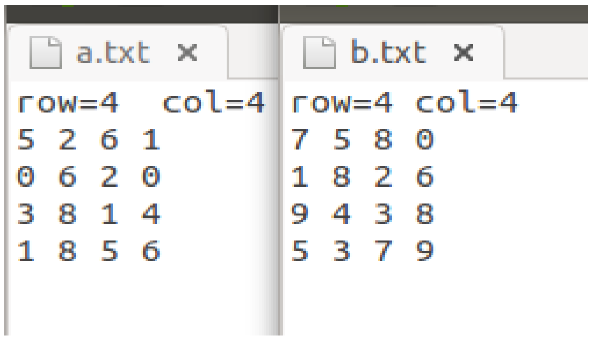
```
row=3 col=3
1 1 2
2 1 1
1 2 1
```



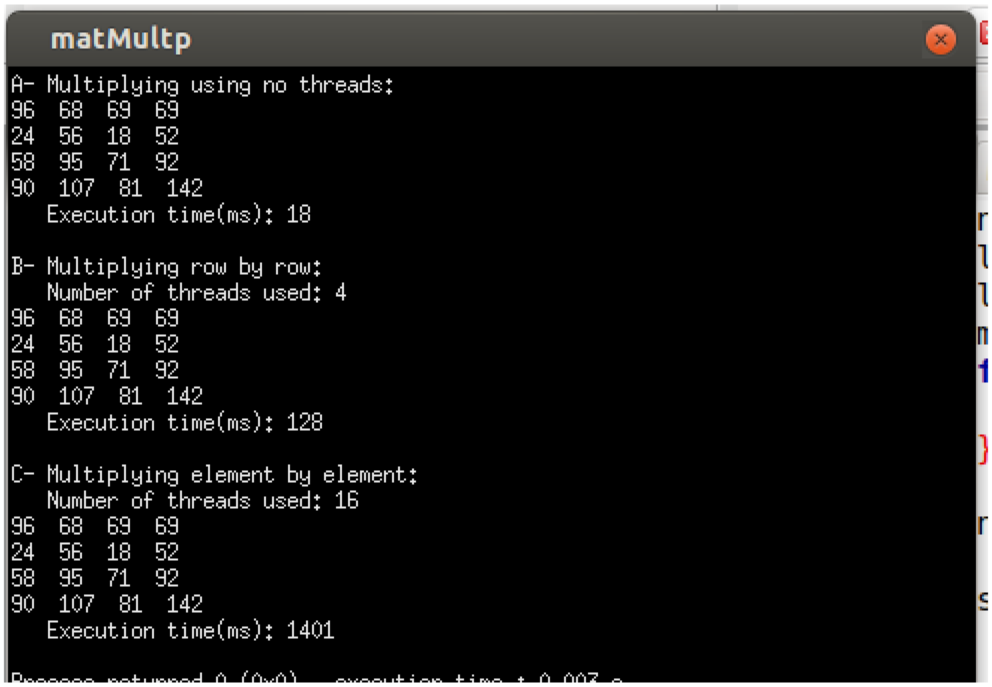
matMultp

```
A- Multiplying using no threads:
5  3  4
3  3  2
3  4  5
   Execution time(ms): 10

B- Multiplying row by row:
   Number of threads used: 3
5  3  4
3  3  2
3  4  5
   Execution time(ms): 59

C- Multiplying element by element:
   Number of threads used: 9
5  3  4
3  3  2
3  4  5
   Execution time(ms): 501

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

2-



```
a.txt              b.txt
row=4   col=4      row=4  col=4
5 2 6 1            7 5 8 0
0 6 2 0            1 8 2 6
3 8 1 4            9 4 3 8
1 8 5 6            5 3 7 9
```



```
matMultp
A- Multiplying using no threads:
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 18

B- Multiplying row by row:
   Number of threads used: 4
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 128

C- Multiplying element by element:
   Number of threads used: 16
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 1401

Process returned 0 (0x0)   execution time : 0.003 s
```

D) Compiling and running :

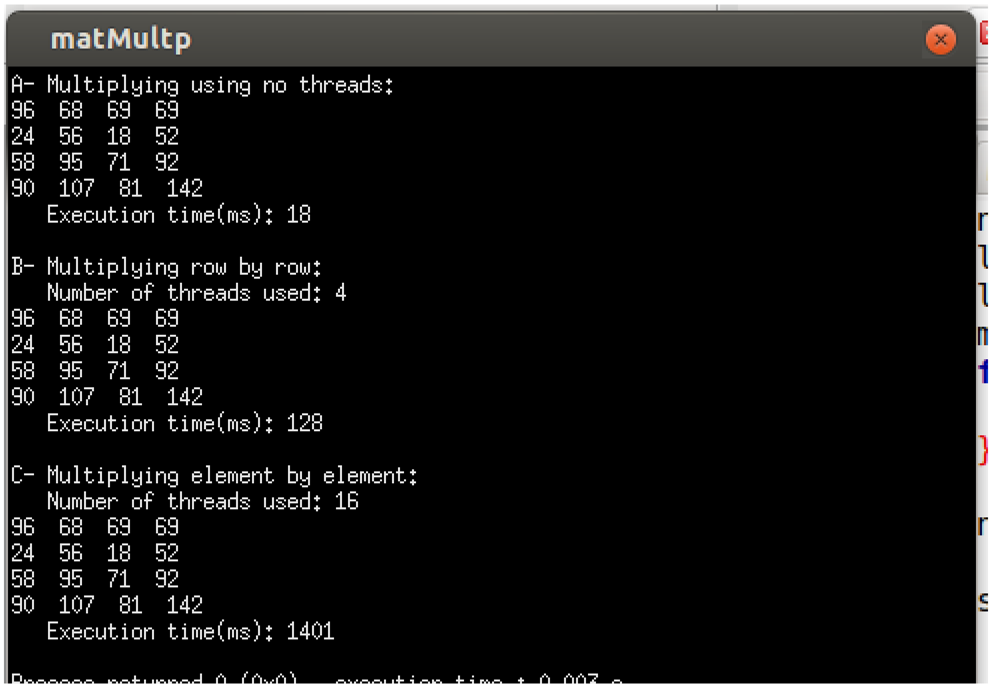
1. Change the directory to the project's directory.
2. Open a terminal.
3. Type “make” command.
4. Execute by typing “**./matmult.out**” either by specifying input and output files such as “**./matmult.out x.txt y.txt z.out**” or just “**./matmult.out**” and the the program would consider the default values which are a.txt , b.txt for input files and c1.out , c2.out for output files.

E) comparing the two methods of matrix multiplication:

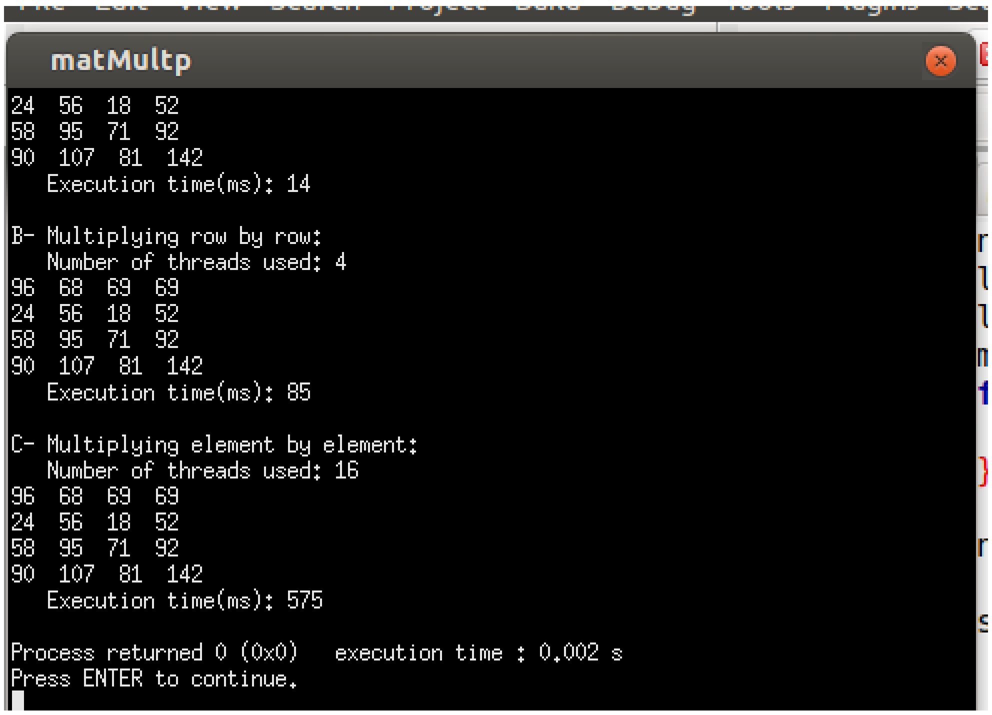
- After several runs for the program it appears that the execution time is as follows:

  No threads < Thread for each row < Thread for each element.

  Here are several runs for the same input showing the execution time:



```
A- Multiplying using no threads:
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 18

B- Multiplying row by row:
   Number of threads used: 4
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 128

C- Multiplying element by element:
   Number of threads used: 16
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 1401

Process returned 0 (0x0)   execution time : 0.003 s
```



```
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 14

B- Multiplying row by row:
   Number of threads used: 4
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 85

C- Multiplying element by element:
   Number of threads used: 16
96  68  69  69
24  56  18  52
58  95  71  92
90  107  81  142
   Execution time(ms): 575

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```