

Name: Salma Saeed Mahmoud Ghareb

ID: 19015779

Matrix Multiplication (Multi-Threading) OS Lab #2

- **How the code is organized:**

- 1) Taking the 2 input matrices from the files that the user enters their names from the terminal.
- 2) The 3 methods are then called to calculate the output matrix and write it in the output file.
- 3) In each method, the time taken is calculated.
- 4) In each method, the threads of each method is created and then joined.

- **main functions:**

- 1) **scanFileToMat** → A function to store the input matrices from their files into a 2D arrays.
- 2) **Read** → A function to read the number of rows and columns from the first or second matrix file according to an integer takes it as an argument, and then call the previous function.
- 3) **method1, method2, method3** → Multiplies the 2 matrices in the normal way, row by row and element by element respectively.
- 4) **thread_method2, thread_method3** → Creates the threads of each method and then joins them.
- 5) **excution1, excution2, excution3** → Calculates the time of each method and printing its output matrix.

- **How to compile and run the code:**

- 1) Open the terminal and type → make
- 2) Type → ./matMultp without arguments or with the input and output files names.

- Sample runs:

```
salma@salma-VirtualBox: ~/Documents/matrix multiplication
make
gcc matMultp.c -o matMultp -pthread
matMultp.c: In function 'read':
matMultp.c:34:36: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
    row = &rowA; col = &colA; mat = &matA;
                           ^
matMultp.c:37:39: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]
    row = &rowB; col = &colB; mat = &matB;
                           ^
matMultp.c:52:25: warning: passing argument 2 of 'scanFileToMat' from incompatible pointer type [-Wincompatible-pointer-types]
    scanFileToMat(file, mat, *row, *col);
                        ^
matMultp.c:24:6: note: expected 'int (*)[1000]' but argument is of type 'int *'
void scanFileToMat(FILE *fp, int mat[1000][1000], int row, int col)
matMultp.c: In function 'method2':
matMultp.c:75:14: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    matC[(int) row][j] = 0;
               ^
matMultp.c:78:18: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    matC[(int) row][j] += matA[(int) row][k]*matB[k][j];
               ^
matMultp.c:78:40: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    matC[(int) row][j] += matA[(int) row][k]*matB[k][j];
               ^
matMultp.c: In function 'thread_method2':
matMultp.c:90:51: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    id = pthread_create(&m2[i], NULL, method2, (void*)i); //i: arg is passed of the start routine "method2"
                                                    ^
matMultp.c:90:42: warning: passing argument 3 of 'pthread_create' from incompatible pointer type [-Wincompatible-pointer-types]
    id = pthread_create(&m2[i], NULL, method2, (void*)i); //i: arg is passed of the start routine "method2"
                                                    ^
```

- With no arguments: the default files are a, b and c

```
salma@salma-VirtualBox:~/Documents/matrix multiplication$ ./matMultp
Method: A thread per matrix
row=2 col=2
15 18
55 68
Number of Threads used: 1
Seconds taken 0
Microseconds taken: 232
-----
Method: A thread per row
row=2 col=2
15 18
55 68
Number of Threads used: 2
Seconds taken 0
Microseconds taken: 652
-----
Method: A thread per element
row=2 col=2
15 18
55 68
Number of Threads used: 4
Seconds taken 0
Microseconds taken: 504
-----
salma@salma-VirtualBox:~/Documents/matrix multiplication$
```

- With arguments:

1) Test1

```
salma@salma-VirtualBox:~/Documents/matrix multiplication$ ./matMultp test1/a test1/b test1/c
Method: A thread per matrix
row=10 col=10
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
Number of Threads used: 1
Seconds taken 0
Microseconds taken: 549
-----
Method: A thread per row
row=10 col=10
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
Number of Threads used: 10
Seconds taken 0
Microseconds taken: 1472
-----
Method: A thread per element
row=10 col=10
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
Number of Threads used: 100
Seconds taken 0
Microseconds taken: 5908
-----
salma@salma-VirtualBox:~/Documents/matrix multiplication$
```

Open ▾



c_per_matrix.txt

~/Documents/matrix multiplication/test1

```
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
```

Open ▾



c_per_row.txt

~/Documents/matrix multiplication/test1

```
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
```

Open ▾




c_per_element.txt

~/Documents/matrix multiplication/test1


```
415 430 445 460 475 490 505 520 535 550
940 980 1020 1060 1100 1140 1180 1220 1260 1300
1465 1530 1595 1660 1725 1790 1855 1920 1985 2050
1990 2080 2170 2260 2350 2440 2530 2620 2710 2800
2515 2630 2745 2860 2975 3090 3205 3320 3435 3550
3040 3180 3320 3460 3600 3740 3880 4020 4160 4300
3565 3730 3895 4060 4225 4390 4555 4720 4885 5050
4090 4280 4470 4660 4850 5040 5230 5420 5610 5800
4615 4830 5045 5260 5475 5690 5905 6120 6335 6550
5140 5380 5620 5860 6100 6340 6580 6820 7060 7300
```

2) Test2


```
salma@salma-VirtualBox:~/Documents/matrix multiplication$ ./matMultp test2/a test2/b test2/c
Method: A thread per matrix
row=3 col=4
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
Number of Threads used: 1
Seconds taken 0
Microseconds taken: 246
-----
Method: A thread per row
row=3 col=4
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
Number of Threads used: 3
Seconds taken 0
Microseconds taken: 569
-----
Method: A thread per element
row=3 col=4
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
Number of Threads used: 12
Seconds taken 0
Microseconds taken: 742
-----
salma@salma-VirtualBox:~/Documents/matrix multiplication$
```

Open  **c_per_matrix.txt**
~/Documents/matrix multiplication/test2

```
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
```

Open  **c_per_row.txt**
~/Documents/matrix multiplication/test2

```
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
```

Open  **c_per_element.txt**
~/Documents/matrix multiplication/test2


```
-1 10 -15 -28
-3 -10 15 -36
5 -2 -9 -20
```

3) Test3


```
salma@salma-VirtualBox:~/Documents/matrix multiplication$ ./matMultp test3/a test3/b test3/c
Method: A thread per matrix
row=5 col=4
175 190 205 220
400 440 480 520
625 690 755 820
850 940 1030 1120
1075 1190 1305 1420
Number of Threads used: 1
Seconds taken 0
Microseconds taken: 324
-----
Method: A thread per row
row=5 col=4
175 190 205 220
400 440 480 520
625 690 755 820
850 940 1030 1120
1075 1190 1305 1420
Number of Threads used: 5
Seconds taken 0
Microseconds taken: 821
-----
```

```
Method: A thread per element
row=5 col=4
175 190 205 220
400 440 480 520
625 690 755 820
850 940 1030 1120
1075 1190 1305 1420
Number of Threads used: 20
Seconds taken 0
Microseconds taken: 1489
-----
```

```
salma@salma-VirtualBox:~/Documents/matrix multiplication$
```

Open  **c_per_matrix.txt**
~/Documents/matrix multiplication/test3

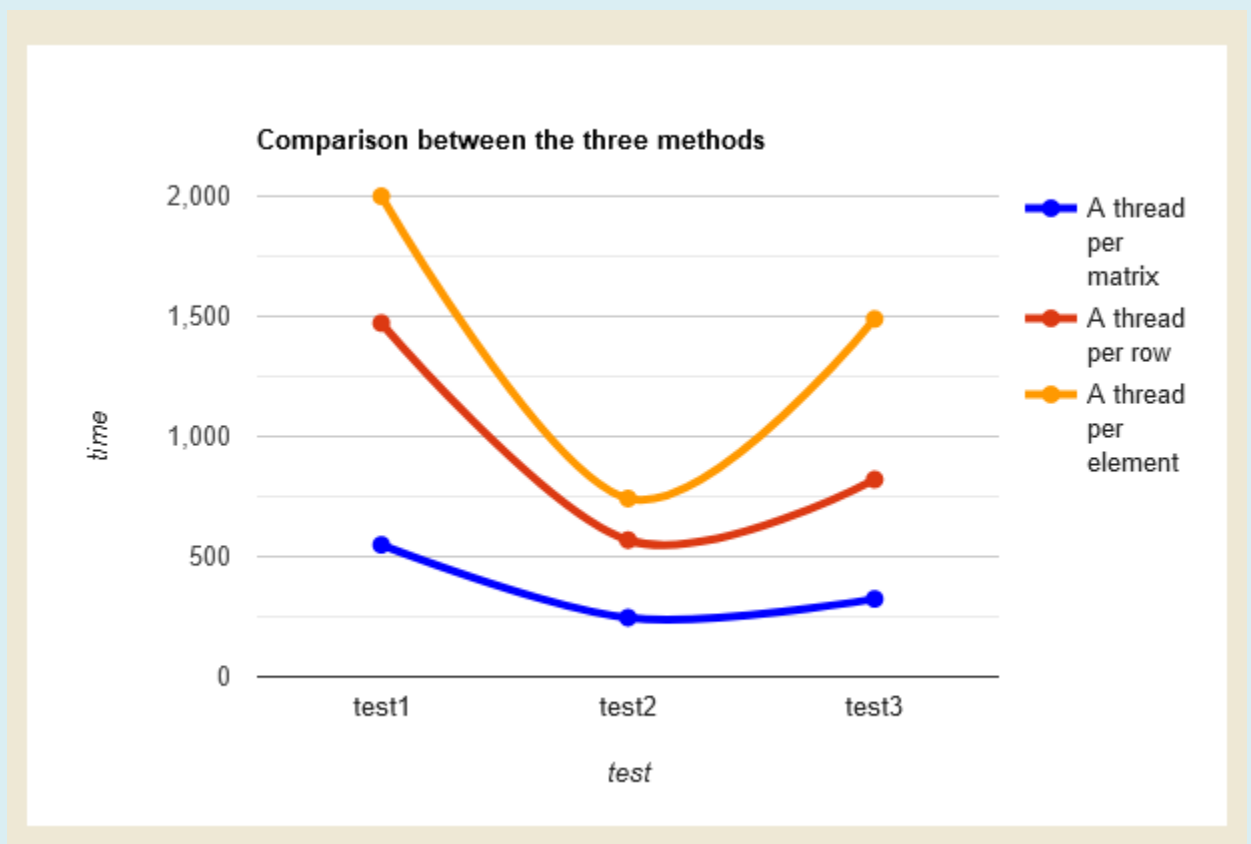
```
175 190 205 220
400 440 480 520
625 690 755 820
850 940 1030 1120
1075 1190 1305 1420
```

Open  **c_per_row.txt**
~/Documents/matrix multiplication/test3

```
175 190 205 220
400 440 480 520
625 690 755 820
850 940 1030 1120
1075 1190 1305 1420
```

```
Open [icon] c_per_element.txt  
~/Documents/matrix multiplication/test3  
175 190 205 220  
400 440 480 520  
625 690 755 820  
850 940 1030 1120  
1075 1190 1305 1420
```

- **A comparison between the three methods of matrix multiplication:**



- The first method (a thread per matrix) is performing better than the second (a thread per row) and the third (a thread per element).
- The idea is, creating and handling threads requires extra overhead and lots of computation, so, it's not always the ideal solution to go for multi-threading, and there's always a tradeoff.

- **Video:**

https://drive.google.com/file/d/1xmQyvpHSnjrhLrfLA_eeVCoOzvDETaMo/view?usp=sharing