Lab 2 report

Matrix multiplication by multi-threading

Name: Ali Hassan Ali Ahmed ELSharawy

ID: 19016013

- ## How code is organized:

  We get size of 2 matrices and check col1 == row2 then we execute 3 functions sequentially and print their threads and time in console.

  In first function for matrix multiplication we use 1 thread and calculate in resMatrix 1 then print it in file.

  In second function for raw multiplication we make row1 threads array and each thread calculate row in resMatrix 2 then we join these threads and print it in file.

  In third function for element multiplication we make row1 * col2 threads matrix and each thread calculate an element in resMatrix 3 then we join these threads and print in file.

- ## Main functions code:

  In main functions we read input from user and check valid input and execute 3 functions and calculate executed time for each function.

  In matrix thread function we calculate resMatrix 1 and print it to file.

  In row thread function we create thread for each row and put arguments (row index, col1, col2) in pointer to argument struct and allocate it in memory and each thread execute raw thread fill function to calculate its row in resMatrix 2 then we join all these threads and print to file.

In row thread fill function we calculate row[row index] of resMatrix2 then we deallocate/free pointer to struct.

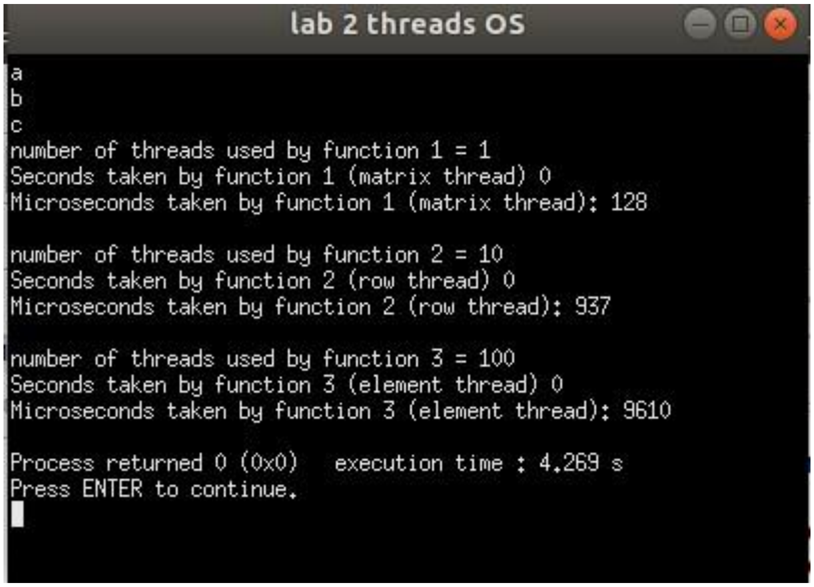
In element thread function we create threads for each element and put arguments (row index, col index, col1, col2) in pointer to argument struct and allocate it in memory and each thread execute element thread fill function to calculate its element in resMatrix 3 then we join all these threads and print to file.

In element thread fill function we calculate element [row index] [col index] of resMatrix3 then we deallocate/free pointer to struct.

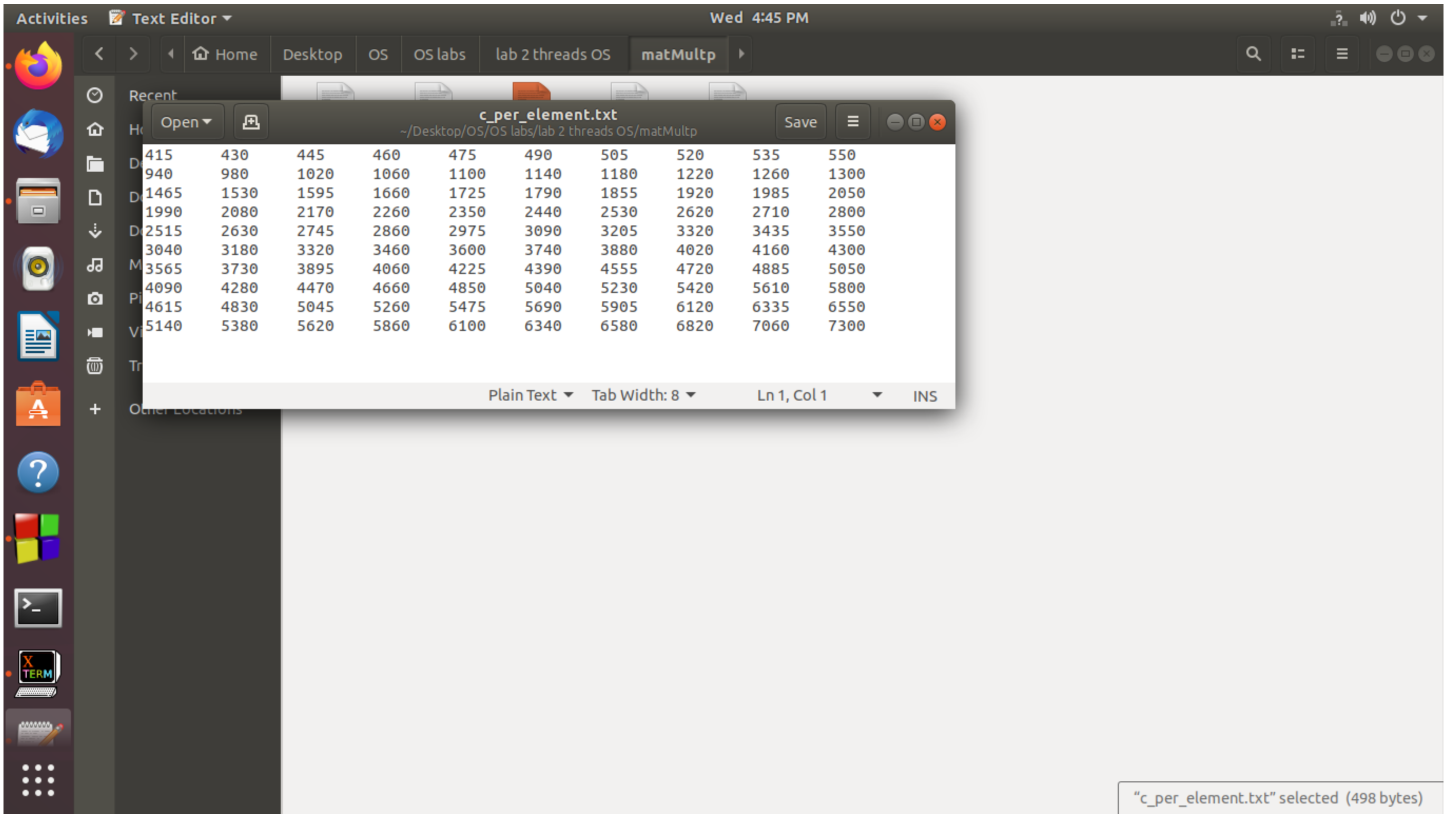
- How to compile and run your code:

1) Make 2 text files for each matrix 1 and matrix 2
2) Open terminal at lab folder
3) make command to compile project
4) ./matMultp file1_name file2_name outFile_name
5) 3 text files with name entered will be created each has function name used
6) The input text files must be in lab folder and exist with same name entered
7) No arguments will have default name a , b , c for files respectively Else the first 3 arguments only will be taken.

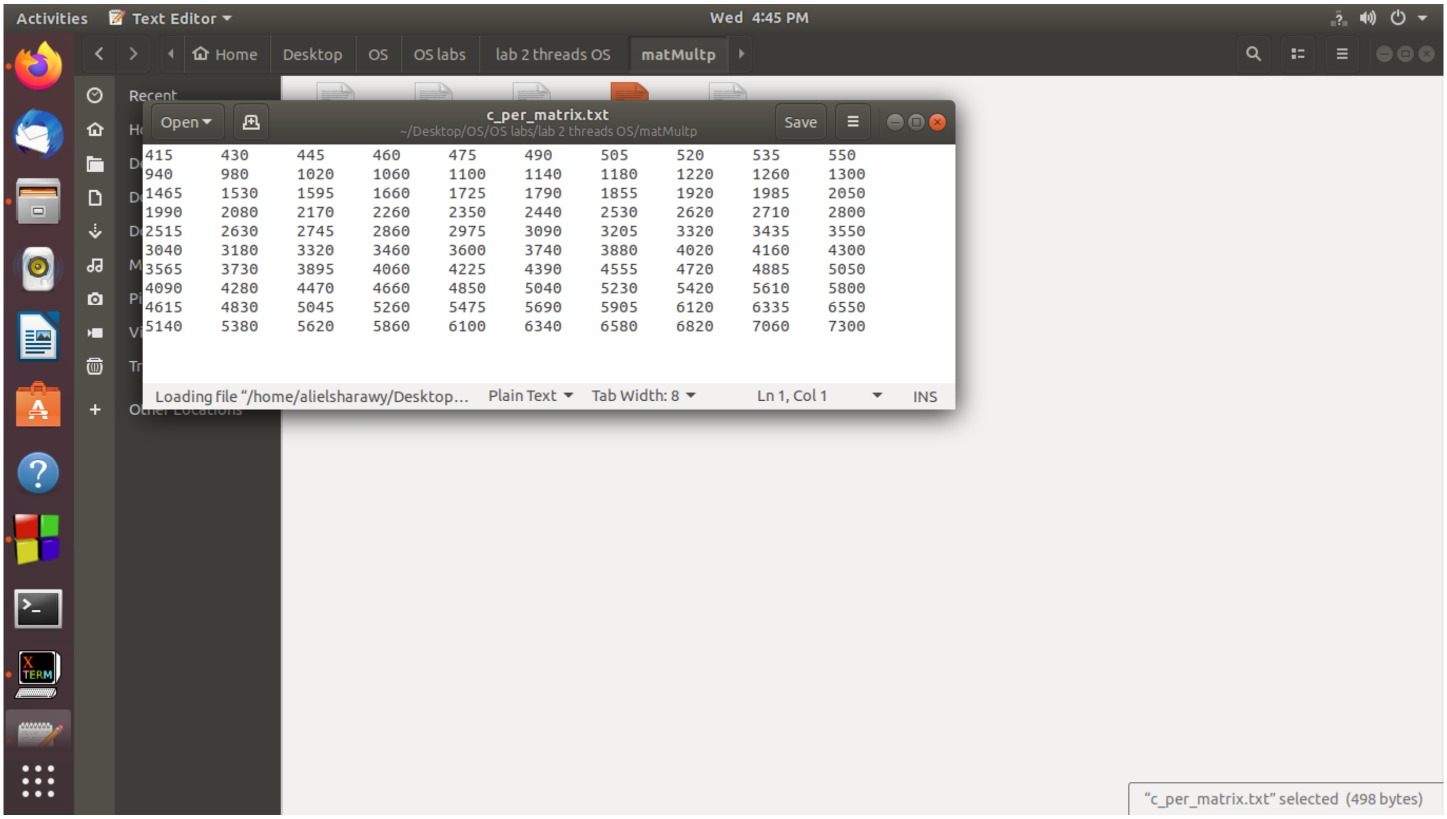
- Sample Runs:



```
a
b
c
number of threads used by function 1 = 1
Seconds taken by function 1 (matrix thread) 0
Microseconds taken by function 1 (matrix thread): 128

number of threads used by function 2 = 10
Seconds taken by function 2 (row thread) 0
Microseconds taken by function 2 (row thread): 937

number of threads used by function 3 = 100
Seconds taken by function 3 (element thread) 0
Microseconds taken by function 3 (element thread): 9610

Process returned 0 (0x0)   execution time : 4.269 s
Press ENTER to continue.
```



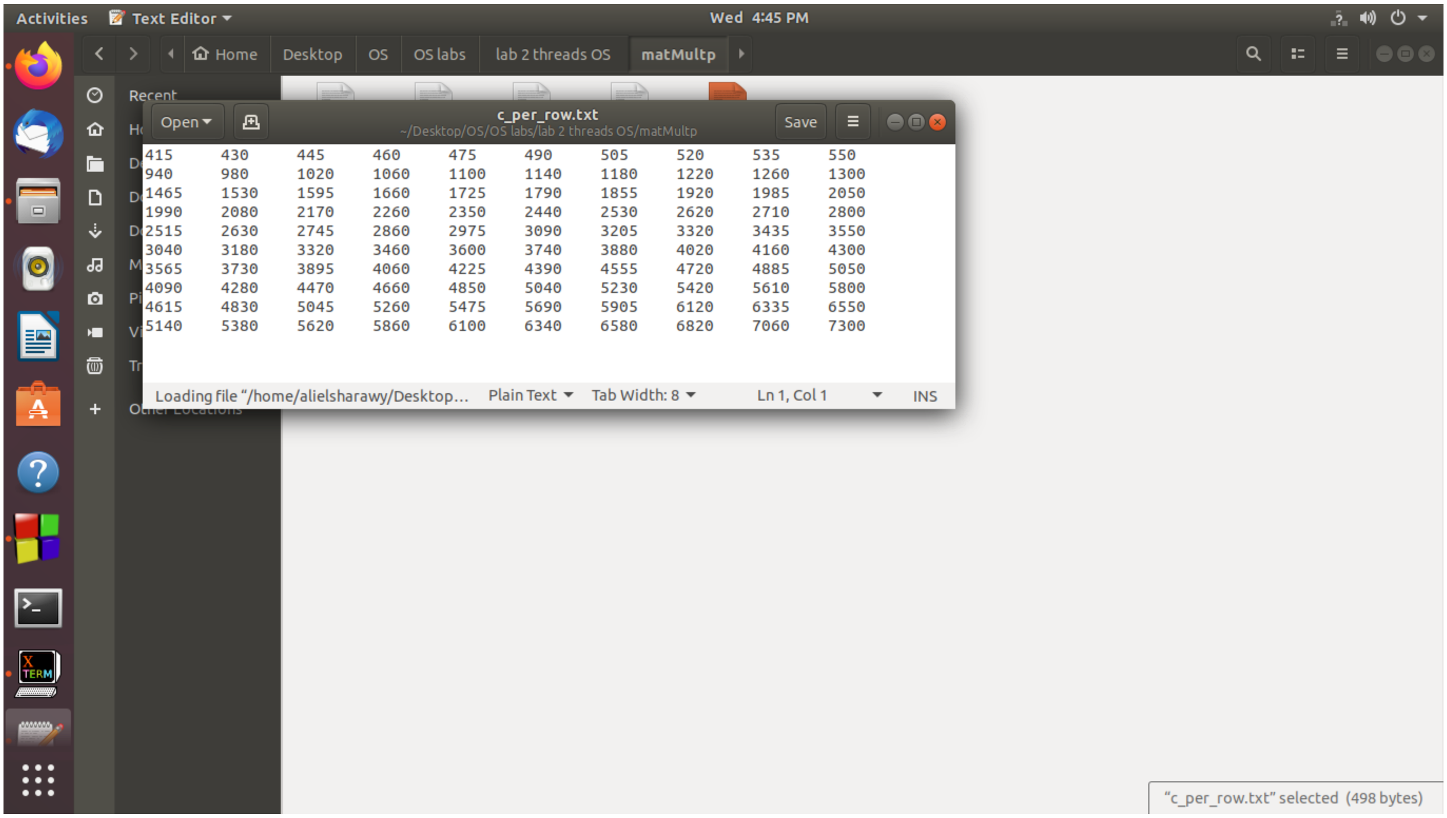
```
415     430     445     460     475     490     505     520     535     550
940     980     1020    1060    1100    1140    1180    1220    1260    1300
1465    1530    1595    1660    1725    1790    1855    1920    1985    2050
1990    2080    2170    2260    2350    2440    2530    2620    2710    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
```



```
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
```

Loading file "/home/alielsharawy/Desktop…

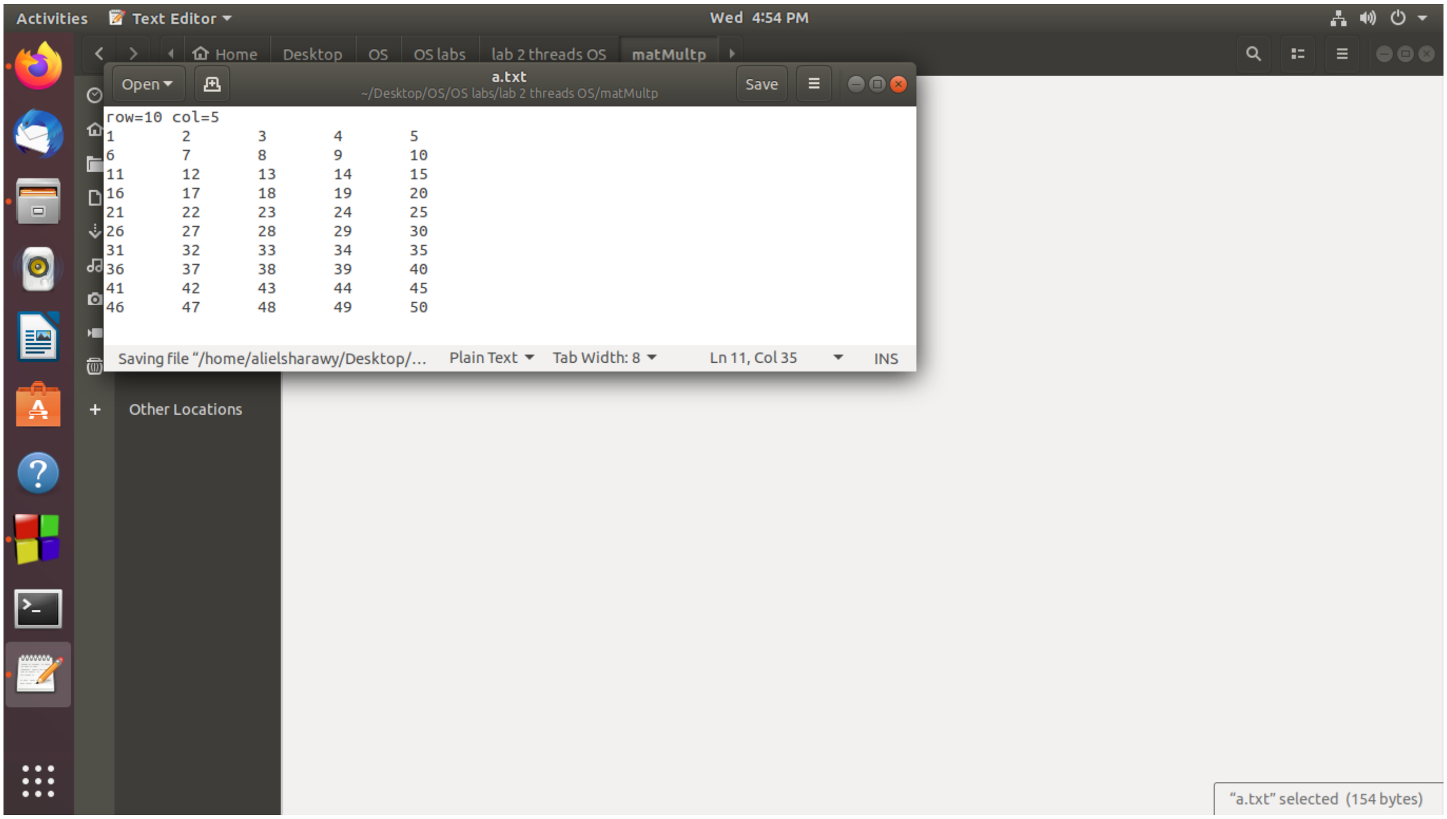
"c_per_matrix.txt" selected (498 bytes)



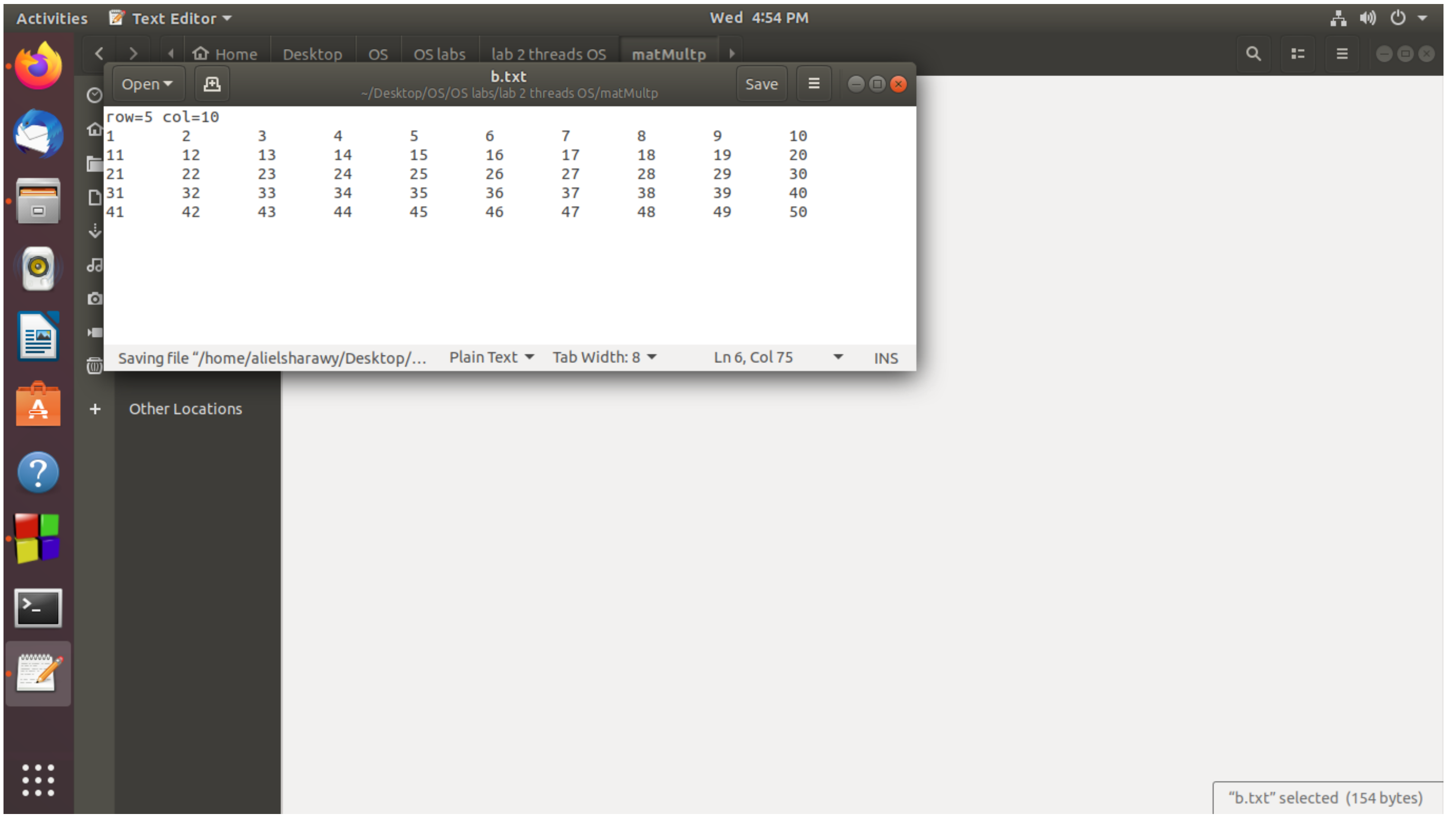
```
415     430     445     460     475     490     505     520     535     550
940     980     1020    1060    1100    1140    1180    1220    1260    1300
1465    1530    1595    1660    1725    1790    1855    1920    1985    2050
1990    2080    2170    2260    2350    2440    2530    2620    2710    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
```

Loading file "/home/alielsharawy/Desktop…

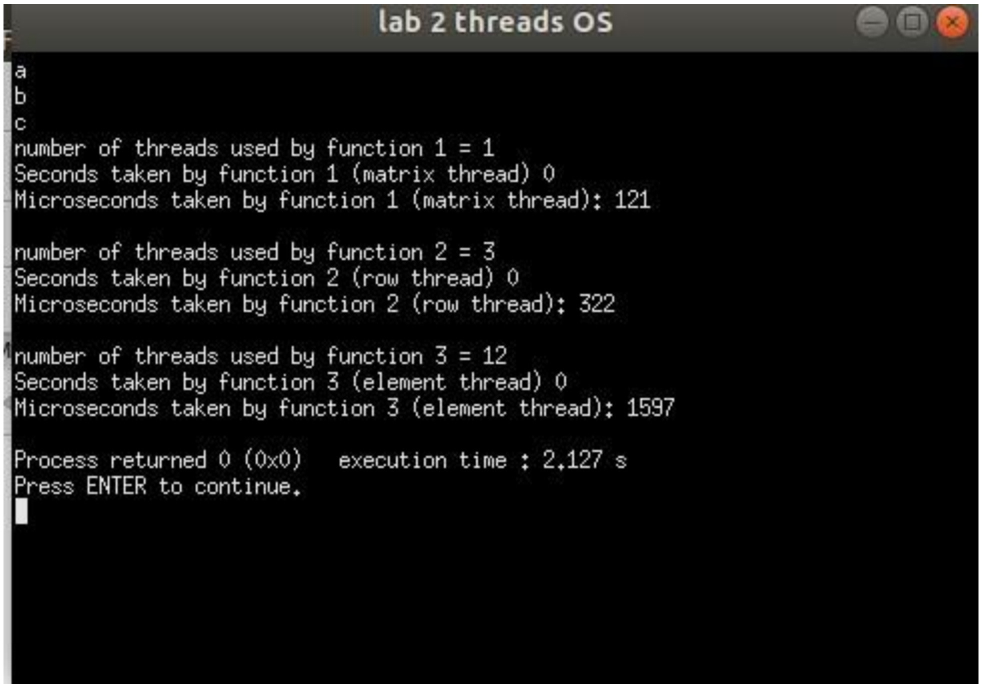
"c_per_row.txt" selected (498 bytes)



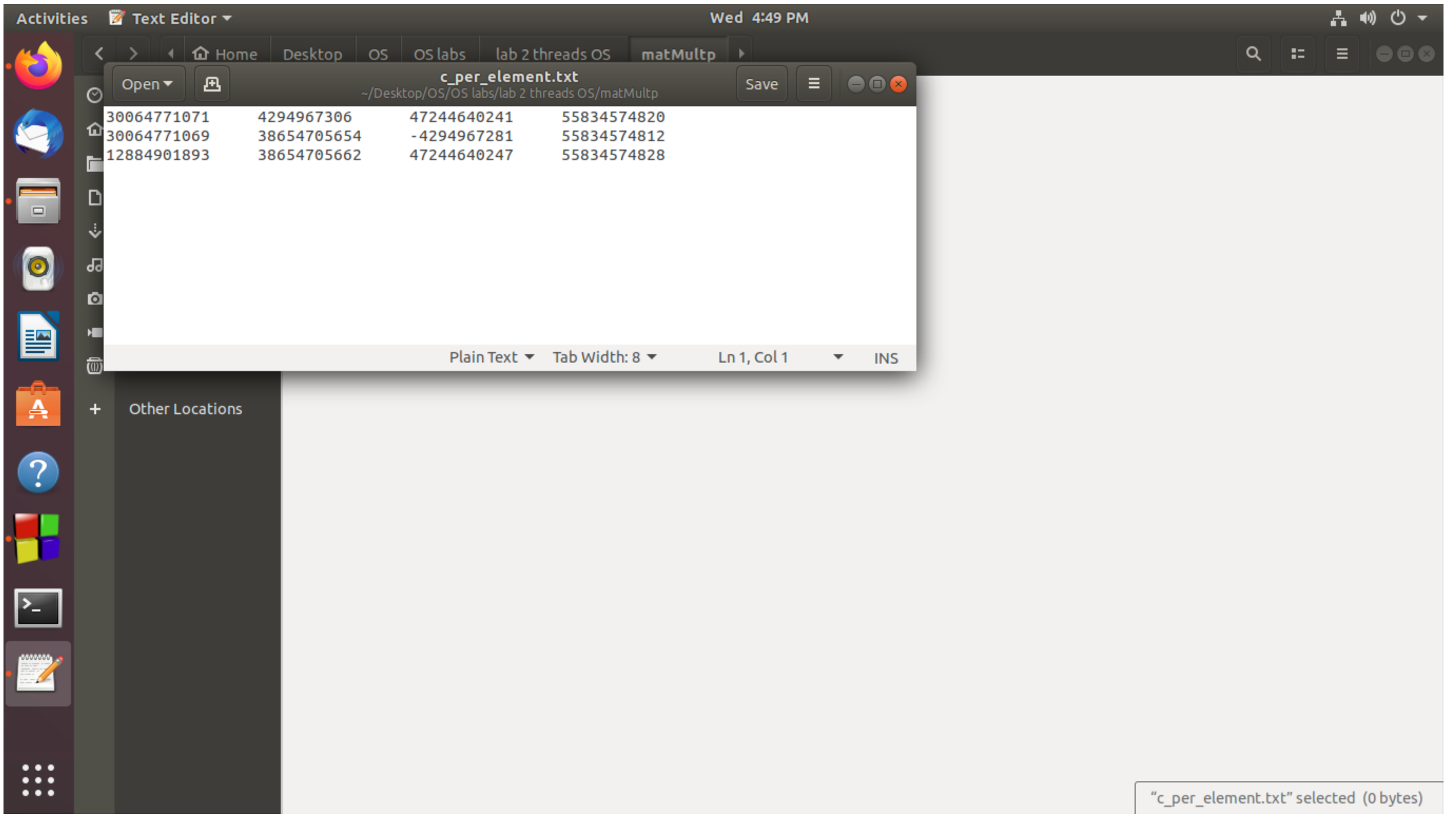
```
row=10 col=5
1       2       3       4       5
6       7       8       9       10
11      12      13      14      15
16      17      18      19      20
21      22      23      24      25
26      27      28      29      30
31      32      33      34      35
36      37      38      39      40
41      42      43      44      45
46      47      48      49      50
```



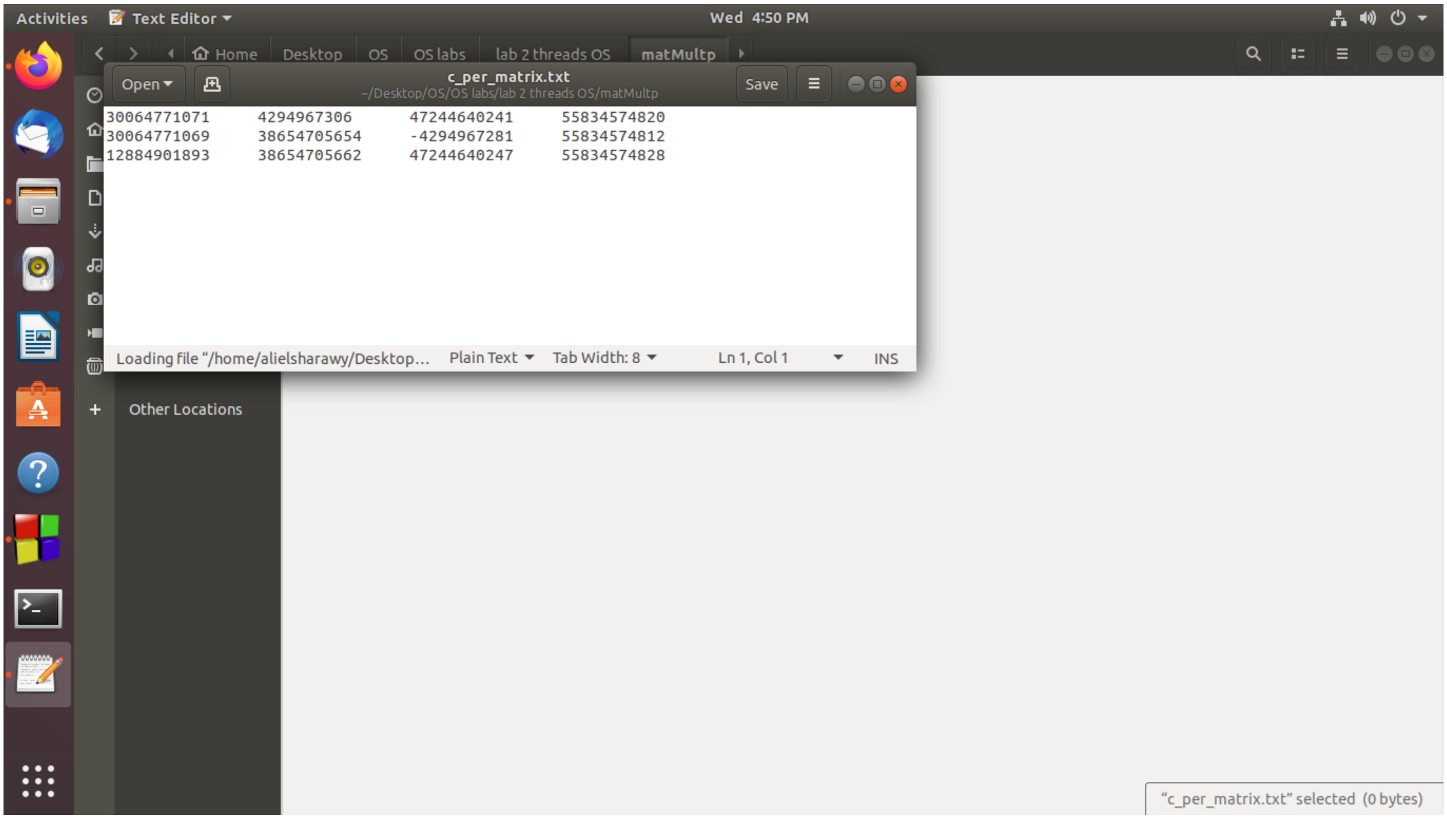
```
row=5 col=10
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
21      22      23      24      25      26      27      28      29      30
31      32      33      34      35      36      37      38      39      40
41      42      43      44      45      46      47      48      49      50
```



```
a
b
c
number of threads used by function 1 = 1
Seconds taken by function 1 (matrix thread) 0
Microseconds taken by function 1 (matrix thread): 121

number of threads used by function 2 = 3
Seconds taken by function 2 (row thread) 0
Microseconds taken by function 2 (row thread): 322

number of threads used by function 3 = 12
Seconds taken by function 3 (element thread) 0
Microseconds taken by function 3 (element thread): 1597

Process returned 0 (0x0)   execution time : 2.127 s
Press ENTER to continue.
```



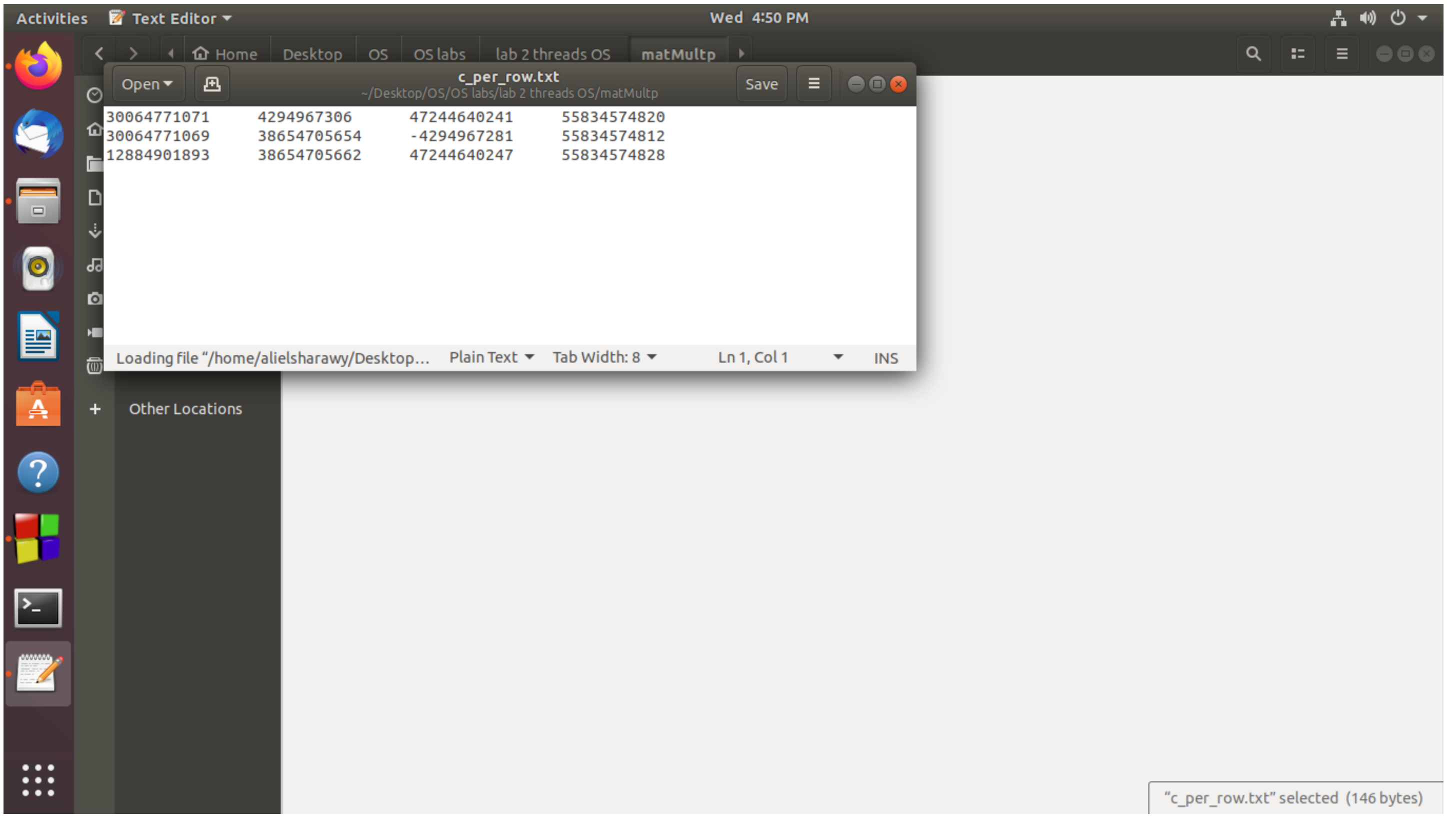
```
30064771071	4294967306	47244640241	55834574820
30064771069	38654705654	-4294967281	55834574812
12884901893	38654705662	47244640247	55834574828
```



```
30064771071     4294967306      47244640241     55834574820
30064771069     38654705654     -4294967281     55834574812
12884901893     38654705662     47244640247     55834574828
```

Loading file "/home/alielsharawy/Desktop…

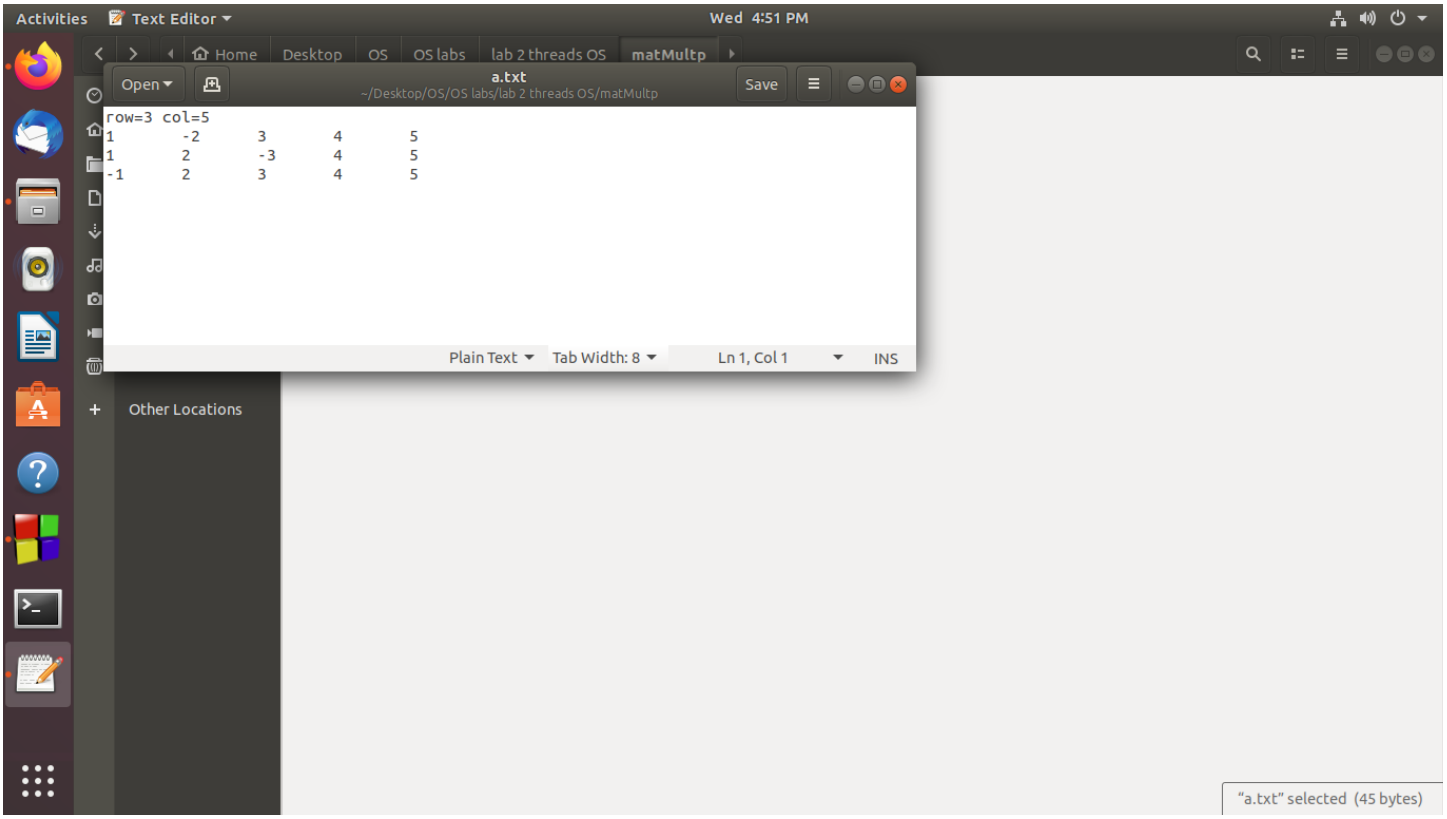
"c_per_matrix.txt" selected (0 bytes)



```
30064771071     4294967306      47244640241     55834574820
30064771069     38654705654     -4294967281     55834574812
12884901893     38654705662     47244640247     55834574828
```

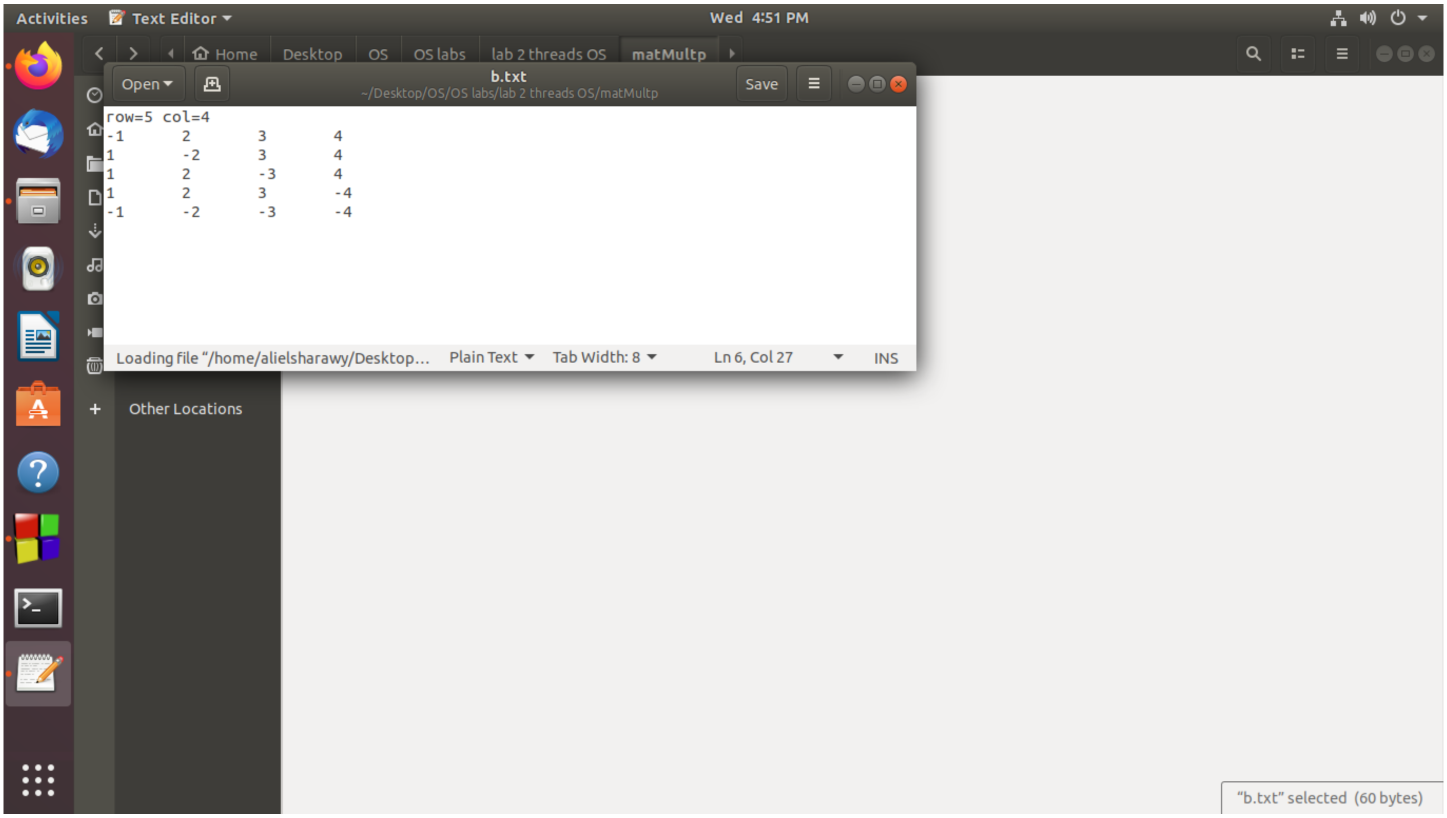
Loading file "/home/alielsharawy/Desktop…

"c_per_row.txt" selected (146 bytes)



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

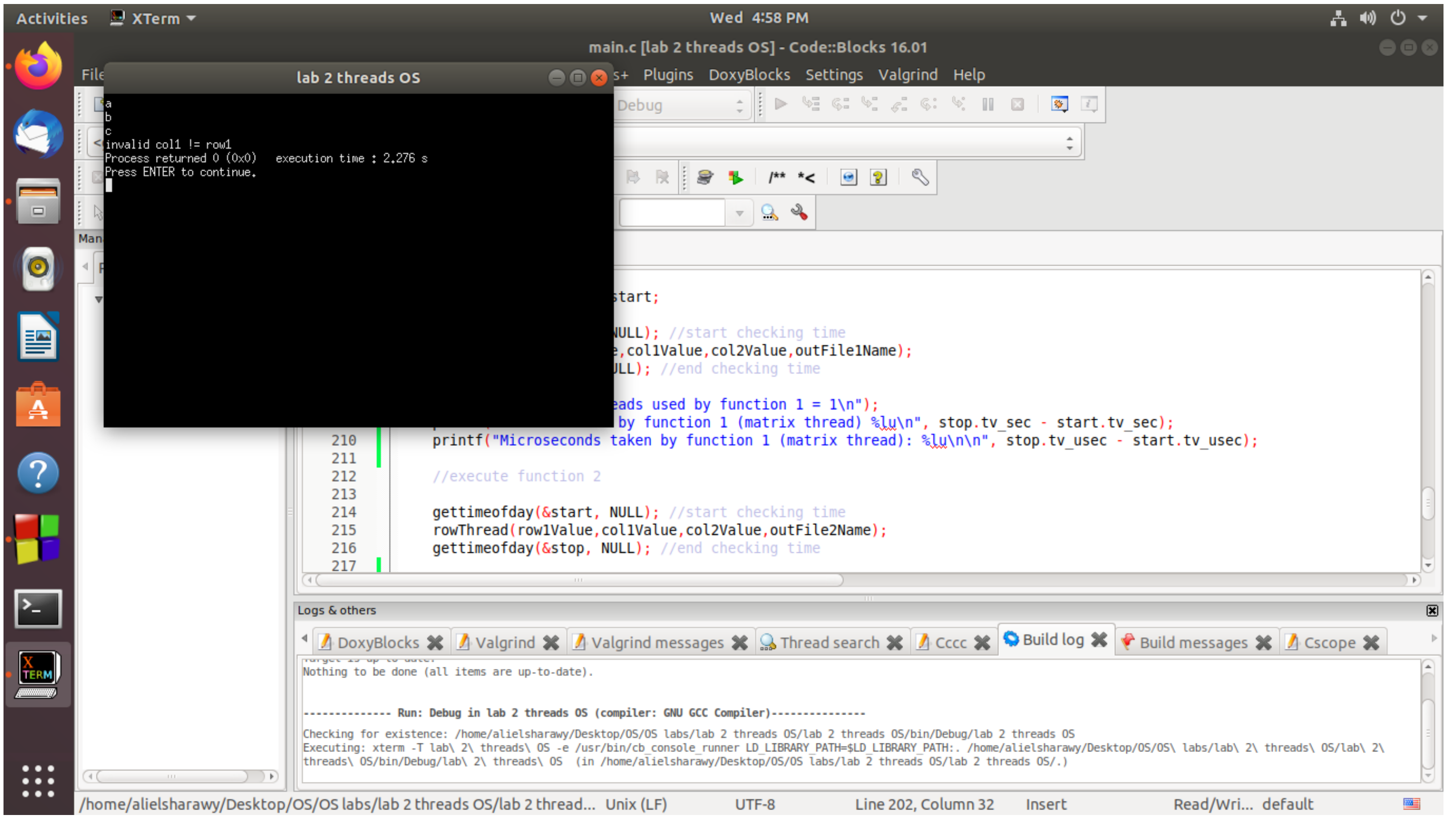
"a.txt" selected (45 bytes)



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

Loading file "/home/alielsharawy/Desktop…

"b.txt" selected (60 bytes)



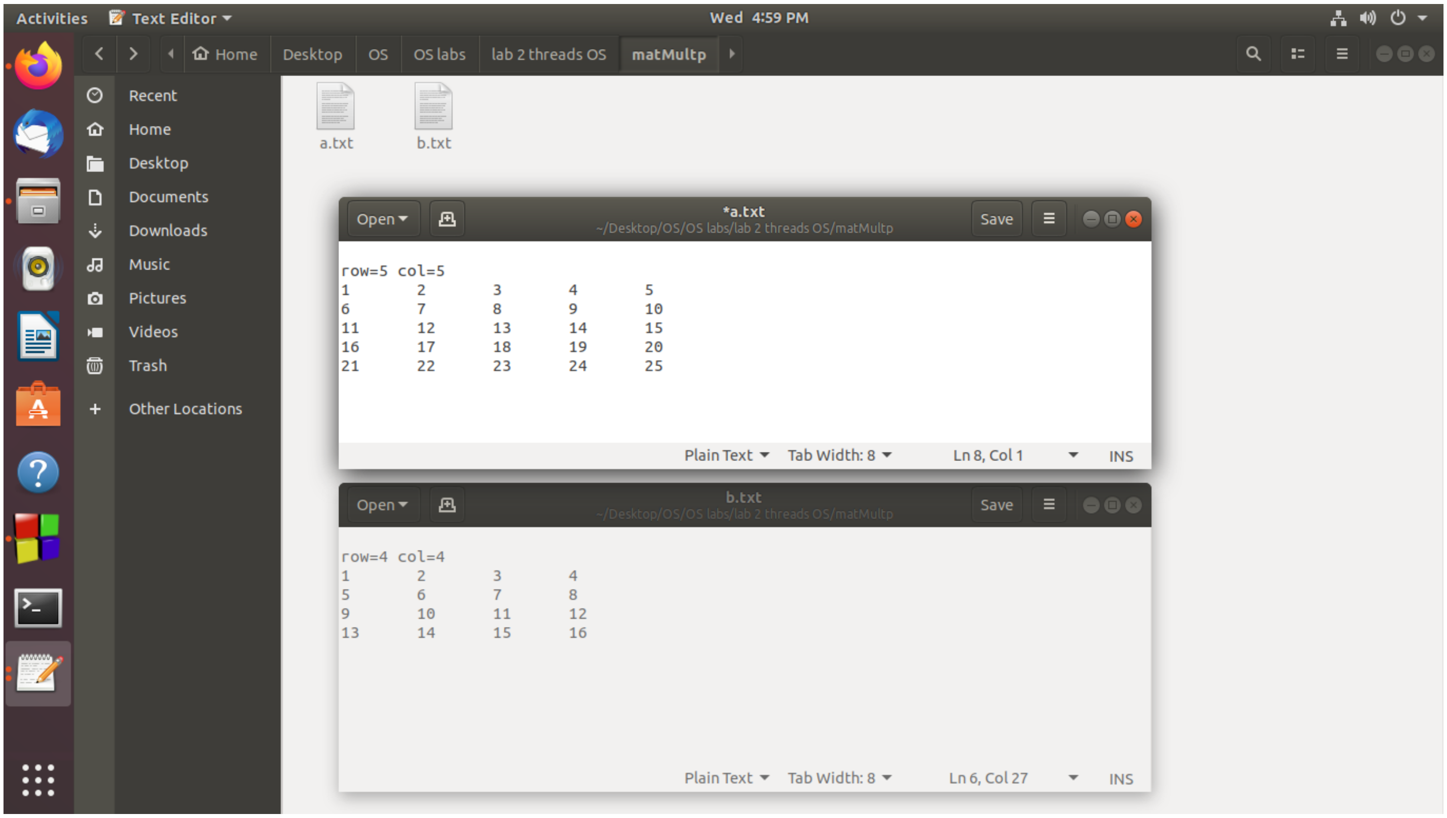
lab 2 threads OS

```
a
b
c
invalid col1 != row1
Process returned 0 (0x0)   execution time : 2.276 s
Press ENTER to continue.
```

```
start;

NULL); //start checking time
e,col1Value,col2Value,outFile1Name);
ULL); //end checking time

eads used by function 1 = 1\n");
 by function 1 (matrix thread) %lu\n", stop.tv_sec - start.tv_sec);
    printf("Microseconds taken by function 1 (matrix thread): %lu\n\n", stop.tv_usec - start.tv_usec);

    //execute function 2

    gettimeofday(&start, NULL); //start checking time
    rowThread(row1Value,col1Value,col2Value,outFile2Name);
    gettimeofday(&stop, NULL); //end checking time
```

```
Nothing to be done (all items are up-to-date).


-------------- Run: Debug in lab 2 threads OS (compiler: GNU GCC Compiler)---------------

Checking for existence: /home/alielsharawy/Desktop/OS/OS labs/lab 2 threads OS/lab 2 threads OS/bin/Debug/lab 2 threads OS
Executing: xterm -T lab\ 2\ threads\ OS -e /usr/bin/cb_console_runner LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. /home/alielsharawy/Desktop/OS/OS\ labs/lab\ 2\ threads\ OS/lab\ 2\
threads\ OS/bin/Debug/lab\ 2\ threads\ OS  (in /home/alielsharawy/Desktop/OS/OS labs/lab 2 threads OS/lab 2 threads OS/.)
```

/home/alielsharawy/Desktop/OS/OS labs/lab 2 threads OS/lab 2 thread... Unix (LF) UTF-8 Line 202, Column 32 Insert Read/Wri... default



*a.txt
~/Desktop/OS/OS labs/lab 2 threads OS/matMultp

```
row=5 col=5
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
```

b.txt
~/Desktop/OS/OS labs/lab 2 threads OS/matMultp

```
row=4 col=4
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
```

- Comparison between the three methods:

First method:

Fastest method as it has no overhead to create threads

Second method:

Slower than first method as it has overhead to create row1 threads

Third method:

Slowest method as it has overhead to create row1*col2 threads

As execution time of matrix multiplication is not very large so overhead to create threads is equivalent to matrix multiplication.