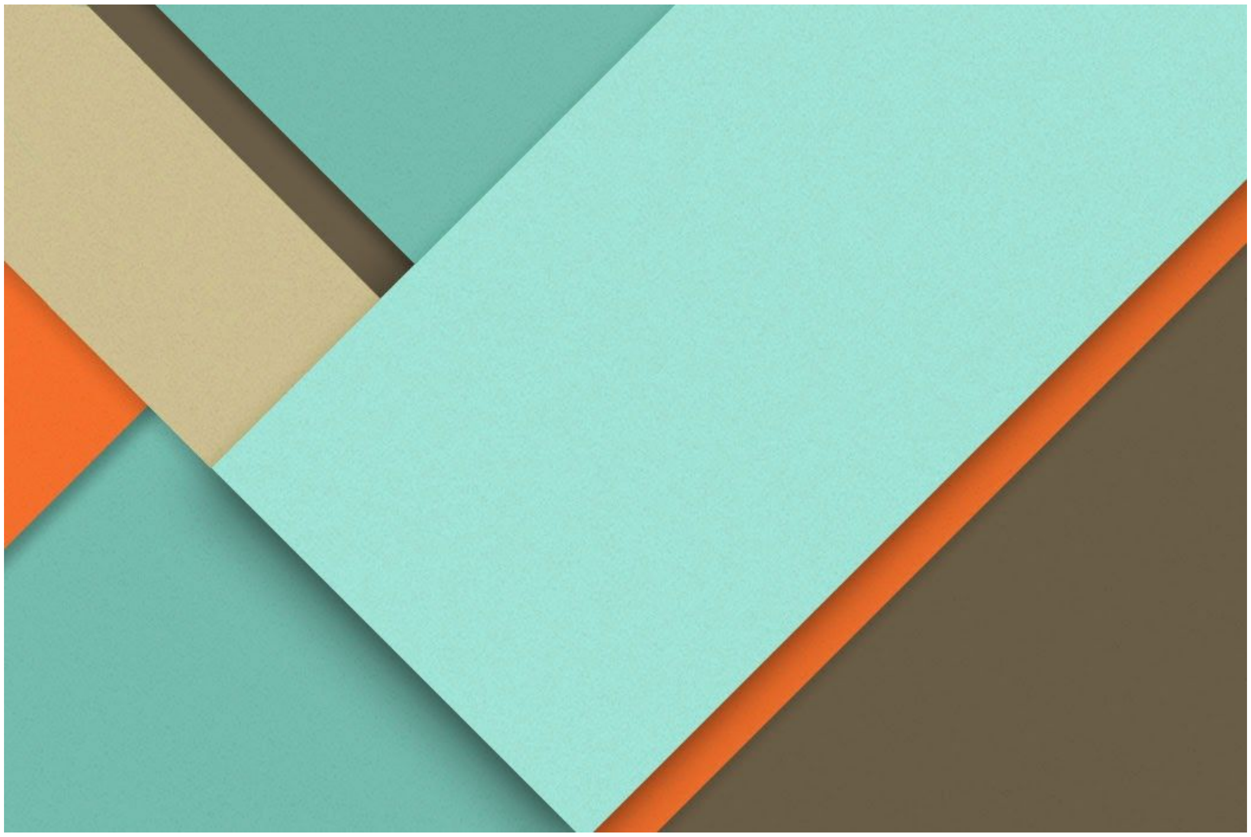


Threads

Ubuntu operating system for development..

Afnan Mousa Mabrouk
ID : 15

Overview

You are required to implement a multithreaded matrix multiplication program. The input to the program is two matrices A(x*y) and B(y*z) that are read from corresponding files. The output is a matrix C(x*z) that is written to an output file.

Goals

- ➢ The program must support the following instructions :
  - Implement the multithreaded matrix multiplication using both methods :
    - ❑ A thread computes each row in the output matrix.
    - ❑ A thread computes each element in the output matrix.
  - Compare the two implementations according to the following:
    - ❑ the number of thread created
    - ❑ the execution time taken.
  - The program handles any errors and terminates gracefully.

Overall organization :

- ➢ The code is separated into two header files and the main class.
- ➢ The follow of the code is to take the input names of the files as arguments from the terminal , if the user doesn't enter any argument so by default the program works on the **A.txt ,B.txt** as matrix one & matrix two ,after opening the files by **ReadFromFile function** which located in the **ReadInput Headerfile** ,the program allocate space in memory to store data of files as **Structure matrix** which contain the number of rows ,columns,data of each matrix .
- ➢ The operation of store data occurs depending on the split data of files in **Split header file**.
- ➢ After storing the data of the matrix in structure the **main function** is called **ControlRows function** which creates the threads according to the concept to calculate each row in the output matrix,then the threads creation call **myThreadFunForRow function** to calculate the Output and store it in **Global struct** .
- ➢ The last step is repeated to calculate each element of the output by calling the **ControlElements function** which calls **myThreadFunForElement function.**
- ➢ Then call **WriteINOutputFile function** to write the output matrix in the **output file**.

- ➢ call **WriteTime function** to write the time which threads take in the two cases in the **output file**.
- ➢ The program terminated.

Major functions:

The major functions :

- ➢ **Control Rows function :**
  - The mainly job of this function to create the threads depend on the number of **Rows of Matrix1** ,by calling the **myThreadFunForRow function** and pass the structure of structure as argument :
    - ❏ This Main struct contain:
      1. Struct of Matrix 1.
      2. Struct of matrix 2.
      3. Int x which id .
      4. Int column to know the index of column to use in **myThreadFunForElement function .**
  - Contain also functions as a **join of the threads**.

- ➢ **Control Elements function :**
  - The mainly job of this function to create the threads depend on the number of **Rows of Matrix1 *Column of Matrix 2** ,by calling the **myThreadFunForElement function** and passing the structure of structure as argument .
  - Contain also functions as a **join of the threads**.

- ➢ **myThreadFunForElement function :**
  - Contain one for loop to calculate the one element of the output matrix depending on the index of Rows of Matrix1 and the index of Column of Matrix 2 which is stored in struct of struct.

- ➢ **myThreadFunForRow function :**
  - Contain two for loop to calculate the total element of the one row in the output matrix depending on the index of Rows of Matrix1 which equal the id thread .

- ➢ **SetValueOfMatrix function :**

  Read from the input file and store the data in the **struct matrix**.

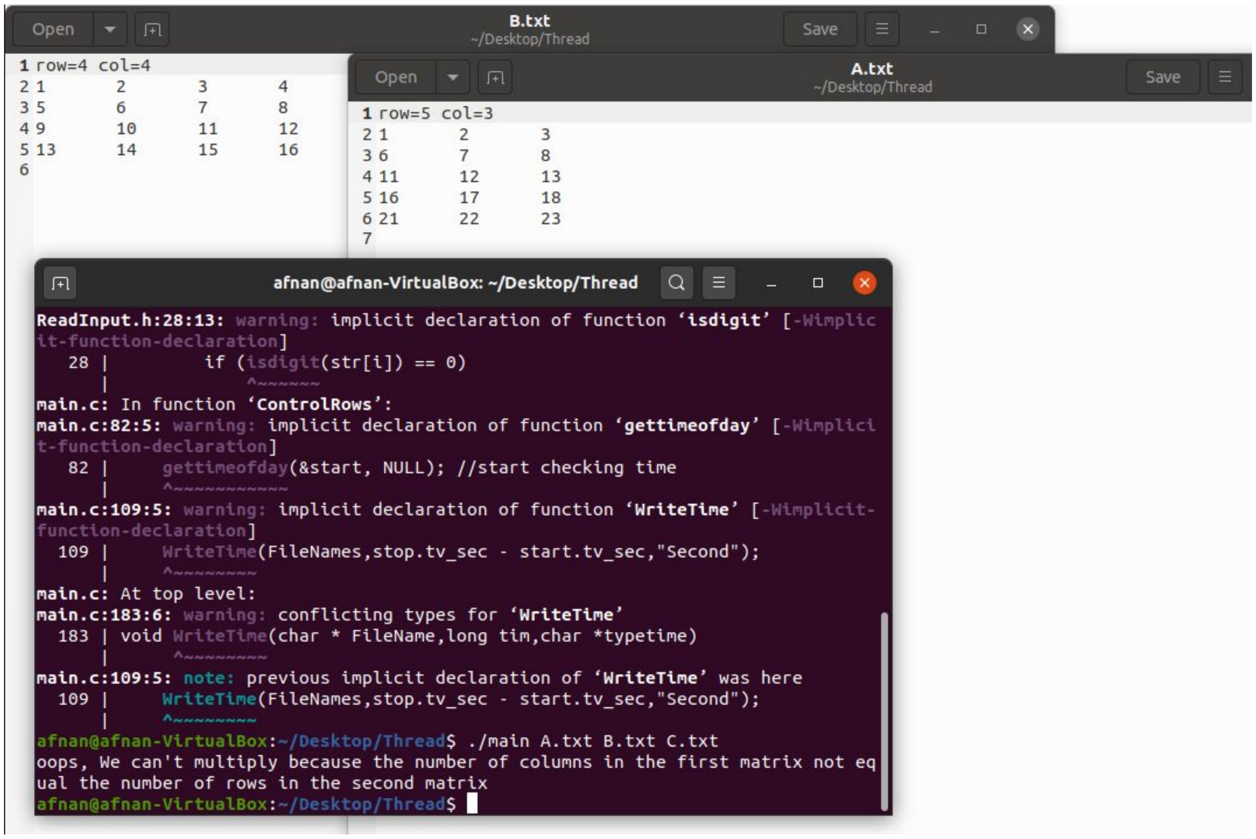
How compile the code :

- ➢ The program runs from terminal or in the console .
- ➢ By terminal throw pass the Names of files .

Sample runs :

I. Sample runs when the user enters two matrices which the number of rows of the first matrix not equal the number of columns of the second matrix .

- ➢ The program prints an error message and terminates.



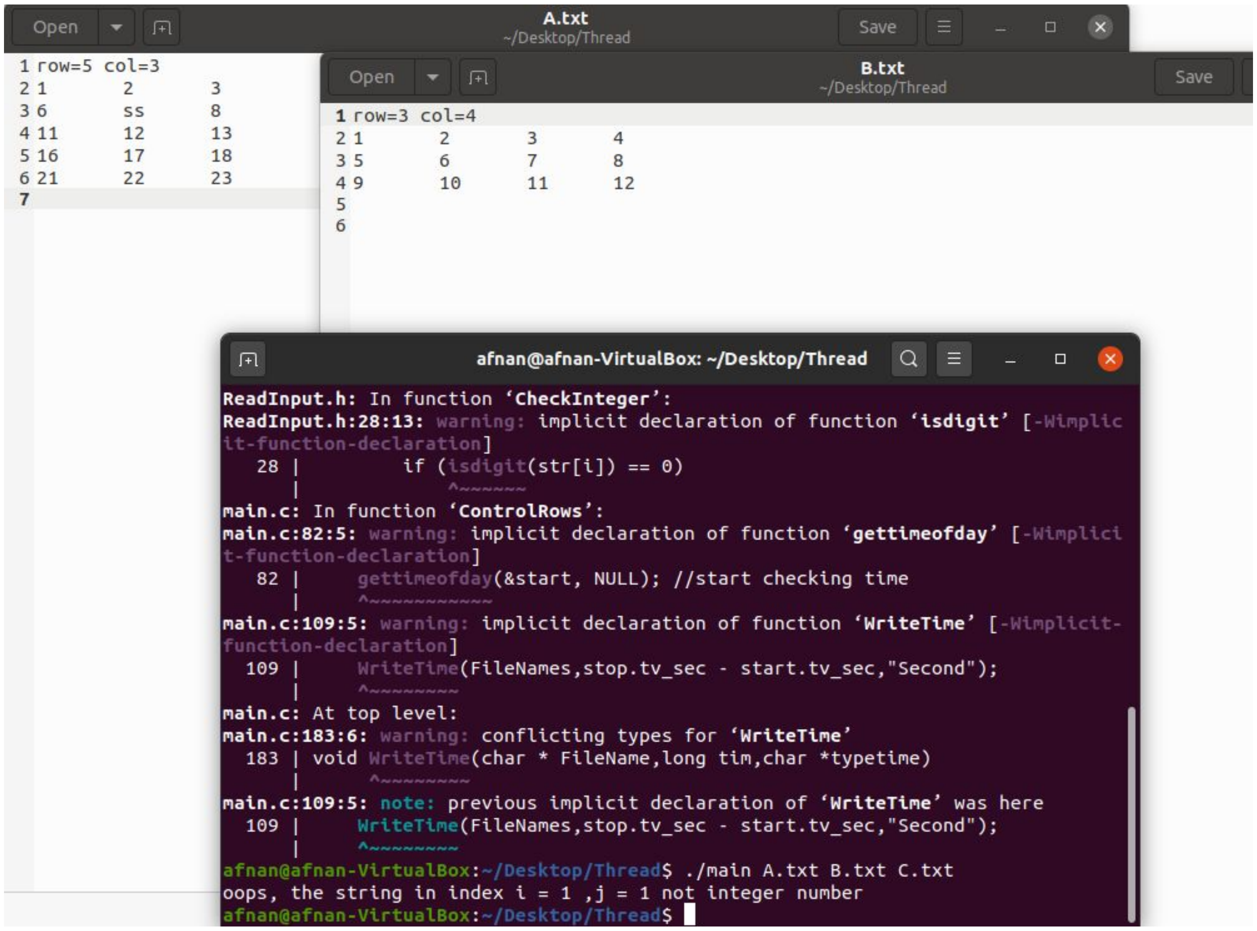
```
row=4 col=4
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      16
```

```
row=5 col=3
1       2       3
6       7       8
11      12      13
16      17      18
21      22      23
```

```
ReadInput.h:28:13: warning: implicit declaration of function ‘isdigit’ [-Wimplic
it-function-declaration]
   28 |          if (isdigit(str[i]) == 0)
      |              ^~~~~~~
main.c: In function ‘ControlRows’:
main.c:82:5: warning: implicit declaration of function ‘gettimeofday’ [-Wimplici
t-function-declaration]
   82 |     gettimeofday(&start, NULL); //start checking time
      |     ^~~~~~~~~~~~
main.c:109:5: warning: implicit declaration of function ‘WriteTime’ [-Wimplicit-
function-declaration]
  109 |     WriteTime(FileNames,stop.tv_sec - start.tv_sec,"Second");
      |     ^~~~~~~~~
main.c: At top level:
main.c:183:6: warning: conflicting types for ‘WriteTime’
  183 | void WriteTime(char * FileName,long tim,char *typetime)
      |      ^~~~~~~~~
main.c:109:5: note: previous implicit declaration of ‘WriteTime’ was here
  109 |     WriteTime(FileNames,stop.tv_sec - start.tv_sec,"Second");
      |     ^~~~~~~~~
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main A.txt B.txt C.txt
oops, We can't multiply because the number of columns in the first matrix not eq
ual the number of rows in the second matrix
afnan@afnan-VirtualBox:~/Desktop/Thread$
```

II. Sample runs when the user enters string in any element of the matrices :

➢ The program prints an error message and terminates.



A.txt
~/Desktop/Thread

```
row=5 col=3
1         2         3
6         ss        8
11        12        13
16        17        18
21        22        23
```

B.txt
~/Desktop/Thread

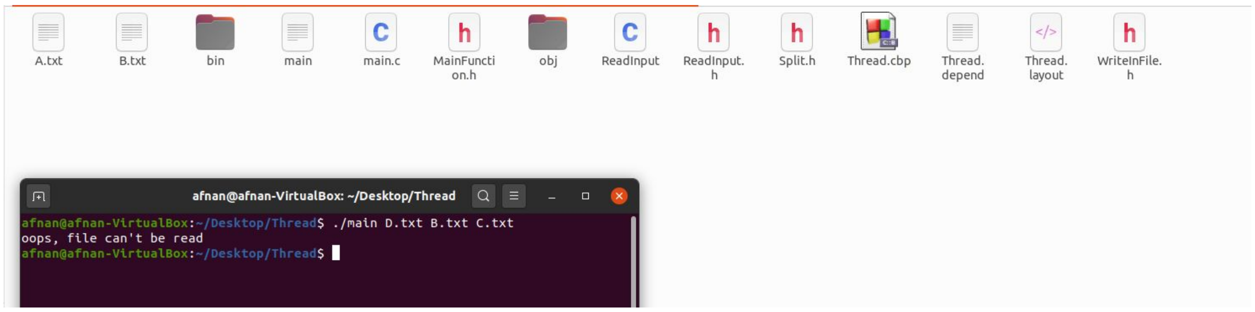
```
row=3 col=4
1         2         3         4
5         6         7         8
9         10        11        12
```

afnan@afnan-VirtualBox: ~/Desktop/Thread

```
ReadInput.h: In function ‘CheckInteger’:
ReadInput.h:28:13: warning: implicit declaration of function ‘isdigit’ [-Wimplic
it-function-declaration]
   28 |         if (isdigit(str[i]) == 0)
      |             ^~~~~~~
main.c: In function ‘ControlRows’:
main.c:82:5: warning: implicit declaration of function ‘gettimeofday’ [-Wimplici
t-function-declaration]
   82 |     gettimeofday(&start, NULL); //start checking time
      |     ^~~~~~~~~~~~
main.c:109:5: warning: implicit declaration of function ‘WriteTime’ [-Wimplicit-
function-declaration]
  109 |     WriteTime(FileNames,stop.tv_sec - start.tv_sec,"Second");
      |     ^~~~~~~~~
main.c: At top level:
main.c:183:6: warning: conflicting types for ‘WriteTime’
  183 | void WriteTime(char * FileName,long tim,char *typetime)
      |      ^~~~~~~~~
main.c:109:5: note: previous implicit declaration of ‘WriteTime’ was here
  109 |     WriteTime(FileNames,stop.tv_sec - start.tv_sec,"Second");
      |     ^~~~~~~~~
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main A.txt B.txt C.txt
oops, the string in index i = 1 ,j = 1 not integer number
afnan@afnan-VirtualBox:~/Desktop/Thread$ 
```

III. when the user enters files not created :

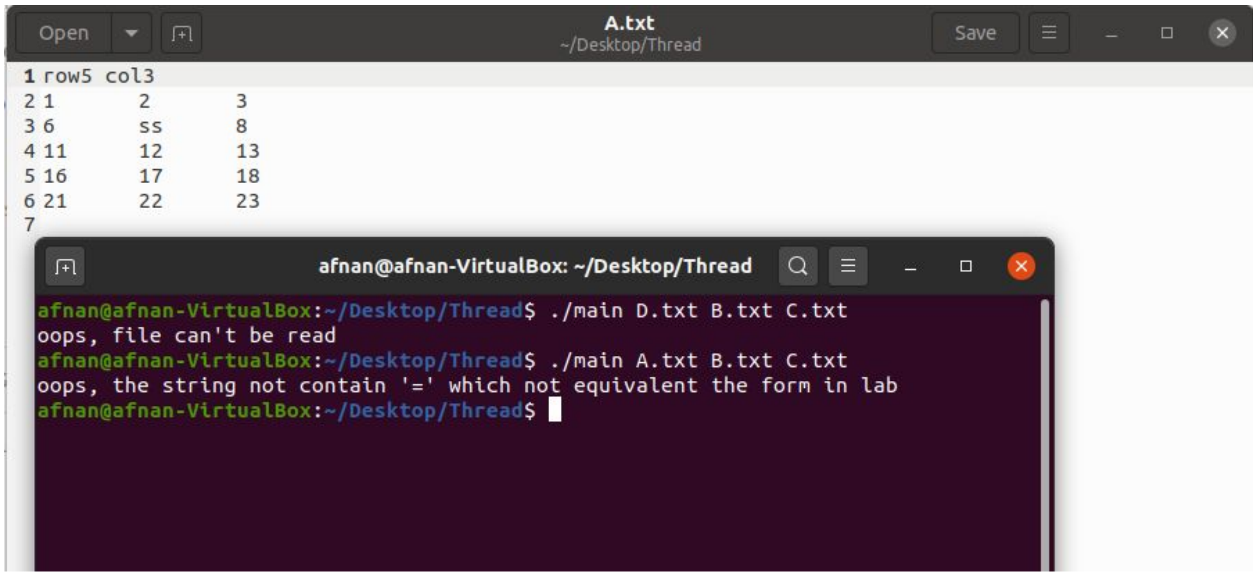
➢ The program prints an error message and terminates.



```
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main D.txt B.txt C.txt
oops, file can't be read
afnan@afnan-VirtualBox:~/Desktop/Thread$
```

IV. when the data in files not accurate :

➢ The program prints an error message and terminates.

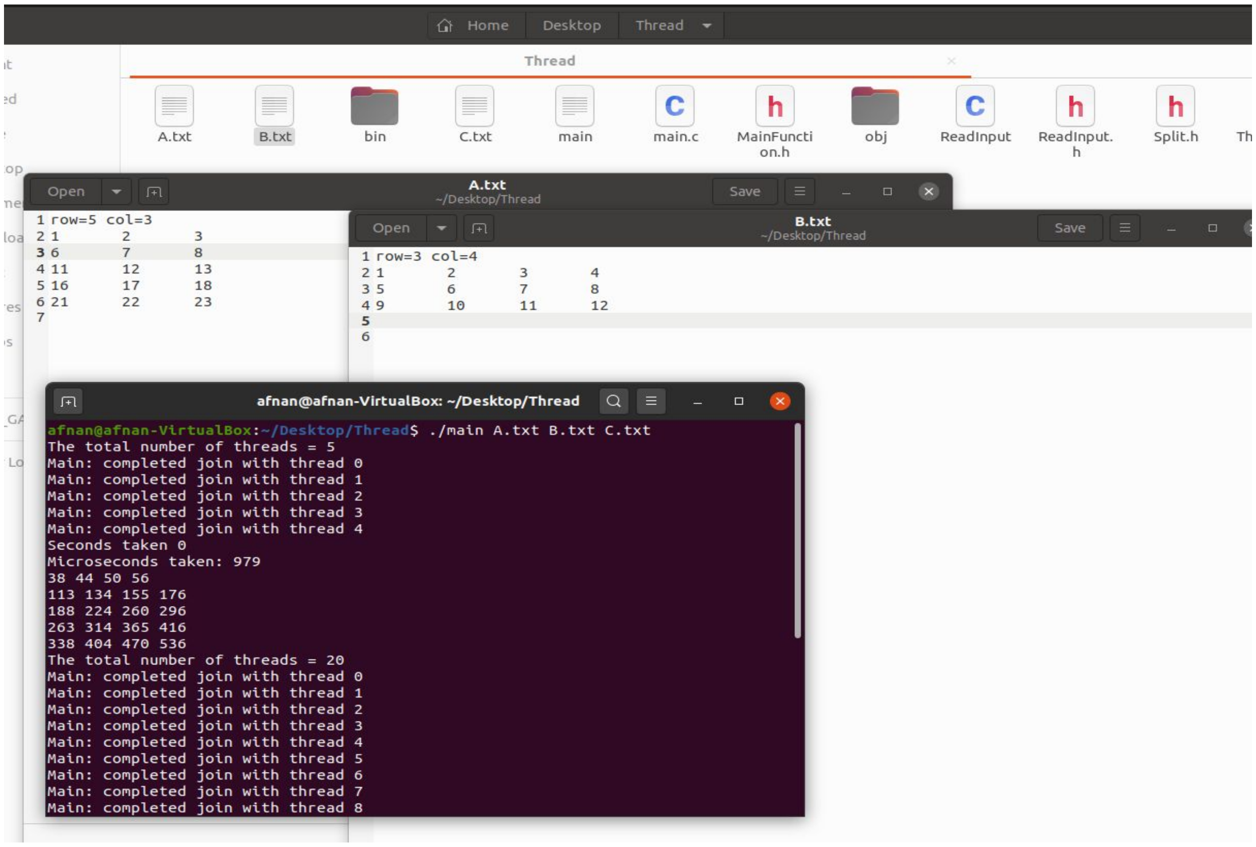


```
1 row5 col3
2 1       2       3
3 6       ss      8
4 11      12      13
5 16      17      18
6 21      22      23
7
```

```
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main D.txt B.txt C.txt
oops, file can't be read
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main A.txt B.txt C.txt
oops, the string not contain '=' which not equivalent the form in lab
afnan@afnan-VirtualBox:~/Desktop/Thread$
```

V. when the data is accurate :

➢ The file C.txt creates and the output is written in it .



A.txt ~/Desktop/Thread

```
row=5 col=3
1       2       3
6       7       8
11      12      13
16      17      18
21      22      23
```

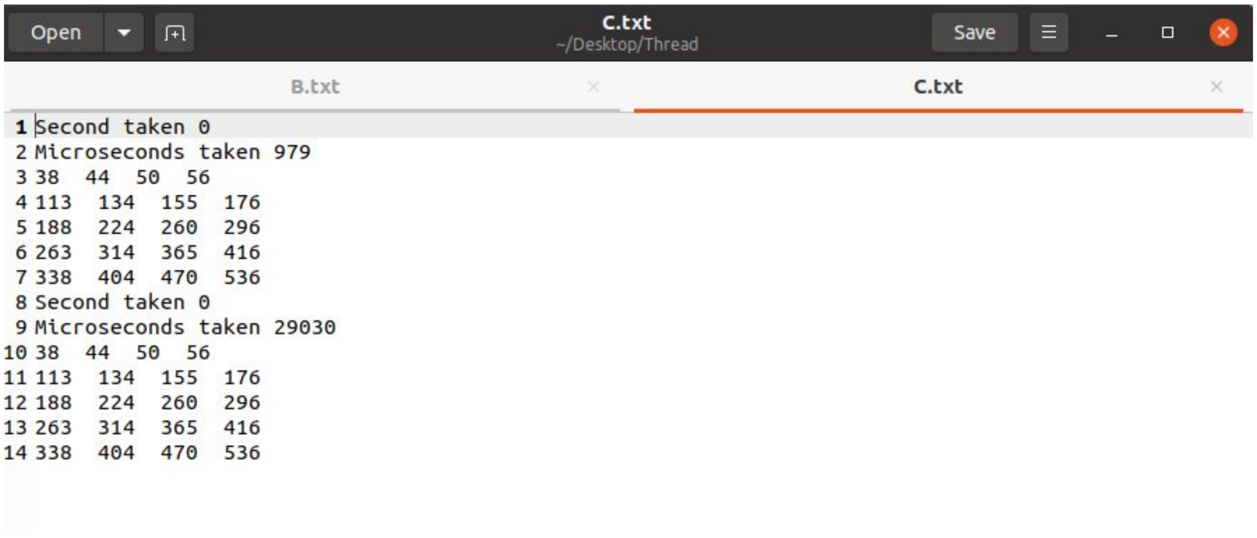
B.txt ~/Desktop/Thread

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

afnan@afnan-VirtualBox: ~/Desktop/Thread

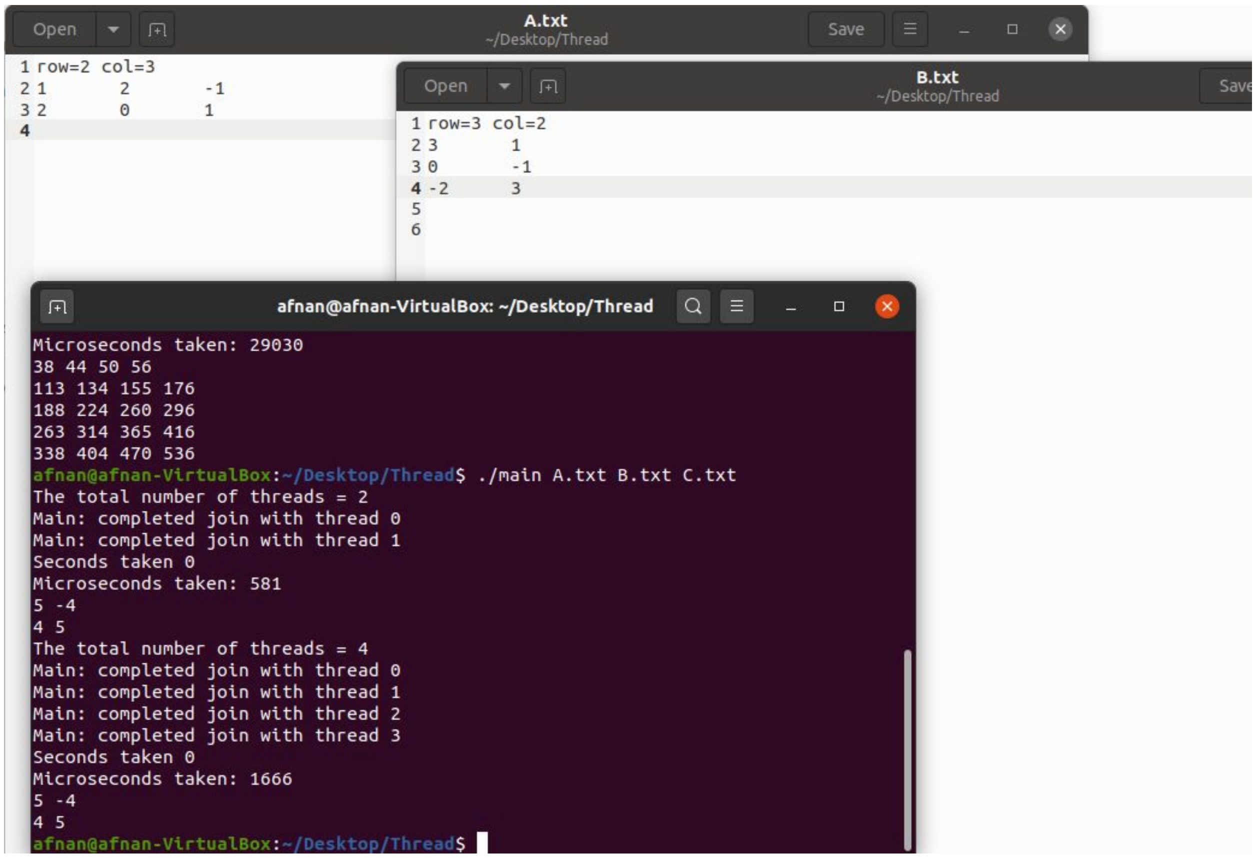
```
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main A.txt B.txt C.txt
The total number of threads = 5
Main: completed join with thread 0
Main: completed join with thread 1
Main: completed join with thread 2
Main: completed join with thread 3
Main: completed join with thread 4
Seconds taken 0
Microseconds taken: 979
38 44 50 56
113 134 155 176
188 224 260 296
263 314 365 416
338 404 470 536
The total number of threads = 20
Main: completed join with thread 0
Main: completed join with thread 1
Main: completed join with thread 2
Main: completed join with thread 3
Main: completed join with thread 4
Main: completed join with thread 5
Main: completed join with thread 6
Main: completed join with thread 7
Main: completed join with thread 8
```

➢ The Output file .



```
Second taken 0
Microseconds taken 979
38   44   50   56
113   134   155   176
188   224   260   296
263   314   365   416
338   404   470   536
Second taken 0
Microseconds taken 29030
38   44   50   56
113   134   155   176
188   224   260   296
263   314   365   416
338   404   470   536
```

➢ **Second example :**



A.txt
~/Desktop/Thread

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

```

B.txt
~/Desktop/Thread

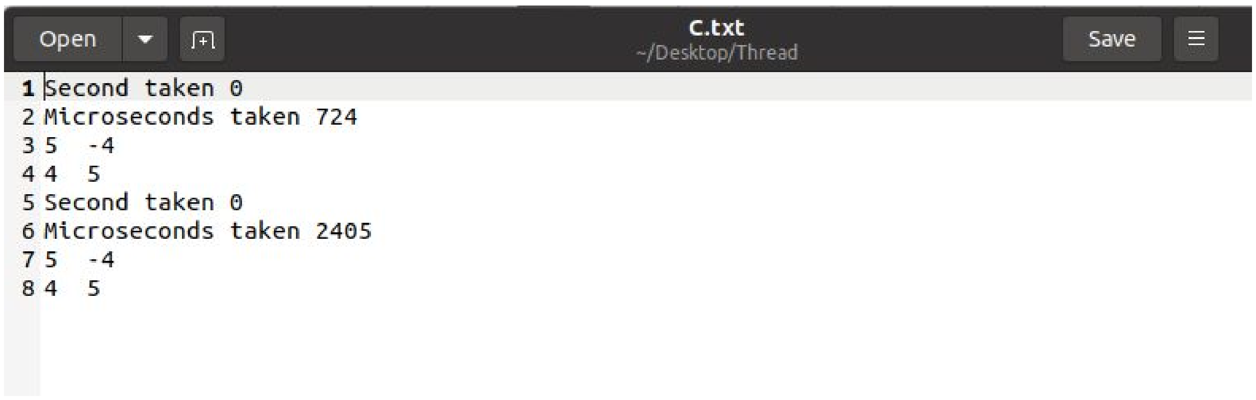
```
row=3 col=2
3       1
0       -1
-2      3


```

afnan@afnan-VirtualBox: ~/Desktop/Thread

```
Microseconds taken: 29030
38 44 50 56
113 134 155 176
188 224 260 296
263 314 365 416
338 404 470 536
afnan@afnan-VirtualBox:~/Desktop/Thread$ ./main A.txt B.txt C.txt
The total number of threads = 2
Main: completed join with thread 0
Main: completed join with thread 1
Seconds taken 0
Microseconds taken: 581
5 -4
4 5
The total number of threads = 4
Main: completed join with thread 0
Main: completed join with thread 1
Main: completed join with thread 2
Main: completed join with thread 3
Seconds taken 0
Microseconds taken: 1666
5 -4
4 5
afnan@afnan-VirtualBox:~/Desktop/Thread$
```

➢ The answer is :



C.txt
~/Desktop/Thread

```
Second taken 0
Microseconds taken 724
5   -4
4   5
Second taken 0
Microseconds taken 2405
5   -4
4   5
```

comparison between the two methods of matrix multiplication :

- The second method takes time greater than the first method .
- The number of threads created in the second method is greater than the number of threads in the first method .
- In the first example the first method takes 979 microsecond ,the second method take 29030 microsecond .
- When the number of threads in the first method =5 threads .
- the number of threads in the second method =20 threads .
- In the second example the first method takes 724 microsecond ,the second method takes 2405 microsecond .
- When the number of threads in the first method =2 threads .
- the number of threads in the second method =4 threads .