CSCI 6905 High Performance Computing
Assignment 01

Anil Adhikari
adhikaria16@students.ecu.edu

Part 1:

This part is solved using two c program files; coordinator.c and worker.c where coordinator.c is responsible for reading matrix from given input file name (file name is read from command line) and worker.c is responsible for doing multiplication.
Since, woker.c does not reads any arguments directly from command line, coordinator provides input for this using execvp() system call.
fork() system call is used to create child process in coordinator. coordinator.c creates child process to compute every element of the resultant matrix(result matrix). For example, if first file has matrix of size 2x2 and second file has matrix of size 2x2, then coordinator creates 4 child process to compute 4 elements of resultant matrix. coordinator.c does not perform multiplication but instead calls execvp() and pass row of first matrix as vector1 and column of second matrix as vector2. And worker.c performs computation and returns result using exit() system call. exit() takes one argument which will be incorporated with the status of the exit() system call.
Now, in parent process in coordinator.c, the status returned from worker is read and right shifted to get the result

Running procedure:

compile coordinator.c:

```
gcc coordinator.c -o coordinator
```

compile worker.c:

```
gcc worker.c -o worker
```

run coordinator:

```
./coordinator a.mat b.mat c.mat
```

The resultant matrix will be written in the c.mat file

Test cases:

Test case 1:

a.mat contains:

```
2   2
1   2
4   5
```

b.mat contains:

```
2   2
2   3
1   2
```

Run the program, for this test case matrices are stored in normal folder:
./coordinator normal/a.mat normal/b.mat normal/c.mat

```
Reading first matrix...
Size of Matrix1: 2 x 2
Matrix1:
1     2
4     5
Reading second matrix...
Size of Matrix2: 2 x 2
Matrix2:
2     3
1     2

In coordinator process: process Id: 4919

In worker process: process Id: 4919
In worker process: vector1:  1   2
In worker process: vector2:  2   1
In worker process: result: 4

In coordinator process: result: 4


In coordinator process: process Id: 4920

In worker process: process Id: 4920
In worker process: vector1:  1   2
In worker process: vector2:  3   2
In worker process: result: 7

In coordinator process: result: 7


In coordinator process: process Id: 4921

In worker process: process Id: 4921
In worker process: vector1:  4   5
In worker process: vector2:  2   1
In worker process: result: 13

In coordinator process: result: 13
```

In coordinator process: process Id: 4922

In worker process: process Id: 4922
In worker process: vector1: 4   5
In worker process: vector2: 3   2
In worker process: result: 22

In coordinator process: result: 22

Size of Resultant Matrix: 2 x 2
Resultant matrix is:
4 7
13 22
Writing to the file...
Done!!

Now, c.mat contains:

```
2    2
15   44
27   60
```

Test case 2:

a.mat contains:

```
2    2
1    4
2    1
```

b.mat contains:

```
3    3
1    2    3
4    3    2
3    2    1
```

Now, run the coordinator, for this test case matrices are stored in nomultiplication folder:

./coordinator nomultiplication/a.mat nomultiplication/b.mat nomultiplication/c.mat

Reading first matrix...
Size of Matrix1: 2 x 2
Matrix1:

1 4
2 1
Reading second matrix...
Size of Matrix2: 3 x 3
Matrix2:
1 2 3
4 3 2
3 2 1
Sorry!! These two matrices can not be multiplied.
Because column1 is not equal to row2 :(

o.mat is empty.

Test Case 3:

a.mat contains:

1 1
5

b.mat contains:

1 1
5

Run the program, for this test case matrices are stored in singleelement folder:
./coordinator singleelement/a.mat singleelement/b.mat singleelement/c.mat

Reading first matrix...
Size of Matrix1: 1 x 1
Matrix1:
5
Reading second matrix...
Size of Matrix2: 1 x 1
Matrix2:
5

In coordinator process: process Id: 5218

In worker process: process Id: 5218
In worker process: vector1: 5
In worker process: vector2: 5
In worker process: result: 25

In coordinator process: result: 25

Size of Resultant Matrix: 1 x 1
Resultant matrix is:
25
Writing to the file...
Done!!

Now, c.mat contains:

1 1
25

Test case 4:

a.mat is empty.

b.mat contains:

3 3
1 2 3
4 3 2
3 2 1

Now, run the coordinator, for this test case matrices are stored in emptyfile folder:
./coordinator emptyfile/a.mat emptyfile/b.mat emptyfile/c.mat

Sorry!!! file emptyfile/a.mat is empty.

o.mat is empty.
In this case, if one of the input file is empty, then the program exits.

Part 2:

This part is solved using pthread library.
Three int pointer array is used where arr is used to store input array, temp is used to temporarily store the value of computation and sum is used to store value and print the final array for each distance. temp array is copied to the sum array at the end of every computation for every increment.
pthread_barrier is used to synchronize every thread created. After the computation is finished is by every thread, then all the thread will be joined to the main thread using pthread_join() method.
Threads are created to computer every element of the array satisfying the condition of increment/distance. So, array of threads is created using pthread_t tids[sizeOfArray].
Input array is stored in file using .mat format.

Running procedure

compile prefix.c:

gcc prefix.c -lpthread -o prefix

run prefix:

./prefix data.mat

Test Cases

Test Case 1:

data.mat contains:

5 6 9 4 7 3

run the prefix using:

./prefix data.mat

Output in console:

Initial Array: 5 6 9 4 7 3

Sum after distance 1: 5 11 15 13 11 10

Sum after distance 2: 5 11 20 24 26 23

Sum after distance 4: 5 11 20 24 31 34

Test Case 2:

array.mat contains:

1 2 3 4 5 6

run the prefix using:

./prefix array.mat

Output in console:

Initial Array: 1 2 3 4 5 6

Sum after distance 1: 1 3 5 7 9 11

Sum after distance 2: 1 3 6 10 14 18

Sum after distance 4: 1 3 6 10 15 21