PROGRAMMING ASSIGNMENT#2 (DUE BY 11:59PM JULY 26TH)

PURPOSE

This assignment intends to familiarize you with Linux Programming using POSIX Pthreads and synchronization mechanism such as mutexes to create multithreading application.

ASSIGNMENT

In your programming assignment#0, you have successfully implemented the `testalphabet` program to count frequency of each alphabet letter in .txt files under a given folder. In this assignment, you will have to speed up your program using multithreading.

You will be provided a *two.zip* which includes the following files:

```
makefile
count.h
alphabetcountmulthreads.c
testmulthreads.c
```

Your assignment is to write/modify code in the following files

`count.h`
`makefile`
`alphabetcountmulthreads.c` to count the frequency of alphabet letters;

Note: Feel free to do any change of these files (fill the code, create new functions and etc). You also can create new .h and .c files if needed.

The instructor will use the following file to test your code, so please DON'T modify this file:

`testmulthreads.c`

Your program must execute correctly on Edoras machine, the instructor will type the following commands to test your code:

```
make// generate testmulthreads executable file
./testmulthreads 3   // run the testmulthreads program using 3 threads in parallel, the result will be stored in the alphabetfreq[] array and written in the file result.txt under result folder (Note: the instructor may vary the number of threads to verify your program)
```

Requirements:

1) Multiple threads are expected to run in parallel to share the workload, i.e. suppose 3 threads to process 30 files totally, then each thread should process 10 files;
2) When a thread is created, a message should be print out showing which files this thread will process, for example:
   ```
   Thread id = 274237184 starts processing files with index from 0 to 10!
   ```

3) When a file is being processed, a message should be print out showing which thread (thread_id = xxx) is processing this file, for example:

```
Thread id = 265844480 is processing file input_11.txt
```

4) When a thread is done with its workload, a message should be print out showing which files this thread has done with work, for example:

```
Thread id = 274237184 is done !
```

5) The array `long alphabetfreq[ ]` should always be up-to-date, i.e. it always has the result of all the threads counted so far. [*You may need to use mutexes to protect this critical region*.]

You should have the screen printing be similar as follows:

```
 Thread id = 274237184 starts processing files with index from 0 to 10!
 Thread id = 265844480 starts processing files with index from 11 to 22!
 Thread id = 257451776 starts processing files with index from 23 to 31!

Thread id = 265844480 is processing file input_11.txt
Thread id = 257451776 is processing file input_22.txt
Thread id = 274237184 is processing file input_00.txt
… …

 Thread id = 274237184 is done !
 Thread id = 265844480 is done !
 Thread id = 257451776 is done !

The results are counted as follows:
a -> 2861232
b -> 494472
c -> 747252
…   …
```

DIRECTIONS TO COMPLETE YOUR ASSIGNMENT

1. Download the source code two.zip from piazza
2. Upload two.zip to Edoras using sftp to programming folder
3. Unzip the two.zip on edoras using the commands:
   `unzip two.zip`
   so you will have one more folders: two (source files) under programming directory on edoras machine
4. Modify source files under folder two to complete this assignment
5. Test your program to make sure it works correctly.

HOW TO SUBMIT YOUR ASSIGNMENT

- The source files under the *two* folder on edoras machine will be considered for grading.
- Please finish your coding by **11:59pm July** 26th and make sure your program must execute correctly on edoras. Your grading may be started immediately after the deadline unless notice otherwise, so please make your files ready by the deadline.
- Excuses of “but it worked on my machine” will not be accepted, so if you develop elsewhere, plan to leave time for any migration problems that might arise.