CSE 333 - OPERATING SYSTEMS

Programming Assignment # 3

DUE DATE: 17/01/2022 - 23:59

In this homework, you will use the POSIX thread (Pthread) library to implement multi-threaded text manipulation program. Your program will take a directory name as an input, read a directory of text files (the files with .txt extension), assign each file to a separate worker thread. Each worker thread will parse all words from the given file and store each parsed unique word into a dynamically allocated array. It should be noted that each file must be read and processed by a separate worker thread.

Use the main thread to orchestrate everything. More specifically, the main thread reads the given directory, assigning threads to regular files that it encounters. The main thread also allocates the initial array of character pointers (or array of structs).

Program Details

a) The main thread is responsible for creating threads and waiting for the completion of them.
b) Each worker thread must open only its assigned file.
c) The directory to be read is specified as the first command-line argument.
d) The number of worker threads is specified as the second command-line argument.
e) Instead of storing just words, you must also store which files the words are in. Therefore, you should probably use a struct for this. This memory must be dynamically allocated.
f) Worker threads are responsible for re-allocating memory, as necessary. More specifically, if a word is to be stored, but the dynamically allocated array is full, then the worker thread detecting this must call the **realloc()** function. The new array size should be doubled in that case. Such operations must be synchronized!
g) It should be noted that the number of files and the number of threads may differ. However, it is ensured that the number of files will be greater than or equal to the number of threads.
h) If the number of files are greater than the number of threads, then each worker thread continues to take a new file given by the MAIN Thread after it completes the first file.
i) Please do not use global variables to synchronize various threads.
j) Please do not just implement the simple approach of merely having all threads share a single “mutex” variable.
k) Instead, implement the following:
    - When a worker thread adds a word, the only part that is synchronized is obtaining the designated index (i.e., such that no worker threads aim

to write to the same array slot). Allocating memory and writing to the designated array slot must be able to occur in parallel with other threads. In other words, multiple threads could be writing to different slots of the array simultaneously.

- When memory for the array needs to be re-allocated (i.e., by doubling its size), only one thread can do this. All other threads must be waited while this re-allocation occurs.

l) Handling Errors

- Your program must ensure that the correct number of command-line arguments are included. If not, display an error message and usage information to stderr as follows:
    i. ERROR: Invalid arguments
    ii. USAGE: ./a.out -d <directoryName> -n <#ofThreads>
- Be sure that all dynamically allocated memory is freed via **free()**.

Required Output

When you execute your program, each worker thread must preface its output with its thread ID, which can be obtained via **pthread_self()** function. Further, each worker thread should display the word that it processed, as well as at which index it places the word. As an example, if you are given a directory called *exampleDirectory* that contains the files *filename1.txt* and *abc.txt* with the following contents:

The content of abc.txt is:

I love CSE333 course a lot.

The content of filename1.txt is:

I like Pthread programming.

….

Hello

…….

Executing the code as follows will display the output shown below. The option “-d” is used for indicating directory name and the option “-n” is used for indicating the total number of threads.

```
$ ./a.out -d exampleDirectory -n 2

MAIN THREAD: Allocated initial array of 8 pointers.
MAIN THREAD: Assigned "abc.txt" to worker thread 1234.
MAIN THREAD: Assigned "filename1.txt" to worker thread 5678.
THREAD 1234: Added "I" at index 0.
THREAD 1234: Added "love" at index 1.
THREAD 1234: Added "CSE333" at index 2.
THREAD 1234: Added "course" at index 3.
THREAD 5678: The word "I" has already located at index 0.
THREAD 5678: Added "like" at index 4.
THREAD 5678: Added "Pthread" at index 5.
THREAD 5678: Added "programming" at index 6.
THREAD 1234: Added "a" at index 7.
THREAD 1234: Re-allocated array of 16 pointers.
THREAD 1234: Added "lot" at index 8.
...
….
THREAD 5678: Re-allocated array of 32 pointers.
THREAD 5678: Added "Hello" at index 16.
...
THREAD 5678: Re-allocated array of 64 pointers.
...
THREAD 5678: Re-allocated array of 128 pointers.
...
THREAD 5678: Re-allocated array of 256 pointers.
...
MAIN THREAD: All done (successfully read 189 words with 2 threads from 2 files).
```

Notes:

- The project will be done in Linux operating system using C programming language.

- You must use PThread library and synchronization appropriately in your code.
- Take into account materials and examples covered in the lab sessions.
- Consider all necessary error checking for the programs.
- No late homework will be accepted!
- In case of any form of copying and cheating on solutions, all parties/groups will get

  ZERO grade. You should submit your own work.

- You have to work in groups of two or three.

What to submit?

A softcopy of your source codes which are extensively commented and appropriately structured should be emailed to cse333.projects@gmail.com. All the files should be submitted as one zip file. You should use your student ids as the name of the file: StudentID1_StudentID2_project3.zip