**Assignment 4 - Word Blast**

Welcome to your fourth homework assignment.

Again, code should be neat, well documented. Variables should have meaningful names and be in a consistent format (use camelCase be consistent. In addition, each file should have a standard header as defined below. Remember your line length limits of generally 80 characters and no more than 100.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Class: CSC-415-0# Spring 2023

\* Name:

\* Student ID:

\* GitHub ID:

\* Project: Assignment 4 – Word Blast

\*

\* File: <name of this file>

\*

\* Description:

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

For this assignment, your program is to read War and Peace (a text copy is included with this assignment) and it is to count and tally each of the words that are 6 or more characters long. And again, you will only use Linux (not library) file functions, i.e. open, close, read, lseek, pread.

BUT, it is to do this using threads. Each thread will take a chunk of the file and process it, returning it’s results to the main which tallies (or if you directly tally to shared memory, that is okay) and then the main will print the ten, 6 or more character, words with the highest tallies, in order highest to lowest, and their associated counts. i.e. The top ten words and the number of times that word appears in the text. Remember that this assignment will be using the pthread functions.

Your program should take two parameters on the command line: FileName and ThreadCount

* FileName is the name of the file to read – WarAndPeace.txt
* ThreadCount is the number of threads you should spawn to evenly divide the work.

That is to say – if the parameter is 1, the entire file would be read and processed by one thread. If the parameter is 5, then you would divide the file into 5 equal parts (accounting for rounding on the last part). So thread one would take the first 1/5 of the file, thread 2 the second fifth and so on. But, these threads should all be launched together in a loop. So that they can execute in parallel.

#include <time.h> in your code and in main, include the code below in your main. This will display how much time your program takes. Your submission should include a run with 1 thread, 2 threads, 4 threads, and 8 threads. How do the times compare?

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// DO NOT CHANGE THIS BLOCK

//Time stamp start

struct timespec startTime;

struct timespec endTime;

clock\_gettime(CLOCK\_REALTIME, &startTime);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\* TO DO \*\*\* start your thread processing

// wait for the threads to finish

// \*\*\*TO DO \*\*\* Process TOP 10 and display

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// DO NOT CHANGE THIS BLOCK

//Clock output

clock\_gettime(CLOCK\_REALTIME, &endTime);

time\_t sec = endTime.tv\_sec - startTime.tv\_sec;

long n\_sec = endTime.tv\_nsec - startTime.tv\_nsec;

if (endTime.tv\_nsec < startTime.tv\_nsec)

{

--sec;

n\_sec = n\_sec + 1000000000L;

}

printf("Total Time was %ld.%09ld seconds\n", sec, n\_sec);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

There is a template main with this code provided. (Don't forget to rename it to the naming conventions). Do not modify in any way the blocks that start and end the timers (including the asterisks). Make sure that other than initialization code and cleanup code that your code is between these blocks where the TO DO's are.

Hints Do not forget to protect critical sections. Make sure you use thread safe library calls. You will need to know how long the input file is, look up lseek. Do not use pipes. You must use mutex locks.

Do a writeup using the template in PDF format that includes a description of what you did, issues, resolutions and an analysis of the results. Explain and reflect on why the times for the different runs are what they are, how does each run compare with the others. Also include the compilation and execution output from your program in the writeup. Your execution output should include at least 4 runs, 1 thread, 2 threads, 4 threads, and 8 threads. Submit all your source code file(s), Makefile and PDF writeup into GitHub. The writeup (PDF) should also be submitted to Canvas.

Note Do not change any of the settings of the virtual machine.

Your main program filenames should be <lastname>\_<firstname>HW<#>\_<optional>.<proper extension>

Sample Output

gcc -o bierman\_robert\_HW4\_main bierman\_robert\_HW4\_main.o -g -I. -l pthread

./bierman\_robert\_HW4\_main WarAndPeace.txt 4

Word Frequency Count on WarAndPeace.txt with 4 threads

Printing top 10 words 6 characters or more.

Number 1 is Pierre with a count of 1963

Number 2 is Prince with a count of 1928

Number 3 is Natásha with a count of 1212

Number 4 is Andrew with a count of 1143

Number 5 is himself with a count of 1020

Number 6 is princess with a count of 916

Number 7 is French with a count of 881

Number 8 is before with a count of 833

Number 9 is Rostóv with a count of 776

Number 10 is thought with a count of 767

Total Time was 1.938852743 seconds