Assignment 4

CS392 Spring 2019

Task Details

Implement a multi-threaded program, in which each thread reads a file, parses the file, and synchronizes the results to global status variables.

Requirement

1. Your submission should produce a program “cs392_thread”:
    a. It defines three global variables “item1_counter”, “item2_counter”, and “item3_counter”. All of them should be initialized as 0.

    b. The program takes three arguments, which are the paths to three files (The instructor will provide those files as test cases).

    c. The program creates three child threads, using the “**pthread_create**” interface. Each of the three threads needs to run the “cs392_thread_run” function
        i. The “cs392_thread_run” function should follow the prototype “void *cs392_thread_run (void *)”.
        ii. For each child thread, you need to pass the path of a unique file (as specified by one of the arguments to the program) to “cs392_thread_run”.
            1. Hint 1: The fourth argument of **pthread_create** can be used to pass argument to “cs392_thread_run”. In this case, you can cast the pointer that points to the file path as a “void*” and pass through **pthread_create.**

    d. Inside the **“**cs392_thread_run” function, you need to open and process the file whose path is passed as its argument:
        i. The file has the following format:
            1. Each line is a string that has one of the following value: “+item1”, “-item1”, “+item2”, “-item2”, “+item3”, “-item3”
        ii. You need to handle each line in the file:
            1. For “+item1”, please increase “item1_counter” by 1. For “-item1”, please decrease “item1_counter” by 1.
            2. For “+item2”, please increase “item2_counter” by 1. For “-item2”, please decrease “item2_counter” by 1.
            3. For “+item3”, please increase “item3_counter” by 1. For “-item3”, please decrease “item3_counter” by 1.

iii. YOU WILL NEED TO USE mutex interfaces to prevent race conditions while you are accessing “item1_counter”, “item2_counter”, and “item3_counter”.
    1. You will need to use **pthread_mutex_init** for mutex initialization.
    2. You will need to use **pthread_mutex_lock** to lock the mutex before global variable accessing.
    3. You will need to use **pthread_mutex_unlock** to unlock the mutex after global variable accessing.
    4. You will need to use **pthread_mutex_destroy** to destroy the mutex after you are done with all the threads.

e. You will need to use **pthread_exit** to terminate inside each of the child threads.

f. Inside the master thread, you will need to use **pthread_join to wait for the finish of each of the child thread.**

g. After all the child threads terminate, you need to print the final values of the three global variables, using the following statement: **printf(“The final value of item1_counter, item2_counter, and item3_counter are %d, %d, and %d”, item1_counter, item2_counter, item3_counter);**.

2. Your project should include the following files:
    a. “cs392_thread.c”
    b. “Makefile”

3. The file “cs392_thread.c” should include
    a. The “main” function
        i. It needs to check the number of arguments
        ii. It needs to start the three child threads
        iii. It needs to wait for the three child threads
        iv. It needs to print the final values
    b. The “cs392_thread_run” function
        i. Details have been explained above.

4. The “Makefile” builds the above source code files into “cs392_thread”
    a. You are expected to include explicit rules to compile the “.c” file into object files (“.o” files)
    b. You are expected to include explicit rule to link the objects files to generate “cs392_thread”
    c. You need to include a “clean” target in your Makefile, which, if executed, can remove the object files and the ”cs392_thread”.

5. Zip all the files into a cs392_ass4.zip and submit the .zip file only

Testcase:

Three test cases are provided at:
https://drive.google.com/file/d/1WTqP8PymQ2MrMUmDpWqqyVTIxi1Dkz_J/view?usp=sharing
https://drive.google.com/file/d/1yFh7dmLOH5066Ym76mUXkV736xpiQJYQ/view?usp=sharing
https://drive.google.com/file/d/128ynfxbrTwuEgAIWahJAmlOrqUFI8czy/view?usp=sharing

Please download the three files as “item_file1.txt”, “item_file2.txt”, “item_file3.txt”

To run the test, put the three test cases in the same folder as your “cs392_thread” program, run:
$ ./cs392_thread ./item_file1.txt ./item_file2.txt ./item_file3.txt

Run the above command multiple times, make sure you always get the same result.
Correct values for item1_counter, item2_counter, and item3_counter are 10055, 4884, and 4995

Grading Policy:

- Cannot compile or run: Will not be graded (I mean it this time)
- No statement of “I pledge my honor that I have abided by the Stevens Honor System.” as comment in the beginning of your code: Will not be graded (I mean it this time)
    - Place the honor statement in the beginning of the “.c” file and the Makefile
- Late submissions without pre-approval will not be accepted !!!
- This must be done individually. Group work will lead to 0 point !!!
- Pass one group of test cases (+8, 40 in total)
    - One group has been provided. The other four groups will **NOT** be released, but will follow the same formats the the given one.
    - **No use of threads to pass the test cases will lead to 0 points in this part**

- Correctly check and handle the arguments to the cs392_thread program (+5)
- Correctly create three threads with pthread_create (+10)
- Correctly use mutex to avoid race conditions (+20)
- Correctly use pthread_exit to terminate threads (+5)
- Correctly use pthread_join to re-cycle the returned threads (+5)
- Correctly do clean-up and handle possible errors (+10)
- Correct Makefile (with cleanup rules) (+5). **If you copy my midterm Makefile, YOU WILL GET 0 POINT HERE, even if you change the file names!!!**