Programming Assignment 2
Implementing Filter and Peterson based Tree Lock Algorithms

Submission Date: 12th February 2022, 9:00 pm

Goal: The goal of this assignment is to implement Filter and Peterson-based tree lock algorithms. Implement both these locking algorithms in C++.

Details. As mentioned above, you have implement two algorithms **in C++**:

1. **Filter Lock** as studied in the class from the book
2. **Peterson based Tree Lock (PTL)** described in Exercise 13 (page 41) of the book.

Each locking algorithm implements a class consisting of two methods: lock and unlock. Note that we did not discuss PTL. But the explanation given in the Exercise 13 is self explanatory.

To test the performance of locking algorithms, develop an application, lock-test is as follows. Once, the program starts, it creates n threads. Each of these threads, will enter critical section (CS) k times. The pseudocode of the test function is as follows:

Listing 1: main thread

```
void main()
{
    ...
    ...
    // Declare a lock object which is accessed from all the threads
    Lock Test = new Lock();
    ...
    ...
    create n testCS threads;
}
```

Listing 2: testCS thread

```

void testCS()
{
    id = thread.getID();
    for (i=0; i < k; i++)
    {
        reqEnterTime = getSysTime();
8          cout << i << "th CS Entry Request at " << reqEnterTime << " by thread
9          " << id << " (mesg 1)";
10         Test.lock();
11         actEnterTime = getSysTime();
12         cout << i << "th CS Entry at " << actEnterTime << " by thread
13         " << id << " (mesg 2)";
14         sleep(t1);
15         reqExitTime = getSysTime();
16         cout << i << "th CS Exit Request at " << reqExitTime << " by thread
17         " << id << " (mesg 3)";
18         Test.unlock();
19         actExitTime = getSysTime();
20         cout << i << "th CS Exit at " << actExitTime << " by thread
21         " << id << " (mesg 4)";
22         sleep(t2);
23     }
24 }
```

Description of the Test Program: The description is self-explanatory. As can be seen, each thread invokes testCS function. When a thread wishes to enter the Critical Section (CS), it stores the time as reqEnterTime (or request Enter Time). And once the thread enters the CS, it records the time as actEnterTime (actual CS Entry Time). The difference between these times actEnterTime - reqEnterTime is the time taken by the thread to enter the CS which we denote as *csEnterTime*.

Similarly, when a thread wishes to leave the CS, it records the time as reqExitTime (request Exit Time). Once it leaves the CS, it again records the time as actExitTime. The difference between these times actEnterTime - reqEnterTime is the time taken by the thread to exit the CS which we denote as *csExitTime*.

If the lock and unlock functions work correctly then the display messages 2 and 3 of every thread will work correctly without any interleaving. Seeing these messages one can be sure of the correctness of the lock and unlock. Note that the message 1 and 4 can interleave and hence may be commented out to check the correctness.

Here $t1$ and $t2$ are delay values that are exponentially distributed with an average of $\lambda1, \lambda2$ seconds (and not milli-seconds). The objective of having these time delays is to simulate that these threads are performing some complicated time consuming tasks.

The $Test$ variable declared in line 6 declared in main is an instance of Lock class and is accessible by all threads.

Input: The input to the program will be a file, named inp-params.txt, consisting of all the parameters described above:$n, k, \lambda1, \lambda2$. A sample input file is: 64 100 10 15.

Output: Your program should output to a file in the format given in the pseudocode for each algorithm. A sample output is as follows:

```
Filter Lock Output:
1st CS Entry Request at 10:00 by thread 1 (mesg 1)
1st CS Entry at 10:05 by thread 1 (mesg 2)
2nd CS Request at 10:01 by thread 2 (mesg 1)
1st CS Exit Request at 10:08 by thread 1 (mesg 3)
1st CS Exit at 10:10 by thread 1 (mesg 4)

.
.
```

.

```
PTL Output:
1st CS Requested at 11:00 by thread 1 (mesg 1)
1st CS Entered at 11:05 by thread 1 (mesg 2)
1st CS Exit Request at 11:07 by thread 1 (mesg 3)
1st CS Exit at 11:08 by thread 2 (mesg 1)
.
.
.
```

As mentioned above the combination of (mesg 2) and (mesg 3) should demonstrate that the mutual exclusion is satisfied.

Report: You have to submit a report for this assignment. The report should first explain the design of your program, specifically the design of PTL algorithm since it is not present in the book. The design of Filter lock can be briefly mentioned.

The report should then show a comparison of the performance of Filter lock and PTL algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph.

You run both these algorithms varying the number of threads from 2 to 64 in the powers of 2 while keeping other parameters same. Please have k, the number of CS requests by each thread, fixed to 10 in all these experiments. You measure the average time taken to enter the CS by each thread.

The graphs in the report will be as follows for each algorithm:

- the x-axis will vary the number of threads from 2 to 64 in the powers of 2 (as explained above).
- the y-axis will show (1) csEnterTime: the average time taken to enter the CS by each thread and (2) csExitTime: the average time taken to exit the CS by each thread.

It must be noted that for each algorithm there will be two curves: entry and exit times. Thus, there will be four curves for the two algorithms.

Finally, you must also give an analysis of the results while explaining any anomalies observed in the report.

Deliverables: You have to submit the following:

- The source file containing the actual program to execute. Name it as SrcAssgn2-⟨rollno⟩.cpp.
- A readme.txt that explains how to execute the program.
- The report as explained above.

Zip all the three files and name it as ProgAssgn2-⟨rollno⟩.zip. Then upload it on the google classroom page of this course. Submit it by the above mentioned deadline.

Please see the instructions given above before uploading your file. Your assignment will NOT be evaluated if there is any deviation from the instructions posted there.

All assignments for this course has the late submission policy of a penalty of 10% each day after the deadline for 8 days Submission after 8 days will not be considered.

Kindly remember that all submissions are subjected to plagiarism checks.