OS2 Programming Assignment 4
Spring 2018
Implementing Barriers using Semaphores
Submission Date: 10/03/2018, 09:00 PM

Goal: The goal of this assignment barrier using semaphores. This problem is exercise 5.41 from the textbook. After implementing this, you have to compare its performance with the barriers provided by pthread library.

Details: As mentioned above, this is exercise 5.41 from the textbook. You have to implement barriers using semaphores. More details are provided in the book.

After implementing you have to test its performance with barriers provided by pthreads library. To test the performance of your implementation, you use the following test program:

```
1  void main()
2  {
3      barrier-init(N);
4      create N test threads;
5  }
```

Listing 1: main thread

```
1  void testBarrier()
2  {
3      int beforeBarrSleep, afterBarrSleep;
4      // Execute the barrier k times
5      for (int barrIter=0; barrIter<k; barrIter++) {
6
7          // Simulate thread doing some time consuming task before the barrier
8          beforeBarrSleep = rand(preλ);
9          cout << "Going to sleep before the barrIterth barrier invocation at time: "
10         << getSysTime();
11         sleep(beforeBarrSleep);
12
13         // Implement this using (1) your implementation (2) pthread barriers
14         cout << "Before the kth barrier invocation at time: " << getSysTime();
15         barrier_point();
16         cout << "After the kth barrier invocation at time: " << getSysTime();
17
18         // Simulate thread doing some time consuming task after the barrier
19         afterBarrSleep = rand(postλ);
20         cout << "Going to sleep after the barrIterth barrier invocation at time: "
21         << getSysTime();
22         sleep(afterBarrSleep);
23     }
24     measure the time taken to complete k barrier invocations;
25 }
```

Listing 2: 'testBarrier' function invoked by each thread

Let us denote your barrier implementation as new-barr while the barrier provided by pthreads as pthread-barr. Please test with both the barrier implementations. The input & output formats are shown below.

Input: The input to the program will be a file, named inp-params.txt, consisting of the all the parameters discussed above: N k $pre\lambda$ $post\lambda$.

A sample input is: 32 100 0.5 0.75

Output: You program should output the following files:

1. You must display the log of all the events as shown in testBarrier method. So your program must generate two output files, one for each algorithm: new-barr-log.txt and pthread-barr-log.txt, consisting of all the events. A sample output is shown below.
2. Average_time.txt, consisting of the average time taken for each thread to complete k barrier invocations. This will consist of N entries, one for each thread and the average of the time taken by all the threads.

Example: A sample output for the barriers would be as follows:

Going to sleep before the 1st barrier invocation at time: 10:00 by Thread 1
Going to sleep before the 1st barrier invocation at time: 10:01 by Thread 2
Before the 1st barrier invocation at time: 10:01 by Thread 1
Before the 1st barrier invocation at time: 10:02 by Thread 2
After the 1st barrier invocation at time: 10:03 by Thread 1
.....
.....

Report: You have to submit a report and readme for this assignment. The report should first explain the design of your program while explaining any complications that arose in the course of programming. The report should contain a comparison graph of the performance of both the barrier implementations - new-barr & pthread-barr. You must run all the algorithms multiple times to compare their performance and display the result in form of two graphs for the following parameters:

Graph1:

1. Y-axis: Average time taken for each thread to complete.
2. X-axis: Varying k from 50 to 100 in the increments of 10. Have the number of threads fixed such as 16. It is better to choose this number based on the number of cores & hardware threads of the system that you plan to test on.

Graph2:

1. Y-axis: Average time taken for each thread to complete.
2. X-axis: Varying N from 2 to 32 in the multiples of 2. Have k to be fixed as 70.

Average each point in the graph by 3 times. The graph must contain two curves for each of the algorithms: new-barr & pthread-barr.

Submission Format: You have to submit the following with this assignment.

1. The source code: Assgn4-newbarr<RollNo>.cpp
2. Readme: Assgn1-Readme-<RollNo>.txt
3. Report: Assgn1-Report-<RollNo>.pdf as explained above.

Name the zipped document as: Assgn1-<RollNo>.zip. If you don't follow these guidelines, then your assignment will not be evaluated. Upload all this on the drive by the specified deadline.

Grading Policy:

1. Design as described in report and analysis of the results: 50%
2. Execution of the programs based on description in readme: 45%
3. Code documentation and indentation: 5%