Synchronization with monitors

Laurent Graebner, Jander Nascimento, Lam

November 26, 2010

Contents

1 Implementing Thread-safe linked list

1.1 Validity of the first test

The first test implemented will not fail duo do its verification. The test was written to catch up situations where the simultaneos change would generate a conflict case and the programming running end up in a *deadlock*. This test is completely valid, but the *deadlock* does not happens constantly with this implementation so we may say that the test is not effective.

1.2 Analyzing the first test output

Output:

1. Thread nmbr 0, won.
2. Thread nmbr 1, won.
3. THREAD 0 TIME: +0s 0.040ms, TYPE : BEGIN WRITE
4. THREAD 0 TIME: +0s 0.041ms, TYPE : END READ
5. THREAD 0 TIME: +0s 0.041ms, TYPE : BEGIN READ
6. THREAD 0 TIME: +0s 0.041ms, TYPE : END READ

7. THREAD 0 TIME: +0s 0.041ms, TYPE : BEGIN WRITE
8. THREAD 0 TIME: +0s 0.041ms, TYPE : END READ
9. THREAD 0 TIME: +0s 0.041ms, TYPE : BEGIN READ
10. THREAD 0 TIME: +0s 0.042ms, TYPE : END READ
11. THREAD 0 TIME: +0s 0.042ms, TYPE : BEGIN WRITE
12. THREAD 0 TIME: +0s 0.042ms, TYPE : END READ
13. THREAD 0 TIME: +0s 0.042ms, TYPE : BEGIN READ
14. THREAD 0 TIME: +0s 0.042ms, TYPE : END READ
15. THREAD 0 TIME: +0s 0.042ms, TYPE : BEGIN WRITE
16. THREAD 0 TIME: +0s 0.042ms, TYPE : END READ
17. THREAD 0 TIME: +0s 0.042ms, TYPE : BEGIN READ
18. THREAD 0 TIME: +0s 0.043ms, TYPE : END READ
19. THREAD 1 TIME: +0s 0.094ms, TYPE : BEGIN WRITE
20. THREAD 1 TIME: +0s 0.097ms, TYPE : END READ
21. THREAD 1 TIME: +0s 0.097ms, TYPE : BEGIN READ
22. THREAD 1 TIME: +0s 0.098ms, TYPE : END READ

The consistency of the call can be analyzed looking at three parameters: the identifier of the thread, type of the call and the order in which they appear in the output.

For instance, using this output we may infer that there is two threads (and the subject says that to us:), analysing the thread 0 (zero) it's possible to see that between write operations it's possible to have read operations.

1.3 Naive implementation, using mutexes

code

1.4 Naive implementation test

code

1.5 Efficiency of naive implementation

This kind of implementation has the same result as a serialization of all operations, since only one thread can access the code at a time. That is the main reason why this kind of implementation is safe.

2 Improving concurrency

2.1 Improving concurrency capability

sec