OS
Assignment 2

Joost Grunwald, s1057493 Martijn van de Wouw, s1052099

October 12, 2021

Deadlocks and starvation

As far as we know there are no possible deadlocks. Because a lock is made in a way that it is always temporary and does not have to wait nor get stuck on an infinite loop. Because of this it is always freed and hence we have no deadlocks in our implementation. Another reason we are so sure about this is the considerable amount of tests we ran.
For starvation, it is idem, we have thoroughly tested it, however it is also ensured by the fact that we have no possible infinite cycles, as we add or remove 1 at the time and not in batches, which means that 1 certain thread can never occupy the shared resources for a considerable amount of time.

Discussing tests

test 1) In buffertest1() we test for normal functionality of the buffer. First we print an empty buffer and add some elements (name 3, 4, 5 in that order). Then we print again so we can see if the buffer has the new elements added to them. Then we remove the first element (being a 3) and print again to see if it worked.
By running the test we find that the normal functionality of the buffer is all good.

test 2) In buffertest2() we test some special cases after first printing an empty log:

- First we read from the log with an out of bounds index. Our error handling for this works.
- Then we try to read from the log from with a negative index. The error handling also prevents this and reports to the user.
- After that we tried to set the buffer bound to 0. This should not be possible since you can not have a buffer of size 0. We prevent this correctly.
- Then we tried to set a buffer with a negative size which also should not be possible. The user gets an error as expected
- Now we try to remove element 0 from the buffer. But since we do not have that element this should not be possible. The user again gets the report that it is not possible.
- Now we tried to remove an element with an out of bounds index. This also does not work. And the programs reports to the user.
- We do the same with a negative index which yields the same result as above.

test 3) In buffertest3() we first add 9 elements to a buffer after which we change the bound to 4 so we can see if the last few elements are dropped like they are supposed to. Then we try to add elements to a full buffer which should not be possible. Now we increase the bound to 8 and add another 5 elements to see if the bound has been set correctly (the last of the 5 should not be added). Now we set the bound to be unbounded and add some elements to see if it truly is unbounded.

test 4) In buffertest4() we test all the above 3 tests but within threads. Which means that we run multiple threads doing these tests and see if this gives any strange results, in this test we also try to show that there are no deadlocks or starvation in our solution. We do this by creating threads running our functions and joining them. We also test all possible combinations of other tests.

test 5) In buffertest5() we do the same as in buffertest4() but now with a lot more threads.