Condition variables

1) What does it mean to say that a data structure is thread safe?

It means that threads won’t clash with each other (race conditions are written out) and the intricacies of synchronization has been taken into account.

2) In the circular buffer implementation of a queue, why is the maximum number of elements in the queue n-1, if n is the size of the array?

We want to avoid the case where next\_in == next\_out, i.e. when the 2 pointers overlap on the same element. There are two cases of this happening, the first is at the onset, when the queue is empty, and the second is when the queue is full and the next\_in point has wrapped completely around and caught up with next\_out. Since we’ve chosen to define next\_in == next\_out to be the condition when the queue empty, we have to find a way to define queue full. In this implementation of circular buffer, queue full to defined as when next\_in = next\_out -1. Hence, there are there are only n-1 spaces in the queue.

3) If there is no mutex to protect the queue, give an example of a sequence of steps that could leave the queue in an inconsistent state.

Queue\_pop can access empty array elements that haven’t been written yet, thus throwing next\_in and next\_out off.

* Consumer is waiting on conditional variable
* Producer sets to array[0]
* Producer writes to array[0]
* Consumer awakes
* Consumer sets to array[0]
* Consumer reads from array[0]
* Producer sets to array[1]
* Consumer sets to array[1]
* Consumer reads from array[1] -> error!

4) When a thread calls cond\_wait, why does it have to unlock the mutex before blocking?

So that other threads can continue execution. Otherwise the mutex remains locked, the conditional thread is waiting for a cond\_signal, and no other threads can proceed.

5) When a thread returns from cond\_wait, what do we know is definitely true?  What do we think is probably true?

We know that that since the new condition is true and the mutex is locked, there is only one thread in the queue. But we can’t be sure that it will remain true up until the moment the thread continues execution (i.e. reads or writes) due to a signal interrupt.

6) What happens if you signal a condition variable when there are no waiting threads?

Nothing. It doesn’t hurt. Its power comes when there are waiting threads, but it’s fine.

7) Do you have to lock the mutex to signal a condition variable?

Let’s assume we do need to lock the mutex. In an ideal case, we lock the mutex, and send a signal to the conditional variable, awakening the threads stuck in conditional limbo. Since the mutex is locked, no thread can execute except those in conditional limbo. But what if there are no threads in conditional limbo? Because of the locked mutex, no threads can run and the program is at an impass.

8) Does the condition have to be true when you signal a condition variable?

A signal to the conditional variable should only be sent when the current state of the condition could true. However, threads must double check this condition immediately before proceeding, since signal interruption can make the condition no longer true when that thread is queued and waiting to be run.