Assignment 3: Conditional Variables

Student 1: S.S. Iyer (866094)
Student 2: X. Teng (851499)

Overview

There exists a single producer thread that will produce an item into a FIFO circular buffer and there exists multiple consumer threads each of which will consume the item produced from the buffer if and only if the item is meant for that consumer. Each item consists of two parts, a sequence number which uniquely identifies the item and a destination which is equal to the id of the consumer it belongs to. The destination of an item is randomly generated. Since, each item is only meant for a specific consumer, the order of retrieval from buffer matters. The buffer is mutex locked between all the threads to ensure mutually exclusive access to it. The producer will add an item to the buffer if sufficient space is available and increment the in index while the consumers will take an item from the buffer if it belongs to them and increment an out index. In both cases, we also keep track of and update the number of elements on the buffer if necessary (known as *bufferSize* in our program).

How we implemented it?

We first initialize all the consumer threads and then the producer thread. Each consumer thread is set to wait for at least one element in the buffer. So they cannot do anything until the producer is also started. Also, the consumer is allowed to peer into the oldest item on the buffer. This property is used to keep track of the destination of the oldest item on the buffer. If there is at least one item in the buffer and the oldest one currently belongs to the consumer checking on it, then that consumer will immediately consume the item from the buffer. Otherwise, if the item does not belong to it, the consumer will first signal the consumer that it belongs to before going back to waiting. Furthermore, after a consumer has consumed an item from the buffer, it checks if there are more items in the buffer. If so, it will signal the consumer corresponding to the next item in the buffer. One thing that becomes apparent at this point is, if consumers signal other consumers, then who signals the first consumer? The producer will signal the first consumer as soon as it adds an element into the buffer.

How do we let all consumers know that production is finished?

Once all items have been produced, the producer produces a special item with a unique sequence number of 0 for each consumer and adds them to the buffer in the normal fashion as described above. The consumers will stop execution upon decoding the sequence number of the special item.

Mutexes used and their purpose

1. *bufferMutex*: This mutex is used to ensure mutually exclusive access to the buffer since all threads will require access to it to produce/consumer an item to/from it.
2. *producerSignalMutex*: This mutex is used to gain mutually exclusive access to the producer signals condition variables array while signaling a specific consumer.

Condition variables used and their purpose

1. *producerSignals[NROF_CONSUMERS]:* This array of condition variable signals is used to signal each consumer uniquely to check if the oldest item in the buffer corresponds to it. The producer uses it to send the progenitor signal to the first consumer which corresponds to the first item generated. Consequently, the consumers will use it to wait for a signal. They will also use it to propagate the signal to the next consumer.
2. *isEmpty*: This condition variable signal is used by the consumers to signal the producer to stop waiting for the buffer to become free once they have consumed an element from it at which point the buffer should be at least free enough for one more item.

Problems faced

- **What happens if producer is too slow compared to the consumers?**
  In this case, the buffer size will become 0 at one point. But if there are more elements to be produced, then the producer will add it to the buffer and again signal the relevant consumer, thereby restarting the process.
- **What happens if the producer is too fast compared to the consumers?**
  The producer is not allowed to produce to more items into the buffer than what the buffer can hold. To enforce this, the producer thread will wait for an isEmpty signal from any consumer thread. It can add a new item only after it receives this signal.
- We faced a weird issue where Peach would reject the order of the produced items if the sleep factor of the consumer threads is by default multiplied by 100. The workaround was to simply remove the 100 times multiplicative factor. Problem could be reproduced by simply setting rsleep(100*NROF_CONSUMERS) in the consumer thread. We need more clarity on this one.