Arjon Arts 1521950
Erwin van Veenschoten 1524348

Assignment 3 - Condition Variables

This document describes the design of the third assignment of the 2INC0 course by Arjon Arts en Erwin van Veenschoten.

There are several global variables used in this program. First, there is of course the buffer. Secondly, the *item_count* keeps track of the number of items that are currently in the buffer. Thirdly, the *production_index* keeps track of the buffer index to add the next produced Item. Fourthly, the *consumption_index* keeps track of the buffer index of the next Item to consume. The indexes are incremented every time the producer puts a new item in the buffer and every time the consumer retrieves an item from the buffer. The modulo operator in combination with *BUFFER_SIZE* converts this value to the appropriate index. Fifthly, a global variable, *called next_item_to_produce*, is used to make sure items are added in ascending order. Lastly, two condition variables are used for conditional signaling between the consumer and producer(s). These are called buffer_empty and buffer_full.

The condition variable buffer_full is used to let producers wait whenever the buffer is full AND/OR the item isn't the next item to be put in the buffer. When the consumer thread obtains an item from the buffer, it signals to one of the producers to continue. The producer checks whether it's obtained item is the next item to put in the buffer. If this is indeed the case, the program will continue. If not, the producer will signal another producer and wait again.

The condition variable buffer_empty is used to let the consumer wait whenever the buffer is empty. When a producer puts an item in the buffer, the consumer is signalled and can continue execution.

To ensure exclusive access to the shared data a mutex is used. Since all the mentioned global variables are referenced in (very) close proximity to each other, the decision has been made to use only a single mutex, called *critical_section.* This will ensure exclusive access to the shared data, while keeping waiting to a minimum.

The program flow is as follows. At startup, the main thread launches the producer thread(s) are along with the consumer threads and wait for them to finish. The producer/consumer threads handle the rest as described above. When a producer obtains an item that is equal to NROF_ITEMS it will put this in the buffer and it will finish executing. The consumer will get the items equal to NROF_ITEMS several times and signal the remaining producers. When this number equals the number of producer threads, this will mean all producers have finished executing. The consumer thread can now finish executing as well and the program ends. This will allow the main thread to exit.