Concurrent Systems II

Practical 2
Producer–Consumer & Condition Variables

February 4, 2014

This practical is worth 3% of your year-end result. Have your program ready one week from today—i.e. have it ready for inspection on February 11 (MCS students) or February 12 (MAI students). Have a printout also.

Write a complete threaded program in C (or non-OO C++) on a Linux machine—e.g. `stoker.cs.tcd.ie`—to implement a simulated producer, a number of simulated consumers and a simulated buffer of limited capacity connecting the producer to the consumers. Use condition variables to prevent buffer overflow and underflow.

Basically the idea is that the producer 'produces' items that are placed in the buffer as they are produced, so long as the buffer is not already full. If the buffer is full, the producer must wait until space becomes available in it. The consumer 'consumes' items by removing them from the buffer and 'using' them. If the buffer is empty, then the consumer must wait, using a condition variable, until an item becomes available.

Use condition variables for signalling between the producer and consumers—do not use sleep-wait polling.

- Devise and implement scenarios and to show that neither overflow nor underflow occurs.
- Show also that use of processor time is minimised.
- Explain whether the consumers are treated "fairly"—if so, how; if not, why not.

The exact details of the items and of their production and consumption are not important, and can be simulated as follows:

- The items themselves don't actually have to exist, but the *number* of items in the buffer does need to be simulated. It can be an integer. 'Adding' and 'removing' items could be simulated by incrementing or decrementing the number of items in the buffer.
- Production and consumption of an item could be simulated by a pseudo-random delay in the producer or consumer at the point of 'production' or 'consumption'.

(`http://www.scss.tcd.ie/CourseModules/CS3015/Assets/Practicals/p2/practical.pdf`)