Computer Science Honours 2016

Distributed & Parallel Processing
Practical Two: JCSP

Bounded-Buffer Producer Consumer using Message Passing

1) If you have not yet done so, first compile and run the simple Producer-Consumer program that we studied in lectures using Java and the JCSP (Java Communicating Sequential Processes) library.

You can download the JCSP library from RUconnected. To install JCSP simply type the following command using a command-line prompt: `jar xvf jcsp1-0-rc4.jar`
For more details, refer to the *jcsp-docs* document structure set up by the installation. See also the example code unpacked by the installation. Additional tutorial material on the JCSP package can also be downloaded from the JCSP website site : `http://www.cs.kent.ac.uk/projects/ofa/jcsp/`
You can also refer to a producer-consumer example in a Java reference of your choice if you wish, but remember that you should use the process creation structures of JCSP for this exercise, and not native Java threads.

2) Once you have your own version of the simple JCSP Producer-Consumer program running, make the following modifications to it, in order:

a) Add a buffer process to the system, which runs concurrently with the producer and consumer, and which catches the output of the producer, and passes it on to the consumer.

b) Modify all 3 concurrent processes so that they handle more than just one item (just use a loop to generate 100 integers in the producer). The point of a buffer is that it should allow the rate of production and consumption to be decoupled, so extend the buffer process to store incoming items in a 10 element array. Remember that to allow the producer and consumer to communicate with the buffer at their own convenience, the buffer needs to include an *alternative construct*. See the `select` method in the `Alternative` class. You will also need to cater for condition synchronisation to avoid buffer underflow or overflow.

c) Now that you have a working buffer to handle one producer and one consumer, it should be a relatively simple task to add one more of each kind of processor. Producer 1 should produce integers in the range 1 to 100, and producer 2 in the range 101 to 200. The two consumers should each consume 100 integers and display them on the console. Use the same code for both Producers (passing a parameter to tell them what to produce).

d) Unless you have already been very careful to keep count of the number of items produced, buffered, and consumed by each concurrent process, you will find that your program hangs after it produces its results and has to be interrupted with a CTRL-C. This is because one or more processes have not terminated cleanly. Clean termination is not a compulsory part of this assignment, but if you do add a mechanism to ensure that all processes terminate cleanly at the end of the program, I will award a bonus mark for it (counting up to 200 in the buffer process and then dying won't qualify for bonus marks — look for a general solution).

e) What can you say about the tasking mechanism of Java and JCSP from the

patterns printed out by each consumer?

To Hand In: You should submit a copy of the final version of the program with the bounded buffer, and two producers and two consumers, with sample output and a short explanation of why the output patterns look the way they do.