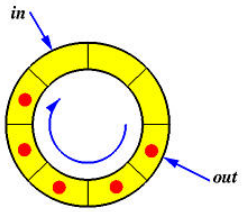
COP 4600 - Operating Systems Design
Spring 2015
Professor Sumi Helal
Lab 1: Process Coordination
Due: 11:59 PM, February 8, 2015

I. Objectives

- Learn about classical synchronization problems known in concurrent processing. Specifically, you will learn about the consumer-producer problem and the Dining Philosopher problem.
- You will implement solutions to these problems under Xinu using Xinu semaphores.

II. The Producer-Consumer Problem

Two processes are sharing a circular buffer (queue), one produces at the tail of the circular queue (the Producer) and the other consumes from the head (the Consumer).

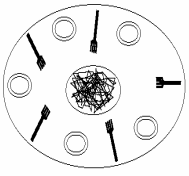


(a) Analyze what can go wrong considering that the two processes are sharing access to the buffer, and are running at possibly different speed (priorities), for varying bursts of CPU processing times. Then, based on your analysis, list all possible problems or situations that may arise and impact the Producer, the Consumer or both.

(b) You are asked to use Xinu semaphores to implement the producer-consumer problem correctly (meaning, no problems or faulty situations). You are to implement a process using *main()*, which defines and allocates the shared circular buffer, then creates both the producer and the consumer processes. You should program both processes with "simulation code" to actually produce or consume. Both processes should output to the console what they are producing or consuming.

III. The Dining Philosopher Problem

You will learn about the dinning philosophers problem in the class. You will learn about deadlock (and starvation).



(c) You are asked to implement a five-philosophers instance of the problem and develop a solution that may or may not involve the use of semaphores. However, your solution should be deadlock free. You should provide a short description as comments within your program to explain why your solution is deadlock free.

IV. Submission

You should submit your two *main()* programs via Sakai. Nothing else in the Xinu code should be changed. The TA's will replace Xinu's main by your mains, one at a time, and will compile and test, in addition of course to inspecting the code itself.