**Operating Systems**

**Homework – Multiplexing I/O**

This homework extends the previous homework: the producer-consumer server.

You recall that in order to enforce a maximum number of simultaneous producer or consumer threads required some sort of concurrency control mechanism, because the total number of producers and the total number of consumers should both be shared (or global variables). They cannot be checked or changed in the main thread, because the server has to do a read from the client (to see if the client is a producer or consumer) and cannot risk blocking. So this read is pushed to the client handler thread; thus the critical section problem appears.

Suppose instead that your server uses multiplexing (with select or poll) to manage all the clients until the point that they identify themselves. Once the main thread reads the message PRODUCER or CONSUMER, which is a fast action, it can pass control to a thread to take care of the longer actions (exchanging the items).

So, write a new server: the server and clients use exactly the same protocol, and the producer-consumer problem is handled exactly as before. The only difference in this server, is that it handles its clients using multiplexing up to the point that they identify themselves. Once it is clear whether a client is a producer, consumer or exceeds one of the limits, the client is either hung up on (close socket) or sent to a thread.

Slides and sample server code is posted.