**Operating Systems**

**Final Project Part 3**

The final project consists of modifications to the producer-consumer system. For part 4, you will make some changes to the server, in preparation for the last part (part 5).

In part 2, you created the possibility that some clients would connect to the server, but not send the welcome message until later (bad or slow clients). In this new server version, the server will be able to reject these misbehaving clients. In the real world, servers might do this because clients who are unresponsive are holding a valuable resource – a socket. Maybe such clients are slow because of evil intent (DDOS) or error conditions or simply a slow network, but the server wants to get rid of them.

You can use the ability of the select system call’s timeout argument to let the server wake up periodically, even if no clients are active, and check for clients that have been accepted but not sent anything. You can read the man page about how to use the timeout argument. You can tell if select has woken up due to a timeout (rather than due to activity on one of the sockets) by examining its return value, which returns the number of file descriptors with activity on them (number in the returned fd\_sets), or -1 on error.

In order to know if a client has been around too long, you must know when it arrived and how much time has passed. There is a new defined constant called REJECT\_TIME in prodcon.h. If a client is found to have been around longer than REJECT\_TIME seconds without announcing if it is a producer or a consumer, then its socket should be closed. This rejection interval does not have to be precise; that is, a server does not have to reject a client immediately upon it reaching REJECT\_TIME idle seconds. This means a busy server does not have to spend time checking every client every time select returns; on the other hand, it should not let idle clients linger too much past their allowed time. You may decide how to make these tradeoffs yourself.

Now that the server is making all these decisions, it should keep track of its status:

* Current producers in the system (already tracked)
* Current consumers in the system (already tracked)
* Current total clients (already tracked)

Since server started:

* Total producers served – who reached the point of placing their item in the item buffer
* Total consumers served – who reached the point of removing an item from the item buffer
* Total clients rejected for exceeding client max
* Total clients rejected for taking too long to identify themselves
* Total producers rejected for exceeding producer max
* Total consumers rejected for exceeding consumer max