Assignment #4 – Memory Management and Critical Regions

Assigned: November 7, 2018

Due: November 23, 2018

In this assignment, the student will create a client/server application pair that will use shared memory, and semaphores, to implement a simulation of a data acquisition system. Your solution should be free of potential race conditions in terms of the use of shared memory between client and server.

The server application will act as a simulation of the data acquisition system. The server will create a block of shared memory, 1024 bytes in size. The application will use this 1024 byte buffer as a circular buffer, constantly writing new data into the buffer at all times. The data will consist of random letters of the alphabet, from A to T (20 different values to select from during the randomization).

These random characters will be placed into the next available spot in the shared memory buffer, and when the end of the buffer is reached, the server will automatically wrap back to the beginning of the buffer.

The server will generate 32 random characters every ¼ second. Thus, the server will need to use Linux APIs to generate ¼ second pauses before the next batch of random characters are generated. Thus it will take 8 seconds for the 1024 byte buffer to be fully refreshed with data.

The server will run forever, or until an appropriate signal arrives to terminate the program.

The client application will perform these actions:

- connect to the shared memory, and if no shared memory exists, launch the server, wait one second, and attempt to connect to the shared memory again. If an error occurs the second time, inform the end user that the client/server application suite is no longer working.
- The client will loop forever, reading data in from the shared memory. The job of the client is to generate a histogram (a character based graphical display of data) that indicates the frequency of occurrence of the 20 different letters of the alphabet. This histogram will be refreshed visually on the screen every 2 seconds. Scale the histogram to fit the width of the terminal window appropriately (for example, an asterisk could represent 16 counts of a letter, allowing for 64 asterisks to represent a worst case of 1024 of the same piece of data in the buffer).
- The client program will run until the SIGINT signal is received. Once this signal arrives, catch the signal, and terminate the server with a similar signal.

At the due date, submit a copy of the makefile and source code to both the client and the server applications to the drop box for this assignment.

Marking Scheme:

Item	Category	Comments	Missing	Poorly implemented or not functional	Meets expectation	Exceeds expectation
1	Server completeness	Features implemented as specified	0	4	8	10

2	Client completeness	Features implemented as specified	0	4	8	10
3	Code Maintainability	Code designed for long term maintainability	0	2	4	5
4	Code Documentation	Appropriate use of commenting for source files, functions, etc.	0	4	8	10
5	Errors / Bugs	1 mark deducted per bug, maximum 5				
6	Total (max 40)					

This assignment is weighted as 8% of final grade.