CS39002: Operating Systems Lab
Spring 2017

Assignment 3
Due: Part 1 on February 6, 2017, 11:59 pm, Part 2 on February 13, 2017, 1 pm

1. In this program, you'll write a program to solve the *m*-producer *n*-consumer problem, *m, n* >= 1. You have a shared circular buffer that can hold 20 integers. Each of the producer processes stores the numbers 1 to 50 in the buffer one by one (in a for loop with 50 iterations) and then exits. Each of the consumer processes reads the numbers from the buffer and adds them to a shared variable SUM (initialized to 0). Any consumer process can read any of the numbers in the buffer. The only constrain is that **every number written by some producer should be read exactly once by exactly one of the consumers**. Of course, a producer should not write when the buffer is full and a consumer should not read when the buffer is empty.

   Write a program that first creates the shared circular buffer and the shared variable SUM using the shm*() calls in Linux. You can create any other shared variable that you think you may need. Also, create any semaphores that you may need. The program then reads in the value of *m* and *n* from the user, and then forks *m* producers and *n* consumers. The producer and consumer codes can be written as functions that are called by the child processes. After all the producers and consumers have finished (the consumers exit after all the data produced by all the producers have been read. How does a consumer know this?), the parent process prints the value of SUM. Note that the value of SUM should be m*25*51 if your program is correct.

   Test your program with (a) m=1, n=1, (b) m=1, n=3, (c) m=3, n=1, and (d) m=3, n=2. Submit only the C file.

2. Consider a file containing a maximum of 100 student records. The record for each student is kept in one line in the file and contains the following, separated by one or more spaces: First_name (ascii string, one word with no spaces in between, max. 20 characters), Last_Name (ascii string, one word with no spaces in between, max 20 charaters), Roll_No (integer), CGPA (float). Roll no. is unique for a student, but others may be the same for two students.

   A process X, when started, first loads all the records in shared memory from a file whose name is passed to it as a command line argument. It then goes to sleep, waking up periodically (say every 5 seconds) to check if any of the data have been modified (how will it know that?). If yes, it writes the entire content of the shared memory back to the file. The process X runs forever.

Any number of other processes can query and update the data by reading/writing the shared memory. Specifically, a querying process Y is allowed to do the following operations on the data:

- Search for a student record by roll no. (should show all data for that student)
- Update the CGPA of a particular student (the student is identified by the roll no., which is unique for each student)
- Exit

The process Y will keep on doing the above two in a loop until the exit option is chosen. Note that any number of copies of Y can be started at one time (think of this as multiple people searching/updating student records at the same time). Also, the following conditions must be ensured:

- If any copy of Y is started before X is started, it should wait for X to be started first.
- Only X should create and initialize all necessary shared memory/semaphores

Write two C programs, X.c for the process X and Y.c for the process Y. Your programs must be well-commented to make it easier for anyone to understand what semaphores you are using and how they are used.

Submit the files X.c and Y.c.

(Suggestion: First write the code assuming X will start first and create and initialize all necessary shared memory/semaphore before any of the Y's start. Test it thoroughly. Then think what needs to change to relax the assumption.)