CSci 5103, Fall 2016
Assignment 4
Programming with POSIX
Due October 25, 2016

This assignment must be done individually.

Objectives:

The objectives of this assignment are the following: Acquire familiarity with using POSIX thread management primitives. Understand and use LINUX functions for creating shared memory between two LINUX processes. How to write POSIX thread programs, where threads execute in different processes and communicate through shared memory. Read the manual pages for the LINUX primitives for creating shared memory between two LINUX processes. These functions are -- *shmget* and *shmat*. Please study and execute the sample programs posted on the course webpage to create shared memory between two LINUX processes using these functions. For sample programs on how to use the POSIX thread library and its shared memory management, look at the Examples section of this class's web page.

Problem Statement:
In this assignment you will write a POSIX program. Consider a system comprising of three producer processes, one consumer process, and a shared buffer of size 2 items. The items to be deposited in the buffer will be strings.

Each producer process randomly creates a specific colored item: *producer_red* creates *RED* colored items, *producer_black* creates *BLACK* colored items, and *producer_white* creates *WHITE* colored items. Associated with each item created by a specific producer is a local LINUX timestamp in microseconds (use the *gettimeofday()* function) which records the time that item was placed into the buffer. Each producer deposits its item into the buffer, and the consumer retrieves the items from the buffer. The items produced by the producers will be of any of the three strings below:
"RED timestamp", "BLACK timestamp", "WHITE timestamp"

Each producer process, after successfully depositing an item into the buffer, will write that item (essentially the "COLOR timestamp" string) into its log file called Producer_COLOR.txt (e.g. Producer_RED.txt). The consumer process will then retrieve the items from this shared buffer. When it retrieves an item, the consumer process will write that item (essentially the "COLOR timestamp" string) into a file called Consumer.txt.

Requirements for the program:

1. Each producer process will produce its respective colored item (represented as a string "COLOR timestamp"). If the producer process finds the shared buffer has space, it will first create a timestamp for that item. The producer process will then deposit the item into the buffer and also write the item into its log file (Producer_COLOR.txt). If the buffer does not have space, the producer will have to wait. Each string (or item) written into the log file will be on a new line.
2. The consumer process will remove an item from the buffer, and write its contents to a text-file named Consumer.txt. Each string should be written in the order it is read from the buffer, and on a new line. The consumer process waits if the buffer is empty.
3. Each producer process should produce 1000 items. Hence, at the end of running the program, the consumer should have written 3000 items into its log file.

Programming Problem 1 (50 points):

Write the code for this system as a single LINUX process containing four POSIX threads. The functions of each producer and consumer should be performed by one separate POSIX thread.

Items to be include in your submission:

(1) Code for your program. This should be in a file named prod_cons.c.
(2) A makefile used to compile your code
(3) A Readme.txt file with instructions on how to run your program

Programming Problem 2 (50 points):

The same problem is now solved using four LINUX processes, one for each of the three producers and one for the consumer. All these four processes will all be executed on the same machine. These processes will use a shared memory space to maintain the bounded buffer. In this shared memory you will have to also maintain synchronization variables such as mutex locks, condition variables, counters, status flags etc.

The main program, in a file named main.c, will fork these four processes. One of these will execute the consumer process code (source for which will be in a file named consumer.c), and the three will execute the producer process code (source for which will be in a file named producer.c). While forking the producer processes, make sure that the main function passes the "color" as a parameter to each of the producer processes, so that each of the three producer processes knows what colored items it will create. The main program will also create a shared memory segment between these four processes. After creating these processes, it will wait for all of them to terminate.

Items to be included in your submission:

(1) Files for code - named main.c, producer.c, and consumer.c.
(2) A makefile used to compile your code
(3) A Readme.txt file with instructions on how to run your program

.

For both the problems your program execution should create four files: Producer_RED.txt, Producer_BLACK.txt, Producer_WHITE.txt , and Consumer.txt.

Submission Instructions:

You will submit only one TAR file which will have two directories "problem1" and "problem2" containing your solutions for the above two problems, with the three items listed above for each of them.

IMPORTANT THINGS TO PAY ATTENTION:

You program must be properly commented, documented, and indented for readability. Up to 15% of the grade is allocated for it. Your program must compile, or significant points will be deducted.
You must test your program for correct execution on a CSELab Linux machine.