CSCE 451/851 Fall 2023 Assignment 3:

FCFS Reader/Writer PROGRAMMING

100 points

Due: Wednesday, Nov 8, 2023, 11:59pm

This assignment consists of writing a C program for implementing semaphore-based for the FCFS version of the reader-writer problem. The FCFS version works as follows: readers and writers access the shared file in the FCFS order; if multiple readers arrive at the system back-to-back, these readers can read the file concurrently when it is their turn to read. It can be assumed that threads in every queue associated with a semaphore is managed with the FCFS policy.

The goal of this assignment is to practice developing semaphores-based applications. Furthermore, practice some system calls and process management by the operating system.

Details

You need to Develop C program to implement your semaphore-based to the FCFS version of the reader-writer problem using the POSIX semaphore. Name your programs as "Assign3_sem.c" to refer to the semaphore solution. The program should be complied correctly on the linux-cse-01 machine. Suppose the name of the compiled executable file is "assign3sem_exe". Your program should have the following functionalities:

1. It accepts 10 arguments from the command-line, denoted as b1, b2, …, b10. Each bi is either 0 or 1. If bi is 0, it means the ith arriving thread is a reader; otherwise, it is a writer. For example, the programs may be launched as:

   ```
   cse ...> assign3sem _exe 0 1 0 0 1 0 0 0 0 1
   ```

   meaning your programs should solve the FCFS version of the reader-writer problems for the situation that 10 reader/writer threads (i.e., the 2nd, 5th and 10th ones are writers while others are readers for this example) arrive at the system (i.e., start running) in the specified order.

2. After accepting the 10 arguments from the command-line, the program should create 10 reader/writer threads and start them in the specified order. Once created, these threads run concurrently. Each reader thread performs one reading operation and each writer thread performs one writing operating.

   Performing a reading or a writing operation is simulated with making the thread sleeps for 1. When a reader (or writer) starts or ends reading (or writing), a message should be printed. For the example shown in (1), the message should be like “Reader 1 starts reading”, …, “Reader 1 ends reading”, …, “Writer 2 starts writing”, …, “Writer 2 ends writing”, ….

The program will need to use **pthreads to create threads**, and **sleep system** call to simulate time elapse, in addition to the semaphore functions for the semaphore solution.

Bonus (Extra 20 points)

Create the same semaphore solution for **processes not pthreads**. Note that semaphores and some variables used in synchronization should be shared by all the reader/writer processes; hence, you will need to use functions in the POSIX shared memory library to create and manipulate the shared semaphores and variables in the memory shared by these processes. Name the programs as “Extra_sem.c” to refer to the semaphore solution. Furthermore, the name of the compiled executable file is “Extrasem_exe”.

Submission

- You should use C to develop the code.
- You can work on this assignment **in Pairs**.
- You need to turn in electronically by submitting a zip file named: LastnameFirstname_Assignment3.zip.
- **Source code must include proper documentation to receive full credit (you will lose 10% of your score, if the code is not well documented).**
- All Assignments require the use of a **make file** such that the grader will be able to compile/build your executable by simply typing “make” or some simple command that you specify in your readme file.
- **A readme file to specify how to use make file to compile, pseudocode for your semaphores, your partner’s name and acknowledgment for others who you made a high-level discussion about the assignment with**.
- **Source code must compile and run correctly on the department machine "linux-cse-01", which will be used by the TA for grading. If your program compiles, but does not run correctly on cse, you will lose 50% of the assignment score. If your program doesn’t compile at all on cse, you will lose 100% of the assignment score.**
- You are responsible for thoroughly testing and debugging your code. The TA may try to break your code by subjecting it to bizarre test cases.

- **Grading criteria:**

  - Semaphore solution: 50 pts
  - Make file: 10 pts
  - Documentation: 10 pts
  - Readme file: 30 pts

- You can have multiple submissions, but the TA will grade only the last one.

Start as early as possible!