Project 2

ESE 333 Real-Time Operating System, Spring 2018
Instructor: Prof. Fan YE
fan.ye@stonybrook.edu

Follow the outline given in Figure $2-28$ of the textbook and implement producer-consumer problem using shared memory and semaphores. Your program should copy a file named file1 to another file named $file2$ byte by byte through a shared buffer. Your main process should fork two processes: one is producer and another is consumer. The main process then pauses waiting for the signals from its children. The producer's job is to move the bytes in $file1$ to a shared buffer (implemented by shared memory), one byte at a time. Insert random delay between two movings. The consumer's job is to move the bytes in the shared buffer to $file2$, one byte at a time. You should also insert random delay between two movings.

Whenever the producer moves an item into the shared buffer, it sends a signal to main process. Main process enters a "P" in the monitoring table. Similarly, whenever the consumer moves an item from the shared buffer, it sends a signal to main process. Main process then enters a "C" in the monitoring table. When producer and consumer have copied the entire file1, the consumer sends a signal to main process. Main process outputs the monitoring table.

Implement the above in the UNIX environment using C or C++ [1]. You may need system calls : **signal, kill, pause, sleep, semget, semctl, semop, shmget, shmat, shmctl**, etc. When sending signals, you should use the two signals defined for users: SIGUSR1 and SIGUSR2 for the producer and consumer respectively.

NOTE: Your code should be well documented so that others may read and understand your code. You should give instructions on how to run your program, as well as indicate the machines you used. The TA will test your program on UNIX Lab machines.
This is an **INDIVIDUAL** project. You should work independently and submit your own program. You should email your program to the TA with subject title "ESE333 Project 2" before deadline.

[1]The following hints can be helpful to your project:
- Read the manual page for a system call, e.g., 'man signal'. A detailed manual explaining the system call API will be shown.
- A good URL on share memory: `http://stackoverflow.com/questions/5656530/how-to-use-shared-memory-with-linux-in-c`
- A book "Advanced linux programming, 2001". Inside the book, the following sections are particularly useful:
  -pg 52-55 3.3 signals: On how to set signal handlers
  -pg 55 3.4 process termination: on how to use kill to send signal to a process
  -pg 96 5.1 shared memory: on how to create, attach, detach a shared memory
  -pg 101 5.2 processes semaphores: on how to create, operate, deallocate semaphores