Lab Assignment (Even PC)

02:00 – 04:00pm, 28-March-2014

Reader-Writer problem

In this assignment, you are asked to implement the Reader-Writer problem with Reader's priority. In this system, you have two readers who read from a shared buffer and one writer who updates the buffer. The readers are to be implemented with the help of child processes and the writer as the parent process. Implement the shared buffer with the help of a shared file called *buffer.txt*. The buffer contains a variable VAR which is initialized to zero. After each access, the writer reads the content of the buffer, prints it (along with its own PID), and increments the content by one. On the other hand, each reader periodically reads the content of the buffer, and prints the current value along with its own identity (PID). Use semaphore(s) in order to ensure proper synchronization between the Reader and Writer processes. Print appropriate messages whenever a process (reader or write) gains control of the buffer. Insert appropriate delays (*sleep()* or *usleep()*) so that the messages become visible. The program terminates when the writer writes MAX to VAR (fix a value in MAX at the beginning). After this happens, the parent kills the reader processes, and itself terminates.

Notice that in the Reader-Priority mode, both the Readers can access the shared content together. The Reader who first makes a read operation should lock the shared content for reading. During the time the shared content is locked for reading, the other Reader can read it. The last Reader done with reading releases the lock. The Writer can write only when no Reader is reading the shared content. Finally, before the Writer finishes writing, no Reader can read.

Sample run (Parent ID: 123, Child IDs: 124, 125)

```
File created.  VAR is 0
First Reader enters
Reader (124) reads the value 0
Last Reader leaves
Writer enters
Writer (123) writes the value 1
Writer leaves
First reader enters
Reader (125) reads the value 1
Reader (124) reads the value 1
Last Reader leaves
```

Submit one single file *rdrwtr.c*.

Lab Assignment (Odd PC)

02:00 – 04:00pm, 28-March-2014

Producer-Consumer Problem

In this assignment, you are asked to implement the producer-consumer problem. In the system, you have one producer process and one consumer process accessing a shared buffer of limited size BMAX. The producer and the consumer are the child and the parent processes, respectively. Implement the shared buffer (circular queue) with the help of a shared memory (initially empty, containing –1). While executing, the producer process produces an item (essentially, the item number, starting from 0) and inserts it in the buffer. The consumer process, while executing, removes the item from the buffer and replaces it by –1. You must ensure that the order, in which items are removed by the consumer, must follow the order of insertion (by the producer). The producer must wait once the buffer is full (number of items reaches BMAX). After the consumer consumes an item, it wakes up the producer. Likewise, the consumer process waits when the buffer is empty, and is woken up by the producer after a data item is inserted.

Print appropriate messages to indicate the production and consumption sequences. Insert appropriate delays (*sleep()* or *usleep()*) so that the messages become visible.

Sample Run (BMAX=5)

```
Producer  inserts item : 0
Producer  inserts item : 1
Producer  inserts item : 2
Producer  inserts item : 3
Consumer consumes item :  0
Producer  inserts item : 4
Producer  inserts item : 5
Producer  inserts item : Buffer  FULL!
Producer waits
Consumer consumes item : 1
Consumer consumes item : 2
Consumer consumes item : 3
Producer  inserts item :  6
Consumer consumes item : 4
Consumer consumes item : 5
Consumer consumes item : 6
Consumer consumes item : Buffer EMPTY!
Consumer waits
```

Submit one single file *prodcons.c*.