```c
/* prodcons-sem.c - Producer Consumer problem using POSIX Semaphores */
/* include main */
#include    <stdio.h>
#include    <unistd.h>
#include    <fcntl.h>
#include    <pthread.h>
#include    <semaphore.h>
#include    <sys/types.h>
#define     NBUFF       10
#define     SEM_MUTEX   "mutex"          /* these are args to px_ipc_name() */
#define     SEM_NEMPTY  "nempty"
#define     SEM_NSTORED "nstored"
int nitems; /* read-only by producer and consumer */
struct {    /* data shared by producer and consumer */
  int buff[NBUFF];
  sem_t     *mutex;
  sem_t     *nempty;
  sem_t     *nstored;
} shared;
void *produce(void *), *consume(void *);
int main(int argc, char **argv)
{
      pthread_t   tid_produce, tid_consume;
      if (argc != 2)
      {
            printf("usage: prodcons1 <#items>");
            exit(1);
      }
      nitems = atoi(argv[1]);
            /* 4create three semaphores */
      shared.mutex = sem_open(SEM_MUTEX, O_CREAT | O_EXCL, 0644, 1);
      shared.nempty = sem_open(SEM_NEMPTY, O_CREAT | O_EXCL, 0644, NBUFF);
      shared.nstored = sem_open(SEM_NSTORED, O_CREAT | O_EXCL, 0644, 0);
            /* 4create one producer thread and one consumer thread */
      pthread_setconcurrency(2);
      pthread_create(&tid_produce, NULL, produce, NULL);
      pthread_create(&tid_consume, NULL, consume, NULL);
            /* 4wait for the two threads */
      pthread_join(tid_produce, NULL);
      pthread_join(tid_consume, NULL);
            /* 4remove the semaphores */
      sem_unlink(SEM_MUTEX);
      sem_unlink(SEM_NEMPTY);
      sem_unlink(SEM_NSTORED);
      exit(0);
}
/* end main */
/* include prodcons */
void *produce(void *arg)
{
      int         i;
      for (i = 0; i < nitems; i++) {
            sem_wait(shared.nempty);      /* wait for at least 1 empty slot */
            sem_wait(shared.mutex);
            shared.buff[i % NBUFF] = i;   /* store i into circular buffer */
            sem_post(shared.mutex);
            sem_post(shared.nstored);     /* 1 more stored item */
      }
      return(NULL);
}
void *consume(void *arg)
{
      int         i;
      for (i = 0; i < nitems; i++) {
            sem_wait(shared.nstored);           /* wait for at least 1 stored item */
            sem_wait(shared.mutex);
            if (shared.buff[i % NBUFF] == i)
```

```
                  printf("buff[%d] = %d\n", i, shared.buff[i % NBUFF]);
            sem_post(shared.mutex);
            sem_post(shared.nempty);            /* 1 more empty slot */
      }
      return(NULL);
}
/* end prodcons */
/*
[vishnu@mannava pxsem]$ ./a.out 5
buff[0] = 0
buff[1] = 1
buff[2] = 2
buff[3] = 3
buff[4] = 4
*/
```