```
#define MAXNITEMS      512000
#define MAXNTHREADS        50

int  nitems;                              /* Read only */
struct {
     pthread_mutex_t  mutex;
     int              buff[MAXNITEMS];
     int              nput;
     int              nval;
} shared = {
     PTHREAD_MUTEX_INITIALIZER
};

void *producer(void *), *consumer(void *);
```

```
int main(int argc, char **argv)
{
     int             i, nthreads, count[MAXNTHREADS];
     pthread_t       tid_prod[MAXNTHREADS], tid_cons;

     nitems = min(atoi(argv[1], MAXNITEMS);
     nthreads = min(atoi(argv[2], MAXNTHREADS);

     for (i=0; i<nthreads; i++) {
         count[i] = 0;
         pthread_create(&tid_prod[i], NULL,
                        producer, &count[i]);
     }
```

```
    for (i=0; i<nthreads; i++) {
        pthread_join(tid_prod, NULL);
        printf("count[%d] = %d\n", i, count[i]);
    }

    pthread_create(&tid_cons, NULL,
                   consumer, NULL);
    pthread_join(tid_cons, NULL);

    exit(0);
}
```

PRODUCER FUNCTION

```
void *producer(void *arg)
{
     for ( ; ; ) {
         pthread_mutex_lock(&shared.mutex);
         if (shared.nput >= nitems) {
              pthread_mutex_unlock(&shared.mutex);
              return(NULL);
         }
         shared.buff[shared.nput] = shared.val;
         shared.nput++;
         shared.nval++;
         pthread_mutex_unlock(&shared.mutex);
         *((int *) arg) += 1;
     }
}
```

CONSUMER FUNCTION

```
void *consume(void *arg)
{
     int     i;

     for (i=0; i<nitems; i++) {
         if (shared.buff[i] != i)
              printf("buff[%d] = %d\n",
                     shared.buff[i]);
     }
     return(NULL);
}
```