

```

/*
 *   File      : pc.c
 *
 *   Title     : Demo Producer/Consumer.
 *
 *   Short    : A solution to the producer consumer problem using
 *             pthreads.
 *
 *   Long     :
 *
 *   Author    : Andrae Muys
 *
 *   Date     : 18 September 1997
 *
 *   Revised  :
 */

#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define QUEUESIZE 10
#define LOOP 20

void *producer (void *args);
void *consumer (void *args);

typedef struct {
    int buf[QUEUESIZE];
    long head, tail;
    int full, empty;
    pthread_mutex_t *mut;
    pthread_cond_t *notFull, *notEmpty;
} queue;

queue *queueInit (void);
void queueDelete (queue *q);
void queueAdd (queue *q, int in);
void queueDel (queue *q, int *out);

int main ()
{
    queue *fifo;
    pthread_t pro, con;

    fifo = queueInit ();
    if (fifo == NULL) {
        fprintf (stderr, "main: Queue Init failed.\n");
        exit (1);
    }
    pthread_create (&pro, NULL, producer, fifo);
    pthread_create (&con, NULL, consumer, fifo);
    pthread_join (pro, NULL);
    pthread_join (con, NULL);
    queueDelete (fifo);

    return 0;
}

void *producer (void *q)
{
    queue *fifo;
    int i;

    fifo = (queue *)q;

    for (i = 0; i < LOOP; i++) {

```

```

pthread_mutex_lock (fifo->mut);
while (fifo->full) {
    printf ("producer: queue FULL.\n");
    pthread_cond_wait (fifo->notFull, fifo->mut);
}
queueAdd (fifo, i);
pthread_mutex_unlock (fifo->mut);
pthread_cond_signal (fifo->notEmpty);
usleep (100000);
}
for (i = 0; i < LOOP; i++) {
    pthread_mutex_lock (fifo->mut);
    while (fifo->full) {
        printf ("producer: queue FULL.\n");
        pthread_cond_wait (fifo->notFull, fifo->mut);
    }
    queueAdd (fifo, i);
    pthread_mutex_unlock (fifo->mut);
    pthread_cond_signal (fifo->notEmpty);
    usleep (200000);
}
return (NULL);
}

void *consumer (void *q)
{
    queue *fifo;
    int i, d;

    fifo = (queue *)q;

    for (i = 0; i < LOOP; i++) {
        pthread_mutex_lock (fifo->mut);
        while (fifo->empty) {
            printf ("consumer: queue EMPTY.\n");
            pthread_cond_wait (fifo->notEmpty, fifo->mut);
        }
        queueDel (fifo, &d);
        pthread_mutex_unlock (fifo->mut);
        pthread_cond_signal (fifo->notFull);
        printf ("consumer: recieved %d.\n", d);
        usleep (200000);
    }
    for (i = 0; i < LOOP; i++) {
        pthread_mutex_lock (fifo->mut);
        while (fifo->empty) {
            printf ("consumer: queue EMPTY.\n");
            pthread_cond_wait (fifo->notEmpty, fifo->mut);
        }
        queueDel (fifo, &d);
        pthread_mutex_unlock (fifo->mut);
        pthread_cond_signal (fifo->notFull);
        printf ("consumer: recieved %d.\n", d);
        usleep (50000);
    }
    return (NULL);
}

#ifdef 0
typedef struct {
    int buf[QUEUESIZE];
    long head, tail;
    int full, empty;
    pthread_mutex_t *mut;
    pthread_cond_t *notFull, *notEmpty;
} queue;
#endif

```

```

queue *queueInit (void)
{
    queue *q;

    q = (queue *)malloc (sizeof (queue));
    if (q == NULL) return (NULL);

    q->empty = 1;
    q->full = 0;
    q->head = 0;
    q->tail = 0;
    q->mut = (pthread_mutex_t *) malloc (sizeof (pthread_mutex_t));
    pthread_mutex_init (q->mut, NULL);
    q->notFull = (pthread_cond_t *) malloc (sizeof (pthread_cond_t));
    pthread_cond_init (q->notFull, NULL);
    q->notEmpty = (pthread_cond_t *) malloc (sizeof (pthread_cond_t));
    pthread_cond_init (q->notEmpty, NULL);

    return (q);
}

void queueDelete (queue *q)
{
    pthread_mutex_destroy (q->mut);
    free (q->mut);
    pthread_cond_destroy (q->notFull);
    free (q->notFull);
    pthread_cond_destroy (q->notEmpty);
    free (q->notEmpty);
    free (q);
}

void queueAdd (queue *q, int in)
{
    q->buf[q->tail] = in;
    q->tail++;
    if (q->tail == QUEUESIZE)
        q->tail = 0;
    if (q->tail == q->head)
        q->full = 1;
    q->empty = 0;

    return;
}

void queueDel (queue *q, int *out)
{
    *out = q->buf[q->head];

    q->head++;
    if (q->head == QUEUESIZE)
        q->head = 0;
    if (q->head == q->tail)
        q->empty = 1;
    q->full = 0;

    return;
}

```