

```

/*
 *   File      : rw.c
 *
 *   Title     : Demo Readers/Writer.
 *
 *   Short    : A solution to the multi-reader's, one writer problem.
 *
 *   Long     :
 *
 *   Author   : Andrae Muys
 *
 *   Date     : 18 September 1997
 *
 *   Revised  :
 */

```

```

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

```

```

#define MAXCOUNT 5

```

```

#define READER1 50000
#define READER2 100000
#define READER3 400000
#define READER4 800000
#define WRITER1 150000

```

```

typedef struct {
    pthread_mutex_t *mut;
    int writers;
    int readers;
    int waiting;
    pthread_cond_t *writeOK, *readOK;
} rwl;

```

```

rwl *initlock (void);
void readlock (rwl *lock, int d);
void writelock (rwl *lock, int d);
void readunlock (rwl *lock);
void writeunlock (rwl *lock);
void deletelock (rwl *lock);

```

```

typedef struct {
    rwl *lock;
    int id;
    long delay;
} rwargs;

```

```

rwargs *newRWargs (rwl *l, int i, long d);
void *reader (void *args);
void *writer (void *args);

```

```

static int data = 1;

```

```

int main ()
{
    pthread_t r1, r2, r3, r4, w1;
    rwargs *a1, *a2, *a3, *a4, *a5;
    rwl *lock;

    lock = initlock ();
    a1 = newRWargs (lock, 1, WRITER1);
    pthread_create (&w1, NULL, writer, a1);
    a2 = newRWargs (lock, 1, READER1);
    pthread_create (&r1, NULL, reader, a2);
    a3 = newRWargs (lock, 2, READER2);

```

```

pthread_create (&r2, NULL, reader, a3);
a4 = newRWargs (lock, 3, READER3);
pthread_create (&r3, NULL, reader, a4);
a5 = newRWargs (lock, 4, READER4);
pthread_create (&r4, NULL, reader, a5);
pthread_join (w1, NULL);
pthread_join (r1, NULL);
pthread_join (r2, NULL);
pthread_join (r3, NULL);
pthread_join (r4, NULL);
free (a1); free (a2); free (a3); free (a4); free (a5);

return 0;
}

rwargs *newRWargs (rwl *l, int i, long d)
{
    rwargs *args;

    args = (rwargs *)malloc (sizeof (rwargs));
    if (args == NULL) return (NULL);
    args->lock = l; args->id = i; args->delay = d;
    return (args);
}

void *reader (void *args)
{
    rwargs *a;
    int d;

    a = (rwargs *)args;

    do {
        readlock (a->lock, a->id);
        d = data;
        usleep (a->delay);
        readunlock (a->lock);
        printf ("Reader %d : Data = %d\n", a->id, d);
        usleep (a->delay);
    } while (d != 0);
    printf ("Reader %d: Finished.\n", a->id);

    return (NULL);
}

void *writer (void *args)
{
    rwargs *a;
    int i;

    a = (rwargs *)args;

    for (i = 2; i < MAXCOUNT; i++) {
        writelock (a->lock, a->id);
        data = i;
        usleep (a->delay);
        writeunlock (a->lock);
        printf ("Writer %d: Wrote %d\n", a->id, i);
        usleep (a->delay);
    }
    printf ("Writer %d: Finishing...\n", a->id);
    writelock (a->lock, a->id);
    data = 0;
    writeunlock (a->lock);
    printf ("Writer %d: Finished.\n", a->id);

    return (NULL);
}

```

```

rwl *initlock (void)
{
    rwl *lock;

    lock = (rwl *)malloc (sizeof (rwl));
    if (lock == NULL) return (NULL);
    lock->mut = (pthread_mutex_t *) malloc (sizeof (pthread_mutex_t));
    if (lock->mut == NULL) { free (lock); return (NULL); }
    lock->writeOK =
        (pthread_cond_t *) malloc (sizeof (pthread_cond_t));
    if (lock->writeOK == NULL) { free (lock->mut); free (lock);
        return (NULL); }
    lock->readOK =
        (pthread_cond_t *) malloc (sizeof (pthread_cond_t));
    if (lock->writeOK == NULL) { free (lock->mut); free (lock->writeOK);
        free (lock); return (NULL); }

    pthread_mutex_init (lock->mut, NULL);
    pthread_cond_init (lock->writeOK, NULL);
    pthread_cond_init (lock->readOK, NULL);
    lock->readers = 0;
    lock->writers = 0;
    lock->waiting = 0;

    return (lock);
}

void readlock (rwl *lock, int d)
{
    pthread_mutex_lock (lock->mut);
    if (lock->writers || lock->waiting) {
        do {
            printf ("reader %d blocked.\n", d);
            pthread_cond_wait (lock->readOK, lock->mut);
            printf ("reader %d unblocked.\n", d);
        } while (lock->writers);
    }
    lock->readers++;
    pthread_mutex_unlock (lock->mut);

    return;
}

void writelock (rwl *lock, int d)
{
    pthread_mutex_lock (lock->mut);
    lock->waiting++;
    while (lock->readers || lock->writers) {
        printf ("writer %d blocked.\n", d);
        pthread_cond_wait (lock->writeOK, lock->mut);
        printf ("writer %d unblocked.\n", d);
    }
    lock->waiting--;
    lock->writers++;
    pthread_mutex_unlock (lock->mut);

    return;
}

void readunlock (rwl *lock)
{
    pthread_mutex_lock (lock->mut);
    lock->readers--;
    pthread_cond_signal (lock->writeOK);
    pthread_mutex_unlock (lock->mut);
}

```

```
void writeunlock (rwl *lock)
{
    pthread_mutex_lock (lock->mut);
    lock->writers--;
    pthread_cond_broadcast (lock->readOK);
    pthread_mutex_unlock (lock->mut);
}

void deletelock (rwl *lock)
{
    pthread_mutex_destroy (lock->mut);
    pthread_cond_destroy (lock->readOK);
    pthread_cond_destroy (lock->writeOK);
    free (lock);

    return;
}
```