```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>


// Shared variables

int readcount = 0;      // Number of active readers

pthread_mutex_t mutex;   // Protects readcount

pthread_mutex_t wrt;     // Controls access to shared resource


// Shared resource (for demonstration)

int shared_data = 0;


// Reader function

void *reader(void *arg)

{

    int reader_id = *(int *)arg;


    while (1) {

        // Entry Section for Reader

        pthread_mutex_lock(&mutex);

        readcount++;

        if (readcount == 1)

            pthread_mutex_lock(&wrt);  // First reader locks writer access

        pthread_mutex_unlock(&mutex);


        // Critical Section for Reader

        printf("Reader %d is reading the shared data: %d\n", reader_id, shared_data);

        sleep(1); // simulate reading time


        // Exit Section for Reader
        pthread_mutex_lock(&mutex);
        readcount--;
        if (readcount == 0)
            pthread_mutex_unlock(&wrt); // Last reader unlocks writer access
        pthread_mutex_unlock(&mutex);

        sleep(2); // simulate time before trying to read again
    }

    return NULL;
}

// Writer function
void *writer(void *arg)
{
    int writer_id = *(int *)arg;

    while (1) {
        pthread_mutex_lock(&wrt); // Writer locks access
        shared_data++;
        printf("Writer %d modified the shared data to: %d\n", writer_id, shared_data);
        sleep(2); // simulate writing time
        pthread_mutex_unlock(&wrt);

        sleep(3); // simulate time before trying to write again
    }

    return NULL;
}

int main()
```

```
{
    pthread_t rtid[3], wtid[2]; // 3 readers, 2 writers
    int reader_ids[3] = {1, 2, 3};
    int writer_ids[2] = {1, 2};

    pthread_mutex_init(&mutex, NULL);
    pthread_mutex_init(&wrt, NULL);

    // Create reader threads
    for (int i = 0; i < 3; i++)
        pthread_create(&rtid[i], NULL, reader, &reader_ids[i]);

    // Create writer threads
    for (int i = 0; i < 2; i++)
        pthread_create(&wtid[i], NULL, writer, &writer_ids[i]);

    // Join threads (this program runs indefinitely)
    for (int i = 0; i < 3; i++)
        pthread_join(rtid[i], NULL);
    for (int i = 0; i < 2; i++)
        pthread_join(wtid[i], NULL);

    pthread_mutex_destroy(&mutex);
    pthread_mutex_destroy(&wrt);

    return 0;
}
```

output

```
Reader 1 is reading the shared data: 0

Reader 2 is reading the shared data: 0

Reader 3 is reading the shared data: 0

Writer 1 modified the shared data to: 1

Reader 1 is reading the shared data: 1

Reader 2 is reading the shared data: 1

Writer 2 modified the shared data to: 2

Reader 3 is reading the shared data: 2

Reader 1 is reading the shared data: 2

Reader 2 is reading the shared data: 2

Writer 1 modified the shared data to: 3

Writer 2 modified the shared data to: 4

Reader 3 is reading the shared data: 4

...
```