```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

int readcount = 0;          // Number of readers currently reading
int data = 0;               // Shared resource
pthread_mutex_t mutex;      // Protects readcount
pthread_mutex_t wrt;        // Controls access to shared resource

void* reader(void* arg) {
  int id = *(int*)arg;
  while (1) {
    pthread_mutex_lock(&mutex);
    readcount++;
    if (readcount == 1)
      pthread_mutex_lock(&wrt);  // First reader locks writers
    pthread_mutex_unlock(&mutex);

    // ---- Reading section ----
    printf("Reader %d: reading data = %d\n", id, data);
    usleep(500000); // Simulate reading time (0.5 sec)

    pthread_mutex_lock(&mutex);
    readcount--;
    if (readcount == 0)
      pthread_mutex_unlock(&wrt);  // Last reader unlocks writers
    pthread_mutex_unlock(&mutex);
```

```
        // -------------------------

        sleep(1); // Reader waits before trying again
    }
    pthread_exit(NULL);
}

void* writer(void* arg) {
    int id = *(int*)arg;
    while (1) {
        pthread_mutex_lock(&wrt);

        // ---- Writing section ----
        data++;
        printf("Writer %d: writing data = %d\n", id, data);
        usleep(700000); // Simulate writing time (0.7 sec)
        // ------------------------

        pthread_mutex_unlock(&wrt);
        sleep(2); // Writer waits before next write
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t rtid[3], wtid[2];
    int rid[3] = {1, 2, 3};
    int wid[2] = {1, 2};
    pthread_mutex_init(&mutex, NULL);
    pthread_mutex_init(&wrt, NULL);

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

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

    // Join threads (optional since infinite loops)
    for (int i = 0; i < 2; i++)
        pthread_join(wtid[i], NULL);
    for (int i = 0; i < 3; i++)
        pthread_join(rtid[i], NULL);

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

    return 0;
}
```