First Reader-Writer Problem: No reader is kept waiting unless a writer has already obtained permission to use the shared object.

semaphore rw_mutex =1; (used as mutual exclusion for writers, also used by the **first** or **last** reader that enters or exits the critical section)

Mutex mutex; (Initially unlock; used for protecting critical section)

int read_count=0;

Reader:	**Writer:**
while (true) { **lock(mutex);** **read_count++;** **if (read_count==1) wait(rw_mutex);** **unlock(mutex);** **/* Reading is performed */** **lock(mutex);** **read_count- -;** **if (read_count==0) signal(rw_mutex);** **unlock(mutex);** **}**	**while(true) {** **wait(rw_mutex);** **/* Writing is performed*/** **signal(rw_mutex);** **}**

Second Reader-Writer Problem: No writer is kept waiting longer than absolutely necessary.

```
int readcount, writecount; (initial value = 0)
semaphore w, r ; (initial value = 1)
Mutex mutex_1, mutex_2, mutex_3; (initially unlocked)
```

Reader:

```
While (true){
  lock(mutex_3);
    wait(r);
      lock(mutex_1);
        readcount := readcount + 1;
        if readcount == 1 then wait(w);
      unlock(mutex_1);
    signal(r);
  unlock(mutex_3);

//  reading is performed

  lock(mutex_1);
    readcount := readcount - 1;
    if readcount == 0 then signal(w);
  unlock(mutex_1);
}
```

Writer:

```
while(true) {
  lock(mutex_2);
    writecount := writecount + 1;
    if writecount == 1 then wait(r);
  unlock(mutex_2);

  wait(w);
     // writing is performed
  signal(w);

  lock(mutex_2);
    writecount := writecount - 1;
    if writecount == 0 then signal(r);
  unlock(mutex_2);
}
```