Reader Writer problem

Solution #1 (Readers with higher priority)

```
int readcount = 0
mutex read_mutex; // used for updating readcount var.
Semaphore write_sem = 1; // used to make sure that only one writer can write at a given time and make sure that writer cannot interfere with reader.

void reader() {
 while (true) {

    read_mutex.lock(); // we want to update the read count
    readcount++;
    if (readcount == 1)
       sem_wait(write_sem); // to make sure no writer can enter while the current reader is inside the CS.
    read.mutex.unlock();

    Read(); // we can do this because it is guaranteed that no writers are in CS.

    read_mutex.lock();
    readcount--;
    if (readcount == 0)    //no reader in CS
       Sem_signal(write_sem); // writers can enter the CS
    read_mutex.unlock(); // reader leaves.
}
```

```
void writer() {
    while (true) {
        Sem_wait(writer_Sem); //requests CS
        Write();
        Sem_signal(writer_sem); //leaves CS.
    }

Void main() {
    readcount = 0;
    parbegin(reader, writer);
}
```

Solution 02 (Writers with higher priority)

```
int readcount;
int writecount;
mutex r_mutex, w_mutex; //used for updating readcount & writecount.
Semaphore readblock=1, writeblock=1, writepending=1;

void writer() {
  while (true) {
    w_mutex.lock();
    writercount++;
    if (writercount == 1)
       Sem_wait(readblock);
    w_mutex.unlock();
   Sem_wait(writeblock);
   Write();
   Sem_signal(writeblock);
   w_mutex.lock();
   writecount--;
   if (writecount==0)
        sem_signal(readerblock)
   w_mutex.unlock();
}
```

```
void reader() {
  while(true) {
    sem_wait(writepending); // If a reader is blocked at readblock then all the other subsequent readers are blocked at writepending semaphore.
    sem_wait(readblock);
    r_mutex.lock()
    readcount++;
    if(readcount==1)
      sem_wait(writeblock); // first reader blocks writers for other readers.
    r_mutex.unlock();
    sem_signal(readblock); // releases readblock so that other readers or writers can use it
    sem_signal(writepending); // writepending=1 again.
    Read();
    r_mutex.lock();
    readcount--;
    if(readcount==0)
      sem_signal(writerblock); // If no other readers are available then writer block is released.
    r_mutex.unlock();
}

void main() {
  readcount, writecount=0;
  parbegin(reader, writer);
}
```