Lin/Snyder, *Principles of Parallel Programming*, Figure 6.12: Fixes

```
1    int readers;                        // Neg value=> active writer
1.5  int readWaiters = 0;
2    pthread_mutex_t lock;
3    pthread_cond_t rBusy, wBusy;        // Use separate conditional vars
4                                        // for readers and writers
5    AcquireExclusive()
6    {
7      pthread_mutex_lock(&lock);
8      while(readers !=0)
9      {
10       pthread_cond_wait(&wBusy, &lock);
11     }
12     readers=-1;
13     pthread_mutex_unlock(&lock);
14   }
15
16   AcquireShared()
17   {
18     pthread_mutex_lock(&lock);

20     while(readers<0)
21     {
21.5     readWaiters++;
22       pthread_cond_wait(&rBusy, &lock);
22.5     readWaiters--;
23     }
24     readers++;
25     pthread_mutex_unlock(&lock);
26   }
27
28   ReleaseExclusive()
29   {
30     pthread_mutex_lock(&lock);
31     readers=0;
31.5   if (readWaiters==0)                    // If there are no waiting readers
31.6       pthread_cond_signal(&wBusy);       // Wake up a writer
31.7   else
32         pthread_cond_broadcast(&rBusy); // Wake up all readers
33     pthread_mutex_unlock(&lock);
34   }
35
36   ReleaseShared()
37   {
38     int doSignal;
39
40     pthread_mutex_lock(&lock);
41     readers--;
42     doSignal=(readers==0)
43     pthread_mutex_unlock(&lock);
44     if(doSignal)                       // Signal executes outside
45     {                                  // of critical section
46       pthread_cond_signal(&wBusy);     // Wake up writer
47     }
48   }
```