

Feb 23, 10 17:14

I11-handout.txt

Page 1/2

```

1 Handout for CS 372H
2 Class 11
3 23 February 2010
4
5 1. Readers/writers
6
7     state variables:
8         AR = 0; // # active readers
9         AW = 0; // # active writers
10        WR = 0; // # waiting readers
11        WW = 0; // # waiting writers
12
13        Condition okToRead = NIL;
14        Condition okToWrite = NIL;
15        Mutex mutex = FREE;
16
17        Database::read() {
18            startRead(); // first, check self into the system
19            Access Data
20            doneRead(); // check self out of system
21        }
22
23        Database::startRead() {
24            acquire(&mutex);
25            while((AW + WW) > 0){
26                WR++;
27                wait(&okToRead, &mutex);
28            }
29            AR++;
30            release(&mutex);
31        }
32
33        Database::doneRead() {
34            acquire(&mutex);
35            AR--;
36            if (AR == 0 && WW > 0) { // if no other readers still
37                signal(&okToWrite, &mutex); // active, wake up writer
38            }
39            release(&mutex);
40        }
41
42
43        Database::write(){ // symmetrical
44            startWrite(); // check in
45            Access Data
46            doneWrite(); // check out
47        }
48
49        Database::startWrite() {
50            acquire(&mutex);
51            while ((AW + AR) > 0) { // check if safe to write.
52                // if any readers or writers, wait
53                WW++;
54                wait(&okToWrite, &mutex);
55                WW--;
56            }
57            AW++;
58            release(&mutex);
59        }
60
61        Database::doneWrite() {
62            acquire(&mutex);
63            AW--;
64            if (WW > 0) {
65                signal(&okToWrite, &mutex); // give priority to writers
66            } else if (WR > 0) {
67                broadcast(&okToRead, &mutex);
68            }
69            release(&mutex);
70        }
71
72        NOTE: what is the starvation problem here?
73

```

Feb 23, 10 17:14

I11-handout.txt

Page 2/2

```

74 2. Shared locks
75
76     struct sharedlock {
77         int i;
78         Mutex mutex;
79         Cond c;
80     };
81
82     void AcquireExclusive (sharedlock *sl) {
83         acquire(&sl->mutex);
84         while (sl->i) {
85             wait (&sl->c, &sl->mutex);
86         }
87         sl->i = -1;
88         release(&sl->mutex);
89     }
90
91     void AcquireShared (sharedlock *sl) {
92         acquire(&sl->mutex);
93         while (sl->i < 0) {
94             wait (&sl->c, &sl->mutex);
95         }
96         sl->i++;
97         release(&sl->mutex);
98     }
99
100    void ReleaseShared (sharedlock *sl) {
101        acquire(&sl->mutex);
102        if (!--sl->i)
103            signal (&sl->c, &sl->mutex);
104        release(&sl->mutex);
105    }
106
107    void ReleaseExclusive (sharedlock *sl) {
108        acquire(&sl->mutex);
109        sl->i = 0;
110        broadcast (&sl->c, &sl->mutex);
111        release(&sl->mutex);
112    }
113
114    QUESTIONS:
115    A. There is a starvation problem here. What is it? (Readers can keep
116        writers out if there is a steady stream of readers.)
117    B. How could you use these shared locks to write a cleaner version
118        of the code in item 1., above? (Though note that the starvation
119        properties would be different.)

```