## Operating Systems Project 14

Haukur Óskar Þorgeirsson hth152@hi.is Matthías Páll Gissurarson mpg3@hi.is

28. febrúar 2013

1. Following is the modified code.

## RaceThreadsSemaphore.java

```
import java.util.concurrent.Semaphore;
2
   public class RaceThreadsSemaphore
3
4
            volatile int in;
5
            public int COUNT MAX;
            private Semaphore sem = new Semaphore(1);
6
7
            //If we had 2 here, 2 threads would enter the
                critical section at the same time.
8
9
            public class CountThread extends Thread
10
                public CountThread(String name)
11
12
                         super(name);
13
14
15
                    public void run()
16
                             for(int i = 0; i < COUNT MAX; i)
17
                                 ++)
                             {
18
                                 sem.acquireUninterruptibly()
19
                                 System.out.print(this.
20
                                     getName());
21
                                 System.out.flush();
22
                                 in++;
23
                                 sem.release();
24
                             }
                    }
25
            }
26
27
28
            public static void main(String[] args)
29
30
                     RaceThreadsSemaphore x = new
                        RaceThreadsSemaphore(Integer.
                        parseInt(args[0]);
```

```
}
31
32
            public RaceThreadsSemaphore(int max)
33
34
35
                      this.COUNT MAX = max;
36
                     Thread counter1 = new CountThread("0");
37
                     Thread counter2 = new CountThread("1");
                      \mathbf{this}.in = 0;
38
39
                      counter1.start();
40
41
                      counter2.start();
42
43
                     \mathbf{try}
44
                              counter1.join();
45
46
                              counter2.join();
47
                      } catch (InterruptedException ie) {
                         return;}
48
                      System.out.println();
                      System.out.println("COUNT MAX_=_" + max
49
                         + "\nin_=_" + in + "\n2*COUNT_MAX_-_
                         in = " + (2*max - in));
50
            }
51
```

- 2. See code in 1.
- 3. The following was the sequence of execution. As we can see, there is some interleaving.

4. As can be seen from the comment in 1, initiating the semaphore with 2 instead of 1 would allow 2 threads to run at once, however, this would break the program and not give us the correct result. To fix that however, one could implement some Mutex for access to the variable, but then we'd have effectively one thread at a time.