Natalie Cluck
04/3/2017

CSCE-313 Spring 2017 Quiz 8 [20 points]

Solutions must be posted by 11:59pm Mon April 3 on eCampus to receive credit. Late submissions are not allowed. You can print, write, scan and upload to eCampus. Or you can type answers and upload. In either case, please upload your answers in PDF format.

Q1. [4 points] A multi-threaded program generates an incorrect answer some of the time raising the possibility of a race condition. Which of the following are steps that can reduce or even eliminate race conditions in the program? **Underline the R, E, or U below to indicate that the given approach can Reduce race conditions, Eliminate them, or is Useless when it comes to race conditions. Write a sentence to explain your answer.**

[✓ R] E U Separate the multithread program into multiple single-threaded programs, and run each thread in its own process. Share data between the programs via named pipes and read and write calls. No other changes to the program.

R E [✓ U] Same as above, but share data between the programs via shared memory segments.

[✓ R] E U Apply the “one-writer” rule. Here you modify the program so that each variable has only one writer, i.e. only one thread can write to a shared variable. Multiple threads can read from it.

R [✓ E] U Ensure that each shared variable is protected by some lock.

Q2. [6 points] Consider three concurrently executing threads in the same process using two semaphores s1 and s2. Assume s1 has been initialized to 1, while s2 has been initialized to 0. **What are all possible values of the global variable x, initialized to 0, after all three threads have terminated? Show your work in reaching the answer.**

B,C,A (x = 6) or C,A,B (x = 36) or C,B,A (x = 18).

```
/* thread A */
s2.P()
s1.P()
x = x*2
s1.V()
```

```
/* thread B */
s1.P()
x = x*x
s1.V()


/* thread C */
s1.P()
x = x+3
s2.V()
s1.V()
```

Q3. Reader-Writer [10 points]: There is 1 thread that writes to a file, and a number of other threads that can simultaneously read the file, but never the readers and the write together (e.g., at one time, 3 reader threads are running, at another time the writer thread is running). Write pseudo code using semaphores for both the Reader() and Writer() functions. Remember that there is no data dependency between the readers and the writer (i.e., it is not a producer-consumer problem), the only condition is that they do not happen at the same time. **Hint**: *The first reader to get access waits until the ongoing writer (if any) finishes, performs the read operation and leaves. The last reader, just before leaving, should give control/access back to the writer (if any) waiting. Think about the initial values of the semaphores.*

```
semaphore s1=1;
semaphore s2=1;
readcount=0;

Writer() {
    s1.start();

    <critical section>

    <exit section>
    s1.end();
}

Reader() {
    s2.start();

    <critical section>

    readcount++;
    if (readcount == 1)
        resource.P();

    <exit critical section>
    s2.end();

    s2.start();

    <CRITICAL Section>
    readcount--;
    if (readcount == 0)
        s1.start();
    <EXIT CRITICAL Section>
    s2.end();
}
```