CSE 430 Summer 2014 - Project 2

Ryan Dougherty - ASU ID: 1203621947

Abstract

This document will cover Project 2, which is about the Readers-Writers problem. We will observe the output produced by the code given, then modify the critical section in Database.java to implement the starvation-free pseudo-code algorithm (but is unfair), and finally modify Database.java further (using RWQueue.java) so that our implementation promotes fairness among the readers and writers.

Note: All modified code (for Parts 2 and 3) is provided at the end of the document.

1 Part 1 Output

The Part 1 output is the following (one sample):

```
reader 2 is sleeping.
writer 1 is sleeping.
writer 0 is sleeping.
reader 1 is sleeping.
reader 0 is sleeping.
writer 1 wants to write.
writer 1 is writing.
reader 0 wants to read.
writer 0 wants to write.
reader 1 wants to read.
reader 0 is reading. Count = 1
writer 1 is done writing.
writer 1 is sleeping.
reader 1 is reading. Count = 2
reader 2 wants to read.
reader 2 is reading. Count = 3
writer 1 wants to write.
```

As we can see, reader 0 reads while writer 1 is still writing (writer 1 finishes writing after reader 0 reads). Therefore, we do not have guaranteed mutual exclusion.

2 Part 2 Output

The Part 2 output is the following (one sample):

```
reader 1 is sleeping.
writer 0 is sleeping.
reader 2 is sleeping.
reader 0 is sleeping.
writer 1 is sleeping.
writer 0 wants to write.
writer 0 is writing.
reader 2 wants to read.
reader 1 wants to read.
writer 1 wants to write.
writer 0 is done writing.
reader 2 is reading. Count = 1
reader 1 is reading. Count = 2
writer 0 is sleeping.
reader 2 is done reading. Count = 1
reader 2 is sleeping.
writer 0 wants to write.
reader 0 wants to read.
reader 2 wants to read.
reader 1 is done reading. Count = 0
reader 1 is sleeping.
writer 1 is writing.
reader 1 wants to read.
writer 1 is done writing.
reader 0 is reading. Count = 1
reader 2 is reading. Count = 2
reader 1 is reading. Count = 3
writer 1 is sleeping.
reader 2 is done reading. Count = 2
reader 2 is sleeping.
```

As we can see, at no point does a writer start writing then have a reader read before the writer finishes. Therefore, we have guaranteed mutual exclu-

sion. Also, neither readers nor writers starve (multiple readers can read simultaneously). Therefore, we have guaranteed no starvation of readers or writers. However, we can see that there are certain readers (such as 1) that get more time reading than other readers, making our implementation of the Readers-Writers problem unfair. This behavior can be further seen if we increase the number of readers (there were 3 readers and 2 writers in this example). Part 3 output solves this problem by allowing fairness.

3 Part 3 Output

The Part 3 output is the following (one sample):

```
reader 0 is sleeping.
writer 0 is sleeping.
reader 2 is sleeping.
reader 1 is sleeping.
writer 1 is sleeping.
reader 1 wants to read.
reader 1 is reading. Count = 1
reader 0 wants to read.
reader 0 is reading. Count = 2
writer 1 wants to write.
reader 2 wants to read.
writer 0 wants to write.
reader 1 is done reading. Count = 1
reader 1 is sleeping.
reader 1 wants to read.
reader 0 is done reading. Count = 0
reader 0 is sleeping.
writer 1 is writing.
reader 0 wants to read.
writer 1 is done writing.
reader 2 is reading. Count = 1
writer 1 is sleeping.
reader 2 is done reading. Count = 0
reader 2 is sleeping.
writer 0 is writing.
writer 1 wants to write.
writer 0 is done writing.
reader 1 is reading. Count = 1
writer 0 is sleeping.
```

```
reader 0 is reading. Count = 2
reader 2 wants to read.
reader 1 is done reading. Count = 1
reader 1 is sleeping.
reader 0 is done reading. Count = 0
reader 0 is sleeping.
writer 1 is writing.
reader 0 wants to read.
reader 1 wants to read.
writer 0 wants to write.
```

As we can see, we still have mutual exclusion and no starvation guarantees as was in Part 2. But now, we can see that no reader reads nor writer writes any more than another. Therefore, we have guaranteed fairness.

4 Part 2 Modified Code

All of the Part 2 modified code can be found in "part2modifiedcode.txt".

```
/**
 ...
 */
public class Database {
        public Database() {
                mutex = new Semaphore(1);
                db = new Semaphore(1);
                rsem = new Semaphore(0);
                wsem = new Semaphore(0);
                wwc = 0;
                wc = 0;
                rwc = 0;
                rc = 0;
        }

        // readers and writers will call this to nap
        public static void napping() {
                // unchanged
        }

        public int startRead() {
                mutex.P();
```

```
        if (wwc > 0 || wc > 0) {
                rwc++;
                mutex.V();
                rsem.P();
                rwc--;
        }
        rc++;
        if (rwc > 0) {
                rsem.V();
        } else {
                mutex.V();
        }
        return rc;
}

public int endRead() {
        mutex.P();
        rc--;
        if (rc == 0 && wwc > 0) {
                wsem.V();
        } else {
                mutex.V();
        }
        return rc;
}

public void startWrite() {
        mutex.P();
        if (rc > 0 || wc > 0) {
                wwc++;
                mutex.V();
                wsem.P();
                wwc--;
        }
        wc++;
        mutex.V();
}

public void endWrite() {
        mutex.P();
```

```
                wc--;
                if (rwc > 0) {
                        rsem.V();
                } else {
                        if (wwc > 0) {
                                wsem.V();
                        } else {
                                mutex.V();
                        }
                }
        }

        Semaphore mutex;  // controls access to readerCount
        Semaphore db;     // controls access to the database
        Semaphore rsem, wsem;
        private int wwc; // writer waiting counter
        private int  wc; // writer counter
        private int  rwc; // reader waiting counter
        private int  rc; // reader counter
        private static final int NAP_TIME = 15;
}
```

5 Part 3 Modified Code

All of the Part 3 modified code can be found in "part3modifiedcode.txt".

```
/**
 ...
 */
public class Database {
        public Database() {
                mutex = new Semaphore(1);
                db = new Semaphore(1);
                rsem = new Semaphore(0);
                wsem = new Semaphore(0);
                queue = new RWQueue();
                wwc = 0;
                wc = 0;
                rwc = 0;
                rc = 0;
```

```
}

// readers and writers will call this to nap
public static void napping() {
        // unchanged
}

public int startRead() {
        mutex.P();
        if (wwc > 0 || wc > 0) {
                rwc++;
                mutex.V();
                queue.enqueue();
                rwc--;
        }
        rc++;
        if (queue.dequeueReader() == false) {
                mutex.V();
        }
        return rc;
}

public int endRead() {
        mutex.P();
        rc--;
        if (rc > 0) {
                mutex.V();
        } else {
                Release();
        }
        return rc;
}

public void startWrite() {
        mutex.P();
        if (rc > 0 || wc > 0) {
                wwc++;
                mutex.V();
                queue.enqueue();
                wwc--;
        }
```

```
                wc++;
                mutex.V();
        }

        public void endWrite() {
                mutex.P();
                wc--;
                Release();
        }

        private void Release() {
                if (rwc > 0 || wwc > 0) {
                        while (true) {
                                if (queue.dequeueWriter() == false) {
                                if (queue.dequeueReader() == false) {
                                        continue;
                                }
                                }
                                break;
                        }
                } else {
                        mutex.V();
                }
        }

        Semaphore mutex;  // controls access to readerCount
        Semaphore db;     // controls access to the database
        Semaphore rsem, wsem;
        RWQueue queue;
        private int wwc; // writer waiting counter
        private int wc; // writer counter
        private int rwc; // reader waiting counter
        private int rc; // reader counter
        private static final int NAP_TIME = 15;
}
```