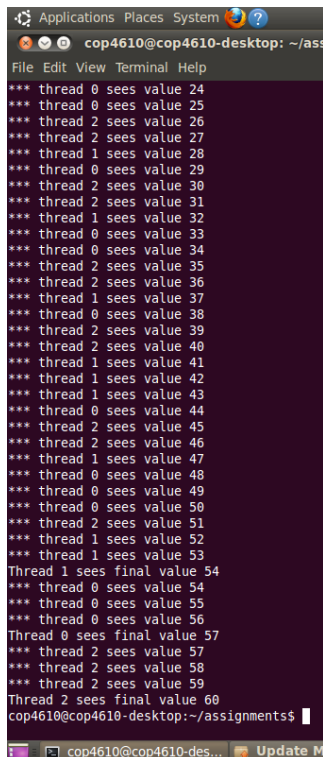


## Operating Systems: Lab 2

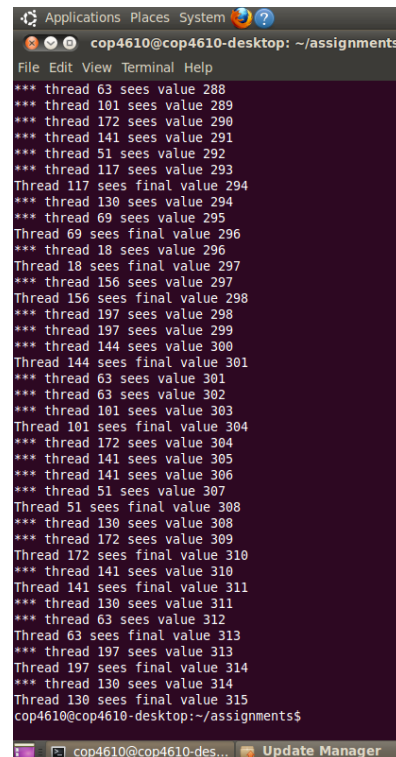
### Report

#### Step 1.1: Simple Multi-threaded Programming without Synchronization



```
*** thread 0 sees value 24
*** thread 0 sees value 25
*** thread 2 sees value 26
*** thread 2 sees value 27
*** thread 1 sees value 28
*** thread 0 sees value 29
*** thread 2 sees value 30
*** thread 2 sees value 31
*** thread 1 sees value 32
*** thread 0 sees value 33
*** thread 0 sees value 34
*** thread 2 sees value 35
*** thread 2 sees value 36
*** thread 1 sees value 37
*** thread 0 sees value 38
*** thread 2 sees value 39
*** thread 2 sees value 40
*** thread 1 sees value 41
*** thread 1 sees value 42
*** thread 1 sees value 43
*** thread 0 sees value 44
*** thread 2 sees value 45
*** thread 2 sees value 46
*** thread 1 sees value 47
*** thread 0 sees value 48
*** thread 0 sees value 49
*** thread 0 sees value 50
*** thread 2 sees value 51
*** thread 1 sees value 52
*** thread 1 sees value 53
Thread 1 sees final value 54
*** thread 0 sees value 54
*** thread 0 sees value 55
*** thread 0 sees value 56
Thread 0 sees final value 57
*** thread 2 sees value 57
*** thread 2 sees value 58
*** thread 2 sees value 59
Thread 2 sees final value 60
cop4610@cop4610-desktop: ~/assignments$
```

Unsynchronized run (3 threads)



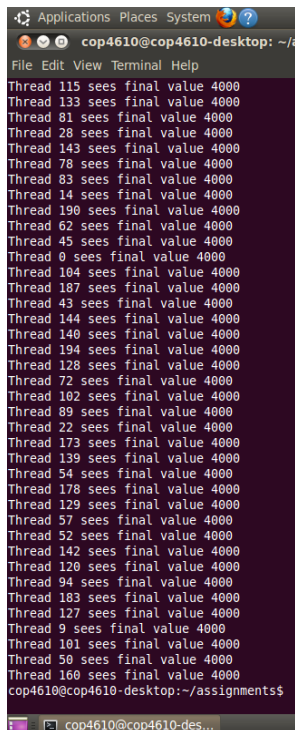
```
*** thread 63 sees value 288
*** thread 101 sees value 289
*** thread 172 sees value 290
*** thread 141 sees value 291
*** thread 51 sees value 292
*** thread 117 sees value 293
Thread 117 sees final value 294
*** thread 130 sees value 294
*** thread 69 sees value 295
Thread 69 sees final value 296
*** thread 18 sees value 296
Thread 18 sees final value 297
*** thread 156 sees value 297
Thread 156 sees final value 298
*** thread 197 sees value 298
*** thread 197 sees value 299
*** thread 144 sees value 300
Thread 144 sees final value 301
*** thread 63 sees value 301
*** thread 63 sees value 302
*** thread 101 sees value 303
Thread 101 sees final value 304
*** thread 172 sees value 304
*** thread 141 sees value 305
*** thread 141 sees value 306
*** thread 51 sees value 307
Thread 51 sees final value 308
*** thread 130 sees value 308
*** thread 172 sees value 309
Thread 172 sees final value 310
*** thread 141 sees value 310
Thread 141 sees final value 311
*** thread 130 sees value 311
*** thread 63 sees value 312
Thread 63 sees final value 313
*** thread 197 sees value 313
Thread 197 sees final value 314
*** thread 130 sees value 314
Thread 130 sees final value 315
cop4610@cop4610-desktop: ~/assignments$
```

Unsynchronized run (200 threads)

In both execution of the programs, it was noticeable that the final value emitted by all the threads were different values, which was different from the expectation that each thread would emit the same value. This is because each thread is allowed to indeterminately sleep and wakeup for an unknown amount of time each loop iteration, which compounds to each thread having a different ending time span size. It was also noticeable in the execution of the program, especially with larger thread amounts, that the value becomes corrupted. For example, considering that on the second case, 200 threads were ran, each with 20 loop iterations encased within the thread, it

was expected that the final value would be  $(\text{thread\_count} * \text{loop iterations}) = 4000$ , but this was simply not the case. This was because the critical section of the program (which accessed the SharedVariable, incrementing it), spun across multiple threads, was not guarded against concurrent execution, nor was there any mechanism to ensure that the critical section was ran as an entire transactional unit.

## Step 1.2: Simple Multi-threaded Programming with Proper Synchronization



```
Applications Places System
cop4610@cop4610-desktop: ~/z
File Edit View Terminal Help
Thread 115 sees final value 4000
Thread 133 sees final value 4000
Thread 81 sees final value 4000
Thread 28 sees final value 4000
Thread 143 sees final value 4000
Thread 78 sees final value 4000
Thread 83 sees final value 4000
Thread 14 sees final value 4000
Thread 190 sees final value 4000
Thread 62 sees final value 4000
Thread 45 sees final value 4000
Thread 0 sees final value 4000
Thread 104 sees final value 4000
Thread 187 sees final value 4000
Thread 43 sees final value 4000
Thread 144 sees final value 4000
Thread 140 sees final value 4000
Thread 194 sees final value 4000
Thread 128 sees final value 4000
Thread 72 sees final value 4000
Thread 162 sees final value 4000
Thread 89 sees final value 4000
Thread 22 sees final value 4000
Thread 173 sees final value 4000
Thread 139 sees final value 4000
Thread 54 sees final value 4000
Thread 178 sees final value 4000
Thread 129 sees final value 4000
Thread 57 sees final value 4000
Thread 52 sees final value 4000
Thread 142 sees final value 4000
Thread 120 sees final value 4000
Thread 94 sees final value 4000
Thread 183 sees final value 4000
Thread 127 sees final value 4000
Thread 9 sees final value 4000
Thread 101 sees final value 4000
Thread 50 sees final value 4000
Thread 160 sees final value 4000
cop4610@cop4610-desktop:~/assignments$
```

### Synchronized run (200 threads)

In this case, the final value was synchronized because of the usage of a `pthread_barrier`, allowing for all threads to cease operation until all threads arrived at that corresponding portion of the code on their individual executions. It was also noticeable that the expected value was not corrupted at all, because of the usage of a mutex lock that allowed us to ensure that a thread's entire critical section didn't run concurrently/interleaved with another thread's critical section code, allowing each critical section portions to run as an entire unit.