

# Contributors

- Cristian Cepeda was the main developer in Part 2 of Assignment-2. However, Marcial Cabrera was jumping in and out since his main focus was on part 1.

## Output for Step 1

- Step 1 of Part 2 of the assignment can be seen by running `./pthreads-1 4`
  - o In this scenario 4 threads start to execute and a clear race condition is noticeable.

```
Shell Shell Shell
:part2 (master) cristiancedas$ ./pthreads-1 4
*** thread 1 sees value 0
*** thread 0 sees value 1
*** thread 0 sees value 2
*** thread 0 sees value 3
*** thread 1 sees value 4
*** thread 1 sees value 5
*** thread 2 sees value 6
*** thread 2 sees value 7
*** thread 3 sees value 8
*** thread 3 sees value 9
*** thread 0 sees value 10
*** thread 2 sees value 11
*** thread 1 sees value 11
*** thread 3 sees value 11
*** thread 0 sees value 12
*** thread 0 sees value 13
*** thread 2 sees value 14
*** thread 2 sees value 15
*** thread 2 sees value 16
*** thread 2 sees value 17
*** thread 3 sees value 18
*** thread 3 sees value 19
*** thread 3 sees value 20
*** thread 3 sees value 21
*** thread 1 sees value 21
*** thread 1 sees value 22
*** thread 0 sees value 23
*** thread 2 sees value 24
*** thread 3 sees value 25
*** thread 3 sees value 26
*** thread 1 sees value 27
*** thread 0 sees value 28
*** thread 0 sees value 29
*** thread 1 sees value 30
*** thread 1 sees value 31
*** thread 2 sees value 32
*** thread 3 sees value 32
*** thread 0 sees value 33
*** thread 0 sees value 34
```

- o The final value in this special case was 71. The real or expected output should have been 80.

```
Shell Shell Shell
*** thread 2 sees value 38
*** thread 2 sees value 39
*** thread 0 sees value 40
*** thread 1 sees value 41
*** thread 1 sees value 42
*** thread 1 sees value 43
*** thread 2 sees value 44
*** thread 3 sees value 45
*** thread 3 sees value 46
*** thread 3 sees value 47
*** thread 3 sees value 48
*** thread 3 sees value 49
*** thread 3 sees value 50
*** thread 3 sees value 51
*** thread 0 sees value 52
*** thread 1 sees value 53
*** thread 2 sees value 53
*** thread 2 sees value 54
*** thread 3 sees value 54
Thread 3 sees final value 55
*** thread 0 sees value 55
*** thread 0 sees value 56
*** thread 1 sees value 57
*** thread 2 sees value 57
*** thread 0 sees value 58
*** thread 1 sees value 59
*** thread 1 sees value 60
*** thread 1 sees value 61
Thread 1 sees final value 62
*** thread 2 sees value 62
*** thread 0 sees value 63
*** thread 2 sees value 64
*** thread 0 sees value 65
*** thread 0 sees value 66
*** thread 2 sees value 67
*** thread 0 sees value 68
Thread 0 sees final value 69
*** thread 2 sees value 69
*** thread 2 sees value 70
Thread 2 sees final value 71
:part2 (master) cristiancedas$
```

- The reasons why we are not getting the expected output for this section of the project are
  - o The critical area of the code doesn't have exclusive access so multiple threads can read and write at the same time causing race conditions.
  - o Some threads read the same value and thus have outdated information.

## Output for Step 2

- Step 2 of Part 2 of the assignment can be seen by running `./pthreads-2 4`
  - o In this scenario 4 threads start to execute and we can see that each thread has mutual exclusion.

```
Shell Shell Shell
:part2 (master) cristiancedas$ ./pthreads-2 4
*** thread 0 sees value 0
*** thread 0 sees value 1
*** thread 2 sees value 2
*** thread 0 sees value 3
*** thread 0 sees value 4
*** thread 0 sees value 5
*** thread 0 sees value 6
*** thread 0 sees value 7
*** thread 3 sees value 8
*** thread 1 sees value 9
*** thread 1 sees value 10
*** thread 2 sees value 11
*** thread 3 sees value 12
*** thread 1 sees value 13
*** thread 1 sees value 14
*** thread 1 sees value 15
*** thread 0 sees value 16
*** thread 3 sees value 17
*** thread 3 sees value 18
*** thread 3 sees value 19
*** thread 3 sees value 20
*** thread 2 sees value 21
*** thread 1 sees value 22
*** thread 1 sees value 23
*** thread 1 sees value 24
*** thread 1 sees value 25
*** thread 0 sees value 26
*** thread 3 sees value 27
*** thread 3 sees value 28
*** thread 3 sees value 29
*** thread 3 sees value 30
*** thread 3 sees value 31
*** thread 3 sees value 32
*** thread 3 sees value 33
*** thread 3 sees value 34
*** thread 3 sees value 35
*** thread 3 sees value 36
*** thread 3 sees value 37
*** thread 3 sees value 38
*** thread 3 sees value 39
```

- o The final value in case is as expected 80. Since we said 4 threads each one adds 20 to the shared variable.

```
Shell Shell Shell
*** thread 2 sees value 44
*** thread 0 sees value 45
*** thread 0 sees value 46
*** thread 0 sees value 47
*** thread 0 sees value 48
*** thread 0 sees value 49
*** thread 0 sees value 50
*** thread 0 sees value 51
*** thread 0 sees value 52
*** thread 0 sees value 53
*** thread 0 sees value 54
*** thread 0 sees value 55
*** thread 2 sees value 56
*** thread 2 sees value 57
*** thread 2 sees value 58
*** thread 2 sees value 59
*** thread 2 sees value 60
*** thread 2 sees value 61
*** thread 2 sees value 62
*** thread 2 sees value 63
*** thread 2 sees value 64
*** thread 2 sees value 65
*** thread 2 sees value 66
*** thread 2 sees value 67
*** thread 2 sees value 68
*** thread 2 sees value 69
*** thread 2 sees value 70
*** thread 2 sees value 71
*** thread 1 sees value 72
*** thread 1 sees value 73
*** thread 1 sees value 74
*** thread 1 sees value 75
*** thread 1 sees value 76
*** thread 1 sees value 77
*** thread 1 sees value 78
*** thread 1 sees value 79
Thread 1 sees final value 80
Thread 0 sees final value 80
Thread 3 sees final value 80
Thread 2 sees final value 80
:part2 (master) cristiancedas$
```

- We got the expected output for this section of part2. This was because mutexes were used to have exclusive access to the shared variable when going into the critical part of the code.
- The last part of this assignment was for each thread to wait for all other threads to finish before they outputted the final seen value. `pthread_barrier_t` could and should have been used for this part of the code. But since my development environment is a Mac. `pthread_barrier_t` is not implemented, or it can't run. So, to solve this issue I basically had each thread check in and loop until all threads had checked in before allowing them to continue.