Mareo Yapp

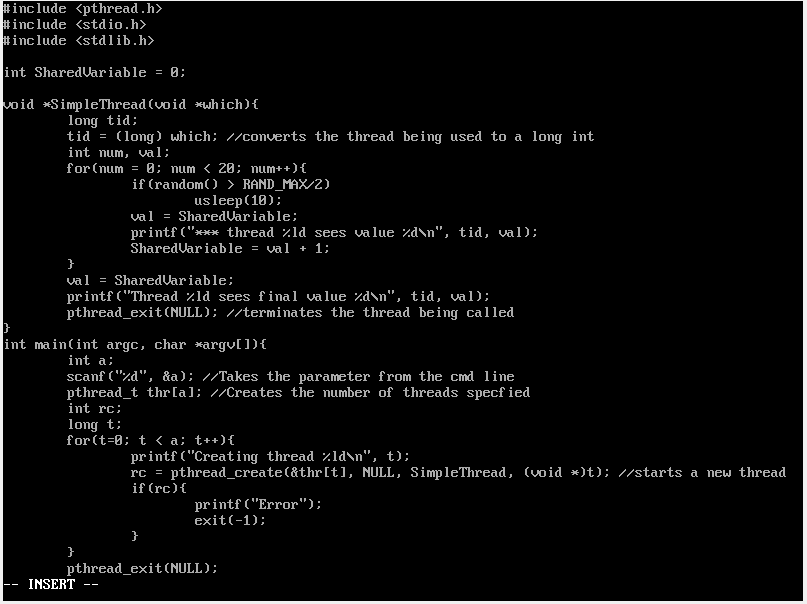
October 2, 2019

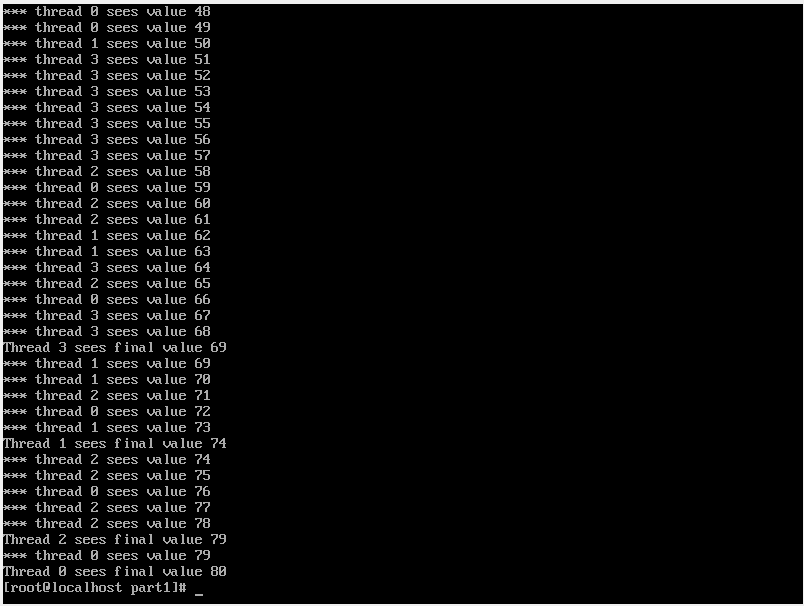
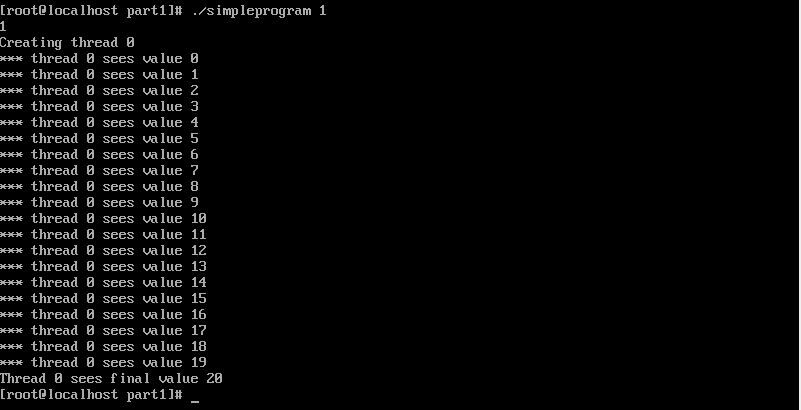
COP 4610

Lab 2

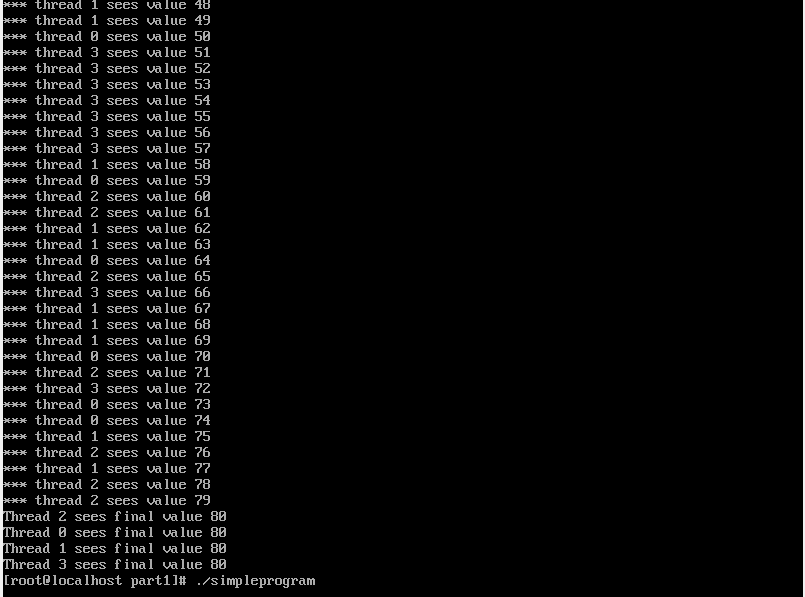
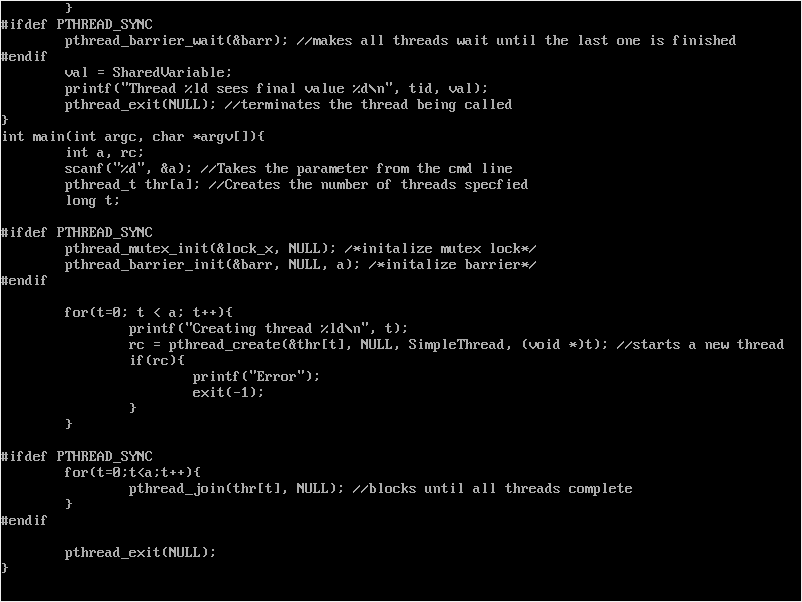
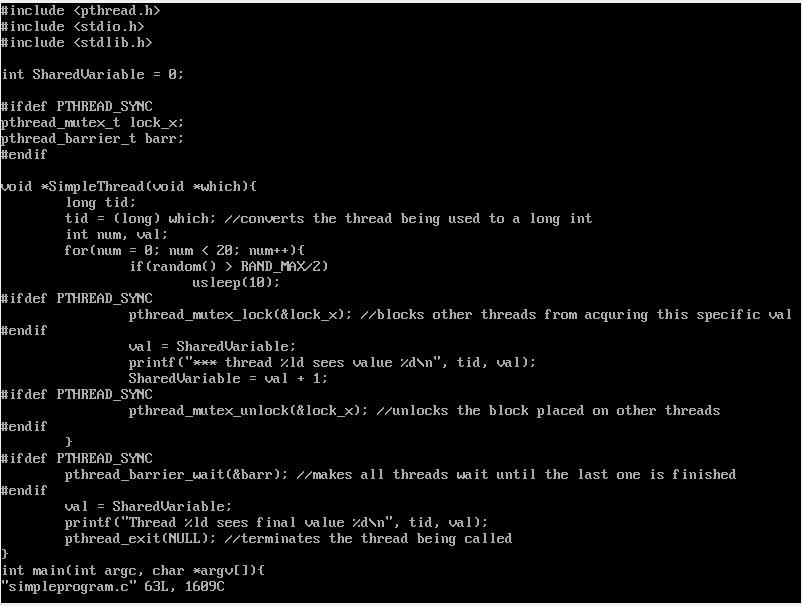
Part 1

For the unsynchronized version the threads didn’t see the same final value. The final value for the last thread would some times be less than what was expected. This is because the threads didn’t wait for each other to finish causing them to overlap on the same values.





For the synchronized version the threads saw the same final value. This is because threads didn’t overlap on the same values. This was achieved by implementing pthread\_barrier\_t and pthread\_mutex\_t. pthread\_mutex\_lock before val = sharedvariable made sure that only thread 0 saw the value 0 and pthread\_mutex\_unlock after sharedvariable = val + 1 allows the next thread to have that specific value preventing overlapping. pthread\_barrier\_wait made all the threads wait until the last one exited the loop to get the final value.



Part 2

Fibonacci works like this: the value of Fib(n) can be found by adding Fib(n-2) and Fib(n-1). To implement the instructions given I created 2 int types a = 0 and b = 1 and changed sum = a + b. When the cmd line parameter was 1 or 2 the program would print an automatic response. I changed i = 3 so that values less than 3 would not run in the for loop. I had to put sum after the print so that the code wouldn’t output an extra number.

./fib 4

a = 0

b = 1

sum = 0 + 1 = 1

printf(“0 ”)

printf(“1 ”)

for(i=3; i<=4; i++){

a = b = 1

b = sum = 1

printf(1)

sum = 1 + 1 = 2

i = i + 1 = 4

a = b = 1

b = sum = 2

printf(2)

sum = 1 + 2 = 3

}

0 1 1 2

