Lab Assignment--Threads and Mutexes

(Please do this lab in LINUX)

This week you will write a program that creates 3 threads. These threads will access a shared resource, an integer called buffer, one at a time. The buffer will initially be set to 0. They will each print their thread ID, process ID and the buffer's current value in one statement, then increment the buffer by one. Use a mutex to ensure this whole process is not interrupted.

Have the threads modify the buffer a total of 24 times. When each thread is done, it should return the number of times it modified the buffer to the main thread. The total number of modifications should be exactly 24.

- Hint 1: the buffer may increase in value while you are waiting to access it. Make sure that you have code that checks the buffer's value after you get access, but before you work with it.
- Hint2: your thread function is defined with void * arguments and returns void *. That means that it can pass in anything (as long as it is a pointer) and return anything (as long as it is a pointer). If you could define it as a regular function, the idea behind your thread function protype would be: int * myfunction()

The threads do not need to access the buffer in any particular order, but over multiple runs it should be obvious that all three are getting a fair chance. If one buffer seems to dominate over multiple runs, try introducing a sleep to simulate work between the mutex lock and unlock, just before the lock, or just after the unlock.
(Tip: if sleep is too slow for you, try nanosleep - it's what I used for my sample runs.)

Did you need to use a sleep to get evenly distributed results? Where did it help the most? Speculate as to why.

Is it possible to control the order of access to the buffer with just a mutex? Justify your answer.

Sample runs:

```
$ ./exercise
TID: 3077897072 PID: 30656 Buffer: 0
TID: 3069504368 PID: 30656 Buffer: 1
TID: 3059014512 PID: 30656 Buffer: 2
TID: 3077897072 PID: 30656 Buffer: 3
TID: 3069504368 PID: 30656 Buffer: 4
TID: 3077897072 PID: 30656 Buffer: 5
TID: 3059014512 PID: 30656 Buffer: 6
TID: 3069504368 PID: 30656 Buffer: 7
TID: 3077897072 PID: 30656 Buffer: 8
TID: 3059014512 PID: 30656 Buffer: 9
TID: 3069504368 PID: 30656 Buffer: 10
TID: 3077897072 PID: 30656 Buffer: 11
TID: 3069504368 PID: 30656 Buffer: 12
TID: 3059014512 PID: 30656 Buffer: 13
TID: 3069504368 PID: 30656 Buffer: 14
TID: 3077897072 PID: 30656 Buffer: 15
TID: 3059014512 PID: 30656 Buffer: 16
TID: 3077897072 PID: 30656 Buffer: 17
TID: 3069504368 PID: 30656 Buffer: 18
TID: 3059014512 PID: 30656 Buffer: 19
TID: 3077897072 PID: 30656 Buffer: 20
TID: 3069504368 PID: 30656 Buffer: 21
```

```
TID: 3059014512 PID: 30656 Buffer: 22
TID: 3077897072 PID: 30656 Buffer: 23
TID 3077897072 worked on the buffer 9 times
TID 3069504368 worked on the buffer 8 times
TID 3059014512 worked on the buffer 7 times
Total buffer accesses: 24

$ ./exercise
TID: 3077978992 PID: 30660 Buffer: 0
TID: 3069586288 PID: 30660 Buffer: 1
TID: 3059096432 PID: 30660 Buffer: 2
TID: 3077978992 PID: 30660 Buffer: 3
TID: 3069586288 PID: 30660 Buffer: 4
TID: 3077978992 PID: 30660 Buffer: 5
TID: 3059096432 PID: 30660 Buffer: 6
TID: 3077978992 PID: 30660 Buffer: 7
TID: 3069586288 PID: 30660 Buffer: 8
TID: 3077978992 PID: 30660 Buffer: 9
TID: 3059096432 PID: 30660 Buffer: 10
TID: 3077978992 PID: 30660 Buffer: 11
TID: 3069586288 PID: 30660 Buffer: 12
TID: 3077978992 PID: 30660 Buffer: 13
TID: 3059096432 PID: 30660 Buffer: 14
TID: 3077978992 PID: 30660 Buffer: 15
TID: 3069586288 PID: 30660 Buffer: 16
TID: 3077978992 PID: 30660 Buffer: 17
TID: 3059096432 PID: 30660 Buffer: 18
TID: 3077978992 PID: 30660 Buffer: 19
TID: 3069586288 PID: 30660 Buffer: 20
TID: 3077978992 PID: 30660 Buffer: 21
TID: 3059096432 PID: 30660 Buffer: 22
TID: 3077978992 PID: 30660 Buffer: 23
TID 3077978992 worked on the buffer 12 times
TID 3069586288 worked on the buffer 6 times
TID 3059096432 worked on the buffer 6 times
Total buffer accesses: 24
```

Deliverables:

- Your answers to the two questions as comments at the beginning of your code
- The code
- A few sample runs