Thread Synchronization Overhead
Andy Tran

1. Measure the execution time of a loop in the given code (synchronization via mutex) std::chrono::high_resolution_clock. Compare the measurements with the code that does the same thing in a single thread. What's the difference and why?

The difference between the threads running with mutex synchronization and all the threads running as a single thread was about .7 seconds. And this difference is apparent because for the threads to switch between each other it takes time, so there is time wasted for switching between the threads' operations.

```
andyqt1@circinus-18 15:36:35 ~/131
$ g++ PartB_mutex_high_res.cpp -o PartB_mutex_high_res -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:36:37 ~/131
$ ./PartB_mutex_high_res
Time(HRC) for thread creation loop and running was 0.727798 seconds
Time(HRC) for running without threads was 0.0550455 seconds
```

2. Measure the overhead of thread creation (get the timestamp before and after each thread is started) and mutex synchronization. Describe in the report, how significant is this overhead compared to the total execution time. Do all threads start at the same time? Why?

The overhead was not very much, they were about thousandths of a second, so it did not affect overall running time too much. The threads did not all start at the same time because it needed time to switch between the threads to start.

```
andyqt1@circinus-18 15:30:25 ~/131
$ g++ PartB_mutex_high_res.cpp -o PartB_mutex_high_res -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:30:27 ~/131
$ ./PartB_mutex_high_res
Time(HRC) for thread creation loop and running was 0.749431 seconds
Thread with mutex took 0.0011432 seconds to begin
Thread with mutex took 0.00195838 seconds to begin
Thread with mutex took 0.00269092 seconds to begin
Thread with mutex took 0.0034396 seconds to begin
Time(HRC) for running without threads was 0.0595329 seconds
```

3. Now use compare_exchange primitive. How different are the measurements and why?

The times for each thread to start with compare and exchange primitive was about the same as they were while using the mutex, however the time to execute the whole loop while using compare and exchange primitive instead of mutex primitive was much slower. But alongside the compare and exchange concept, I implemented two different methods of it through the vector size and boolean. And we can also see that when comparing and exchanging vector size as opposed to boolean, the vector size checks were much faster. Both compare and exchanges were still slower than running the threads sequentially. This could be because the overhead for the compare and exchange primitive was slower and took longer to switch between the threads.

```cpp
void worker_thread() {
    mtx.lock();
    thread_starttimes_hrc.push_back(std::chrono::high_resolution_clock::now());
    mtx.unlock();

    for (int i = 0; i < kNumIterations; ++i) {
       int expected = size;
       while (!size.compare_exchange_weak(expected, expected + 1)) {
          expected = size;
       }
       shared_vector[expected] = i;

}
}
```

```
andyqt1@circinus-18 15:30:41 ~/131
$ g++ PartB_comp_exc_high_res.cpp -o PartB_comp_exc_high_res -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:30:58 ~/131
$ ./PartB_comp_exc_high_res
Time(HRC) for thread creation loop and running was 0.946287 seconds
Thread with compare and exchange took 0.00109414 seconds to begin
Thread with compare and exchange took 0.00179016 seconds to begin
Thread with compare and exchange took 0.00246963 seconds to begin
Thread with compare and exchange took 0.00318967 seconds to begin
Time(HRC) for running without threads was 0.0560093 seconds
```

```cpp
void worker_thread() {
    mtx.lock();
    thread_starttimes_hrc.push_back(std::chrono::high_resolution_clock::now());
    mtx.unlock();

    bool expected;
    for (int i = 0; i < kNumIterations; ++i) {
        expected = false;
        while (!lock.compare_exchange_weak(expected, true)) {
            expected = false;
        }
        shared_vector.push_back(i);
        lock = false;
    }
}
```

```
andyqt1@circinus-18 15:31:33 ~/131
$ g++ PartB_comp_exc_bool_high_res.cpp -o PartB_comp_exc_bool_high_res -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:31:35 ~/131
$ ./PartB_comp_exc_bool_high_res
Time(HRC) for thread creation loop and running was 1.24477 seconds
Thread with compare and exchange took 0.0010723 seconds to begin
Thread with compare and exchange took 0.00200563 seconds to begin
Thread with compare and exchange took 0.00275589 seconds to begin
Thread with compare and exchange took 0.00348781 seconds to begin
Time(HRC) for running without threads was 0.0579633 seconds
```

4. Use clock() function to measure the execution time of the code. Are the measurements different from the previous ones? If so, which way of time measurement is correct? Why?

The measurements were different but the order ranking in completion times were the same, single thread, then mutex, compare and exchange with vector size, then compare and exchange with boolean. Overall for mutex and compare and exchange primitives while using the clock instead of the high resolution clock took longer to start the threads and also complete the whole loop with threading. The time to start the threads was 2 times that measured by the high resolution clock. Another difference is the gap between the mutex and compare and exchange primitives. Before with the high resolution clock, compare and exchange took about 2 times the

mutex primitive, however with the clock, compare and exchange took about 3 times the mutex primitive. The high resolution clock is a better measurement tool because it has the shortest tick period so the measurements are more accurate and precise.

```
andyqt1@circinus-18 15:32:04 ~/131
$ g++ PartB_mutex_clock.cpp -o PartB_mutex_clock -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:32:06 ~/131
$ ./PartB_mutex_clock
Time(Clock) for thread creation loop and running was 1.6055 seconds
Thread with mutex took 0.001943 seconds to begin
Thread with mutex took 0.004266 seconds to begin
Thread with mutex took 0.006454 seconds to begin
Thread with mutex took 0.009046 seconds to begin
Time(Clock) for running without threads was 0.054918 seconds
```

```
andyqt1@circinus-18 15:32:27 ~/131
$ g++ PartB_comp_exc_clock.cpp -o PartB_comp_exc_clock -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:32:29 ~/131
$ ./PartB_comp_exc_clock
Time(Clock) for thread creation loop and running was 3.55284 seconds
Thread with compare and exchange took 0.001942 seconds to begin
Thread with compare and exchange took 0.003891 seconds to begin
Thread with compare and exchange took 0.006261 seconds to begin
Thread with compare and exchange took 0.010131 seconds to begin
Time(Clock) for running without threads was 0.055354 seconds
```

```
andyqt1@circinus-18 15:32:47 ~/131
$ g++ PartB_comp_exc_bool_clock.cpp -o PartB_comp_exc_bool_clock -Wall -std=c++11 -fsanitize=thread -fPIE -pthread -pie
andyqt1@circinus-18 15:32:52 ~/131
$ ./PartB_comp_exc_bool_clock
Time(Clock) for thread creation loop and running was 5.41008 seconds
Thread with compare and exchange took 0.001776 seconds to begin
Thread with compare and exchange took 0.004051 seconds to begin
Thread with compare and exchange took 0.006605 seconds to begin
Thread with compare and exchange took 0.010304 seconds to begin
Time(Clock) for running without threads was 0.057626 seconds
```