CSE 437 REAL TIME SYSTEM ARCHITECTURES
HOMEWORK 2 REPORT
Fatih Selim YAKAR – 161044054

The Requirements,Constraints And Assumptions Of Timer

Versions

- Source code runs in C++ version 11.
- The source code's cmake runs in version 3.0.

Constraints

There are constraints from the methods defined in ITimer.h.

- **registerTimer (1)** → After the timer waits for the specified time, it runs the given method and ends.
- **registerTimer (2)** → The timer runs the specified method indefinitely (actually until main ends) for the specified period.
- **registerTimer (3)** → Timer runs the specified method for the period given as a parameter up to the limit given as parameter.
- **registerTimer (4)** → The method given as timer predicate runs the specified method during the period given as a parameter until it returns false.
- There should be 1 timer thread and 1 main thread
- If there is more than 10ms delay in the period, the deadline will print a miss warning.

Assumptions

- The first callback for the second version of the registerTimer starts from the first period after the registerTimer method runs.
- The first callback for the 3rd version of the registerTimer starts from the first period after the registerTimer method runs. However, it does not execute the callback when the specified time has passed or it is equal.
- The first callback for the 4th version of the registerTimer starts from the first period after the registerTimer method runs. But as soon as the predicate method is false, the timer does not execute the callback.
- All timers are finished when the main thread ends.

The Design Of The Timer

A class with the name TimerTask was written in order to keep the variables coming as parameters in the registerTimer method. Timer was implemented in the main Timer class, which extends from ITimer.h. Inside the Timer class was an STL Vector holding the TimerTask variables. Each time registerTimer function runs, a new TimerTask object is added to this vectore and notified. In general, the Timer is designed to create a thread when the timer object is created and this thread will terminate when the timer object is deleted. The most recent callback is found in the TimerTask vector in the Timer thread and waited until then with sleep for. After the wait, whichever Timer type will work with certain if-else sequences, it worked. The ending Timers are deleted from the TimerTask vector. At first, condition variable was used with flag in order to add a new timer when there is no timer or to add a new timer when main does not end. For the synchronization between Main thread and Timer thread, unique_lock was used.

Test

In order to test, 8 timers in total were run simultaneously, 2 of each timer type. Simultaneous operation was tested by setting the start time of some of these timers in reverse order. Test code and output was as follows:

```
int main(){

    Timer timer;
    std::this_thread::sleep_for(std::chrono::seconds(1));
    T0 = CLOCK::now();
    logCallback(-1, "main starting.");
    auto t1 = CLOCK::now() + std::chrono::seconds(1);
    auto t2 = t1 + std::chrono::seconds(1);
    timer.registerTimer(t2, [&]() {logCallback(1, "callback str (type 1 timer)"); });
    timer.registerTimer(t1, [&]() {logCallback(2, "callback str (type 1 timer)"); });

    timer.registerTimer(Millisecs(700),
    [&]() { logCallback(3, "callback str (700ms period - type 2 timer)"); });
    timer.registerTimer(Millisecs(1400),
    [&]() { logCallback(4, "callback str (1400ms period - type 2 timer)"); });

    timer.registerTimer(t1 + Millisecs(3000), Millisecs(1000),
    [&]() { logCallback(5, "callback str (1000ms period - type 3 timer)"); });
    timer.registerTimer(t1 + Millisecs(2200), Millisecs(1100),
    [&]() { logCallback(6, "callback str (1100ms period - type 3 timer)"); });

    timer.registerTimer([&]() {
        static int count = 0;
        return ++count < 3;
    }, Millisecs(3000), [&]() { logCallback(7, "callback str (3000ms period - type 4 timer)"); });

    timer.registerTimer([&]() {
        static int count = 0;
        return ++count < 5;
    }, Millisecs(800), [&]() { logCallback(8, "callback str (800ms period - type 4 timer)"); });
    std::this_thread::sleep_for(std::chrono::seconds(10));
    logCallback(-1, "main terminating.");
}
```

```
fatihselimyakar@Fatih-MacBook-Air build % ./run
[0] (cb -1): main starting.
[701] (cb 3): callback str (700ms period - type 2 timer)
[805] (cb 8): callback str (800ms period - type 4 timer)
[1001] (cb 2): callback str (type 1 timer)
[1001] (cb 5): callback str (1000ms period - type 3 timer)
[1105] (cb 6): callback str (1100ms period - type 3 timer)
[1402] (cb 4): callback str (1400ms period - type 2 timer)
[1403] (cb 3): callback str (700ms period - type 2 timer)
[1606] (cb 8): callback str (800ms period - type 4 timer)
[2004] (cb 1): callback str (type 1 timer)
[2004] (cb 5): callback str (1000ms period - type 3 timer)
[2106] (cb 3): callback str (700ms period - type 2 timer)
[2205] (cb 6): callback str (1100ms period - type 3 timer)
[2410] (cb 8): callback str (800ms period - type 4 timer)
[2805] (cb 4): callback str (1400ms period - type 2 timer)
[2807] (cb 3): callback str (700ms period - type 2 timer)
[3001] (cb 7): callback str (3000ms period - type 4 timer)
[3005] (cb 5): callback str (1000ms period - type 3 timer)
[3215] (cb 8): callback str (800ms period - type 4 timer)
[3512] (cb 3): callback str (700ms period - type 2 timer)
[4017] (cb 8): callback str (800ms period - type 4 timer)
[4207] (cb 4): callback str (1400ms period - type 2 timer)
[4213] (cb 3): callback str (700ms period - type 2 timer)
[4918] (cb 3): callback str (700ms period - type 2 timer)
[5609] (cb 4): callback str (1400ms period - type 2 timer)
[5619] (cb 3): callback str (700ms period - type 2 timer)
[6003] (cb 7): callback str (3000ms period - type 4 timer)
[6322] (cb 3): callback str (700ms period - type 2 timer)
[7013] (cb 4): callback str (1400ms period - type 2 timer)
[7024] (cb 3): callback str (700ms period - type 2 timer)
[7729] (cb 3): callback str (700ms period - type 2 timer)
[8416] (cb 4): callback str (1400ms period - type 2 timer)
[8433] (cb 3): callback str (700ms period - type 2 timer)
[9006] (cb 7): callback str (3000ms period - type 4 timer)
[9138] (cb 3): callback str (700ms period - type 2 timer)
[9821] (cb 4): callback str (1400ms period - type 2 timer)
[9841] (cb 3): callback str (700ms period - type 2 timer)
[10005] (cb -1): main terminating.
Timer::thread function terminating...
```

Build

Open the terminal in the directory where the source codes are located. Create new file with "mkdir build", "cd build" and enter it. Run cmake with "cmake ..", "make" and do make. Run the resulting executable file with ./run.