

CSE 437 REAL TIME SYSTEM ARCHITECTURES

HOMEWORK 2 (Due to 29th May. 2022)

Implement `gtu::mutex` class that has priority ceiling protocol capability using C++.

Requirements:

1. `gtu::mutex` shall have the properties described in Priority Ceiling Protocol in Lecture 5: “Real-Time Operating Systems”.
2. `gtu::mutex` shall implement `std::mutex` interface that is required by `std::lock_guard` and `std::unique_lock`. That is, the user of this mutex shall be able to use the mutex in the following way:
`gtu::mutex m;`
...
`std::lock_guard<gtu::mutex> lock(m);`
...
3. `gtu::mutex` shall provide an interface to register threads that are capable of locking this mutex in advance. During runtime, if a thread tries to lock the mutex, but hasn’t registered itself before, a runtime exception shall be raised.
4. As described in priority ceiling protocol, the designed mutex shall be able to avoid deadlocks. Therefore, the following code shall not create a deadlock:

<pre>// Thread 1 .. std::lock_guard<gtu::mutex> lock1(m1); std::lock_guard<gtu::mutex> lock2(m2); ...</pre>	<pre>// Thread 2 .. std::lock_guard<gtu::mutex> lock2(m2); std::lock_guard<gtu::mutex> lock1(m1); ...</pre>
---	---

5. As described in priority ceiling protocol, if a lock attempt is blocked due to priority ceiling protocol rule (that is, the thread trying to lock does not have higher priority than the highest ceiling of all acquired locks), the priority of the thread holding the lock will be elevated if necessary.

Homework Details:

1. Design and implement the required mutex class and helper classes, if necessary. The interface of the mutex shall be the same as “`std::mutex`” as described in requirement 1. The mutex interface shall be embedded in “`gtu` namespace”.
2. Design a test application to test the class designed in (1), verifying all requirements described in section “Requirements”.
3. Write a 2-page report containing descriptions of:
 - a. The design of your priority ceiling protocol system.
 - b. How to build and test your project

Notes:

- Required threading, timing and synchronization features should be implemented with native C++ (11, 14, ...). No OS specific function calls are allowed (i.e. `posix` threads in linux, multimedia timer in Windows...)
- The homework should be archived in a zip file containing sources and be uploaded Teams. The archive should only contain the report, source files and Makefile.