GTU Computer Engineering Department

CSE 437 Real Time System Architectures, Fall 2019

Homework 3 Titles (Due: 26th December 2019)

C++ core guidelines is a document containing a set of guidelines for using C++ well. The guideline covers various topics of C++ programming. In this homework, you will prepare and make a 10-minute presentation in the class about some of the guidelines that have significant impact on the way we program real-time systems.

You will cover at least the following titles in your presentation:

- Detailed explanation and reasoning of the problem at hand and the concepts involved in the problem.
- 'Bad examples', containing code that should be avoided after understanding the guideline.
- 'Good examples' containing code that follows the guideline rule.

Reference: http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

You should assume that the audience knows nothing about reasoning and concepts you cover. Get prepared to answer questions that are too basic or too complex.

1. "Resource acquisition is initialization – RAII" principle. Explain the principle, and the importance of this programming technique. Then, explain the following guideline titles
   - R.1: Manage resources automatically using resource handles and RAII
   - CP.20: Use RAII, never plain lock ()/unlock ()
2. The effect of memory access patterns on cache performance. Explain what a bad and good access pattern are, and the following guideline title:
   - Per.19: Access memory predictably
3. Writing functions considering that your function may be called concurrently. Explain the following guideline title
   - CP.1: Assume that your code will run as part of a multi-threaded program
4. Sharing data among threads and avoiding race conditions. Explain when and how a data race occurs and how to avoid. Explain the following guideline titles:
   - CP.2: Avoid data races
   - CP.3: Minimize explicit sharing of writable data
5. Atomic access to shared data and volatile variables. Explain the following guideline titles ;
   - CP.8: Don't try to use volatile for synchronization
   - CP.200: Use volatile only to talk to non-C++ memory
6. Using multiple mutexes and avoiding deadlocks: Explain std::lock and std::scoped_lock. Explain the following guideline title:
   - CP.21: Use std::lock() or std::scoped_lock to acquire multiple mutexes
7. Joining and detaching threads. Explain why and how we join threads, why we shouldn't detach and the related guideline support library function:
   - CP.23: Think of a joining thread as a scoped container
   - CP.25: Prefer gsl::joining_thread over std::thread
   - CP.26: Don't detach() a thread
8. Condition variables and using predicates: Explain condition variables and waiting with predicates. Explain the following guideline title:
   - CP.42: Don't wait without a condition
9. Lambdas and how to use them. Explain how lambdas access variables, incorrect usage causing runtime errors (i.e., caused by lost references to variables on stack). Also, have at least one example where a thread is passing a lambda, and another is calling it. Explain the following guideline title:
   - F.53: Avoid capturing by reference in lambdas that will be used nonlocally, including returned, stored on the heap, or passed to another thread