ECE 3574: Applied Software Design

Course Summary

Today we are going to wrap up the course.

- Course Summary
- Final Exam Format
- SPOT

Announcement

The course GitHub repositories will be deleted on May 15th!

If you want to record your work for the semester, be sure to clone the repositories before then.

Course Objectives

Having successfully completed this course, you should be able to:

- Use software design patterns and application programming interface (API) specifications to implement efficient and portable software.
- Design and implement multi-threaded and multi-process applications that rely on standardized inter-process communication and synchronization mechanisms.
- Design and implement complex software applications based on portable software frameworks and event-driven programming.
- Design, implement, and perform testing strategies including unit and integration testing.

Course Summary: Tools

- Use version control: git, mercurial, fossil, subversion
- Use a cross-platform build system: CMake, scons, qmake
- Use a testing framework: Catch, QTest, etc.

Course Summary: Design Principles

- DRY Principle
- Single-Responsibility Principle
- Orthogonality
- Coupling and Law of Demeter
- Principle of Least-Astonishment

Course Summary: Testing

- Unit tests
- Integration tests
- Test automation

Course Summary: Polymorphism

- Static Polymorphism: Templates
- Dynamic Polymorphism: Inheritance
- has-a versus is-a relationships
- Composition versus Inheritance
- virtual keyword
- Interfaces: pure abstract base classes
- member overloading
- dynamic_cast

Course Summary: Design Patterns

- RAII
- copy-swap
- copy-on-write
- State Machines

Course Summary: Model-View-Controller (MVC)

Separation of concerns:

- Model holds and provides access to data
- View accesses the model to display it
- Controller coordinates the model and view

Communication can occur using object pointers and member functions, or signals/messages.

Course Summary: Process Concurrency

- Process model of OS
- Basic OS functionality: virtualization
- IPC using pipes and shared memory

Course Summary: Thread Concurrency

- threads and C++ concurrent memory model
- asynchronous calls and future/promise
- thread communication using synchronization
- thread communication using message passing

C++ threads and QThreads.

Course Summary: Message Passing

- Object Serialization
- Thread-safe queues
- Qt Signals/Slots

Course Summary: Concurrency Patterns

- Message Queues
- Producer/Consumer
- Actors and the Singleton Pattern

Course Summary: Embedded Programming

- Subset of C++ suitable for embedded programming
- memory management
- stacks and pools
- event-driven designs
- preemptive kernels and RTOS

Final Exam

The final exam is 1:05-3:05 PM on Wednesday May 10, 2017 in Torg 2150

- The format will be similar to the in-class exercises.
- You will download from Canvas a zip file with three programming problems in it.
- You will have 2 hours to complete the assignments and check they compile and work as expected.
- You will zip up your code and upload to Canvas.

The final is open book, open notes, and you can refer to code you wrote this semester as well as access online documentation as needed (e.g. C++ and Qt).

- The exam assumes that you have the course development environment setup correctly on your laptop.
- You can also use the virtual machine distributed as part of projects 2 or 3 to test your code.

Student Perspectives on Teaching

Please respond the the SPOT survey for the course.

Any final questions?

I will have regular office hours the remainder of the week.