ECE 3574: Applied Software Design

Chris Wyatt

Spring 2017

Welcome to ECE 3574: Applied Software Design

- Website: https://filebox.ece.vt.edu/~ECE3574
- Instructor: Chris Wyatt, clwyatt@vt.edu
- TAs:
  - Walker Sensabaugh, walkerks@vt.edu
  - Rammohan Chowdary Chintha, rammohan@vt.edu

Today's Schedule:

- Description of the course
- Administrative details
- Expectations
- Course Tools Setup

Communication

Course Website: https://filebox.ece.vt.edu/~ECE3574

- syllabus, schedule, notes, etc.
- primary way materials are *distributed*.

GitHub: https://github.com

- distribute starter code
- demonstrate progress
- share code with instructors/TAs
- how you submit your assignments

Canvas: https://vt.instructure.com

- grades posted

Piazza: https://piazza.com/

- forum/wiki like software for QA, polls, announcements
- replaces email listserv, but has a configurable email digest
- good mobile apps, embedded in Canvas
- use it to ask (and answer) questions

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 Topics

We will be covering the following core topics.

- Generics and containers
- Inheritance and polymorphism
- Unit and integration testing
- Design patterns
- Using class-based software libraries
- Event-driven programming
- Concurrency: processes and threads
- Communication using shared memory and messages

Prerequisites

ECE 2574 Data Structures and Algorithms. You are expected to be competent in the basics of programming with C++ and the use of data structures and algorithms to solve problems.
It is helpful to be familiar with Unix systems (e.g. taken 2524) but it is not considered a prerequisite.

Text and Resources

Readings will be assigned from the following books. They can be read online through the library, although there is limited access, or you can purchase your own (both are very good and offer solid, practical advice on programming). There will also be assigned readings from various online sources.

- Clean Code, Robert C. Martin, Prentice Hall 2009
- The Pragmatic Programmer, Andrew Hunt and David Thomas, Addison-Wesley, 2000

Additional Resources

- Pro Git book https://git-scm.com/book/en/v2
- CMake Tutorial http://www.cmake.org/cmake-tutorial/
- C++ Reference http://en.cppreference.com/w/
- QT Documentation http://doc.qt.io

Software

A modern C++ compiler with sufficient C++11 support is required. Only specified compiler extensions and libraries may be used. We will use the open source CMake application for managing the build process (see www.cmake.org) and the git source code management tool (see git-scm.com).
Recommended Compilers:

- GCC >= 4.8
- Clang >= 3.5
- VC++ == 19 (Visual Studio 2015)

Development Environments

- For development you can use your favorite editor and a command console to invoke the compiler toolchain, or you can use any integrated development environment (IDE) supported by CMake, including Visual Studio on Windows and XCode on the Mac.
- There are several other options including QT Creator and CLion.
- Use whatever works for you but note each project will define a reference environment using a virtual machine that, **for grading purposes**, is the final arbiter of working code.

Whatever environment you choose, you should be adept at using it to write and debug code.

Grading and Honor Code

Coursework consists of in-class exercises, four projects, and a final exam. The grades will be computed as follows:

- Exercises: 10%
- Projects: 75%
- Final Exam: 15%

All graded work, other than the in-class exercises, is expected to be the original work of the individual student.

- Copying of specific assignment program-code is an honor code violation.
- I do use a code comparison system.

Exercises

The exercises are worked through in-class after a short lecture on the material for that day, and are due by midnight the day of class. Credit for exercises is assigned based on good-faith effort.

Projects

The projects (except project 0) have two due dates.

- The first due date is the beta version, which is graded and returned to you with feedback.
- The second due date, no shorter than 1 week after the feedback is returned, is the final version and should correct any mistakes or quality issues identified in the feedback.
- The total grade is distributed among the two versions on a per-project basis.

All projects must be turned in by the time/date indicated on the assignment. No late project work will be accepted.

Prior Expectations

You are expected to understand basic computer organization and C++ syntax from ECE 1574

- Types, including references and pointers
- How to write and call a function, passing arguments and returning values
- How to write and use a basic class
- How to read and write files, including parsing techniques
- How to do proper memory allocation and handling

Prior Expectations

You are expected to understand selection and use of common data structures and algorithms from ECE 2574

- Array-based and Link-based lists, stacks, queues, deques, and priority queues
- Tree and Hash Table based dictionaries
- Algorithms and used for sorting and searching

Prior Expectations

You are expected to be able to write, compile, and debug C++ code using good engineering practices.

- Perform the mechanics of compiling a program
- Understand how to read and correct compile-time errors
- Understand runtime-errors and how to identify why they are occurring
- Use an incremental development technique
- Have good debugging skills (hypothesis testing)
- Be able to identify and use appropriate reference material to solve problems

Prior Expectations

A Readiness Exercise and Project 0 will be used to ascertain your competence in these areas.
If you do poorly, you will be asked to attend one or more special help sessions early in the semester.

Project 0

https://filebox.ece.vt.edu/~ECE3574/projects/00-hexdump/

- The goal of this project is to get used to the course workflow and be sure you understand how to submit code for grading.
- You will write a simple hex dump program
- Due 1/27 by 12:59 pm

Questions?

Exercise 01: Setup

See Website.

Next Actions

- Take the Readiness Exercise
- Make sure you complete today's Exercise 01.
- Read Chapter 1 and 2 of the Pro Git Book (you can skip sections 1.2 and 1.5)
- Start on Project 0