Special Topics Course Proposal
Advanced Programming Techniques for Engineering Applications
George F. Riley

Introduction - The proposed course will cover a number of advanced topics in programming methods, data management, distributed computing, and advanced algorithms used in typical engineering applications. All class projects and in–class examples will use the *C++* programming language. It is designed to be a 4000 level course cross listed with a 6000 level course, taken by both advanced undergraduate and beginning graduate students. The undergraduate and graduate versions will meet in the same room at the same time. The graduate students will be expected to complete two or three additional aspects of each assignment, over and above what the undergrates are required to complete. The proposed format is 2–3–3, with the majority of the learning activities focused on out–of–class programming assignments. The course is planned to be offered once annually.

The topics are diverse, and each could merit its own course. Instead, this course will cover each topic from a conceptual standpoint, and discuss in some detail a small number of specific instances of the programming techniques used to implement programs using that topic. One programming assignment for each topic will be provided to give students practical experience in each topic, and to improve the students overall programming skill via substantial practice in coding and debugging.

Grading - The grading breakdown is:

50% Programming Projects
20% Mid-Term exam
20% Final Project
10% Class participation

Textbook - There is no suitable textbook for this course. The lectures will make extensive use of available on–line resources, as well as handouts prepared by the instructor.

Prerequisite - The prerequisite is *ECE32036*, or experience with the *C++* programming language.

Syllabus - The tentative outline for the course is as follows:

1. Distributed programming with *MPI* (3 lectures)
   (a) Synchronous and Asychronous communications
   (b) Group Communication and Synchronization
2. Parallel programming with *pthreads* (3 lectures)
   (a) Mutual Exclusion
   (b) Thread Synchronization
3. Object–Oriented code templates (2 lectures)
   (a) Typesafe callbacks with templates
   (b) Re–usable code with templates
4. Introduction to Data Mining using Map–Reduce (3 lectures)
   (a) Google's approach to managing large datasets
5. Event–based Programming (2 lectures)
   (a) Typesafe event handlers.
6. Introduction to graphics programming using *OpenGL* (3 lectures)
   (a) 2-D and 3-D coordinate transformations
7. Using web services (3 lectures)
   (a) Introduction to *SOAP*

   (b) Performance considerations with web servcies

8. Using non–blocking system I/O (2 lectures)
   (a) Asynchronous input-output programming
   (b) Handlng multiple sockes with `select`

9. Introduction to database programming using *MYSQL* (2 lectures)
   (a) The *MYSQL* database access API
   (b) Security Issues with database programming.

The objectives for this course are:

1. Become familiar with various methods for concurrent and distributed programming methods.
2. Program complex engineering applications in the C or C++ programming language.
3. Become familiar with three-dimensional graphics library interfaces.
4. Implement advanced encryption techniques using multi-precision math libraries.
5. Become familiar with both the creation and use of the popular “Smart Pointers” approach for memory management in C++ programs.
6. Program client-server applications using non-blocking system I/O calls.

The Educational Outcomes for this course are as follows.

The student will be able to :

1. Determine when to use distributed computing methods or parallel computing methods to solve complex engineering applications.
2. Create high-quality visual 3-D images of complex objects using the OpenGL graphics interface.
3. Implement several multi-precision public key and private key encryption methods using the GNU multi-precision mathematical library.
4. Create programs without memory leaks using the Smart Pointers approach.
5. Implement client/server applications using the sockets API and using the non-blocking appraoch for handling multiple clients simultaneously.
6. Manage large programming tasks using Makefiles.