CSE 333 Systems Programming

Credits
4.0 (3 hrs lecture, 1 hr section)

Lead Instructor
Steve Gribble

Textbook

- *C++ Primer*, Lippman

Course Description
Includes substantial programming experience in languages that expose machine characteristics and low-level data representation (e.g., C and C++); explicit memory management; interacting with operating-system services; and cache-aware programming.

Prerequisites

CSE 351.

CE Major Status
Selected Elective

Course Objectives

At the end of this course, students should:

1. have an in-depth understanding of the C and C++ programming languages and competence at programming to strict style guidelines;
2. understand manual memory management, pointer-based data structure construction and manipulation, and their debugging challenges;
3. learn the file-system and networking system call API and use it to construct systems such as an on-disk index and a client/server application;
4. be familiar with programming tools like debuggers, unit test frameworks, code coverage, profilers, and code review systems; and,
5. gain rudimentary familiarity with concurrent programming, including processes, threads, and event-driven code.

ABET Outcomes
(a) an ability to apply knowledge of mathematics, science, and engineering
(c) an ability to design a system, component, or process to meet desired needs within realistic

constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability
(e) an ability to identify, formulate, and solve engineering problems
(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

Course Topics

- C programming (2 weeks)
    - pointers, structs, casts; arrays, strings
    - dynamic memory allocation
    - C preprocessors, multifile programs
    - core C libraries
    - error handling without exceptions
- C++ programming (4 weeks)
    - class definitions, constructors and destructors, copy constructors
    - dynamic memory allocation (new / delete), smart pointers, classes with dynamic data inheritance, overloading, overwriting, and dynamic dispatch
    - C++ templates and STL
- Tools and best practices (2 weeks)
    - compilers, debuggers, make
    - leak detectors, profilers and optimization, code coverage
    - version control
    - code style guidelines; code review
- Systems topics: the layers below (OS, compiler, network stack) (2 weeks)
    - concurrent programming, including threading and asynchronous I/O
    - fork / join, address spaces, the UNIX process model
    - file-system API
    - network sockets API
    - understanding the linker / loader