Introduction to Systems Programming

Introduction and Overview

Prof. Dan Connors
OS Staff

Instructor:
  Prof. Dan Connors
  Office: MW 12-2
  CS Building 320
  Email: dconnors@cs.du.edu

Teaching Assistant:
  TA: Swayanti Das
  Email: swaydas@cs.du.edu
Course logistics and details

- Course Web page -
  - http://web.cs.du.edu/~dconnors/courses/comp2355
- Lectures - MW 2:00-3:50

- Textbook:
  System Programming with C and Unix (Paperback) by Adam Hoover
  ISBN-10: 0136067123

- Computer Systems: A Programmer's Perspective (CS:APP)
  Randal E. Bryant and David R. O'Hallaron
  http://csapp.cs.cmu.edu/
Course logistics and details

- **Homework and Assignments**
  - ~5 written homework assignments in the quarter, and 5 programming assignments (some knowledge of C). Late homework will not be accepted.

- **Tests**
  - Midterm - date to be announced, tentatively Week 5 in class
  - 10:00 am - 12:50 pm       Friday       John Greene
    Hall 316    Mar 12, 2010 - Mar 12, 2010
Grading Policy

- Homework - 35%
- Midterm - 30% of the final grade
- Final exam - 35% of the final grade

Final assignment of grades will be based on a curve.
Lecture Schedule

Week 1
Class 1: Introduction  Class 2: Datatypes and Control Flow in C

Week 2
Class 3: C Preprocessor, Compiler, Linker, Loader and Debugger  Class 4: Arrays

Week 3
Class 5: Fundamental System V APIs  Class 6: Pointers

Week 4
Class 7: The GNU C Build System  Class 8: Symbols and Attributes

Week 5
Class 9: Profiling and Optimizing C Code  Class 10: Function Pointers and Closures

Week 6
Class 11: Signals  Class 12: Process Management

Week 7
Class 13: Terminal Programming  Class 14: Introduction to C++

Week 8
Class 15: C++ classes  Class 16: C++ templates and C++ operator overloading

Week 9
Class 17: C++ Standard Template Library  Class 18: Buffer Overflows and Memory Corruption

Week 10
Class 19: Concurrent Programming  Class 20: Final Review
Topics

- Specific topics that will be covered:
  - Datatypes in C
  - The C preprocessor
  - Linking and loading
  - System V APIs
  - Pointers
  - C development and build systems
  - Input-output in C
  - Process management
  - C++ namespaces, classes, templates, operators
  - C++ STL (Standard Template Library)
Overview

- A system is a set of interacting or interdependent entities. Managing complexity is the key problem in systems engineering.

- Systems programming is the writing of system software.
  - Systems software is software supporting application software.

- Examples of Systems Software: Operating systems / Firmware, Programming tools (compilers, linkers, loaders), (Essential) runtime libraries of programming languages
System Programming

Systems programming is:

• essential to any kind of computing
• an important foundation for application development
• constantly evolving to accommodate changes in hardware (job security!)
• hacking with unparalleled freedom
• fun!
Why Study C?

• C is the language of the systems programmer

• All major Operating Systems are written in C (BSD, Linux, OS X, Windows)

• C is a subset of C++ (so you cannot know C++ and not know C)

• C and C++ together are used for more than $\frac{1}{3}$ of all open-source projects at SourceForge

---

1http://www.cs.berkeley.edu/~flab/languages.html
What is Special About C?

- C is essentially abstract assembly language of many historical processors (like the PDP series)

⇒ You can emulate features from high-level languages (such as Java, OCAMl, PHP) in C!

- C has hardly any runtime system

⇒ Small footprint, easily ported to new architectures
Why Study C++?

C++ adds popular features from other languages to C.

+ No need to emulate certain features manually
+ Nice, standardized syntax for those features makes code easier to read
+ Still all of the power of C “under the hood”
  - Much larger language, harder to learn and use well
  - C++ has a non-trivial runtime system
  - C++ still evolves; C is much more stable as a language
Beyond “Introduction to C”

This course is also about good programming practice:

- Use of tools and principles behind those tools (subversion, compilers)
- Use of libraries and API design (GNU libc, STL)
- Understanding of UNIX design principles (System V, Security)
- Solving problems as a team