Introduction to System Programming Course

2016 Spring
Euiseong Seo
(euiseong@skku.edu)
Overview

- What this course is about
- Who teaches this course
- Why you have to take this course
- What you will learn in this course
- What you will earn in this course
- How to succeed in this course
What This Course is About

System Programming

- Information representation
- Assembly language
- Processor architecture
- Compilers, linkers, and loaders
Administrative Information

- **Course Code**
  - SWE 2001

- **Class Hour**
  - Tuesday: 10:30 AM ~ 11:45 AM
  - Thursday: 09:00 AM – 10:15 AM

- **Lecture Room**
  - #21534 (located on 5F of Engineering Bldg. I)
Textbook

- Computer Systems: A Programmer’s Perspective (3rd Ed.)
  - Randal E. Bryant and David R. O’Hallaron
  - 2015 Prentice-Hall
  - Authors’ site: http://csapp.cs.cmu.edu
Course Components

- Lectures
  - Concepts
  - Backgrounds

- Projects
  - Mostly on assembly programming
  - Design, implementation, measurement and optimization
Course Web Page

- http://csl.skku.edu/SWE2001S16
- Check the web site regularly
- Class material, project information and other useful things will be posted
Grading

- Proportion of Activities
  - Participation 10%
  - Projects 30%
  - Midterm 30%
  - Final 30%

- If you miss any exam, you will fail
- No lateness is allowed
- Up to four absences will be tolerated
Project

- You will work on each project alone
- The submission status will be posted on the course web page
- Only the assignments submitted before the deadline will receive credit
Ethical Code

- No academic misconduct will be tolerated
  - Zero-tolerance policy
  - One who is found guilty will be kicked out of my class immediately
Lecturer

• Euiseong Seo
  • Associate professor, Software Dept.
  • E-Mail: euiseong (at) skku.edu
  • Office: #85564
  • Phone: (031) 299-4953
Why You Have to Take This

- Abstraction is Good, But Don’t Forget Reality
  - Most CSE courses emphasize abstraction
    - Abstract data types
    - Asymptotic analysis
  - These abstractions have limitation
    - In performance optimization
    - In detection and elimination of bugs
Great Reality

- **Ints are not Integers, Floats are not Reals**
  - **Example 1: Is \( x^2 \geq 0? \)
    - **Float’s**: Yes!
    - **Int’s**:
      » 40000 * 40000 = 16000000000
      » 50000 * 50000 = ??
  - **Example 2: Is \( (x + y) + z = x + (y + z) \)?
    - **Unsigned & signed Int’s**: Yes!
    - **Float’s**:
      » (1e20 + -1e20) + 3.14 = 3.14
      » 1e20 + (-1e20 + 3.14) = ??
Great Reality

- **Does not generate random values**
  - Arithmetic operations have important mathematical properties

- **Cannot assume all “usual” mathematical properties**
  - Due to finiteness of representations
  - Integer operations satisfy “ring” properties
    - Commutativity, associativity, distributivity
  - Floating point operations satisfy “ordering” properties
    - Monotonicity, values of signs

- **Observation**
  - Need to understand which abstractions apply in which contexts
  - Important issues for compiler writers and serious application programmers
Great Reality

- When you measure the required time to execute a part of your code, what can you do?
  - Measuring time in clock cycle accuracy is necessary

- Time Stamp Counter
  - A special 64-bit register in Intel-compatible machines
  - Incremented every clock cycle
  - Read with `rdtsc` instruction
  - Why couldn’t it be possible with a C-function?
Great Reality

- **Memory is not unbounded**
  - It must be allocated and managed
  - Many applications are memory dominated

- **Memory referencing bugs especially pernicious**
  - Effects are distant in both time and space

- **Memory performance is not uniform**
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements
Great Reality

typedef struct {
    int a[2];
    double d;
} struct_t;

double fun(int i) {
    volatile struct_t s;
    s.d = 3.14;
    s.a[i] = 1073741824; /* Possibly out of bounds */
    return s.d;
}

fun(0) ➞ 3.14
fun(1) ➞ 3.14
fun(2) ➞ 3.1399998664856
fun(3) ➞ 2.00000061035156
fun(4) ➞ 3.14
fun(6) ➞ Segmentation fault
What You Will Learn

- A little bit of computer organization
- A little bit of processor internals
- Assembly language
- Code optimization techniques
- A little bit of system software
  - Compiler, linker, loader, OSs and so on
What You Will Learn

Software

Operating Systems

Application

Architecture

Hardware

CPU
Mem
I/O Devices
Application programs
Data structures & algorithms
Programming languages & compilers
Operating System
Architecture
Microarchitecture
Hardware Description Languages
Digital logic
VLSI layout
Processing, Fabrication
Chemistry, Physics
You will become more effective programmers
- Able to find and eliminate bugs efficiently
- Able to understand and tune for program performance

You will be prepared for later “systems” classes in CSE
- Compilers, Operating Systems, Computer Architecture, Embedded Systems and etc.
Keys to Success

- Think with your butt, not with your brain
- Invest as many hours as possible
- There’s no royal road to become a hacker