Course Overview

CS230: System Programming
1st Lecture

Instructors:

Jaehyuk Huh

Overview

- Course theme
- Academic integrity
- Five realities

This course is

- CS230 System Programming
- The course is an “introduction to computer systems” from the perspective of programmers
- If you expected System Programming, this is not a right course.

Course Perspective

- **Most Systems Courses are Builder-Centric**
  - Computer Architecture
    - Design pipelined processor
  - Operating Systems
    - Implement sample portions of operating system
  - Compilers
    - Write compiler for simple language
  - Networking
    - Implement and simulate network protocols

Course Perspective (Cont.)

- **Our Course is Programmer-Centric**
  - Purpose is to show that by knowing more about the underlying system, one can be more effective as a programmer
  - Enable you to
    - Write programs that are more reliable and efficient
    - Incorporate features that require hooks into OS
      - E.g., concurrency, signal handlers
  - Cover material in this course that you won’t see elsewhere
  - Not just a course for dedicated hackers
    - **We bring out the hidden hacker in everyone!**

Cheating: Description

- **Please pay close attention**
- **What is cheating?**
  - Sharing code: by copying, retyping, **looking at**, or supplying a file
  - Describing: verbal description of code from one person to another.
  - Coaching: helping your friend to write a lab, line by line
  - Searching the Web for solutions
  - Using code-generation tools such as ChatGPT
  - Copying code from a previous course or online solution
    - You are only allowed to use code we supply, or from the CS:APP website
- **What is NOT cheating?**
  - Explaining how to use systems or tools
  - Helping others with high-level design issues
- **Ignorance is not an excuse**

Cheating: Consequences

- **Penalty for cheating:**
  - Minimum penalty: one letter grade downgrade (A0 → B0)
  - Possible removal from course with failing grade (F)
  - Your instructors' personal contempt

- **Detection of cheating:**
  - We have sophisticated tools for detecting code plagiarism
  - In the prior semester, >30 students were caught cheating

- **Commit your updates to your git repository as often as possible**

- **Don't do it!**
  - Start early
  - Ask the staff for help when you get stuck

Textbooks

- **Randal E. Bryant and David R. O'Hallaron,**
  - *Computer Systems: A Programmer's Perspective*, **Third Edition** (CS:APP3e), Pearson, 2016
  - http://csapp.cs.cmu.edu
  - Highly recommend the original English version (you'll have to get used to textbooks in English as soon as possible to survive CS)
  - This book really matters for the course! (for labs and exams)
  - **Reading the textbook is NOT optional, but required**
    - I will post required sections in KLMS.
    - There will be occasional quizzes about the reading assignments.
- **Optional textbook: Brian Kernighan and Dennis Ritchie,**
  - *The C Programming Language*, Second Edition, Prentice Hall, 1988
  - Still the best book about C, from the originators

Course Components

- **Lectures**
  - Higher level concepts
- **Labs (5 assignments)**
  - **The heart of the course**
  - About 2-3 weeks each
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement
- **Exams (midterm + final)**
  - Test your understanding of concepts & mathematical principles

Getting Help

- **Class Web page: KLMS**
  - Complete schedule of lectures, exams, and assignments
  - Copies of lectures, assignments, exams
  - Clarifications to assignments
  - **Update your email address (Important announcements can be made via emails and KLMS postings)**

- **Use the Piazza Q&A board to share questions and answers**
  - **We will instruct how to use the Piazza board by email**
  - **The Piazza board will be shared by Section A and B.**
  - **Students are also encouraged to answer questions**

Getting Help

- **Staff mailing list: cs230_ta@casys.kaist.ac.kr**
  - Use this for all communication with the teaching staff
  - Always CC staff mailing list during email exchanges

Policies: Labs and Exams

- **Work groups**
  - You must work alone on all lab assignments
- **Handins**
  - Labs due at 11:59pm on Tues or Thurs
  - Electronic handins (no exceptions!)
- **Exams**
  - Midterm + Final (traditional exams)
  - **At least 2/3 of exam questions are related to the lab assignments**
- **Appealing grades**
  - In **writing** to Prof Jaehyuk Huh within 7 days of completion of grading
  - Follow formal procedure described in syllabus

Timeliness

- **Lateness penalties**
  - Get penalized **30% per day**
  - No handins later than **1 day after due date**
- **Catastrophic events**
  - Major illness, death in family, …
  - Formulate a plan to get back on track
- **Advice**
  - Once you start running late, it's really hard to catch up

Policies: Grading

- **Exams (70%): midterm (35%), final (35%)**
- **Labs (25%): weighted according to effort**
- **Attendance (5%)**
  - Attendance of the first week (8/29 and 8/31) will not be checked as the enrollment changes during the period.
  - If you miss 1/3 of class meetings (9 class meetings), the final grade will be automatically F.

Labs

- *Tentative* lab schedule

- Lab 1: Data lab (9/26)
- Lab 2: Bomb lab (10/12)
- Lab 3: Attack lab (11/2)
- Lab 4: Tsh lab (11/16)
- Lab 5: Malloc lab (11/30)

Lab Environments and Requirements

- **Lab system**
  - A remote linux account will be provided.
  - Your submission must be running on the account.
- **Programming environments**
  - C/C++ is the lab language. If you are not familiar with it, you must learn it in 2 weeks.
  - Vscode (Visual Studio code) and git will be used for IDE and repository.
  - We will post introductory materials for the environments for self-study.

Programs and Data

- **Topics**
  - Bits operations, arithmetic, assembly language programs
  - Representation of C control and data structures
  - Includes aspects of architecture and compilers

- **Assignments**
  - L1 (datalab): Manipulating bits
  - L2 (bomblab): Defusing a binary bomb
  - L3 (attacklab): The basics of code injection attacks

Exceptional Control Flow

- **Topics**
  - Hardware exceptions, processes, process control, Unix signals
  - Includes aspects of compilers, OS, and architecture

- **Assignments**
  - L4 (tshlab): Writing your own Unix shell.
    - A first introduction to concurrency

Virtual Memory

- **Topics**
  - Virtual memory, address translation, dynamic storage allocation
  - Includes aspects of architecture and OS

- **Assignments**
  - L5 (malloclab): Writing your own malloc package
    - Get a real feel for systems-level programming

Networking, and Concurrency

- **Topics**
  - High level and low-level I/O, network programming
  - Internet services, Web servers
  - concurrency, concurrent server design, threads

Lab Rationale

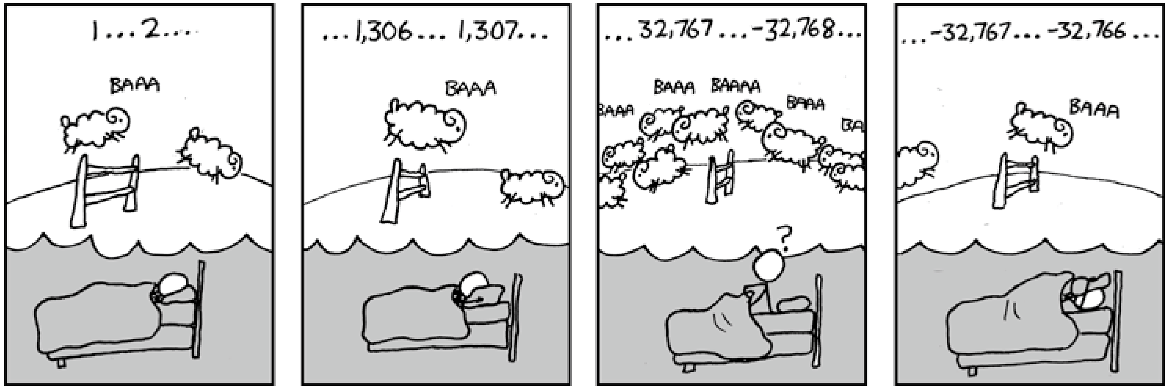
- **Each lab has a well-defined goal such as solving a puzzle or winning a contest**
- **Doing the lab should result in new skills and concepts**
- **We try to use competition in a fun and healthy way**
  - Set a reasonable threshold for full credit

Course Theme: Abstraction Is Good But Don’t Forget Reality

- **Most CS courses emphasize abstraction**
  - Abstract data types
  - Asymptotic analysis
- **These abstractions have limits**
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
- **Useful outcomes from taking cs230**
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
    - Able to understand and tune for program performance
  - Prepare for later “systems” classes in CS
    - Compilers, Operating Systems, Networks, Computer Architecture, Embedded Systems, Storage Systems, etc.

Great Reality #1:
Ints are not Integers, Floats are not Reals

- **Example 1: Is $x^2 \geq 0$?**
  - Float's: Yes!



  - Int's:
    - 40000 * 40000 -> 1600000000
    - 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) --> ??

Source: xkcd.com/571

Great Reality #2: You've Got to Know Assembly

- **Chances are, you'll never write programs in assembly**
  - Compilers are much better & more patient than you are
- **But: Understanding assembly is key to machine-level execution model**
  - Behavior of programs in presence of bugs
    - High-level language models break down
  - Tuning program performance
    - Understand optimizations done / not done by the compiler
    - Understanding sources of program inefficiency
  - Implementing system software
    - Compiler has machine code as target
    - Operating systems must manage process state
  - Creating / fighting malware
    - x86 assembly is the language of choice!

Great Reality #3: Memory Matters

Random Access Memory Is an Unphysical Abstraction

- **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 #4: Computers do more than execute programs

- **They need to get data in and out**
  - I/O system critical to program reliability and performance

- **They communicate with each other over networks**
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
    - Cross platform compatibility
    - Complex performance issues

Welcome
and Enjoy!