CS 61

- Fall 2011, Tuesdays and Thursdays 2:30pm–4pm
- Prereqs: CS 50 (or C programming experience)
- An *introduction* to computer systems
  - Not an “advanced” course.
  - Don’t need to be a CS concentrator to take this class.
  - Will set you up for CS161 (OS), CS153 (compilers), and CS141 (architecture)

- CS concentrators:
  - Need 2 out of 3 of CS 50, CS 51, and CS 61

- CS as a secondary field
  - Counts as one of the 4 half-courses
What is CS61 about?
What is CS61 about?

• What happens when I run a program?
  • Delving into mysteries of how machines really work
  • Get “under the hood” of programming at machine level
  • Understand what affects performance of your programs
What is CS61 about?

• What happens when I run a program?
  • Delving into mysteries of how machines really work
  • Get “under the hood” of programming at machine level
  • Understand what affects performance of your programs

• Need to understand how machines work in order to grasp
  • Operating Systems
  • Databases
  • Processor Architecture
  • Compilers
  • Networks
What is CS61 about?

• What happens when I run a program?
  • Delving into mysteries of how machines really work
  • Get “under the hood” of programming at machine level
  • Understand what affects performance of your programs

• Need to understand how machines work in order to grasp
  • Operating Systems
  • Databases
  • Processor Architecture
  • Compilers
  • Networks

• … and to be a good programmer!
  • i.e., to write efficient, robust, portable, maintainable code
What we’re going to cover

• Learn how computer systems work
  • How processors work and what affects their performance
    • Linking, loading, execution of programs
    • Memory, caches, heap, stack
  • Machine representation of programs and information
    • Compilation
    • x86 assembly code

• Learn about OS-level programming
  • UNIX system programming: files, processes, pipes, signals
  • Concurrency: threads and synchronization
Workload

• CS 61 is not intended to be a heavy workload course
  • Challenging, but fun
  • Suitable for anyone who has taken CS 50, not just CS concentrators

• One midterm, one final, 2 lectures + 1 section per week

• ~5 assignments
  • Defusing a binary bomb
  • Exploiting buffer overrun vulnerabilities
  • Implementing your own shell
  • Writing concurrent programs
  • Implementing dynamic memory allocation
    (can work in pairs on the programming assignments)
A taste...

- Why is it important to understand how computers work?
- Ken Thompson, *Reflections on Trusting Trust*
  - Co-inventor of UNIX
  - Won Turing Award in 1983
  - During award lecture, revealed surprising exploit...
The Thompson Hack
The Thompson Hack

- Thompson put backdoor into `login.c` to allow easy access
  - Early days of UNIX
  - Allow Ken Thompson access to any UNIX system by e.g., entering username “ken” and password “magic”
  - Helpful for debugging
The Thompson Hack

- Thompson put backdoor into `login.c` to allow easy access
  - Early days of UNIX
  - Allow Ken Thompson access to any UNIX system by e.g., entering username “ken” and password “magic”
  - Helpful for debugging
- But anyone looking at code for `login.c` would see the backdoor and be able to use it!
The Thompson Hack

• Thompson put backdoor into login.c to allow easy access
  • Early days of UNIX
  • Allow Ken Thompson access to any UNIX system by e.g., entering username “ken” and password “magic”
  • Helpful for debugging
• But anyone looking at code for login.c would see the backdoor and be able to use it!
• So, Thompson hacked the C compiler
  • C compiler notices when it is compiling login.c
  • C compiler inserts backdoor code
The Thompson Hack

- Thompson put backdoor into `login.c` to allow easy access
  - Early days of UNIX
  - Allow Ken Thompson access to any UNIX system by e.g., entering username “ken” and password “magic”
  - Helpful for debugging
- But anyone looking at code for `login.c` would see the backdoor and be able to use it!
- So, Thompson hacked the C compiler
  - C compiler notices when it is compiling `login.c`
  - C compiler inserts backdoor code
- Now `login.c` code looks normal, but code for the C compiler is suspicious
The Thompson Hack
The Thompson Hack

- So Thompson hacked the C compiler again
  - C compiler notices when it is compiling *itself*
    - The C compiler was written in C
  - C compiler inserts code that will notice when *login.c* is being compiled and will insert back door
  - Then delete the hacked compiler source code
The Thompson Hack

• So Thompson hacked the C compiler again
  • C compiler notices when it is compiling itself
    • The C compiler was written in C
  • C compiler inserts code that will notice when login.c is being compiled and will insert back door
  • Then delete the hacked compiler source code

• Now compiler code and login.c code look normal
  • The backdoor only noticeable when you look at the binary executable for the compiler and the login program!
The Thompson Hack

- So Thompson hacked the C compiler again
  - C compiler notices when it is compiling itself
    - The C compiler was written in C
  - C compiler inserts code that will notice when login.c is being compiled and will insert back door
  - Then delete the hacked compiler source code
- Now compiler code and login.c code look normal
  - The backdoor only noticeable when you look at the binary executable for the compiler and the login program!

- Moral: computers may not do what you expect.
  - Take CS 61 and hone your expectations!
CS 51 or CS 61?

- Take both! They’re complementary...
- CS51 focuses on concepts of program design, data structures, and algorithms
  - Sets you up for later theory and programming classes
- CS61 is more “nuts and bolts” – how machines work
  - Sets you up for later systems, architecture, and compiler classes
Questions?

- Email me (chong@seas.harvard.edu)
- Look at the CS 61 website: http://cs61.seas.harvard.edu/
- Hope to see you in the Fall!