A brief intro to

CS61

Systems Programming and Machine Organization

Prof. Matt Welsh

Harvard University

November 16, 2009

CS61

Fall 2010: Tuesday/Thursday 2:30-4:00

Prerequisites: CS50 (or C programming experience)

Can be used for CS concentration breadth requirement (“middle digit”)

Can be used for CS secondary area requirement

You can, and should, take both CS51 and CS61 at the same time!

What is CS61 all about?

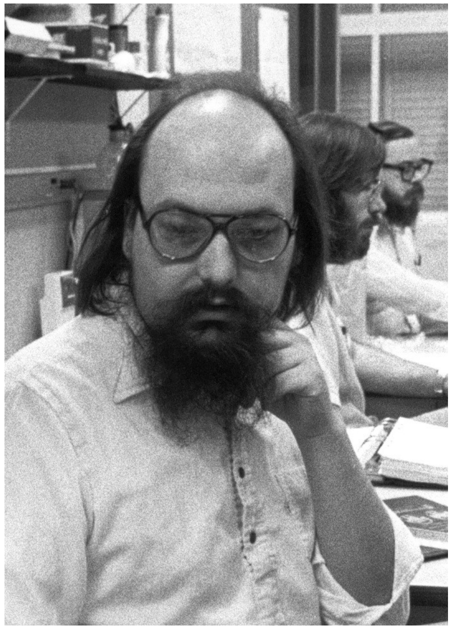
- Revealing the mystery of how machines really work!
- Getting “under the hood” of programming at the machine level
- Understanding what affects the performance of your programs:
  - Processor architecture
  - Caching and memory management
  - Processes, threads, and synchronization
- Writing rock solid (and fast) systems code

Why CS61?

- Huge gap between the *concepts* of programming and the *reality*
- This gap is more profound when you start programming in higher-level languages: Java, C++, Scheme, etc.
- Need to understand how machines really work to grasp:
  - Operating Systems
  - Databases
  - Processor Architecture
  - Compilers
  - Networks
- ... and even just to be a good programmer, even if you don't become a Computer Scientist.

Ken Thompson's Compiler Hack

- Ken Thompson – Co-inventor of UNIX
- Won Turing Award in 1983 (with Dennis Ritchie)
- During his award lecture, made a stunning admission...



Thompson's Compiler Hack

- Early days of UNIX: Thompson hacked the “login” program
  - Would accept a “magic” password to let him login on any UNIX system
  - Really helpful for debugging ...
- Problem: The source code for “login.c” was widely distributed
  - The whole system was “open source” (before we had that term...)
  - So, anyone could find the backdoor code!
- So, he hacked the C compiler...
  - C compiler would recognize that it was compiling “login.c”
  - Insert the backdoor code in at compile time

Thompson's Compiler Hack

- Now the backdoor was in the compiler code.
  What if someone read that?
- He hacked the ***compiler*** to recognize when it was compiling ***itself***
  - The compiler was itself implemented in C.
  - (Chicken and egg problem: How did they write the first C compiler?)
- The compiler would insert the backdoor code into itself!
  - So when the compiler compiles itself, it would insert the backdoor code to recognize when it was compiling login.c, to insert the backdoor code to check for the magic password. Got it?
- He then deleted the original compiler source code.
  - The backdoor could only be found in the ***binary!***

Why take CS61?

- Learn how machines really work.
  - Use gdb and objdump like an expert.
- Debug the hardest (and most interesting) bugs.
  - Stuff that only makes sense when you can read assembly.
- Hacking binaries for fun and profit.
  - How did the iPhone get jailbroken? The Code Red virus spread so quickly?
- Measure and improve the performance of your programs.
  - Understand memory hierarchies, processor pipelines, and parallelism.
- Write concurrent, multi-threaded programs like a pro.
  - The basis for every application and server on the Internet today.

Er, this sounds really hard...

- CS61 is **not** intended to be a heavy workload course.
  - Challenging, but fun.
  - Intended for everyone who has taken CS50 – not just CS concentrators
- Five lab assignments – can work in pairs:
  - 1) Defusing a binary bomb
  - 2) Hacking a buffer overrun bug
  - 3) Implementing dynamic memory allocation
  - 4) Writing your own UNIX shell
  - 5) Building a concurrent Internet service.
- One midterm, and a final. That's it.

Topics to be covered

- Intel x86 assembly language programming
  - Registers, memory, control flow, procedures, data structures
- Performance measurement and program optimization
- Linking and loading
- Memory hierarchy, caching, and dynamic memory allocation
- UNIX systems programming: files, pipes, signals, processes
- Threads and synchronization
- UNIX sockets programming
- Implementing concurrent servers

Or take both!

Questions?

- Email me! mdw@eecs.harvard.edu
- Or drop by Maxwell Dworkin 233