Slide 1

Administrivia

- Homework 6 on the Web. Due date December 10. No homework accepted past 11:59pm on that date.
- Sample solutions for first four homeworks on the Web (soon). Solutions for others also soon.
- Grades . . . coming by mail when I have them. If you've turned in all the homeworks, more or less on time, and your code compiles and passes your tests, and you've attended class, you will likely make an A.

  If you haven't been turning in homeworks, or you have but the programs don't compile or don't work, let's talk. I'd rather grade working code!

Slide 2

Minute Essay From Last Lecture

- How other languages with C in the name (C++, Objective C, C#) came from C.
- Major differences between C and C++.
- More about pointers.
- How to get more practice with C.
- More about `structs` and faking object-oriented programming.

Slide 3

Course Topics — Recap

- Basic C programming, for people who already know how to write programs in some other language. Especially useful (I think!) for those who start in a very abstract/high-level language.
- Review of the Linux/UNIX command-line environment and command-line development tools.
- Basics of computer arithmetic and data representation.

Slide 4

Why Learn C? (For Java/Python/Scala Programmers — Recap)

- Scala and Python (and Java, less so) provide a programming environment that's nice in many ways — lots of safety checks, nice features, extensive standard library. But they hide a lot about how hardware actually works.
- C, in contrast, has been called "high-level assembly language" — so it seems primitive in some ways compared to many other languages. What you get (we think!) in return for the annoyances is more understanding of hardware — and if you do low-level work (e.g., operating systems, embedded systems), it may well be in C. (Performance *may* also be better, though "measure and be sure".)

Slide 5

Quotes of the Day/Week/?

- From a key figure in the early days of computing:

  "As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs." (Maurice Wilkes: 1948)

- From someone in a discussion group for the Java programming language:

  "Compilers aren't friendly to anybody. They are heartless nitpickers that enjoy telling you about all your mistakes. The best one can do is to satisfy their pedantry to keep them quiet :)"

Slide 6

Minute Essay

- How did the course compare to your expectations/goals? Did you learn what you hoped to learn?