Slide 1

Administrivia

- Homework 6 on the Web. Due date next Wednesday. We need a "not accepted past" date for all homeworks. Tuesday 5/14 at 11:59pm?
- Grades . . . coming by mail when I have them. (As a preview, I mailed points/comments on part of hw1 today.) More on the next slide.
- I will post sample solutions for homeworks.
- I plan to have office hours next week, but probably not every day. I will send e-mail when I've settled details. We could schedule an "open lab" session for one day next week?

Slide 2

Grades

- Most of grade based on homeworks, with a few points for attendance.
- 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've turned in all or most of the homeworks, but some of them didn't work, you're welcome to submit revised versions of anything I haven't graded yet. I'd rather grade working code!
- If you didn't turn in a homework, it's not too late to get *some* points (maximum of half credit, but better than zero!).

Slide 3

Minute Essay From Last Lecture

- Other languages derived from C? C++, Objective C, etc.? (Notice also that several languages build on basic C syntax. Not a bad idea to build on syntax many programmers know.)
- GUIs and graphics in C? (Lots of libraries, but none standard, and many not portable.)
- Heaps? Trees? Linked data structures? (Same algorithms you use in a higher-level language, but you have to deal with more details.)
- Headers and recursion.

Slide 4

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.
- Review of basics of computer arithmetic and data representation.

Slide 5

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 6

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 7

Minute Essay

- Would you be interested in an open-lab session sometime next week? If so, what are times you are *not* available (finals or other scheduled events)?