Slide 1

Administrivia

- Homework 5 due Monday, May 7, and not accepted late. For all other homeworks, you will get *some* credit for anything you turn in by 11:59pm May 1 (end of reading days).
- Sample solutions for earlier assignments will be on the Web soon. Grades/comments coming by e-mail, when I have them.
- Should we have an optional meeting (“review session”?) Friday or during reading days?

Slide 2

Minute Essay From Last Lecture

- More about pointers? (As time permits.)
- “*Anything* dynamics you can do?” Yes, but you have to do it yourself.
- Networking in C? Library functions exist, but system-specific (though POSIX defines a standard for UNIX-like systems — fairly low-level). `man socket` is a starting point.
- GUIs in C? Even more variation. Base in “X Window System”, and then there are many “widget” libraries that build on it.
- C versus C++? C++ intended in part as “a better C” and also as “C with classes”. *Almost* a strict superset of C, plus syntax for object-oriented programming and standard libraries somewhat similar to collections found in Python/Scala. Strongest C influence is probably explicit pointers and memory management.

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?