Slide 1

Administrivia

- Homework 6 due date extended to May 12 (11:59pm). *Final deadline for all homeworks.* (If you missed an earlier homework, it's not too late: you will get some points for anything you turn in by this deadline.)
- I can (will?) post sample solutions for all assignments on the course Web page.
- Office hours from now through May 12? I will send mail when I know (probably tomorrow).

Slide 2

Grades

- Most of grade based on homeworks, with a few points for attendance. (Homework grades coming by e-mail as I finish 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'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

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 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

A Very Little History

- Initial development by Ken Thompson and Dennis Ritchie at Bell Labs, circa 1970. At least in part motivated by their work on UNIX: Many (most?) operating systems then were written in assembly language, hence difficult to port to new architectures. An O/S mostly written in something higher-level and more portable, however . . .
- First formal standard 1989. Continues to evolve, though slowly.
- Why so influential? Good question. Possibly a case of “right place and right time”, and then once something becomes popular it’s apt to stay that way, deservedly or not.

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 :)”

Minute Essay

- None — sign in.