Slide 1

Administrivia

- Reminder: Homeworks 7 and 8 due next Tuesday (Web page says today but "accepted without penalty" through Tuesday). If you can't do both, do Homework 8; it's more important in terms of learning outcome.
- I'm grading Homework 6 (and almost everyone is getting full or nearly full credit!) and will e-mail grades soon. I'll also send out a preliminary grade summary.
- I can put together an extra-credit assignment if there's interest. (I'll ask you in the minute essay whether you're interested.) Would be totally optional, could only help your grade, and would be due near or at the end of finals period.
- I plan to have regular office hours next Monday, and then after that I'm not sure, but probably sometime Wednesday and/or Friday. I'll send mail later with details. (If you need/want to talk to me, it might be good to send me an e-mail so we can pick a time.)

Slide 2

Administrivia

- (Anything else?)

Slide 3

This and That From Minute Essays — How C Is Used

- My guess is — niche market. But it *is* used — a recent alumnus worked for a few years with an "embedded system", programmed in C (with some GNU extensions). He moved on the graduate school, and his position has been filled by another alum!
- At one time, C was the way to go for best performance, but C++ compilers are pretty good now, and for general-purpose programming it's really probably better. (Either that or a more "managed" language such as Scala!)

Slide 4

This and That From Minute Essays — Scripting

- One person asked about "scripting", meaning shell scripts I think.
- As I may have mentioned earlier, what you type in a terminal window is actually a rather crude programming language interpreted and executed by a shell. So it has variables, conditional execution, and even loops. You can collect commands into a file and run them. (Examples?)

Slide 5

Learning C++

- Very different language from C, despite its origins as “C with classes” or “a better C”.
- Big complicated language that used to be almost *too* complicated — many features that experts could use to do amazing things but non-experts could struggle with. Current version has some features that seem to help. As with C, some things “for historical reasons”.
- A design goal (according to its inventor, paraphrasing mine) — make it possible to write “nice” programs while also making it possible to maximize(?) efficiency.
- My opinion — knowing both Scala and C is a good background, more so than one or the other.

Slide 6

C Programming using Non-Standard Features and Libraries

- C’s standard library is pretty limited, in keeping with the idea that the language should be implementable on a very wide range of platforms of varying capabilities.
- So if you want to write completely standard and portable C, there are a lot of things you can’t do.
- However, a lot of real-world uses of C require going outside the standard (e.g., programming those embedded systems, where you have to interact with hardware in low-level way).
- And there are a *lot* of non-standard libraries, some platform-specific, that do interesting or useful things . . .

Slide 7

C Non-Standard Features and Libraries

- Multithreading available with the OpenMP extensions. Fairly cross-platform.
- Parallel programming over a network available with MPI ("Message-Passing Interface"). Also fairly cross-platform.
- Parallel programming using GPU available via OpenCL. Somewhat cross-platform.
- Text-based GUI-ish features available for UNIX/Linux (most systems?) via the `ncurses` library.
- "Real" GUIs available via a lot of libraries. "X11" available for most UNIX/Linux but pretty primitive. Other "toolkits" available.
- (I could put examples on the "sample programs" page?)

Slide 8

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 9

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. A little more about floating-point representation.

Slide 10

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.

Slide 11

Minute Essay

- How are you doing with Homeworks 7 and 8? anything noteworthy to report yet?
- And best wishes for a good holiday!