Slide 1

Administrivia

- Extra-credit assignment on the Web. Totally optional and can only help your grade. Due just after finals (May 10 *at noon*).
- I will send out grade summaries as soon as I grade Homework 7 (this week I hope). (I just did a quick computation of grades based on everything else, and most people are doing quite well. Yay!)
- I will likely have office hours sometime next week; 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.)
- (Anything else?)

Slide 2

This and That From Minute Essays — How C Is Used

- My guess is — niche market. But it *is* used — a recent alumnus has been working with an "embedded system", programmed in C (with some GNU extensions).
- 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.

Slide 3

This and That From Minute Essays — C and its Successors

- As far as I know, C has had a big impact, as much as anything with regard to basic syntax — constructs for expressions and conditionals in particular.

Slide 4

This and That From Minute Essays — Programming Style

- (This is mostly about how to write for human readers as well as a compiler.)
- There's not One True Way to format code, but there are at least several good ways. Several widely-used conventions for indentation, all of which help if applied consistently.
- Choosing good variable names can help a lot. For loop counters, not so much, but in other cases, it can really help to think about "what does this variable represent?" I say this helps with programming logic too — make sure that how you use it matches its intended meaning.
- Comments are often helpful, *but only if they're consistent with the code*. With good names for variables and functions code can be sort of "self-documenting", but commments about interface can be good.

Slide 5

This and That From Minute Essays — 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

Just For Fun — “Extreme” ASCII Art?

- Some of you may have heard of “ASCII art”? a truly over-the-top example, from quite a while ago, can still be found, via

  ```
  telnet towel.blinkenlights.nl
  ```

  (to interrupt control-`]` then “quit” or control-d — although this doesn’t seem to work in a terminal window??)

  (For a while recently the site wasn’t working. Seems okay now?)
- (What some people choose to do with their time can be — interesting?)

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.

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 9

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 10

Minute Essay

- None really — just sign in (unless you have parting remarks?).
- And best wishes for a good summer break!