Slide 1

Administrivia

- Reminder: Homework 8 was due last week. I'll accept revisions at least through next Monday.
- Reminder: Homework 9 due May 8 (firm deadline this time).
- Apologies for the delay in grading Homework 7. I should be able to do that in the next few days.
- About getting help with these last few assignments: I can check, but usually the ACM tutoring stops after classes are over. I may be on campus a few days next week (details by e-mail early next week) and either way will plan to do some virtual office hours (times TBA). And I do respond to e-mail pretty well!

Slide 2

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! (He recently got in touch to say, however, that they're moving to C++.)
- 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 3

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 4

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 5

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.
- Many other things — multithreading, GUIs/graphics, networking, etc. — are typically done with platform-specific libraries, though some cross-platform libraries exist.

Slide 6

C Non-Standard Features and Libraries — UNIX/Linux

- Text-based GUI-ish features available (most systems?) via the `ncurses` library.
- "Real" GUIs and graphics available via a lot of libraries. "X11" is very low-level; other "toolkits" available.
- Networking available via "sockets".
- (Examples of multithreading and text-based "GUIs"?)

Slide 7

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?)

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

- I asked you about the video lectures and most (but not all!) found them useful. If you've taken another course in which lectures were all on video and class time was used for something else, how did that work for you? If you haven't, does it sound like something you'd like? (Now that I know how to make video lectures I'm speculating about how to use them as a supplement rather than substitute.)
- How are you doing with Homeworks 8 and 9? anything noteworthy to report yet?
- And best wishes for a good summer break!