Slide 1

Administrivia

- Reminder: Homework 6 due today.
- There will be one more short assignment — one problem using pointers — to be assigned tomorrow and due a week later. That will also be the "not accepted past" deadline for all assignments.

  Should there be extra-credit problems too?
- Everything up to Homework 6 graded. I will send out grade summaries . . . after grading this assignment too?
- Sample solutions to all quizzes and most homeworks available online. I will post one for Homework 6 soon.

Slide 2

More Administrivia

- Reminder: Final Friday.

  Draft review sheet online. I'm considering relaxing the rule about not using the classroom computers except to browse. Thoughts?

  Should there be a review session? I could do Wednesday afternoon.

Minute Essay From Last Lecture

- (Question was about what else you would need in order to write programs you might find useful.)
- More/better understanding of what's been covered? yes — practice helps with that, though.
- Networking, images, GUIs? all helpful in writing "real" application programs, yes. standard C provides none of these, alas, though there are many non-standard or platform-dependent libraries available.

Course Recap

- Course is an "introduction to programming."
- Ideally, a first course would focus more on ideas of programming than details — except that, in the words of a colleague

  "Programming is not a spectator sport."

  so we have to choose a programming language, and an environment, and then it's difficult *not* to get caught up in the details.

Slide 5

Course Recap, Continued

- Course intended as introduction to programming for students majoring in Engineering Science, taught in a language acceptable to them, with some exposure to Linux command-line environment.
- Choice of examples and assignments meant to slant toward those of use in STEM field.
- Some material normally covered in a first course for majors omitted/skimmed.

Slide 6

What I Hope You Got From This Course

- A basic understanding of what programming is — expressing a problem and its solution as “an algorithm” and turning that into code.

  In particular I tried to make at least some assignments not-totally-trivial, to give a sense of what you can do with programming skills.
- A basic knowledge of C and its quirks.
- Exposure to Linux command-line tools.

Slide 7

"Why C?", Revisited

- C would not be most people's choice as a beginning language — must learn both programming basics in general and C quirks. (But our department used it in CS1 at one time!)
- But traditionally it's a "universal language" with implementations on pretty much every platform (though that may be changing?). So you may need it at some point, particularly for "embedded systems" work.

Slide 8

"Why Not C"

- On many occasions I've mentioned "more-recent languages" as being easier to use, safer, etc. Also many of them include extensive standard libraries that support GUIs, graphics, networking, etc., etc.
- In my thinking, for general-purpose/application programs one of these is the way to go. Popular choices include C++, Java, and Python (particularly the latter, for people outside CS). We like Scala but it is not (yet?) as widely-used.
- Does that mean it was useless to learn C? I say no! good to have in your "bag of tricks", and once you know *one* programming language, the next is easier, and the one after that is easier still . . .

Minute Essay

- None really — sign in, mostly, but also tell me:
- Are you interested in a review session Wednesday? (if we didn't decide already).
- Are you possibly interested in extra-credit problems?