Slide 1

Administrivia

- Reminder: Homework 5 due today. Okay to turn in incomplete version today, better version soon.
- Quiz 3 scores were, well, "disappointing"? I'll ask you why on the minute essay, but to me this feels like a bad sign. ?

Slide 2

"Random" Numbers — Recap

- To me this is kind of deep and fascinating topic. A not-so-deep view:
- C's `rand()` and `srand()` functions provide a way to generate sequences of integers that look "random" (can't easily predict the next one from previous).
- Usual procedure is to first call `srand()` to initialize and then call `rand()` repeatedly to generate the sequence. Parameter to `srand()` can either be chosen in some way meant to be unpredictable or in some way that lets you repeat tests, possibly with different "seed" values.
- (Finish example from last time.)

Slide 3

Character Data

- As mentioned previously, in C we can represent characters as type `char`.
- Simplest way to input/output a single character is with `getchar` and `putchar`. Note that `getchar` returns an `int`; this is so there can be a "special" value (`EOF`) for "end of file". (For input from a terminal, signal with something system-dependent, control-D on Linux machines.)
- Functions in `ctype.h` classify characters as alphabetic, digits, etc. Functions `toupper()` and `tolower()` do what their names suggest.

Slide 4

Files and C

- Why files? You probably already know: Things stored in memory vanish when you turn the computer off; to preserve them, usually save them as *files*.
- We know one way for a C program to get its input from a file, or write its output to a file — I/O (input/output) redirection. But this makes it difficult to get input from more than one source, or save output in more than one place.
- So C (like many other programming languages) provides ways to work more generally with files.

Slide 5

Streams

- C's notion of file I/O is based on the notion of a *stream* — a sequence of characters/bytes. Streams can be *text* (characters arranged into lines separated by something platform-dependent) or *binary* (any kind of bytes). Unix doesn't make a distinction, but other operating systems do.
- An input stream is a sequence of characters/bytes coming into your program (think of characters being typed at the console).
- An output stream is a sequence of characters/bytes produced by your program (think of characters being printed to the screen, including special characters such as the one for going to the next line).

Slide 6

Streams in C

- In C, streams are represented by the type `FILE *`. `FILE` is something defined in `stdio.h`. (As usual, the `*` means pointer — discussed a bit already, more later.)

  (`FILE` is an example of an "opaque data type" — something defined in a library, the details of which might vary among implementations and which should not matter to users.)
- A few streams are predefined — `stdin` for standard input, `stdout` for standard output, `stderr` for standard error (also output, but distinct from `stdout` so you can separate normal output from error messages if you want to).
- To create other streams — next slide.

Slide 7

Creating Streams in C

- To create a stream connected with a file — `fopen`.
- Parameters, from its `man` page:
  - First parameter is the name of the file.
  - Second parameter is how we want to access the file – read or write, overwrite or append — plus a `b` for binary files.
  - Return value is a `FILE *` — a somewhat mysterious thing, but one we can pass to other functions. If `NULL`, the open did not succeed. (Can you think of reasons this might happen?)

Slide 8

Working With Streams in C

- To read from an input stream — `fscanf` or `fgetc`, almost identical to `scanf` and `getchar`. To write to an output stream — `fprintf` or `fputc`, almost identical to `printf` and `putchar`.
- When done with a stream, `fclose` to tidy up. (Particularly important for output files, which otherwise may not be completely written out.)
- (Simple examples.)
- How to get names of files from user? Well . . .

Slide 9

Text Input in C

- We've seen how to read text a character at a time. Many programming languages provide nice ways to get a whole line at a time. C isn't really one of them (and why that is may become clearer after we talk about arrays a week or two from now).
- Many books show various not-perfect approaches; what I prefer instead for filenames is to "pass them as command-line arguments" (more next time).

Slide 10

Minute Essay

- If you didn't do well on Quiz 3, what do you think went wrong?
- Anything noteworthy to report about Homework 5?
- Anything you particularly want me to review Monday?