Administrivia

- Next homework coming soon.

Slide 1

Review — Strings and Pointers

- Strings in C are null-terminated arrays of `char`s.
- Pointers are in some ways a less abstract and less safe version of Java references. They're also in some respects interchangeable with arrays.

Slide 2

Arrays of Text Strings and Command-Line Arguments

- If you can have arrays of `int` and `char` and so forth — can you have arrays of text strings? Sure! They look like two-dimensional arrays of `char`, or like arrays of `char *`.
- Further, this is how C programs get input “from the command line” (e.g., when you write `gcc myprogram.c`, `gcc` somehow gets `myprogram.c`, right?):

  `main` can also be defined as

  ```
  int main(int argc, char * argv[]) { ....  }
  ```

  where `argc` is the number of arguments, plus one, and `argv` is an array of strings containing the arguments. Example — let’s write a simple “echo” program.

I/O in C — Basics

- We talked already about single-character I/O (`getchar` and `putchar`).
- You already know about a function to write output to “standard output”, `printf`. Many options, allowing a lot of control over what’s printed.
- How about input? Counterpart of `printf` is `scanf` (skim man page). Simple to use, though error detection is somewhat crude, and reading text strings can be hazardous.
- One way to work with files is I/O redirection. Is there something more general? Yes . . .

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 *` — i.e., a pointer to a `FILE`, which is something defined in `stdio.h`.
- 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 (for now, text in double quotes).
  - 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`, almost identical to `scanf`. To write to an output stream — `fprintf`, almost identical to `printf`. `fgetc` and `fputc` may also be useful.
- When done with a stream, `fclose` to tidy up. (Particularly important for output files, which otherwise may not be completely written out.)

Minute Essay

- None — sign in.