Slide 1

Administrivia

- Reminder: Homework 8 due today.
- (Next homework to be posted by next Monday. Due the following Monday.)

Slide 2

Minute Essay From Last Lecture

- (Most people who attempted to give answers gave the right ones. That's good!)

Slide 3

UNIX Tip — Input Redirection Revisited

- I've said that "standard input" doesn't have to mean the keyboard? One option, mentioned previously, is to get input from a file (using redirection operator <).
- Another option is to get input from a "pipe" — connect output of one program with input of another. Together with the simple program `echo` . . .

  ```
  echo 1 2 3 4 | a.out
  ```

  which means that if `a.out` uses `scanf` to read input, values come from the "echoed" text.
- Very useful (I think!) during program development, so you don't have to type the same input over and over.

Slide 4

Bit Manipulation in C — Review/Recap

- Last time I talked about various C operators for working with integers at the level of individual bits.
- One example — show bits of integer (`showlong` program).
- Another example — functions to set, clear, and test individual bits (`flags` example — finish).

Slide 5

Non-Text (“Binary”) Files in C

- Up to this point we’ve dealt only with text files. But surely not all files are plain text? right, so how to work with them?
- In C, can have “binary files”. Open as usual, but include `b` in “mode” parameter (e.g., `"rb"` rather than just `"r"`). What’s different about this? on UNIX systems, nothing really, but on other systems, lines can be separated with two-character sequences, and processing of text streams automatically converts to/from single `'\n'`. Opening files as binary avoids that.

Slide 6

Binary Files in C, Continued

- Open with `fopen` and close with `fclose` as usual, except include `b` in mode when opening.
- Read with `fread`; write with `fwrite`. (Look at `man` pages for description of parameters.)
- Probably worth noting that while text files are relatively portable (many tools deal with converting between different conventions for line separators), binary files probably aren’t.

Non-Sequential File Access

Slide 7

- We've also been reading and writing files sequentially, and often that's what you want, but not always — consider for example a big database of some sort, where it would be nice to somehow represent the data in some way that avoids having to read the whole file to find one piece of information. Exactly how to do that might be complicated, but to do it at all, need some way to read file non-sequentially — "random access".
- In C, functions `fseek` and `ftell` make this possible.
- (Also note `rewind`.)

Minute Essay

Slide 8

- Anything noteworthy about Homework 8?
- (And best wishes for a good holiday!)