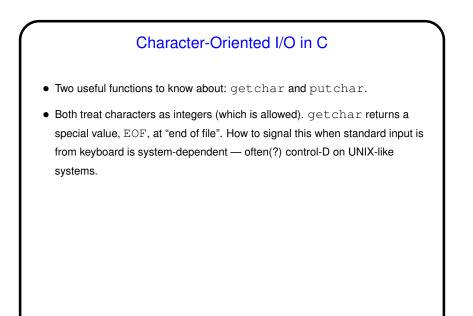
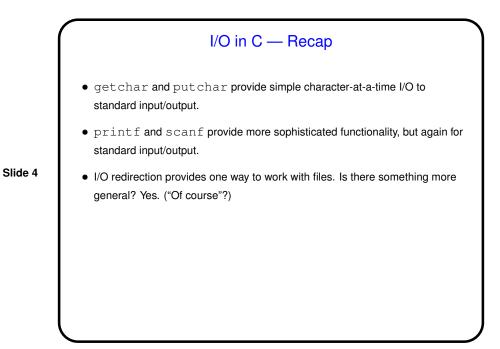


Minute Essay From Last Lecture
No complaints about too much work (good!).
Several people found the second problem (working with "random" numbers) more difficult than previous ones. Not a surprise, and in a programming class

it's probably good if problems get more difficult as the semester goes along?



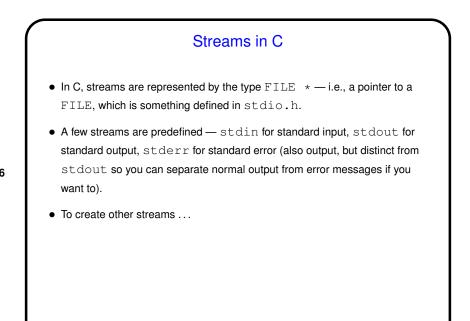


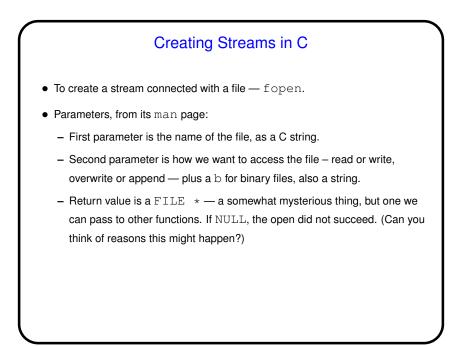
## File I/O — Streams

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/Linux doesn't make a distinction, but some other operating systems do.

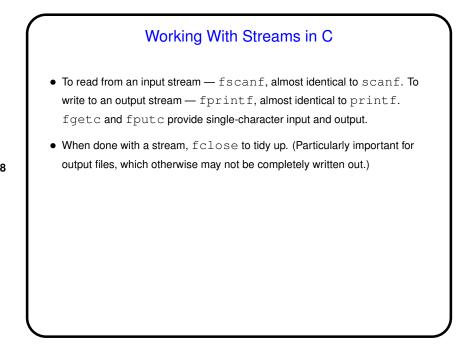
Slide 5

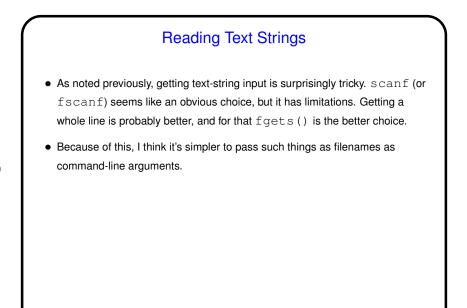
- An input stream is a sequence of characters/bytes coming into your program (think of characters being typed at the console).
- An output stream is a sequence of characters/bytes produced by your program (think of characters being printed to the screen, including special characters such as the one for going to the next line).





## Slide 7





Slide 9

## Simple Examples First do a simple example of character-oriented I/O, using getchar and putchar for a first version and then fgetc and fputc. Then try an example (a revised program to sum inputs) of using fscanf and fprintf to read/write integers. Notice that fscanf "fails" in two situations — end of file and bad input. One way to tell which has happened is with feof(), which returns "true" at EOF. Notice that this function only returns "true" after you've tried to read something but EOF was detected.

