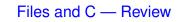


Slide 1

More Administrivia As mentioned in mail, apparently there's some confusion about computer use for quizzes (and exams): It's okay to do whatever you need to do to look at the course Web site and your notes and graded work, but that's all — in particular, using the compiler to find out what a piece of C code does or to write programs is *NOT* allowed. (I think I did say this before the first quiz but I guess many didn't remember?) (What would be the point of asking "what does this code do?" if you could type it in and try it?)

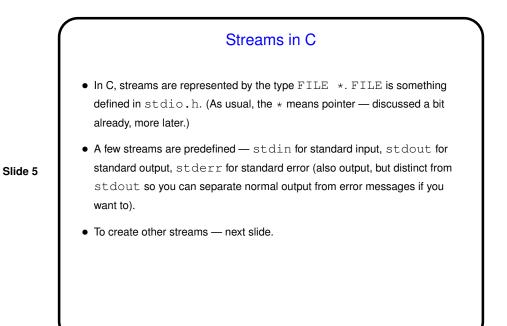


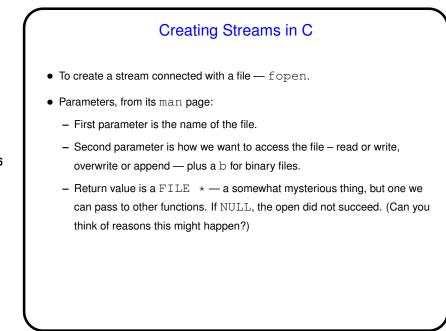
- Why files? You probably already know: Things stored in memory vanish when you turn the computer off; to preserve them, usually save them as *files*.
- We know one way for a C program to get its input from a file, or write its output to a file — I/O (input/output) redirection. But this makes it difficult to get input from more than one source, or save output in more than one place.
- Slide 3
- So C (like many other programming languages) provides ways to work more generally with files.

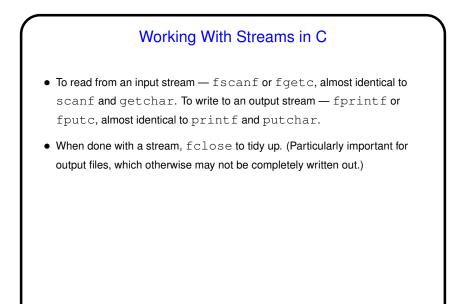
Streams

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 7

Another Way to Get Input - Command-Line Arguments

- (We can't completely discuss this until a bit later, but it's so useful for working with files that we'll do just a bit now.)
- You may have observed that most of the commands you use don't prompt you for input, but instead decide what to do based on what you type on the command line after the command name? so the program must be getting that information somehow, but how? "command-line arguments" (e.g., for the command gcc -Wall hello.c there are two command-line arguments).

(And those commands? Many of them are C programs!)

 Most programming languages provide a way to access this text, often via some sort of argument to the main function/method.

