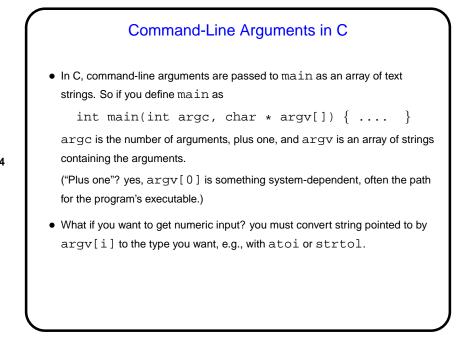


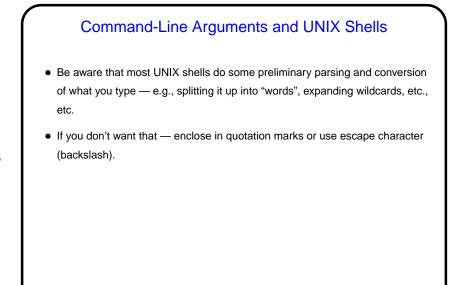


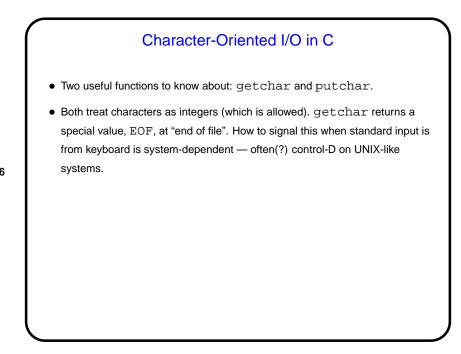
Now that we know about arrays, pointers, and text strings, we can talk about command-line arguments. What are they? text that comes after the name of the program on the command line (e.g., when you write gcc -Wall myprogram.c, there are are two command-line arguments), possibly modified by the shell (e.g., for filename wildcards).

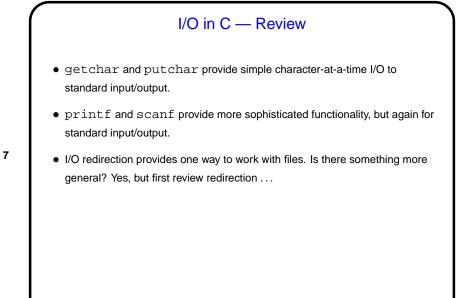
Slide 3

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

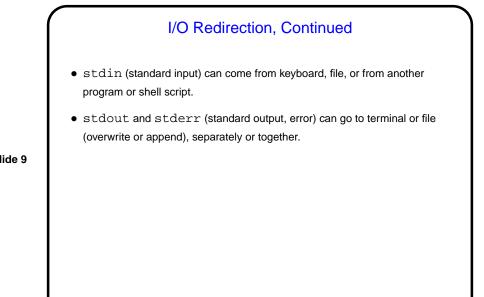


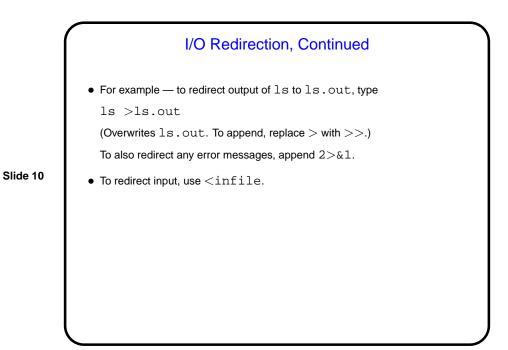






Sidebar — Input/Output Redirection in UNIX/Linux In programming classes I talk about "reading from standard input" rather than "reading from the keyboard", and "writing to standard output" (or "writing to standard error") rather than "writing to the screen". What's the difference?



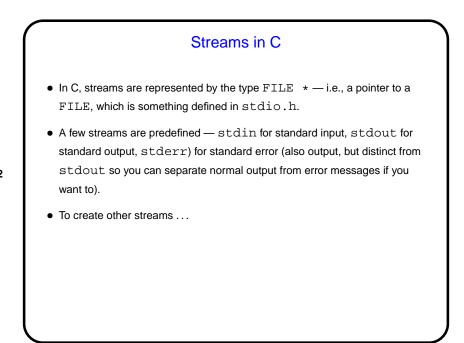


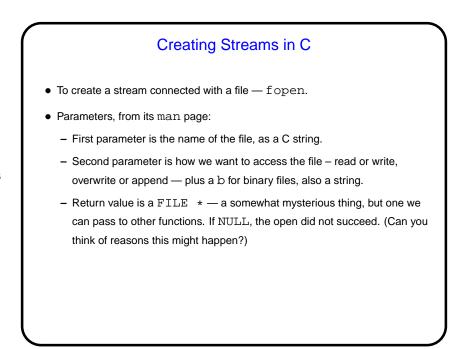


C's notion of file I/O is based on the notion of a *stream* — a sequence of characters/bytes. Streams can be *text* (characters arranged into lines separated by something platform-dependent) or *binary* (any kind of bytes). UNIX/Linux doesn't make a distinction, but some other operating systems do.

Slide 11

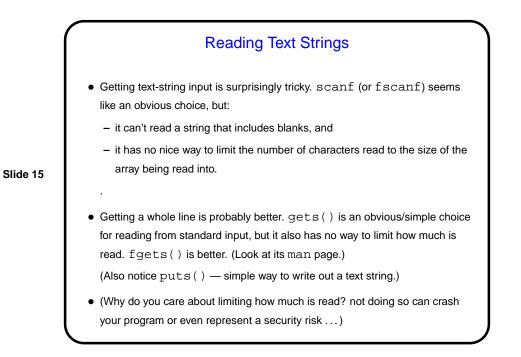
- An input stream is a sequence of characters/bytes coming into your program (think of characters being typed at the console).
- An output stream is a sequence of characters/bytes produced by your program (think of characters being printed to the screen, including special characters such as the one for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

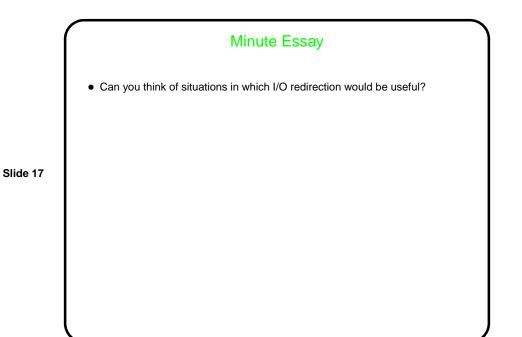




Working With Streams in C

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 provide single-character input and output.
When done with a stream, fclose to tidy up. (Particularly important for output files, which otherwise may not be completely written out.)





Minute Essay Answer