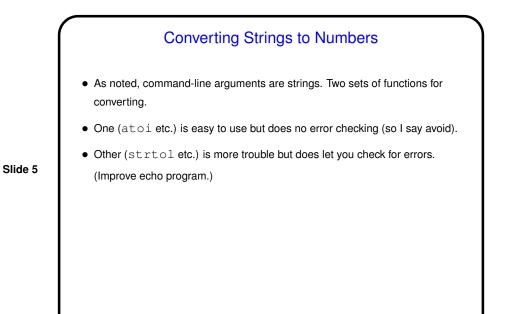
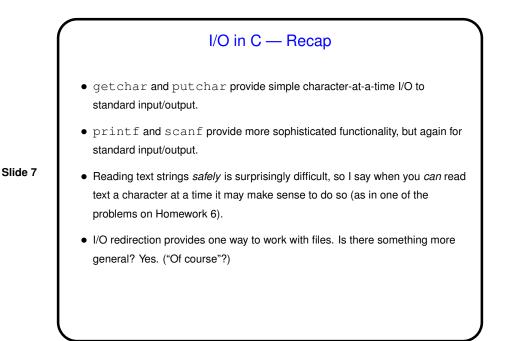


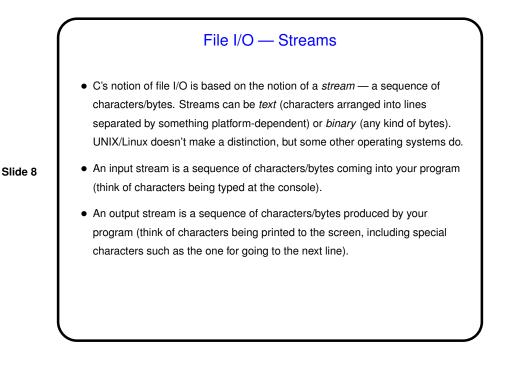
Command-Line Arguments and UNIX Shells

- Be aware that most UNIX shells do some preliminary parsing and conversion of what you type e.g., splitting it up into "words", expanding wildcards, etc., etc.
- If you don't want that: Enclose in quotation marks or use escape character (backslash).



Character-Oriented I/O in C
Two useful functions to know about: getchar and putchar.
Both treat characters as integers (which is allowed). getchar returns a special value, EOF, at "end of file". How to signal this when standard input is from keyboard is system-dependent — often(?) control-D on UNIX-like systems.
(Sample program echo-text.c illustrates using these — not shown in class.)





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.
(FILE is an example of an "opaque data type" — something defined in a library, the details of which might vary among implementations and which should not matter to users.)
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 ...

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, as a C string.
Second parameter is how we want to access the file – read or write, overwrite or append — plus a b for binary files, also a string.
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?)

