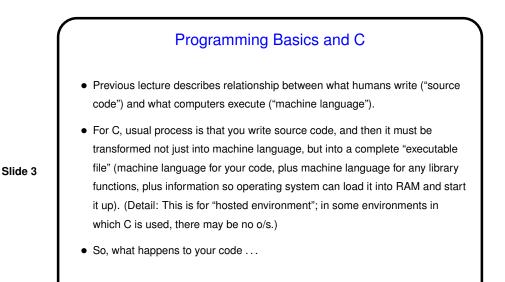


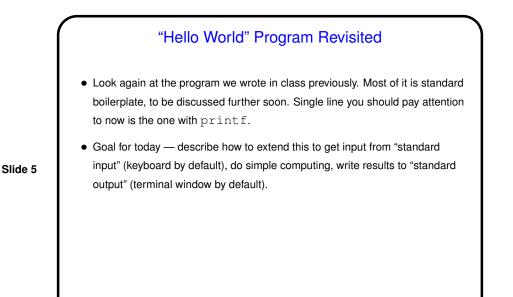
Slide 2

Most people thought pace was pretty much okay, though having more of a chance to practice in class would be good. So — maybe. Several people aren't clear on the roles of various tools. Brief review shortly. "What language do I learn after C"? Depends on what kind of programs you want to write! ask me, or ask someone who's doing a kind of programming that appeals to you. One person has a developer friend who on hearing we were learning vim said "good luck with that". Yeah. People love it or they hate it.

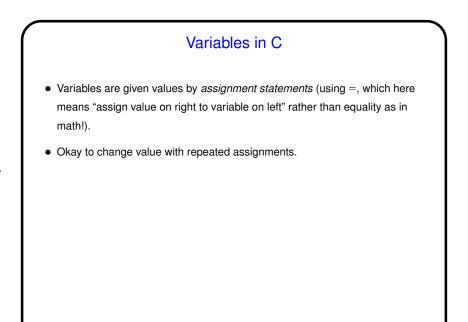


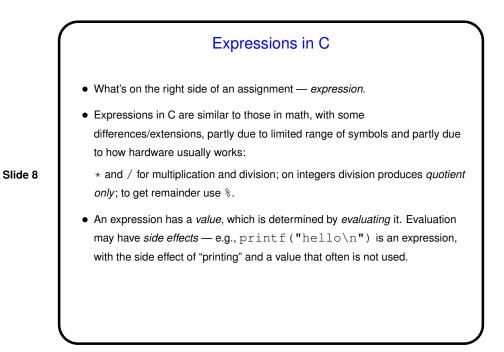
Programming Basics and C, Continued

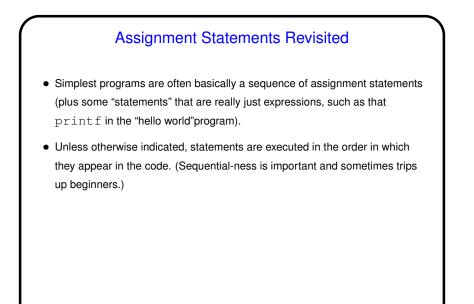
- Your code is first "compiled" into "object code" (machine language).
 Then it's "linked" with any library object code to form "executable file".
 Sometimes (as for examples we're doing now) both steps happen as a result of a single command.
- Slide 4
- To recap, we're basically using two tools, vim (to write/edit programs) and gcc to compile them, and a command-line environment (terminal window) to run both tools and also programs we write.



Variables in C
In C as in most/many other programming languages, you need temporary storage for data — e.g. someplace to save an input value and/or intermediate results. For this we use *variables*.
Again in C as in many others — *variables*. In C variables must be *declared*, each with both a name and a *type*. Effect of declaring a variable is to reserve RAM for a value of the specified type and give it a name that can be referenced. (Similar to Matlab, except for choice of types?) What a name can look like is somewhat restricted (see textbook).
Types in C are pretty basic — integers, "floating-point numbers" (for now, real numbers), and characters. Integer types are represented as fixed-size binary numbers and include various "sizes". More about the others later.







Simple I/O in C

- Use printf to display predefined text and values of variables. Syntax is that of "function call" (more later) with first parameter a "format string" that may include "conversion specifications". Followed by zero or more expressions, one for each conversion specification. When statement is executed, expressions are evaluated and the results turned into something printable using those conversion specifications.
- Use scanf to get input. (It's not really very good for interactive programs, but it's what almost all intro texts use, so we will too, but keep in mind that it has limitations and annoyances.) Syntax very similar to that of printf except that rather than expressions you have *pointers* that say where to store value(s). More about pointers later; for now usually name of variable preceded by &.

Slide 10

