

Slide 2



• Functions in C are conceptually much like functions in other procedural programming languages. (Functions in object-oriented languages are similar but have some extra capabilities.)

I.e., a function has a *name*, *parameters*, a *return type*, and a *body* (some code).

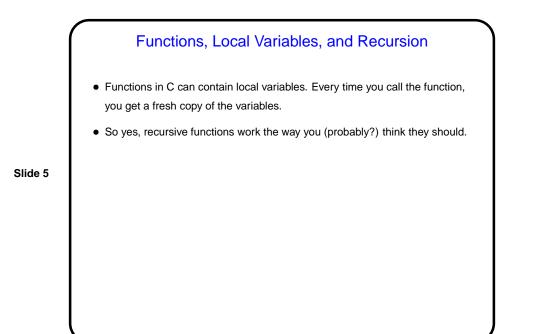
- One difference between C and higher-level languages: You aren't supposed to use a function before you tell the compiler about it, either by giving its full *definition* or by giving a *declaration* that specifies its name, parameters, and return type. The function body can be later in the same file or in some other file.
- Also, C functions are not supposed to be nested (though some compilers allow it.)

Parameter Passing in C

- In C, all function parameters are passed "by value" which means that the value provided by the caller is copied to a local storage area in the called function. The called function can change its copy, but changes aren't passed back to the caller.
- An apparent exception is arrays no copying is done, and if you pass an array to a function the function can change its contents (as we'll do in the next example program). Why "apparent exception"? because really what's being passed to the function is not the array but a pointer! so the copying produces a second pointer to the same actual data. (More about pointers soon.)

Slide 3

Slide 4



Library Functions in C
C does include a library of standard functions, though it's not as extensive as that of some languages.
At least on UNIX-like systems, for each library function there should be a man page that tells you about it, including information about #include files you need and link-time options (e.g., -lm for sqrt). For now, be advised that asterisks in types denote pointers, which we will talk about soon.

Slide 6

