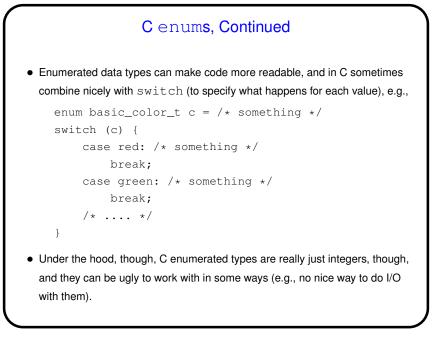
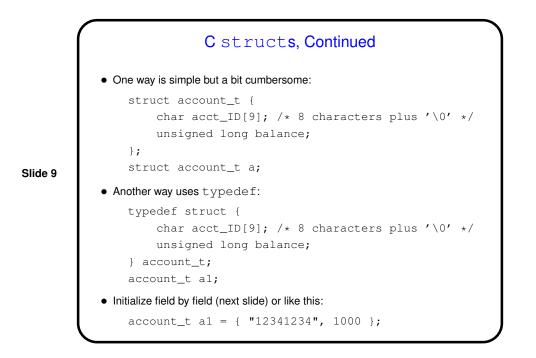
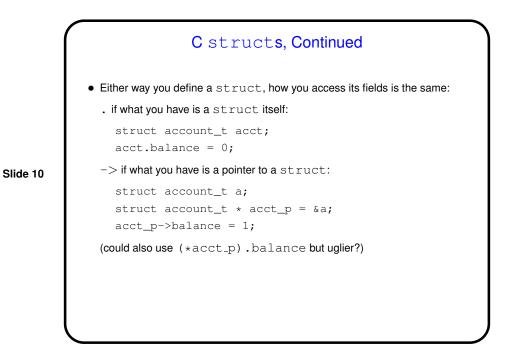


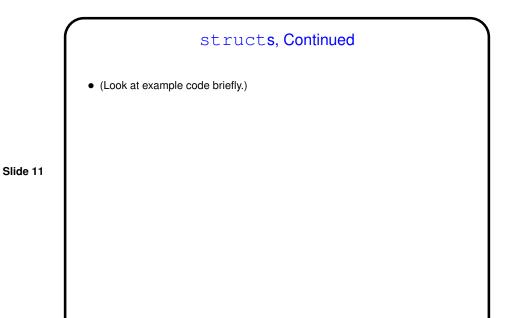
Slide 7



User-Defined Types in C — struct
More complex (interesting?) types can be defined with struct, which lets you define a new type as a collection of other types — something like a class in an object-oriented language, but with no methods and no way to hide fields/variables.
Two versions of syntax (next slide) ...



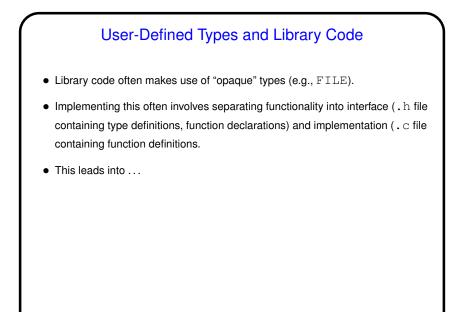




User-Defined Types in C — union
For completeness, we should mention that C also provides a way of defining a structure that can contain one of several alternatives ("this OR that", as opposed to the "this AND that" of struct) — union.a
For example, the following declares a data type that can hold either a float or an int:

union thing {
float f;
int i;
;
union thing t1;
t1 can hold either a float (t1.f) or an int (t1.i) but not both.

More in textbook about this; it can be useful, but can also make code more difficult to understand.



Slide 13

Separate Compilation and ${\tt make}$ — Review

- C (like many languages) lets you split large programs into multiple source-code files. Typical to put function declarations (headers), constants, etc., in file ending . h, function definitions (code) in file ending . c. Compilation process can be separated into two steps: "compile" (convert source to object code) and "link" (combine object and library code to make executable).
- make can help manage compilation process. (Can also be useful as a convenient way to always compile with preferred options.)

