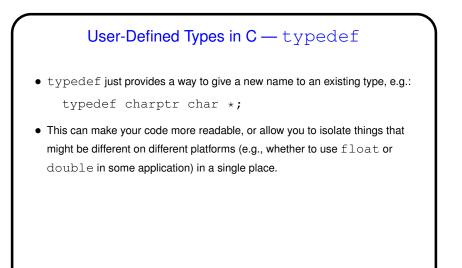
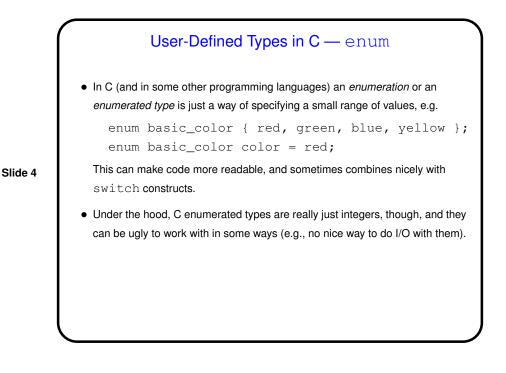
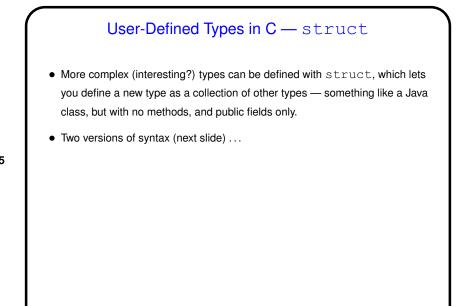
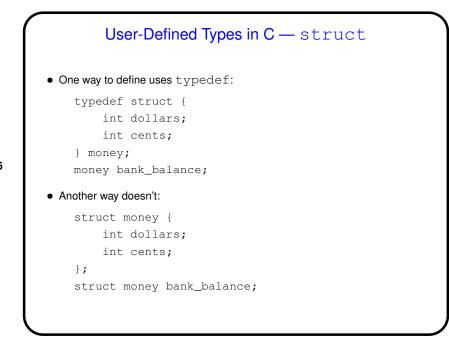


User-Defined Types
So far we've only talked about representing very simple types — numbers, characters, text strings, arrays, and pointers. You might ask whether there are ways to represent more complex objects, such as one can do with classes in Java.
The answer is "yes, sort of" — C doesn't provide nearly as much syntactic help with object-oriented programming, but you can get something of the same effect. But first, some simpler user-defined types ...

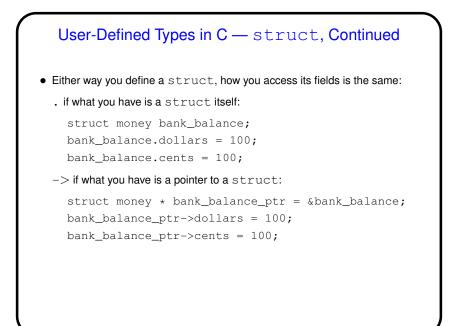


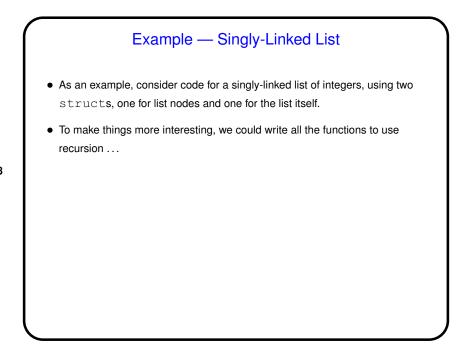




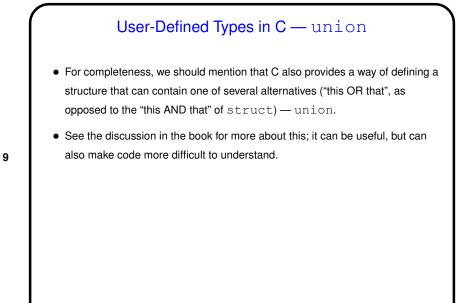


Slide 6





Slide 8



Slide 10

A Little More About gcc Many, many compiler options for gcc. One of the most useful is -Wall. To automate using them every time, you can use the UNIX utility make ...



 Motivation: Most programming languages allow you to compile programs in pieces ("separate compilation"). This makes sense when working on a large program — when you change something, just recompile parts that are affected.

Slide 11

Slide 12

• Idea behind make — have computer figure out what needs to be recompiled and issue right commands to recompile it.

Makefiles First step in using make is to set up "makefile" describing how files that make up your program (source, object, executable, etc.) depend on each other and how to update the ones that are generated from others. Normally call this file Makefile or makefile. Simple example (assuming main.c #includes defs.h and foo.h): main: main.o foo.o gcc -o main main.o foo.o main.o: main.c defs.h foo.h gcc -c main.c foo.o: foo.c gcc -c foo.c When you type make, make figures out (based on files' timestamps) which files need to be recreated and how to recreate them.

