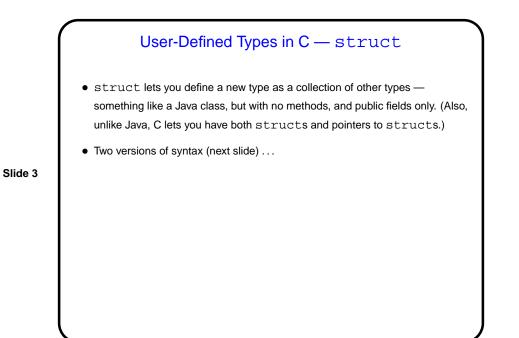
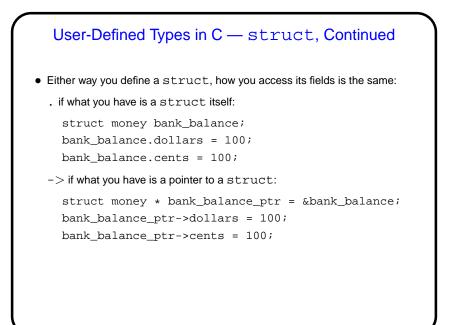


Slide 2

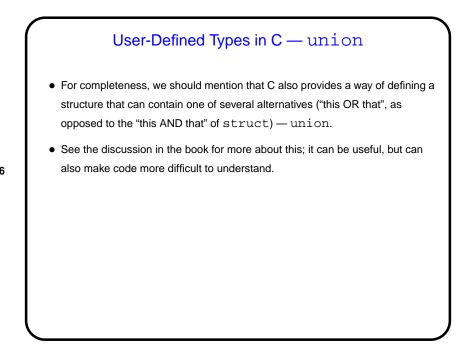


User-Defined Types in C — struct
• One way to define uses typedef:
 typedef struct {
 int dollars;
 int cents;
 } money;
 money bank_balance;
• Another way doesn't:
 struct money {
 int dollars;
 int cents;
 };
 struct money bank_balance;

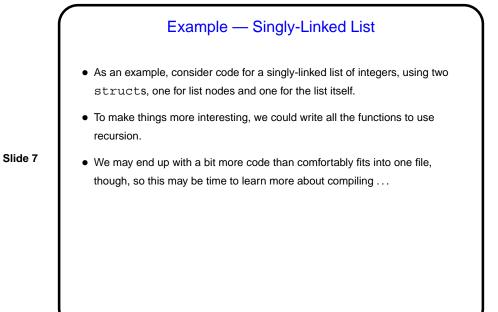
Slide 4

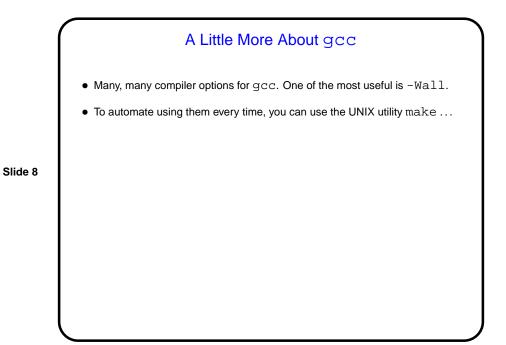


Slide 5



Slide 6







 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 9

Slide 10

 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.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.

