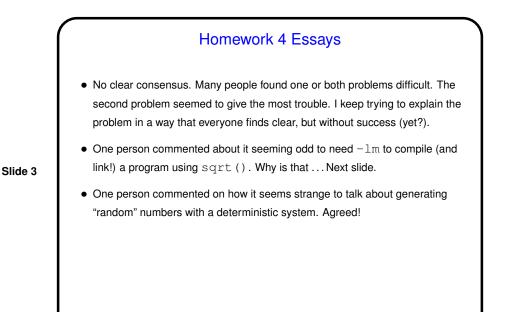
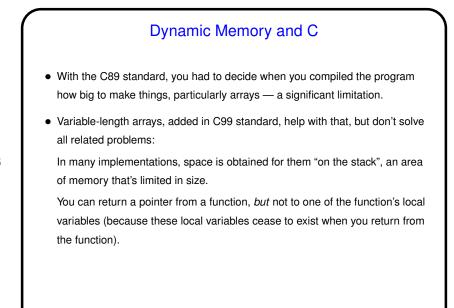


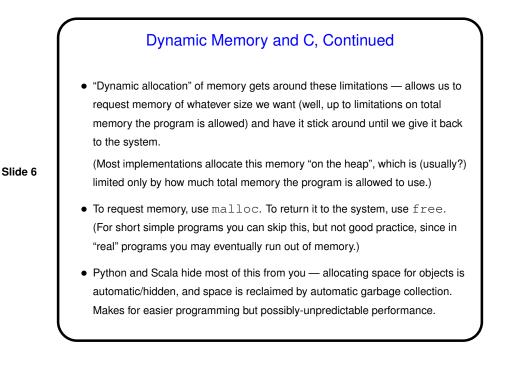
Minute Essay From Last Lecture
No clear consensus; some people compiled using gcc with no flags; others made one of the other choices.
Some said they started with just gcc and then added -Wall if they ran into problems; others that they liked the extra warnings all the time. Be advised that it's a quirk of C (or gcc?) that sometimes warnings indicate problems you'd only observe if you compile with optimization (-O).
A few people commented that it was nice not to have all programs compile to a.out.
If you're interested in using make but don't know how, it's in the slides for 9/05.

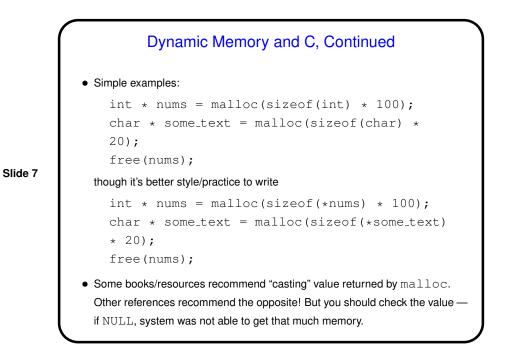


Compiling/Linking — Review/Clarification

- When you type gcc hello.c, gcc is actually doing two steps:
- In the first step (compiling), it converts your code into "object code" (binary
  machine instructions plus some extra info about, e.g., function names). At this
  point it doesn't need access to code for library functions; it just needs to know
  their names and signatures, typically via a . h file.
- In the second step (linking), it combines your code with library code. At this
  point it needs the actual library code (as object code). For some reason,
  probably historical, most of the standard C library functions are in a place
  where the linker looks be default. The math functions aren't, and -lm tells it
  where to look.







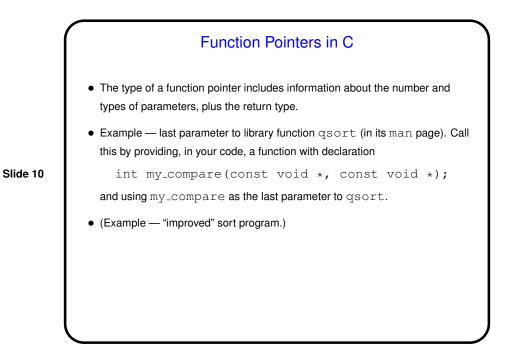
• (Example — "improved" sort program.)
Slide 8

## **Function Pointers**

• You know from more-abstract languages that there are situations in which it's useful to have method parameters that are essentially code. Some languages make that easy (functions are "first-class objects") and others don't, but almost all of them provide some way to do it, since it's so useful — e.g., providing a "less-than" function for a generic sort.

Slide 9

• In C, you do this by explicitly passing a pointer to the function.



## More vim Tips

• To edit multiple files at once, vim followed by their names. :next takes you to the next file, :rew back to the first one. :q exits only from the current file; :qall to exit from all.

Or use "split the screen" (:split) to show two files (or two parts of the same file) at once; control-W twice switches between them. :split followed by filename splits the screen and puts the other file in the new "window".

- You (probably? maybe?) know about diff to compare contents of two files.
   What you might not know about is vimdiff, which shows files side by side (or one above the other with −o) using colors to highlight differences.
- If you don't like the colors, there are options: Type :colorscheme and a space and press "tab" repeatedly to cycle through choices, enter to try one. If you find one you like, put command in .vimrc file.

