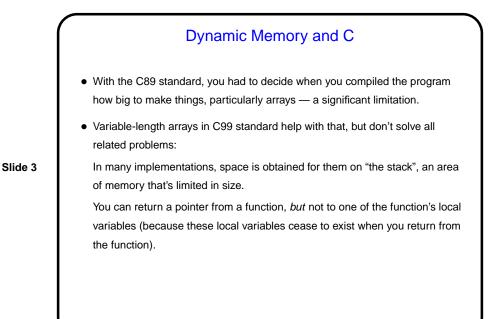
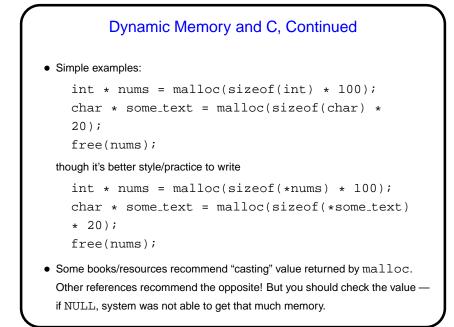


A Little About the C Preprocessor
C logically divides the process of producing an executable into distinct phases. First phase is "preprocessing".
Preprocessing makes use of "preprocessor directives", which start with a #.
Examples you've seen — #include to include information about library functions, #define to define constants.
Other functionality includes macros and "conditional compilation". More in chapter 14, some beyond the scope of this course. Focus is on relatively simple text manipulation.



Dynamic Memory and C, Continued "Dynamic allocation" of memory gets around these limitations - allows us to request memory of whatever size we want (well, up to limitations on total memory the program can use) and have it stick around until we give it back to the system. (The trick here is that most implementations differentiate between two areas of memory, a "stack" used for local variables, and a "heap" used for dynamic memory allocation. Usually the former is more limited in size.) • To request memory, use malloc. To return it to the system, use free. (For short simple programs you can skip this, but not good practice, since in "real" programs you may eventually run out of memory.) • Python and Scala hide most of this from you - allocating space for objects is automatic/hidden, and space is reclaimed by automatic garbage collection.



(Example — improved sort program.)

Slide 5

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.

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

Slide 7

Function Pointers in C
The type of a function pointer includes information about the number and types of parameters, plus the return type.
Example — last parameter to library function qsort (in its man page). Call this by providing, in your code, a function with declaration

int my_compare(const char *, const char *);
and using my_compare as the last parameter to qsort.

(Example — improved sort program.)

