Separate Compilation, Project 1 Array Walkthrough

Bryce Boe

2013/07/09

CS24, Summer 2013 C

Outline

- Finish copy.c
- Libraries and Separate Compilation
- Project 1 Array Walk Through

Finish copy.c (File I/O example)

- <In class completion of copy.c>

LIBRARIES AND SEPARATE COMPILATION

What?

- A **library** (also referred to as modules) is a collection of structures and functions that perform some function
  - stdio: Provides the FILE struct and input and output routines
  - list (project 1): Provides a List struct and associated operations

Example

- <In class example using the following files:>
  - pre_library.c
  - library_usage.c
  - cs24lib.c and cs24lib.h
  - cs24lib_ext.c and cs24lib_ext.h

Notes from the example

- In order to re-use functions they need to be in their own files
- Use MACRO conditionals to prevent #including the same *code* more than once
- Separate structure definitions and function declarations into .h files to support *separate compilation*

Library Components: Header File (.h)

- Provides the *interface* for the module
- Defines data structures (e.g., FILE, List, Node)
- Declares function prototypes
  - int get_at(struct List *list, int index);
- Uses macros (#define, #ifndef, #endif) to prevent duplicate declarations

Library Components: Implementation File (.c)

- Provides the *implementation* for the module
- Uses the #include macro to include the associated header
- Provides the function definition (i.e., the completed source code)

Questions

- Why should you never #include a “.c” file?
  - Doing so doesn’t allow for *separate compilation*
- What is the purpose of the “#ifndef … #define … #endif” guard around the content of “.h” files?
  - Avoids structures and functions from being declared more than once

Another Question

- What is the primary purpose of separate compilation?
  - To reduce subsequent compilation time by reusing *object* files

PROJECT 1 ARRAY WALKTHROUGH

Time to move around

- Everyone seated is a chunk of memory in the heap
- If you are *allocated* we'll represent that by having you come to the front of the class
- If you represent a pointer (or contain a pointer), you should use your hand to point to the address (another person)

Array-implementation walk through

- struct List* list_construct()
- void list_destruct(struct List *list)
- int list_size(struct List *list)
- int list_is_empty(struct List *list)
- char *list_at(struct List *list, int position)
- int *list_push_back(struct List *list, char *ite)
- char *list_remove_at(struct List *list, int pos)