CSC161 2010F *Imperative Problem Solving*

Class 23: Building Libraries

Held: Tuesday, 5 October 2010

Summary: We consider why and how to put C code in separate files.

Related Pages:

- EBoard.
- Lab: Simple Libraries in C.
- Reading: K&R 4.3-4.6.

Notes:

- For tomorrow: Read Unit Testing 101 for Non-Programmers.
- Remember: You can vote on campus today!

Overview:

- Why We Need Code Libraries.
- Simple Code Libraries in C.
- Lab.

Libraries

- We often write functions that we can reuse in other situations and other contexts.
- We could copy those files from program to program.
- But that makes it hard to propagate updates.
- So different languages provide different ways to build libraries or other collections of useful stuff.

- In C, you can break a program into separate files.
- You can compile those files together.
- You can compile those files separately and then link them together.
- There are also ways to build libraries in C.

- How does one compile a file separately?
  - One builds a `.o` file.
  - `make` knows about `.o` files,
- How does one join all the .o files together?
  - With `cc`.
  - Or with an appropriate make command.

- How does one file know about the code from another file?
  - It can rely on the standard C perspective of “If you don’t tell me anything about a function, I’ll assume that it’s available and returns an `int`.”
  - You can explicitly indicate that the function is available.
- Often, we group the information about a library file into a *header* which we can then include in any file that uses it.
- Headers have a suffix of `.h`.

Lab

- Do The lab.
- Be prepared to reflect.
