$Id: lab7c-headers-adts.mm,v 1.19 2016-02-04 13:05:19-08 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps012b-wm/Labs-cmps012m/lab7c-headers-adts
URL: http://www2.ucsc.edu/courses/cmps012b-wm/:/Labs-cmps012m/lab7c-headers-adts/

1. Overview

This lab will introduce you to header (`.h`) files, also known as **`#include`** files, which are C's equivalent of interfaces, and implementation files (`.c`) which are C's equivalent of implementation files. C has no keyword **`private`**, but its equivalent is to put hidden information only in implementation files.

In this program you will use separate compilation to implement a simple queue of lines in C.

2. Subdirectory cbox

Sample code has been provided. First, study the trivial **`cbox`** application which shows how to do separate compilation using header files and implementation files. Inspect the following files :

`cbox.h` The interface to the **`cbox`** ADT showing the constructor, destructor, accessors, and mutators. In Java these would be the public functions.

`cbox.c` The implemention of the ADT, showing the private functions and fields. Note that the **`struct`** fields are thus hidden from the client.

`main.c` Shows how to create and access an ADT. The application itself is trivial.

`Makefile` Builds each object (`.o`) file and then links them into an executable image. Note how it creates dependencies by editing itself.

3. Subdirectory queue

Provides a simple queueing application, where each line of the input is read into a queue. The queue is then printed and the program stops. The idea of a ***stub*** is introduced. Incomplete code is just represented by a print statement which must eventually be replaced by actual code. The following command will tell you where all of the stub calls are : **`grep STUB *.[hc]`**

You must implement **`new_queue`**, **`insert_queue`**, **`remove_queue`**. your implementation must be done via the several files, whose object modules are linked together into a single executable. Remove any complaints **`valgrind`** may have about memory access or memory leaks. Run **`checksource`**.

4. What to Submit

The **`Makefile`** and all source code, but no object files or executable binaries. Do a build in your own directory to verify that it works. Look in the submit volume to verify that there are no missing files. If you do pair programming, also follow the instructions in **`/afs/cats.ucsc.edu/courses/cmps012b-wm/Syllabus/pair-programming/`**.