CSC 374: Computer Systems 2, 2010 Fall, Assignment #5

Last modified 2010 Nov 12

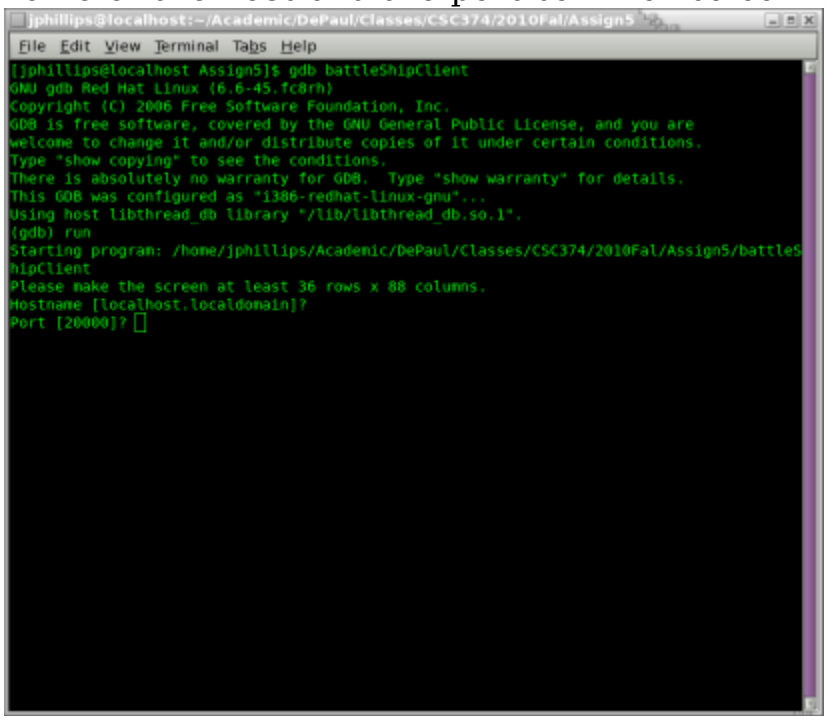
Purpose:

To go over system calls related to sockets and cursor control with the ncurses package.

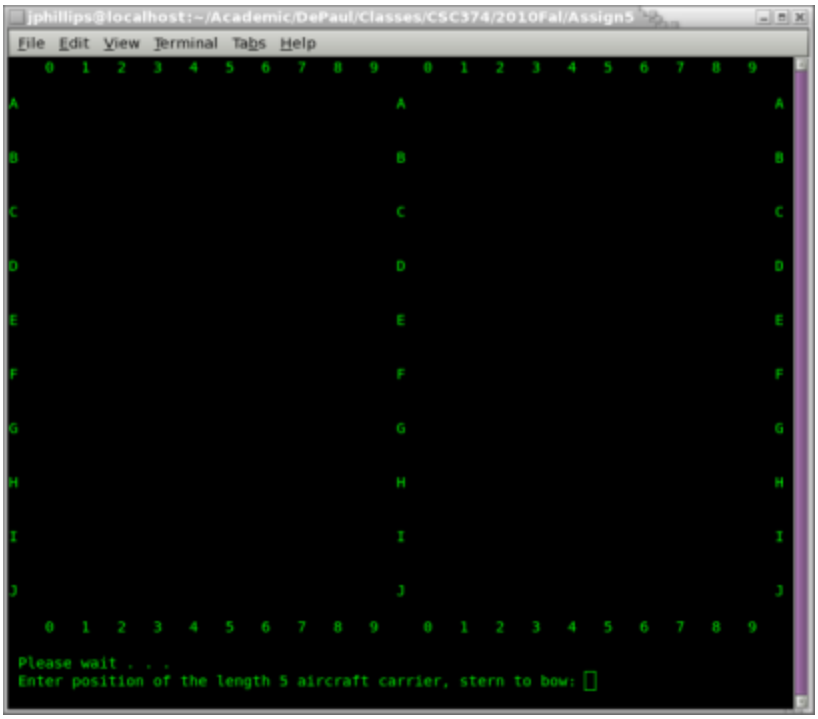
Assignment:

We will write the client program for "Battleship" which will:

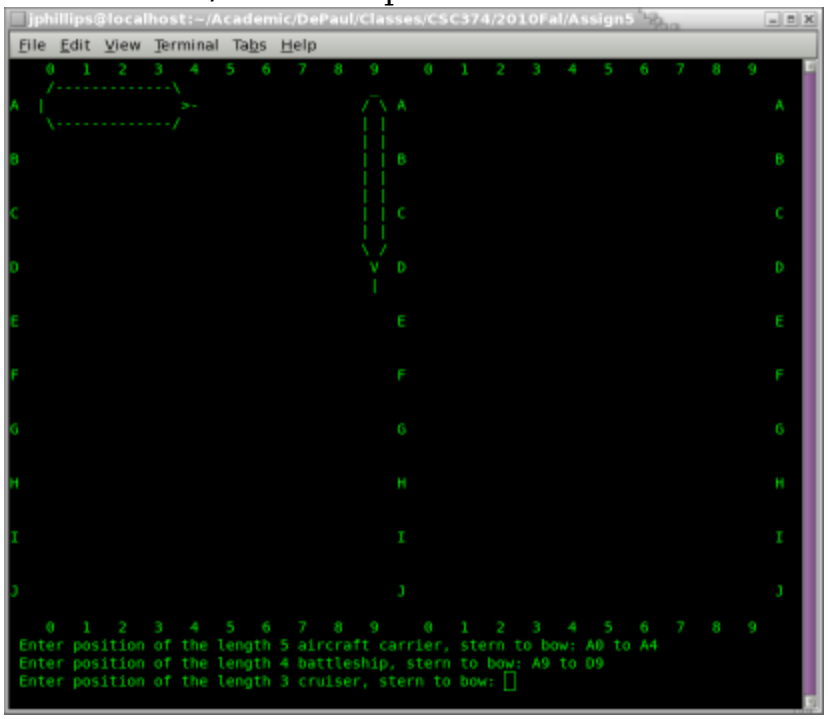
- Ask for the name of the host and the port to which to connect,



- Attempt to connect to the server using sockets
- Initialize the screen with ncurses to show both one's own fleet positions on the left board and what is known about the opposing fleet on the right
- Initialize and status text screen below both boards



- Let the user choose his/her fleet positions



- Play the game

- Inform the user about the outcome

Design of the Battleship client program

The Battleship client program uses 7 files:

- `headers.h`: Used by all .cpp files because it includes most of the required library headers, as well as files `constants.h`, `declare.h`, `StatusDisplayWindow.h`, and

`BattleShipBoard.h`. (**No need to edit this**)

- `constants.h`: Defines many constants (and few basic types) used through-out the program. (**No need to edit this**)
- `declare.h`: *Declares* functions that `battleShipClient.cpp` (and the server too) also use that are defined in `battleShipCommon.cpp`. Also defines some very simple inline functions. (**No need to edit this**)
- `StatusDisplayWindow.h`: Manages the text window used for text communication to/from the user and the server. (**You must complete this!**)
- `BattleShipBoard.h`: Manages the positioning of the fleet, the status of the fleet (who's been hit, who's still afloat), and the display of the fleet. (**You must complete this!**)
- `battleShipCommon.cpp`: Has miscellaneous functions used both by client and server for both socket communication and display. (**You must complete this!**)
- `battleShipClient.cpp`: Has the code directly relating to the operation of the client. (**You must complete this!**)

What you need to do

1. Download `2010Fall_CSC374_Assign5_code.zip` from Course On-Line
2. Download the executable `battleShipServer` from Course On-Line
3. Upload them both to a Linux environment (if you are not already on one). Please use an sftp program like Filezilla (http://filezilla-project.org/) to do so. Just use:
   a. *Host*: `ctilinux1.cstcis.cti.depaul.edu`
   b. *Username*: (Your username)
   c. *Password*: (Your CDM password)
   d. *Port*: 22
4. Login to `ctilinux1.cstcis.cti.depaul.edu`
5. Unzip `2010Fall_CSC374_Assign5_code.zip` with:

   ```
   linux$ unzip 2010Fall_CSC374_Assign5_code.zip
   ```

6. Edit the files with `nano`. The main commands are **Ctrl-O** to save (write-*O*ut) and **Crtl-X** to quit (e*X*it).
7. Start coding! Compile and link with:

   ```
   linux$ g++ -g -c battleShipClient.cpp
   linux$ g++ -g -c battleShipCommon.cpp
   linux$ g++ -o battleShipClient battleShipClient.o battleShipCommon.o -lncurses
   ```

   Look for the comments that say *YOUR CODE HERE*
8. **WARNING:** We are doing socket programming, and we need both clients and the server to be *on the exact same machine!* Please do *not* use `cdmlinux.cdm.depaul.edu` because it is too non-specific. Please use either:

- 3 `ctilinux1.cstcis.cti.depaul.edu` windows (1 server, 2 clients), or
- 3 `ctilinux2.cstcis.cti.depaul.edu` windows (1 server, 2 clients)

Please start the server first, then start up one client, then the other.