CPSC 341 Operating Systems and Networks

Spring 2012

Extra Credit - Homework 6

Due: Saturday, June 2nd at 11:00pm

For this assignment, you must work alone.

# Description

In this assignment, you will be creating a client and server program to play a simple treasure hunt game. Client programs can connect to the server to play the game. Clients send their guess to the server. The server responds by sending the current game status back to the client.

The game is played on a 200 by 200 ft. grid with the origin (0, 0) in the center as follows:

Here is the process to play the game:

1. The player (client) is prompted to enter the name and is sent to the server.
2. The treasure is placed randomly in the grid by the server. It is located at a coordinate with an integral number of feet such (89, -37).
3. The player enters an (*x*, *y*) coordinate as a guess.
4. The client makes sure the guess is valid and sends the guess to the server.
5. The server first checks if the player has found the treasure. If so, go to step 8.
6. If the player did not find the treasure, the server responds to the client with the distance (as the crow flies) from the guess location to the treasure location.
7. Repeat steps 3-6 until the game has won.
8. When the game is won, the server will respond with a victory message that indicates the number of turns it took to guess the treasure location.
9. In addition, the server will send a leader board indicating the name of the top five players along with the number of turns it took to find the treasure (fewer turns are better).

# Treasure Hunt Client (hw6\_client.cpp)

Here is a more detailed description of the client:

1. Start the client will have two command line arguments (both required):
   * IP address of the server
   * port of the server process (use one of the ports assigned to you)
2. Connect with the server. Exit with an appropriate error message if a connection could not be established.
3. Ask the user for their name.
4. Send the name to the server.
5. Ask the user for a guess.
6. Check if the guess is valid (fits within the coordinates). If the guess is invalid, print an error message and go back to step 5.
7. Send the guess to the server.
8. Receive the distance back from the server.
9. If the distance is not zero, print the distance to the screen and go back to step 5.
10. If the distance is zero, receive an additional victory message from the server.
11. Print the victory message to the screen.
12. Receive leader board information from the server.
13. Print leader board information to the screen.
14. Close the connection with the server.

# Treasure Hunt Server (hw6\_server.cpp)

Here is a more detailed description of the server:

1. Start the server will have one command line arguments (both required):
   * port number (use one of the ports assigned to you)
2. Listen for a client.
3. When a client connects, create a new thread to process that client.
4. The first step is to receive the player's name from the client.
5. Receive a guess from the client.
6. If the guess corresponds to the treasure location:
   1. Send a distance of 0.
   2. Send a victory message indicating the number of turns it took.
   3. Update the leader board if the player made the top 5.
   4. Send the current leader board to the client.
   5. Close the connection with the client.
7. If the guess does not respond to the treasure location:
   1. Send the distance (as the crow flies) as a floating point number to the client.
   2. Go back to step 5.
8. The server will run forever until aborted (Ctrl-C).

# Communication

For the most part, it is up to you to determine how the various information is transmitted over the network. However, there are a few rules:

* The guess must be sent as two integers in network order.
* The response must be a single floating point number. It does not require any conversion.
* You may send strings using any method you would like. You may not place a limit on the number of characters on the user’s name.
* The leader board can be sent as a series of records or as a long string (easier).

# Leader Board

A few notes about the leader board:

* When the server is started, the leader board initially starts empty.
* If the leader board contains fewer than five entries, only display the current number of entries.
* Ties are broken by whoever played first.
  + For example, let’s say the following people played in the following order from earliest to latest with number of turns in parentheses: Anna (19), Maya (15), Eric (23), Beth (23), Mark (19), John (15). Then the leader board would be:

**1. Maya 15**

**2. John 15**

**3. Anna 19**

**4. Mark 19**

**5. Eric 23**

* + The ordering is determined by when the leader board is updated at the end of the game.
  + Do not make this more complicated than necessary. This rule actually makes tie handling very simple.
* The output of the client should display each person on the leader board on its own line (like the example illustrating ties above).
* Did you remember that the data structure storing the leader board is shared among the different client threads? Your program should protect against any issues regarding the leader board.
* Assignments that implement the leader board using five variables such as top1, top2, top3, top4, and top5 will be penalized. Use an appropriate data structure instead.

# Sample Playing of the Game

For this sample playing, assume that the treasure is at (5, -12).

Welcome to Treasure Hunt!

**Enter your name: Eric**

Turn: 1

Enter a guess (x y): 0 0

Distance to treasure: 13 ft.

Turn: 2

Enter a guess (x y): -5 0

Distance to treasure: 15.62 ft

Turn: 3

Enter a guess (x y): -5 -12

Distance to treasure: 10 ft.

Turn: 4

Enter a guess (x y): 5 -12

Distance to treasure: 0 ft.

Congratulations! You found the treasure!!

It took 4 turns to find the treasure.

Leader board:

**1. Maya 2**

**2. Mark 4**

**3. Eric 4**

**4. Beth 5**

**5. Eric 12**

# Testing your Program

To test your server and client, it is necessary to start the server before the client. The server and client can either be run on different machines or on the same machine.

10.124.72.20 is the IP address for cs1

10.124.70.21 is the IP address for css2

Use the ports assigned to you in class. If either program terminates (either Ctrl-C or program error) while the connection is still active, you may get a bind error. If this happens, select a different port. In this situation, the OS thinks the port is still being used and it takes a minute or two for the OS to figure out the process using that port has terminated.

# Submitting your Program

# Run your program and capture at least two sample output runs in a file and name it as hw6\_output.txt

# On cs1, run the following script in the directory with your program:

/home/fac/ulmustar/submit/cpsc341/hw6\_submit

This will copy the files hw6\_client.cpp and hw6\_server.cpp and hw6\_output.txt to a directory that can be accessed by the instructor. Please be sure to keep the same file names or the submission program will not work.

The submission program will attempt to compile your program. Your program must properly compile or the submission program will reject your program. Programs that fail to compile will not be graded.