CSE 109: Systems Programming

Fall 2018

Program 7: **Due on Sunday, December 9th at 9pm on CourseSite.**

Collaboration Reminder:

1. You must submit your own work.
2. In particular, you may not:
   (a) Show your code to any of your classmates
   (b) Look at or copy anyone else's code
   (c) Copy material found on the internet
   (d) Work together on an assignment

Assignment: Preparation

1. Make a *Prog7* directory in your class folder.
2. You will be writing a *Client.cpp* (and possibly *Client.h*) file for this assignment.
3. The *prog7student* folder will have a sample Client that allows you to manually enter in commands to interact with the Server.
4. All source code files must have the comment block as shown at the end of this document. All files must be contained in your *Prog7* directory.
5. You are expected to check return values (except for printf/scanf and their variants) for errors via errno and report them (usually aborting when this happens).

Assignment:

In this assignment, you will be interacting with a basic network server and handling responses. We will refer to the components of this assignment Server (which you don't have to code but have to interact with) and Client (which you do have to code). The following page has an example of some working Client/Server code: http://math.msu.su/∼vvb/Java/samples/Invert/Invert.html be aware that you may need to check errors that it does not check and that the code is a bit sloppy. You can adapt portions of the code there for our purposes. You will be expected to understand what functions you end up using and what some of the arguments are what they are.

Here is an additional link that goes into more detail if you are interested: http://www.cs.rpi.edu/∼moorthy/Courses/os98/Pgms/socket.html.

1. Server

   (a) The Server will be running on some sunlab machine. It is possible that the Server ends up running on different machines and different port numbers during the assignment. Therefore, you must read the file *connection.txt* from the *prog7student* folder. The format of *connection.txt* is as follows:

      i. The first two bytes are a short, reflecting the port number.

      ii. The next eight bytes are a size_t reflecting the length of the hostname, the machine that Server is running on.

      iii. The next 'length of the hostname' bytes are a non-NULL terminated string reflecting the hostname.

      iv. Any additional bytes must be ignored, if any.

   (b) Once connected, the Server will send along data regarding the state of the game you are playing on the Server.

      By default, it will load a 10x10 map and you will be able to see the entire map. In fact, for any game with 10 rows in the map, you will always be able to see the map. However, that is for your benefit in coding. You will generally not be able to see the map.

   (c) By sending commands to the Server, you will move from the top left position to the bottom right position by navigating through a maze. The basic commands are first letters of the cardinal directions: "North", "West", "East", "South".

   (d) There is a guarantee that a path exists, at the beginning, from start to end.

   (e) Your score is based on the number of moves taken to reach the end.

(f) You can ignore items, however, coding to handle these can allow for some extra credit if your program can use them to beat the maze faster.

(g) You should use the sample Client given and run through the mazes to get a feel for how things work.

2. Commands (Basic)

   The Server accepts a number of commands that it will accept after it prompts the user. This just lists the basic set of commands, you can get full credit only handling the basic set.

   (a) "N"/"North": Moves you up (row - 1).
   (b) "S"/"South": Moves you down (row + 1).
   (c) "W"/"West": Moves you left (column - 1).
   (d) "E"/"East": Moves you right (column + 1).
   (e) "L"/"Load": Loads a new map, at start of game only (see Loading).

3. Loading

   (a) The load command can only be used on turn 1.
   (b) It takes 3 parameters, as size_t elements.
   (c) The first is the seed that determines the map layout
   (d) The second is the number of rows, max 5000.
   (e) The third is the number of columns, max 5000.
   (f) You will resume the game in the top left corner (0,0), of the newly loaded map.
   (g) Odd seeds will generate maps with items. Even seeds will not.
   (h) Maps with 10 rows will always be displayed

4. TestClient

   (a) TestClient is provided so you can get a feel for how the game works.
   (b) It will automatically find the Server and connect to it.
   (c) You are free to explore the maze at your own pace.
   (d) Use this to figure out what strings you end up getting so you know how to parse things!

5. Server
   (a) The Server will be running independently of your assignment.
   (b) The Server will send messages in the following format:
      i. The first 8 bytes will reflect a size_t that indicates how much data (text) is being sent.
      ii. The remaining bytes will be the text sent. There is no guarantee of NULL-termination of this text.
   (c) The Server will continue to send text until it sends a prompt to the user. The prompt is in the form: *command>*

      Note that there is a single space after > in the prompt.
   (d) After the prompt, the Server will expect the Client to send a command in the following format:
      i. The first 8 bytes will reflect a size_t that indicates how much data (text) is being sent.
      ii. The remaining bytes will be the text sent. You may NULL-terminate this, you don't have to.
   (e) When you reach the end of the maze, the server will automatically disconnect after providing a score.

6. Following the Side of the Maze
   (a) A basic algorithm to solve the maze:
      i. Start by facing North.
      ii. Continue moving in the direction you face until you hit a wall.
      iii. If there is a wall in the direction to your right, face to your right and go to step ii.
      iv. Move to your right, but do not change the direction you face.
      v. If there is a wall in the direction you face, face to your right and go to step ii.
      vi. Move in the direction you face, then face to your left.
      vii. Go to step ii.
   (b) The key thing to notice is how this algorithm attempts to "hug" the wall. Let's look at an example of behavior.

      Let * be our starting location and G be the Goal.

```
* # #   #
     #   #
  #   # #
  ##  # #
 ##  ##
   ##

 #     #
 ### #
  #   #  G
```

(c) i) Initially, we are in 0,0 and facing North.

(d) ii) We hit a wall. iii) No wall to our right (East).

(e) iv) We move to 0,1 and are facing North.

(f) v) There is a wall to the North, we turn East and go back to step ii.

(g) ii) We hit a wall. iii) No wall to our right (South)

(h) iv) We move to 1,1 and are facing East.

(i) v) There is no wall to the East.

(j) vi) Move to 1,2 (East) and face left (North). vii) go to step ii.

(k) ii) We hit a wall. iii) No wall to our right (East)

(l) iv) We move to 1,3 and are facing North.

(m) v) There is no wall to the North.

(n) vi) Move to 0,3 (North) and face left (West). vii) go to step ii.

(o) ii) We hit a wall. iii) There is a wall to our right (North), turn North and go to step ii.

(p) ii) We hit a wall. iii) There is a wall to our right (East), turn East and go to step ii.

(q) ii) We hit a wall. iii) No wall to our right (South).

(r) iv) We move to 1,3 and are facing East.

(s) v) There is no wall to the East.

(t) vi) Move to 1,4 (East), then face to our left (North).

(u) etc, this will eventually get to the end G (9,9). It will take about 60 moves to do so.

1. Commands (Extended)

   For students who want an additional challenge, the maze contains items that can be used while the maze is traversed (or different/better algorithms can be used).

   (a) "I"/"Inventory": Displays your inventory.

   (b) "C"+<item>: Try to use <item>. For example "s" represents the *ring of scare walls*, therefore "Cs" will use that item. Some items, such as food, will be automatically used.

1. Item List

   (a) 'a': box
   (b) 'b': sandwich
   (c) 'c': handful of salt
   (d) 'd': potato
   (e) 'e': shoe
   (f) 'f': chicken nugget
   (g) 'g': rare painting
   (h) 'h': key
   (i) 'i': folder of evidence
   (j) 'j': scroll of walk through walls
   (k) 'k': scroll of polymorph items
   (l) 'l': compass
   (m) 'm': cup of instant noodles
   (n) 'n': sekrit
   (o) 'o': scroll of teleportation
   (p) 'p': scroll of time stop
   (q) 'q': segmentation violation
   (r) 'r': ten-leaf clover
   (s) 's': ring of scare walls
   (t) 't': cabbage

Style:

1. **Review the Style document on Coursesite**
2. For assignments, we follow the Allman style of braces and indentation.

Testing:

1. You should be able to run your code and it will connect to the Server and handle the given maze. It should not assume that it will always get the default maze. We will run it with different mazes as the starting situation - with loading of new mazes disabled.

Submission:

1. Once ready to submit, you can package up the assignment as a .tgz file

   ```
   tar -czvf Prog7.tgz Prog7
   ```

   You must use this command in the directory that contains the *Prog7* folder, not within the directory.

2. Transfer *Prog7.tgz* to the Program 7 submission area of CourseSite.

Comment Block:

```
/*
    CSE 109: Fall 2018
    <Your Name>
    <Your user id (Email ID)>
    <Program Description>
    Program #7
*/
```