COMP 206 Fall 2018 – Assignment 4

November 21st, 2018

Objectives:

Work with structs and linked lists. Practice with networked software systems through the socket library. Writing code to create dynamic web-pages with CGI. Final practice with all the C basics.

On your marks:

Read the slides on sockets and the web up to the CGI section. We have provided most of the necessary networking code for you, but it will be hard to understand what's happening unless you have a good grasp on that content prior to starting.

Get Set:

Download A4.zip and unzip. I recommend that you read carefully through all the files provided. Then, change to the Assignment4 directory. Type "make test". Ensure you see:

```
dmeger@DESKTOP-CDQUCTF:~/COMP206_Fall2018_A4_dev/Assignment4$ make test
./server  &
sleep 1
Game server ready on port 8888.
echo LOGIN,dave,1234 | ./client 127.0.0.1
Enter message : Server response: LOGIN UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo CREATE,dave,1234,mygame | ./client 127.0.0.1
Enter message : Server response: EXAMPLE GAME CREATED. COMPLETE THIS IMPLEMENTATION.
a    |   |
  ---|---|---
b    |   |
  ---|---|---
c    |   |

   1   2   3

Enter message : Failed to get input from stdin.
echo LOGIN,dave2,4321  | ./client 127.0.0.1
Enter message : Server response: LOGIN UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo JOIN,dave2,4321,mygame,b2  | ./client 127.0.0.1
Enter message : Server response: JOIN UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo MOVE,dave,1234,mygame,a1  | ./client 127.0.0.1
Enter message : Server response: MOVE UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo MOVE,dave2,4321,mygame,b1  | ./client 127.0.0.1
Enter message : Server response: MOVE UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo MOVE,dave,1234,mygame,a2  | ./client 127.0.0.1
Enter message : Server response: MOVE UNIMPLEMENTED
Enter message : Failed to get input from stdin.
echo MOVE,dave2,4321,mygame,b3  | ./client 127.0.0.1
Enter message : Server response: MOVE UNIMPLEMENTED
Enter message : Failed to get input from stdin.
dmeger@DESKTOP-CDQUCTF:~/COMP206_Fall2018_A4_dev/Assignment4$
```

You might see errors such as "bind failed" or "port in use". This is because only one student can test their server on each port of any given machine (mimi will be the congested one). As you develop, crashes on your server may also tie up ports for a few minutes and require that you switch often until the crash is fixed. To deal with this, add an argument to change the port, for example: $ make PORT=48888 test.

The finish line: December 3rd, 11:59pm hard deadline

Test everything at least once on mimi, including Q1.

Create a zip file with the two files you must modify to create the solution: ttt_server.c and ttt_cgi.c.

$ zip A4_solution.zip ttt_server.c ttt_cgi.c

Submit the zip file to My Courses.
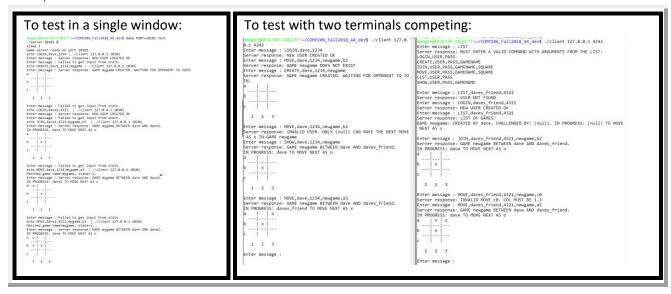
Question #1 – A tic-tac-toe server (70 marks)

We have set up a very basic client server interaction that allows two processes to communicate, but nothing interesting is happening yet. You must complete the server portion so that it implements a tic-tac-toe game system following this protocol:

Client messages consist of a comma separated list of fields used to interact with the game. Each message starts with one of 6 COMMANDs that you must complete and associated arguments:

- LOGIN,USER,PASS which logs an existing user in if the password matches, creates a user for the first time if they did not exist, or returns error if the password is wrong. All following commands must be run with a valid user/pass, so logging in is required as a first step to play.
- CREATE,USER,PASS,GAMENAME which creates a new game with a blank board and the specified user listed as creator.
- JOIN,USER,PASS,GAMENAME,SQUARE which allows the user to join a game that has just been created, as challenger, and lets them play a first move (must be a move on the board, between a1 and c3). Only 2 users can play, so trying to join a game that already has 2 players should fail.
- MOVE,USER,PASS,GAMENAME,SQUARE which allows the user to make a subsequent move. The server should check if it is the users turn (challenger plays all odd numbered moves as "x" and creator all even moves as "o"). Check the outcome of the game after the move. Has someone won or is it a draw?
- LIST,USER,PASS which allows all the games on the server to be listed. This should include both "live" games that are in play, as well as completed games.
- SHOW,USER,PASS,GAMENAME shows the current gameboard requested as well as the game status (who won, or who's turn comes next)

The server replies with a plain text output that will be printed for the user. For the precise outputs to every case, run "daves_compiled_ttt_server_example", provided in the zip file. I compiled that on mimi, so it should work there and on other compatible machines, but may fail elsewhere.

Example Runs:

Question #2 – Playing on the Web (30 marks)

For this portion, you must replace the terminal-based client with a dynamic web page. We have given you an html entry page, but you must complete the CGI script from scratch (or by careful copy/pasting) that allows your website to connect to your game server and play.

Ensure that every subsequent page has the same input form elements as the provided html, and that every value the user enters for one command is auto-filled in the next generated page.

Print the server's response below the form but ensure to transform the text so that it displays in a more readable fashion on the browser. In particular:

- Replace newlines with "<br>"
- Replace spaces with " "
- Replace dashes with "–"

To test your GCI:

- Place the provided ttt.html in the public_html directory in your home folder on mimi (any other SOCS computer in 3rd floor Trottier works as well).
- Create a "cgi-bin" directory within the public_html folder
- Compile your ttt_cgi.c file with the command "$ gcc ttt_cgi.c ~/public_html/cgi-bin/ttt.cgi"
- Run your ttt_server somewhere and take note of both the address and port.
  - If you run it at mimi, the address will be 132.206.51.22.
  - If you try to run it elsewhere use the ifconfig command to discover the relevant address.
- Point your browser to your page on the web by typing into the address bar
  www.cs.mcgill.ca/~<your_username>/ttt.html
- Make sure you can play the game correctly.

Hints:

- Apart from the CGI specifics, like reading the query string, you can re-use a lot of networking functionality from the ttt_client.c provided file. Copy pasting from that source is completely allowed and encouraged (you just have to be smart about exactly what to take) and what needs to be adapted for the CGI context.
- Because your CGI will be run by the web server rather than directly by you on the terminal, debugging can be even more difficult. I strongly recommend that you come up with a very good plan before starting any code only implement small pieces at a time. For every 5-10 lines plan to print some debug result to the web output to confirm things are working as you plan.
- If you have trouble connecting from home to mimi, try to connect using McGill's VPN:
  http://kb.mcgill.ca/kb/?ArticleId=1212&source=article&c=12&cid=2#tab:homeTab:crumb:8:artId:1212:src:article

Example Runs:

A finished game played between two users in browser via CGI:	A browser user in progress of fighting a terminal user: