Assignment 5

Assignment 5: Client/Server TicTacToe

Background

You will create a tictactoe client/server application that is limited to run only when both client and server are on the same host. In effect, you will write two programs: a tictactoe server and a tictactoe client. The reason that the programs are required to run on the same host is to make it a bit easier; rather than using sockets to communicate, the programs will use *named pipes*.

The client program is what a user invokes to play the game. The client program uses the NCurses library to create the tictactoe board and update its state at every turn. Each square in the tictactoe board is represented by a (row,column) pair in the range from (1,1) to (3,3), where (1,1) is the upper-left corner and (3,3) is the lower right corner, as is shown below:

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

When the user starts the game, she should see a screen something like the following:

```
                T * I * C * T * A * C * T * O * E


                  Welcome to the tictactoe game!
               To play, move the cursor to an unplayed
                   position and press the spacebar.
                   You always go first and play x's.
                    I play o's.  Good luck!

    Enter any key to start the game:
```

When the user presses a key, the screen should change to the game screen, which is shown on the next page. The user uses the arrow keys to move the cursor, and then presses spacebar to play the given cell of the board. The user should be prevented from playing a cell that has already been played. Pressing the spacebar outside of the board should have no effect. Pressing 'q' should terminate the game. Pressing any other key should have no effect.

During game play, the client sends the user's move to the server and displays the move on the screen as well. The server generates a valid (row,column) pair for its move and sends it back to the client, which updates the board. This goes back and forth until the game is over. A tictactoe game ends when a player wins or there are no more empty cells.

Both the client and the server will need to determine when the game is over and whether the user won, lost, or tied, because the user might enter a move that wins, in which case the server should not move again, or the server might make a winning move, in which case the user cannot enter another move. The server should have logic to choose winning moves and moves that block the user from winning. It should be able to tell whether the user's move is valid, and if it is not valid, should send a message to the client that the move is invalid and that a new move is requested. This is just added redundancy in case a client is not validating its moves. The server should always generate valid moves.

The server should be able to play multiple clients at a time. This means that each time a client connects to it, it should fork a new process to play against that client.

```
                    T * I * C * T * A * C * T * O * E



                    __________ Current Game __________


                               |     |
                          -----------
                               |     |
                          -----------
                               |     |


  It's your turn. Make a move or type q to quit.
```

Program Requirements

You must use a named pipe as the means of initial communication between each client and the server. The server needs to create this pipe and put itself in the background, like a daemon. The pipe should be in `/tmp` so that it is publicly writable. When a tictactoe client starts up, it creates a pair of named pipes and sends a message to the server that it wants to play a game, along with the names of the pipes it just created. These names must be unique to this client. (It uses two pipes so that one can be for the server to write to and the other, for the client to write to.) The server forks a process to play with the child and this child process plays the game with the child, using the two pipes for exchanging moves.

The client process handles all of the screen activities. To avoid conflicts with other people's solutions, the pipe created by the server must have a name related to your user name, so that multiple tictactoe servers do not share a pipe. Therefore, the name of the public pipe you create should be `/tmp/TICTACTOE_<username>`.

You are free to make whatever decisions need to be made regarding the look and feel of the game.

Submitting the Assignment

You are to create a zip file containing all of your source code and put that zip file in the directory

```
/data/biocs/b/student.accounts/cs493.66/projects/project5
```

naming it `username_hwk5.zip`. Give it permissions 600 so that only you have access to it. Whether you have a single file or multiple files, you are to create a directory named `username_hwk5`, putting all files into it and using the command

```
zip -r username_hwk5.zip username_hwk5
```

to create the zip file. Make sure that you create a Makefile and that it creates two executables, one named `username_tttd` and the other, `username_ttt_client`.

The programs must be well-documented and must conform to my programming guidelines for full credit. They must be placed in the directory no later than midnight of the due date.