CSE 4510/5241 HW5
Submit server: course=dc, project=hw5
Due 5pm, Apr 29, 2009

Design and implement a distributed 6x6 tic-tac-toe game:

- 6x6 board
- The first who makes 3 in a row, column, or diagonal wins.
- When a player leaves the game, all his/her positions are removed.

Properties:

1. Multiple hosts
2. Multiple players (let's say at most 4)
3. Players can enter the game at ANY time.
4. Players can leave the game at ANY time.
5. Whenever there is a change to the board, the updated version is displayed for each player.
6. A schedule to determine who makes the next move—at anytime, all the original players move before the newly joined player can make a move.
7. A player can't make a move until all players have received the previous move or the positions are removed for a player who left the game.

Implementation:

1. A player joins the game by starting a client (`ttt`) and leaves by exiting from the client.
2. A server (`tttd`) coordinates all the players/moves concurrently.

The client should be able to take at least two commands at ANY time:

- make a move (print error if it's not the player's turn)
- exit (Control-C is not acceptable to exit the client unless your client can catch it and exit normally).

When someone wins, everyone knows someone wins, no one can make a move, and the game ends. A **third** command on the client is to ask whether the user wants to keep playing in the next game—only available after a game ends, the other two commands can be issued at any time.

The server allows a client to exit at ANY time (not wait till the exiting client's turn). That is, for example, if the client ordering is A, B, C, D, it's currently A's turn, and D wants to exit, D doesn't need to wait for B and C to make their moves before D can exit. The server can let A finishes its move and then cleans up D's moves. When it's B's turn, D should not exist on the board.

The server handles:

- normal client exit (exit message from client) and
- abnormal client exit (lost connection due to network, host, client failures (e.g., control-c)...)

Your implementation involves mulitple threads for concurrency in the server and client. You may not use sleep(timePeriod) or wait(timePeriod) unless it's well justified; if I see *unjustified* suspenison of execution for some predetermined amount of time, I'll deduct points.

CSE 5241 students only
Peer to peer, instead of client-server:

- peer: up to 4 peers
- daemon: (similar to HW2) initial set up for peers, only peer-to-peer communication afterwards

---

What to turn in:

1. Detailed description of the concurrency design for the client and the server and how you implement Properties 3 through 7 (README file).
2. Compilation instructions (preferably makefile)
3. Source code
4. Sample session (`script` on unix)