CS69003: Computing Systems Lab I
Autumn 2008

Assignment 8

Design of a 2-player network game

Due: November 12, 2008

In this assignment, two players connect to a server in order to play the game of tic-tac-toe interactively. A $k \times k$ board is stored in a shared memory segment in the server. Moves alternate between two remote clients until a player wins or the game ends in a draw. The winning position is indicated by a row or a column or a major diagonal being filled up with the same symbol (X or O). All connections are made by TCP. The server processes synchronize by using semaphores.

The server waits for two clients to request for a new game. Upon the request of each player (client), a new process in spawned to handle the client. When the game is over, the server resumes waiting for two more players to connect for starting a new game.

Suppose that C_1 and C_2 are the two client processes that are allowed access to play the game. The corresponding server processes are called S_1 and S_2. For simplicity, you may assume that the client process which connects earlier (say, C_1) plays X and makes the first move.

The board is initialized by the server in the shared memory segment and is accessible to both the spawned processes S_1 and S_2. These processes synchronize by using two semaphores σ_1 and σ_2 both initialized to zero. Initially, S_1 waits on σ_1. When C_2 connects, the parent process forks S_2 and signals on σ_1 to start S_1.

When it is the turn of C_1 to make the move, S_1 first sends the current board to C_1 and then waits for a valid move from C_1. The other process S_2 waits on the semaphore σ_2. When C_1 supplies a move, S_1 updates the board in the shared memory, signals the semaphore σ_2 and then waits on the semaphore σ_1 again. The process S_2 now wakes up (since σ_2 is signaled) and serves C_2 in an analogous fashion.

Before sending any board position to the client, a server process always checks whether the game is over. If so, it terminates after sending the board position. A client process, upon reception of a board position, checks whether the game is over. If so, it prints an appropriate message and exits.

The parent server process waits for both the child processes S_1 and S_2 to terminate and then waits for a new game.

Submit two C source files: `tttserver.c` and `tttclient.c`.