cab403 assignment

Process Management and Distributed Computing

Due Date: 28th October 2018  
(APPROVED ASSIGNMENT EXTENSION)

Group Members: Davina Tan (N9741127)  
Quintus Cardozo (N9703578)

# Statement of Completeness

**Task 1: Client-Server Computing**

Completed

**Task 2: Multithreaded Programming & Process Synchronisation**

Completed

**Task 3: Thread Pool**

Incomplete

# Team Information

Davina Tan – N9741127  
Quintus Cardozo – N9703578

# Team Contribution

Davina Tan – 45%  
Quintus Cardozo – 55%

# Data Structures

## Representing the Playfield

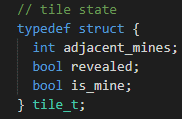
  
*Figure 1 – Tile state data structure*

Figure one displays a data structure used to store a tile’s state. The integer variable adjacent\_mines stores the number of adjacent mines there is of the tile. The two boolean variables revealed and is\_mine store the states of the tile. If the tile is revealed, revealed will be true, and if the tile is a mine, is\_mine will be true.

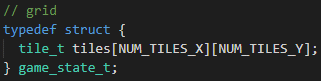
  
*Figure 2 – Game state data structure*

Figure two displays a data structure which stores the game state. The data structure holds the playfield of the game (a 9x9 grid of tiles), indicating there are tiles in a 2D array of [NUM\_TILES\_X] and [NUM\_TILES\_Y]. The number of tiles for x and y are both defined (see Figure 3) as 9, as stated in the specification.

  
*Figure 3 – Number of tiles defined*

## Leaderboard Implementation

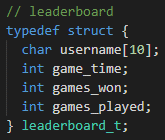
  
*Figure 4 – Leaderboard data structure*

Figure four displays the data structure which contains the main variables which compromise the leaderboard. username is a char array which holds the player’s username. The integer variables: game\_time, games\_won, games\_played, store the time of the completed game, the amount of games the user won and the amount of games the user played.

# Task 2

## Critical-Section Problem Management

The Critical-Section problem involves multiple concurrent connections to the leaderboard, each of which may read or write to the leaderboard. This problem is managed using mutex locks, which enables the writer the ability to write to the leaderboard and disables the ability of other connections to write to the leaderboard.  
pthread\_mutex\_lock() and pthread\_mutex\_unlock() are both used to mutex lock and unlock multiple sections throughout the code in the server where the code contains information of the leaderboard.

# Task 3

## Thread Pool Creation and Management

As specified in the assignment brief, the server will allow up to 10 clients to use the system at the same time. In server.c, the constant BACKLOG is defined as 10 and a global array of threads is created as pthread\_t THREAD\_IDS[BACKLOG]. Within the main function, a for loop is created which loops through variable i and creates a new thread with pthread\_create() and passes it to the function requests\_loop().

requests\_loop()handles each individual thread in a while loop and gets a new request from the function get\_request(). When a connection is closed, the thread is deleted using pthread\_exit().

# Compile and Run Instructions

1. To compile, open a terminal and type ‘make’ in the directory, this will build the server and client
2. Type ‘./server (portnumber)’ to run the server. If no port number has been specified, the server will run on a default port of 12345.
3. Open a separate terminal
4. Type ‘./client localhost (portnumber)’to run the client, using the same portnumber as the server
5. The game will begin immediately on the client terminal.