CAB403 Alexander Brimblecombe

(Individual)

n10009833

Major Assignment

Statement of completeness / Report

Statement of completeness

All of the features of the assignment have been implemented, except for network byte order used for transmitting multi-byte data types. This was due to the fact that the program was already largely implemented and I did not have time to make appropriate alterations as the transmitted data types vary in size by a large margin.

The signal handling for closing down the server is also something which can be improved in my implementation. I opted for cancelling the threads rather than joining them, as threads are blocked when waiting on a client with the recv command. All of the resources used by the threads are either globally accessible or defined on the stack so there should be no memory leaks in this implementation. An improvement to this situation would be creating a socket receiivng thread on the client’s side dedicated to filling a buffer of communications from the server. This buffer could then be tested for a quit command which could in turn allow to the client to also send a quit command.

TEAM AND CONTRIBUTION: This assignment was completed individually.

Data Structures used for Game

For the implementation of the game, I opted for making a dedicated header file to declare the structures for the game types. These types are shared by the client and the server which means that a full Tile struct cant be sent over a socket, and interpreted on the client’s end.

typedef struct {

u\_int8\_t adjacent\_mines;

u\_int8\_t is\_revealed;

u\_int8\_t is\_bomb;

u\_int8\_t flagged;

u\_int8\_t x;

u\_int8\_t y;

}Tile;

*/\* Struct for game type \*/*

typedef struct {

bool game\_over;

Tile tiles[NUM\_TILES\_X][NUM\_TILES\_Y];

int remaining\_mines;

int start\_time;

}Game;

A variation from the assignment specification is the fact the the x and y position of each tile is stored in the struct. This is to allow a single Tile to contain all the information that the client needs to know when it receiving from the server. (This wasn’t necesarry when one tile was sent, however in my implementation it is necessary when tiles are sent recursively as it allows the client to successively know the position of each surrounding tile when there are 0 surrounding mines).

Compile Instructions

1. Navigate to folder with make file - run ‘make’ to compile server\_program and client\_program

2. IMPORTANT: Change directory to ‘bin’ folder. (If not, server will not find authentication.txt)

2. Run 'server\_program' inside ‘bin’ with either a argument for port number - or no argument (default port: 12345) i.e. *.*/server\_program 12345

3. run instances of 'client\_program' inside /bin with IP address and port number as arguments.

i.e. ./client\_program 127.0.0.1 12345

\*\*NOTE: if the server program is run from outside /bin directory - the authentication.txt file will not be found\*\*