Unix and C Programming Assignment Report

Alec Maughan 19513869

Contents

Files

tictactoe.c

This is the ‘main’ file of the assignment, it contains the main method, and handles all user interface (apart from the in-game UI). It stores the fundamental variables of the program, like m,n,k and the log entry linked list in the main method. The file includes the implementation of displaying and editing settings, displaying logs to the terminal, and saving logs to file with the help of file_io.c. The main method also uses file_io.c to read in the settings file given as a command line parameter.

file_io.c

This file is responsible for any file IO needed in the assignment. It handles reading the settings file and saving logs to file. Although it finds errors and produces detailed error messages, it’s responsibility is not UI so it passes error messages back to tictactoe.c for it to display.

game.c

This file is the implementation of the tic tac toe game itself. The ‘main’ function is playTTT, which stores the 2D malloc array of the board and uses the many helper functions of game.c to handle the many aspects of the game. The file is responsible for a bit of UI, in displaying the board, and having the user take a turn. The header file contains the typedef of the LogEntry struct, and the .c file contains the function to free this struct for use in freeLinkedList().

linked_list.c

Implementation of a generic double ended, doubly-linked list. This is used to store the log entries of the game. Uses typedef to define structs for the list nodes and the list itself, contains print and free functions that take in function pointers to deal with generic values.

Logging

Logs are stored in a generic linked list that is stored in the main method of the program. The log itself is a struct, defined in game.h. It contains the game number, turn number, player (as a char), the x position and the y position of each turn. When a player takes a turn in the game, we store all required info on the turn and call addLog(), which takes all this in, constructs a new log entry, and uses the linked list function insertLast() to insert it into the list.

When displaying the logs (To terminal and to file), we need to iterate through the linked list. A complication I came across is that if we do this through the standard linked list functions, such as removeFirst(), we would be removing the entries every time we display it, and would lose all information. Instead we take the head of the list, typecast it’s value to a seperate value, take the info from this, go to the ‘next’ of the current node, and repeat. We also must print a new “GAME: X” header for every new game, so before printing the other info, we check if the game number of the current log is different from that of the previous log, if so, we print the new header.

Demonstration & Usage

Settings file

The settings file can be any file extension and must contain values for M, N and K. The game board has a width of M and a height of N. A player needs to match K tiles in a row to win. In the file a setting is declared with `M=<value>`, this is case insensitive. The order the settings are in does not matter but they must be separated by new lines.

Compilation

If the program has been previously compiled, the files can be cleaned up with `make clean`. To compile the program normally, use `make`. To run it in editor mode, which allows the changing of the game settings at runtime, use `make EDITOR=1`. To run in secret mode, which disables the ability to save logs to file, use `make SECRET=1`.

Running the program

When running the program, you must provide the name of the settings file as a command-line argument. To do this, use `./tictactoe <filename>`. An error will be displayed if there is a problem with the settings file and the program will close.

Using the main menu

When the program is run, the main menu is shown, listing all available options. Each option is lead by a number in square brackets (eg. `[1]`). This is the number you must enter to select that menu option.

Playing the game

When the game is running, each turn a graphical representation of the board will be printed to the screen. The board is displayed as a grid, with empty tiles empty and filled tiles with a coloured ‘X’ or ‘O’. Player O goes first. To take a turn, type the coordinates of the tile you wish to place your symbol, comma separated. The coordinates of each square are shown on the sides of the board for your reference. For example, to play at coordinates (1,2), enter `1,2`. If for any reason the selection was invalid, you will be prompted to choose again. Once the game is over, either from a player winning or a stalemate from the board being full, you are sent back to the main menu.

The logfile

When you choose to save your logs to file, the program will display the filename in which they are saved. This filename includes the settings of the game and the date and time. The file shows the settings of the game (if compiled in editor mode, the last used settings) and the logs for each game. Each game is assigned a number, and each log has a turn number for that game, a player and x and y coordinates. This exact information can also be displayed to the terminal with the view logs option in the main menu.