Unix & C Programming Assignment

Tic-tac-toe Game

Reece Jones

[0] - Overview

The goal of this assignment is to produce a basic, but functional, M-N-K tic-tac-toe game, as a command-line game. The game supports user-provided settings for board size and win condition, as well as game action logging. The game is written in C89 and builds using Make.

[1] – Usage

[1.1] – Command line

The command-line usage of the application is as follows:

tictactoe <settings\_file\_path>

where tictactoe is the path to the application’s executable (typically ./tictactoe from the same directory as the application), and <settings\_file\_path> is the path to the containing the settings to use.

[1.2] – Settings file

The settings file is an ASCII plaintext file, with one option specified per line. The syntax of a line is as follows:

<option>=<value>

where <option> is a single character (case-insensitive) specifying the option to set, and <value> is the desired value of the option. Whitespace before and after <option> and <value> is ignored. Lines consisting of only whitespace are also ignored.

The following table outlines the allowed options:

|  |  |  |  |
| --- | --- | --- | --- |
| Option character | Description | Value type | Required? |
| M | Width of the game board | Integer >0 | Yes |
| N | Height of the game board | Integer >0 | Yes |
| K | Consecutive tiles to win | Integer >0 | Yes |

The application loads and validates the settings file immediately upon startup. If there are any invalid values, error message(s) will be shown and the application will abort.

Some values or combinations of values will still allow the application to run, but may affect usability. These will produce warnings.

K=1 – allowed, however the game will always be won on the first move.

K > M and K > N – also allowed, however the game cannot be won and will always end in a draw.

[1.3] – Main menu

After loading the settings, the application will enter the main menu, which contains the following options:

Start a new game - starts a new game of tic-tac-toe using the current settings.

Edit settings (*editor mode only1*) - allows modification of the application settings.

View current settings – displays the current application settings.

View game logs – displays the game logs from this session.

Save game logs to file (*not in secret mode1*) – exports this session’s game logs and settings to file.

Exit application: terminates the application.

*1See section* 2.1 – Building the application *for specification of the editor and secret modes.*

Each menu option is identified by an integer, which the user enters in order to make a selection.

[1.4] – Gameplay

After choosing the “Start a new game” option from the main menu, the application runs a standard game of M-N-K tic-tac-toe. Player “X” always moves first. The player will be prompted for a coordinate in the format

<x>,<y>

where <x> is the column number and <y> is the row number of the tile to be marked for that turn. 0,0 is the top left corner of the board.

---------------------  
| X | | | | |  
---------------------  
| | O | | | |  
---------------------  
| | | | | |  
---------------------  
| | | | X | |  
---------------------

The following is an example rendering of a 5x4 board, with 3 tiles already marked:

Once a player gets *K* (the ‘K’ option from the settings file) tiles in a row (horizontally, vertically, or diagonally), they have won and the game ends. The application then returns to the main menu.

[1.5] Logging

TODO

[2] – Build system

[2.1] – Building the application

The application is built with Make, using the makefile in the project’s root directory.

The Make target tictactoe creates the application executable. It can be invoked with

make tictactoe

which will produce the executable tictactoe in the project’s root directory.

There are two build options available, which modify the behaviour of the application:

EDITOR\_MODE – when set, enables the “Edit settings” option in the main menu.

SECRET\_MODE – when set, disables the “Save game logs to file” option in the main menu.

These are specified by assigning them a value in the Make command, for example:

make tictactoe SECRET\_MODE=1

[2.2] Building the unit tests

Alongside the main application, there is a separate executable available to build that unit tests code for the main application. It is also built with Make, under the tictactoe\_test target:

make tictactoe\_test

which will produce the executable tictactoe\_test in the project’s root directory.

Usage of the previously mentioned application build options will not affect this build.

Please see section TODO for more information on the unit tests.

[3] – Project structure

[3.1] – Root directory

* example\_invalid\_settings\_1.txt – an example invalid settings file showing incorrect formatting and invalid options.
* example\_invalid\_settings\_2.txt – an example invalid settings file showing invalid option values.
* example\_valid\_settings\_1.txt – an example valid settings file.
* example\_valid\_settings\_2.txt – an example valid settings file showing a configuration that will cause a warning.
* Makefile – the Makefile used by Make to build the application and unit tests.
* Report.pdf – this report.
* tictactoe – the main application executable, present when built.
* tictactoe\_test – the unit test executable, present when built.

[3.2] - src/main directory

Contains the C source code of the main application. Both header and source files are stored here.

* board.c/.h – sources and header files for the “board” module. See section TODO.
* common.c/.h – source and header files for the “common” module. See section TODO.
* interface.c/.h – source and header files for the “interface” module. See section TODO.
* linked\_list.c/.h – source and header files for the “linked list” module. See section TODO.
* log.c/.h – source and header files for the “logging” module. See section TODO.
* main.c – source file for the main module. See section TODO.
* settings.c/.h – source and header files for the “settings” module. See section TODO.

[3.3] src/tests directory

Contains the C source code of the unit tests. Both header and source files are stored here.

* board\_test.c/.h – source and header files for the “board” module unit tests. See section TODO.
* common.c/.h – source and header files for the “test common” module. See section TODO.
* common\_test.c/.h – source and header files for the “common” module unit tests. See section TODO.
* linked\_list\_test.c/.h – source and header files for the “linked list” module unit tests. See section TODO.
* log\_test.c/.h – source and header files for the “logging” module unit tests. See section TODO.
* main.c – source file for the unit test main module. See section TODO.
* settings\_test.c/.h – source and header files for the “settings” module unit tests. See section TODO.

[3.4] test\_data/readUntil directory

Contains test data files used in the “common” module unit tests for the readUntil() function. See section TODO.

[3.4] test\_data/settings directory

Contains test data files used in the “settings” module unit tests for the readSettings() function. See section TODO.

[3.5] obj/main directory

Stores the main application object files when they are built by Make. All files in here are temporary and can safely be deleted.

[3.5] obj/test directory

Stores the unit test object files (just the test code itself, not the units being tested) when they are built by Make. All files in here are temporary and can safely be deleted.