Exercises — Tic-Tac-Toe

version #7be580532266ed398481e31366afcc24b1950c2a

The way is lit. The path is clear.
We require only the strength to follow it.

Assistants C/Unix 2023 <assistants@tickets.assistants.epita.fr>

Contents

*https://intra.assistants.epita.fr

File Tree

```
tic_tac/
├─ Makefile  (to submit)
├─ empty.tt
├─ filled.tt
├─ result_tic_tac.c  (to submit)
├─ tic_tac.c  (to submit)
└─ tic_tac.h
```

Makefile

- library: Produce the libtictac.a archive
- clean: Delete everything produced by make

Authorized functions : You are only allowed to use the following functions

- fclose(3)
- fopen(3)
- fputs(3)
- fputc(3)
- fgetc(3)
- freopen(3)

Authorized headers : You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- stddef.h

Compilation : Your code must compile with the following flags

- -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function : None

1 Goal

Given a file format representing a Tic-Tac-Toe grid, you have to implement multiple functions to enable players to play a Tic-Tac-Toe and compute the result (which player won the Tic-Tac-Toe game). The Tic-Tac-Toe file format is the following: The only characters allowed in a Tic-Tac-Toe grid are : 'X', 'O', '|' and ' '. All the lines are followed by a line feed. There are exactly 3 lines of 5 characters each. Each value of the Tic-Tac-Toe ('X', 'O' or ' ') are separated by a '|'.

For example:

```
X|O|O
O|X|O
O|X|X
```

Two examples Tic-Tac-Toe grids are provided on the intranet.

2 Functions

2.1 init_tic_tac

The first function you have to implement is `init_tic_tac`.

```c
int init_tic_tac(const char *file);
```

This function will take a file name in parameter and create an empty Tic-Tac-Toe grid. An empty grid is composed of alternating spaces and '|' characters. An example of an empty grid should be in the provided files on the intranet. If any error occurs, returns `-1`, otherwise returns `0`.

2.2 read_tic_tac

```c
int read_tic_tac(char buffer[3][3], FILE *stream);
```

The function takes a buffer corresponding to the Tic-Tac-Toe grid and fills it up. `buffer[1][2]` corresponds to the second line third column. The function returns `0` in case of success and `-1` if the Tic-Tac-Toe format is not valid.

2.3 write_tic_tac

In a similar way, `write_tic_tac` takes a buffer of allowed characters and writes it to the file given in parameter. The `buffer[0][0]` case correspond to the top left of the grid. If any error occurs, returns `-1`, otherwise returns `0`.

```c
int write_tic_tac(char buffer[3][3], FILE *stream);
```

2.4 fill_tic_tac

`fill_tic_tac` fills an element of the Tic-Tac-Toe grid in the given file.

```c
int fill_tic_tac(const char *file, size_t line, size_t col, char value);
```

Here are the expected return codes:

- **0** : Everything is OK.
- **-1** : A parameter is not adequate.
- **-2** : The grid in the given file does not have the Tic-Tac-Toe format.
- **-3** : The element to be filled is not empty.

2.5 result_tic_tac

The last function to write is the `result_tic_tac` that will take a file with a filled Tic-Tac-Toe grid and output which player won the game on standard output.

```c
int result_tic_tac(const char *file);
```

The three possible outputs are:

- "Winner: X"
- "Winner: O"
- "Winner: none"

followed by a newline. If any error occurs, for example if a player cheated (played more times than he should) or if both player win, returns `-1`, otherwise returns `0`.

The way is lit. The path is clear. We require only the strength to follow it.