Tui

The terminal user interface is not a necessary component of the system but at the same time is the most used by the user. Our goal was to offer a good user experience despite the terminal environment. The first problem was about the low variety of libraries. We decided to use termios, a very low level library that allows to change the terminal in raw mode.

User interface

The user interface is divided in 7 specific blocks, each of them has different refresh level in order to improve the usability. A little brief:

- **User input**: user input block takes file or directory path from the user.
- **Quick help**: quick help block displays all possible commands for user input block.
- **Tree input**: tree input block allows to move in the file system.
- **Current directory**: current directory block displays the current directory.
- **File system**: file system block displays files and directories inserted in user input block of the current directory.
- **Percentages**: percentages block displays the percentages of some groups of char over the total.
- **Help message/Table presentation**: this block displays the help message or the table that contains the count

Implementation

In order to draw everything on the screen we take the current width and height of the terminal and we create a chars grid width*height. Then there is a thread that:

1. cleans the screen
2. prints all the grid
3. sleeps for a specific amount of time
4. repeat

A huge difference was made by the raw terminal mode and by the ansi escape code. Raw terminal mode allows us to take keyboard events and with them we could write what the user types directly on the grid. Ansi escape code instead allows us to clear the screen, move the cursor and print some char in different colors.

All TUI is focused on the concept of moving inside the file system and toggling files. When a file is toggled/untoggled a request is sent to the analyzer through reporter in order to get the table that represents the analysis of all the toggled files.

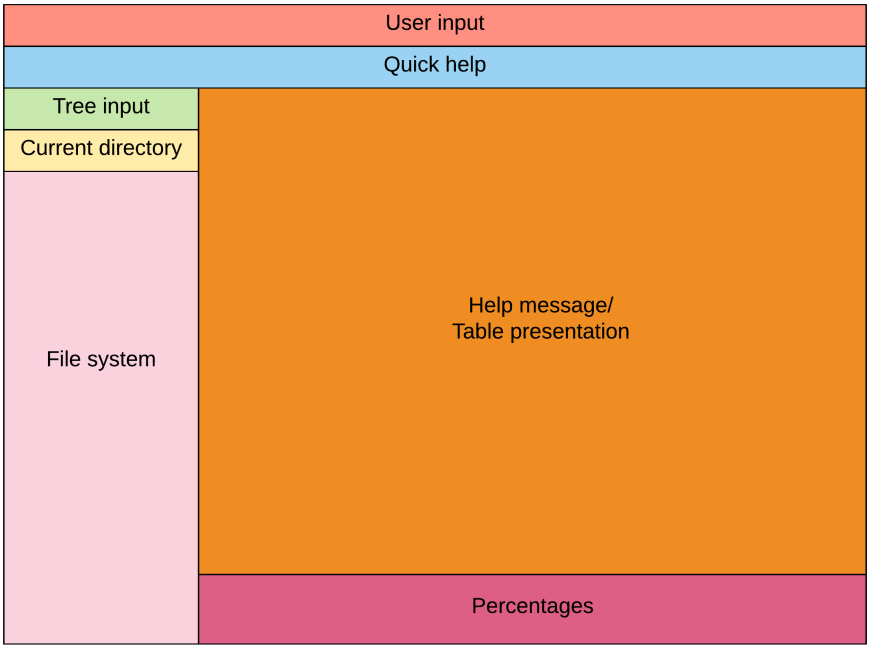

Figure 1: User interface

User input

User input block is populated after a key is pressed. When line feed is pressed the text is parsed. If the processed text is an input it is passed to reporter that sends the input to the analyzer. Instead if the input is a command some changes are made. The commands are:

- **quit**: quits the application
- **help**: displays the help message
- **n**: changes the worker's amount
- **m**: changes the manager's amount
- **tree**: switches in tree mode and move cursor in tree input block
- **maiuscole**: highlights only upper case letters on the table
- **minuscole**: highlights only lower case letters on the table
- **punteg.**: highlights only punctuation letters on the table
- **cifre**: highlights only digits on the table
- **tutto**: highlights all ASCII letters on the table

The path written by the user can be relative or absolute and supports . (current directory reference) and .. (previous directory reference).

Tree input

Tree input block is populated after a key is pressed only if the tree command was typed on the user input. The tree mode area has four different types of input:

- file: when a file is written it becomes toggled if previously wasn't, otherwise becomes untoggled
- directory: when a directory is written the program change the working directory and file system block change according to them
- commands: there are two commands
  - . dot toggles all untoggled files in the current directory (and vice versa)
  - .. double dot moves backward in the file system
- down/up arrows: if the number of children of the current directory is greater than the lines of the file system block, with the down and up arrow the user can easily scroll between them

File system

The file system displayed in TUI's dedicated block shows only the files that the user passed as input to the system and are really stored on the disk. For this specific reason, the user can toggle only the analyzed files.

Considerations

Developing this component was very challenging and due the small amount of time given for this project and our inexperience with low level system call some

parts of the code may be a little hard coded. Also our implementation may not be the most effective and the easiest to write but suits perfectly our needs.

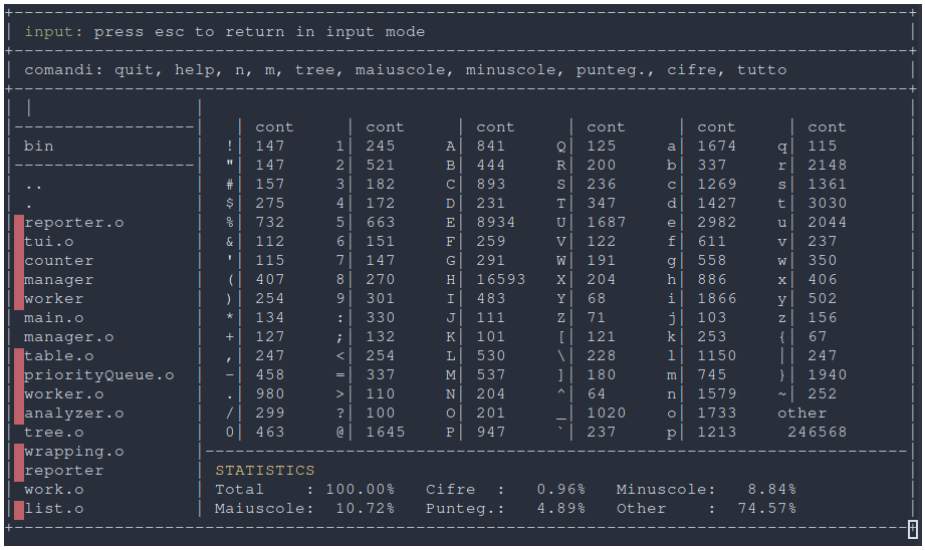

Figure 2: Screenshot