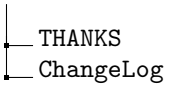
Making of Hangman2

Rémi Nicole

Chapitre 1

Directory tree of the project

- [root]
  - configure.ac ..... Autoconf main configration file
  - Makefile.am
  - src
    - Makefile.am
    - hangman2 ..... Executable
    - hangman2.c ..... Main source file
    - utils
      - dict.c
      - dict.h
      - words.c
      - words.h
      - pics.c
      - pics.h
      - inc.h ..... Common `#include` and `#define`
    - pics ..... Default pictures directory
      - 1.txt
      - 2.txt
      - 3.txt
      - 4.txt
      - 5.txt
      - 6.txt
      - 7.txt
      - 8.txt
      - 9.txt
      - 10.txt
  - doc ..... Documentation directory
    - Makefile.am
    - report.tex
    - report.pdf
  - NEWS
  - AUTHORS
  - COPYING
  - INSTALL
  - LICENSE
  - README


THANKS
ChangeLog

Chapitre 2

The Game

2.1 Main game

2.1.1 Picking a random word

In order to pick a random word in a file consisting of a single word per line, the file is first scanned to count the number of files (n), then a random number k is picked in the $[\![1; n]\!]$ interval. After that, the k^{th} word of the file is picked and returned.

2.1.2 Game word and hidden word

Two sting variables are instantiated for the random word : one which is the random word (game word) and another which is originally composed of underscores (`_`) only (hidden word). The second variable will be the text displayed to the player and will serve to check if the player won : if this string does not contains any underscores, then the player won. Each time the player inputs a character, the inputted character is searched through the game word and if found, any occurrences of this character will replace the corresponding underscore(s) in the hidden word. If not found, the player will lose one attempt.

2.1.3 Tried letters

The string typed variable named `triedLetters` will contains in order the letter already tried (with or without success) and each time the user will input a character, the program will check that the letter was not already tried, meaning that the letter is not contained in the `triedLetters` variable. If it is contained, then the error message `"You already tried it!"` is displayed.

2.1.4 Endgame

For the end game, two parameters are taken into account : the hidden word variable and the remaining attempts variable. If the hidden word is no more hidden, the player wins. Else if the remaining attempts is zero, then the player looses.

2.2 Details

2.2.1 Command line arguments

Command line arguments can be passed to the program. The `-h` and `--help` options will make the program display an help for the program command line usage and exit. The `-v` or `--version` options will display the version number of the program and exit. The `-d` or `--display` option takes one argument which is the

dictionary file containing the words to pick from. The header `unistd.h` is used in order to be able to make use of the `getopt` facility.

2.2.2 Console clear

For each lines printed by the program each turn, the escape sequence code `"\033[A\033[2K"` is printed which will make the terminal cursor go one line up and any printed text after this escape sequence code will be printed over these lines.

2.2.3 Invalid characters

Each time the player inputs a character, the character inputted is checked through the function `validLetter` and display `"Invalid character"` if the character inputted is not a letter. If the letter inputted is a `'\n'`, the console curser goes up one line more.

2.2.4 `malloc` aversion

In order to prevent a `malloc` from being used in the function `searchWord` of the `utils/dict.c` file, a `static` string is returned and reset each time the function is called by putting the `'\0'` at the first character.