Elementary programming

get_next_line

Astek in charge astek_resp@epitech.eu

Abstract: This document is the subject of the get_next_line Elementary Programing project

Contents

Chapter I

Instructions

- Your code shall comply with the norm
- Turn-in:
  `Turn-in directory : CPE_2013_getnextline`

`Pay attention to the permission of your folders and files`

Chapter II

Subject

- The goal of this project is to write a function that returns a line read from a file descriptor.
- You must define one macro in your `get_next_line.h` file
    - The macro indicates the number of characters read at each call of `read`
- You must use one (several) static variable(s) to save the characters that were read but not sent.
- You shall return two files (`get_next_line.c` and `get_next_line.h`)
- The turn-in directory shall not contain any `Makefile` nor `main` function.
- The macros, and the `get_next_line` prototype, must be located in a `get_next_line.h` file.
- `get_next_line` shall return its result without the `\n`. If there is no line left to be returned or if there is an error during reading, the function returns `NULL`.
- The function shall be prototyped as follows:

```
1   char *get_next_line(const int fd);
```

- Turn-in :
`get_next_line.h` et `get_next_line.c`
- Example :

```
1   /*
2   ** main.c for get_ next_line_main_test in
3   **
4   ** Made by tek assistant
5   ** Login <astek@epitech.net>
6   **
7   ** Started on Mon Nov 5 14:59:09 2001 tek assistant
8   ** Last update Mon Nov 5 14:59:09 2001 tek assistant
9   */
10
11  #include "my.h"
12  #include "get_next_line.h"
13
14  int main()
15  {
16    char *s;
17
18    while ((s = get_next_line(0)))
19      {
20        my_putstr(s);
21        my_putchar('\n');
22        free(s);
23      }
24    return (0);
25  }
```

Chapter III

Authorized functions

- read
- malloc
- free