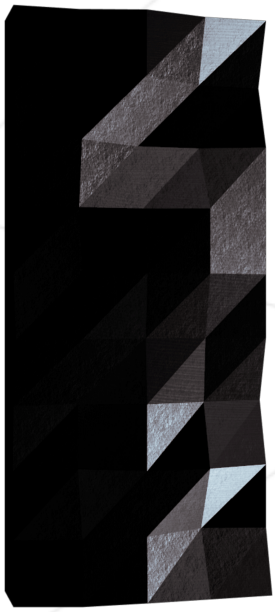

b_libft

Your first own library

Gaetan gaetan@42.us.org

Summary: The aim of this project is to code a C library regrouping usual functions that you'll be allowed to use in all your other projects.

Contents

Chapter I

Foreword

This first project marks the beginning of your training to become de software engineer. To accompany you during this project, here is a list of oustanding music groups. It's highly probable that you won't like any of those. This will mean that you have poor music taste. I'm sure that you have some other qualities such as being able to hold your breath for more than 3 minutes or maybe you know by heart the names of the 206 United Nations' signatory states. The groups aren't listed in any particular order and the list does not need to be exhaustive. Click on the links to find out more.

- Between The Buried And Me
- Between The Buried And Me, c'est bon, mangez-en
- Tesseract
- Chimp Spanner
- Emancipator
- Cynic
- Kalisia
- O.S.I
- Dream Theater
- Pain Of Salvation
- Crucified Barbara

Chapter II

Introduction

The libft project builds on the concepts you learned during Day-06 of the bootcamp ie code a library of useful functions that you will be allowed to reuse in most of your C projects this year. This will save you a lot of precious time. The following assignments will have you write lines of code you already wrote during the bootcamp. See the libft project as a Bootcamp reminder and use it wisely to assess your level and progress.

Chapter III

Goals

C programming can be very tedious when one doesn't have access to those highly useful standard functions. This project makes you to take the time to re-write those functions, understand them, and learn to use them. This library will help you for all your future C projects. Through this project, we also give you the opportunity to expand the list of functions with your own. Take the time to expand your libft throughout the year.

Chapter IV

General instructions

- You must create the following functions in the order you believe makes most sense. We encourage you to use the functions you have already coded to write the next ones. The difficulty level does not increase by assignment and the project has not been structured in any specific way. It is similar to a video game, where you can complete quests in the order of your choosing and use the loot from the previous quests to solve the next ones.

- Your project must be written in accordance with the Norm.

- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the defence.

- All heap allocated memory space must be properly freed when necessary.

- You must submit a file named author containing your username followed by a newline at the root of your repository.

```
$>cat -e author
xlogin$
```

- You must submit a C file for each function you create, as well as a libft.h file, which will contain all the necessary prototypes as well as macros and typedefs you might need. All those files must be at the root of your repository.

- You must submit a `Makefile` which will compile your source files to a static library libft.a.

- Your `Makefile` must at least contain the rules $(NAME), all, clean, fclean et re in the order that you will see fit.

- Your `Makefile` must compile your work with the flags -Wall, -Wextra and -Werror.

- Only the following libc functions are allowed : `malloc(3)`, `free(3)` and `write(2)`, and their usage is restricted. See below.

- You must include the necessary include system files to use one or more of the three authorized functions in your .c files. The only additional system include file you are allowed to use is string.h to have access to the constant NULL and to the type size_t. Everything else if forbidden.

- We encourage you to create test programs for your library even though this work won't have to be submitted and won't be graded. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.

Chapter V

Mandatory part

V.0.1 Technical considerations

- Your `libft.h` file can contain macros and typedefs if needed.
- A string must ALWAYS end with a '\0', even if it is not included in the function's description, unless explicitly stated otherwise.
- It is forbidden to use global variables.
- If you need sub-functions to write a complex function, you must define these sub-functions as static as stipulated in the Norm.

Check out this link to find out more about static functions:
http://codingfreak.blogspot.com/2010/06/static-functions-in-c.html

- You must pay attention to your types and wisely use the casts when needed, especially when a void* type is involved. Generally speaking, avoid implicit casts. Example:

```
char *str;

str = malloc(42 * sizeof(*str));          /* Wrong! Malloc returns a void * (implicit cast) */
str = (char *) malloc(42 * sizeof(*str));   /* Right (explicit cast) */
```

V.0.2 Libc Functions

In this first part, you must re-code a set of the libc functions, as defined in their man. Your functions will need to present the same prototype and behaviors as the originals. Your functions' names must be prefixed by "ft_". For instance strlen becomes ft_strlen.

Some of the functions' prototypes you have to re-code use the "restrict" qualifier. This keyword is part of the c99 standard. It is therefore forbidden to include it in your prototypes and to compile it with the flag -std=c99

You must re-code the follwing functions:

- memset
- strlen
- strdup
- strcpy
- strncpy
- strchr
- strcmp
- isdigit
- toupper
- tolower

Chapter VI

Submission and peer-evaluation

Submit your work on your GiT repository as usual. Only the work on your repository will be graded.

Once your have completed your defences, Deepthought (the “moulinette”) will grade your work. Your final grade will be calculated taking into account your peer-correction grades and Deepthought’s grade.

Deepthought will grade your assignments in the order of the subject : Part 1, Part 2 and Bonus. One error in one of the sections will automatically stop the grading.

Good luck to you and don’t forget your author file !