C Piscine

Day 08

Staff 42 pedago@42.fr

Summary: This document is the subject for Day08 of the C Piscine @ 42.

Contents

Chapter 1

Instructions

- The exercises are carefully laid out in order of difficulty, from easiest to hardest. An exercise is only graded if all previous ones are correct. In other words: the grading for a day stops at the first mistake.
- Be mindful of the submission procedures indicated at the start of every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. Be as thorough as possible!
- Moulinette relies on a program called **norminette** to check if your files respect the Norm. An exercise containing files that do not respect the Norm will be graded 0.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- If **ft_putchar()** is an authorized function, we will compile your code with our **ft_putchar.c**.
- You'll only have to submit a **main()** function if we ask for a program.
- Moulinette compiles with these flags: **-Wall -Wextra -Werror**, and uses **gcc**.
- If your program doesn't compile, it will be graded 0.
- You should not leave any additional file in your directory than those specified in the subject.

For this day, norminette is launched without any particular flag!

The forewords are entirely unrelated to the subjects and can safely be ignored.

Chapter 2

Topics

Today, you will have to learn about:

- C preprocessor macros
- Structures

Chapter 3

Foreword

Here's what **Wikipedia** have to say about **Platypus** :

The platypus (Ornithorhynchus anatinus), also known as the duck-billed platypus, is a semiaquatic egg-laying mammal endemic to eastern Australia, including Tasmania. Together with the four species of echidna, it is one of the five extant species of monotremes, the only mammals that lay eggs instead of giving birth. The animal is the sole living representative of its family (Ornithorhynchidae) and genus (Ornithorhynchus), though a number of related species have been found in the fossil record.

The unusual appearance of this egg-laying, duck-billed, beaver-tailed, otter-footed mammal baffled European naturalists when they first encountered it, with some considering it an elaborate hoax. It is one of the few venomous mammals, the male platypus having a spur on the hind foot that delivers a venom capable of causing severe pain to humans. The unique features of the platypus make it an important subject in the study of evolutionary biology and a recognisable and iconic symbol of Australia; it has appeared as a mascot at national events and is featured on the reverse of its 20-cent coin. The platypus is the animal emblem of the state of New South Wales.

Until the early 20th century, it was hunted for its fur, but it is now protected throughout its range. Although captive breeding programs have had only limited success and the platypus is vulnerable to the effects of pollution, it is not under any immediate threat.

This subject is absolutly not talking about platypus.

Chapter 4

Exercise 00 : ft_split_whitespaces

Turn-in directory : `ex00/`
Files to turn in: `ft_split_whitespaces.c`
Allowed functions: `malloc`

---

42 - Classics : Theses exercises are key assignments that do not earn points, but are mandatory to validate in order to access to the real assignments of the day.

- Create a function that splits a string of characters into words.
- Words are separated by spaces, tabs and line breaks.
- This function returns an array of strings, each of these strings being a word from the argument **str**. The last element of this array should be equal to 0 to mark the end of the array.
- There can't be any empty strings in your array. Draw the necessary conclusions.
- The given string can't be modified.
- Here's how it should be prototyped :

```
char **ft_split_whitespaces(char *str);
```

Chapter 5

Exercise 01 : ft.h

Turn-in directory : `ex01/`
Files to turn in: `ft.h`
Allowed functions: None

- Create your **ft.h** file.
- It contains all prototypes of your **libft.a** functions.

Chapter 6

Exercise 02 : ft_boolean.h

Turn-in directory : ex02/
Files to turn in: ft_boolean.h
Allowed functions: None

- Create a **ft_boolean.h** file so that the following compiles and runs appropriately:

```
#include "ft_boolean.h"

void        ft_putstr(char *str)
{
        while (*str)
                write(1, str++, 1);
}

t_bool        ft_is_even(int nbr)
{
        return ((EVEN(nbr)) ? TRUE : FALSE);
}

int                main(int argc, char **argv)
{
        (void)argv;
        if (ft_is_even(argc - 1) == TRUE)
                ft_putstr(EVEN_MSG);
        else
                ft_putstr(ODD_MSG);
        return (SUCCESS);
}
```

- This program should display

```
I have an even number of arguments.
```

- or

```
I have an odd number of arguments.
```

- followed by a line break when adequate.

Chapter 7

Exercise 03 : ft_abs.h

Turn-in directory : `ex03/`
Files to turn in: `ft_abs.h`
Allowed functions: `None`

- Create a macro **ABS** which replaces its argument by its absolute value:

```
#define ABS(Value)
```

Chapter 8

Exercise 04 : ft_point.h

Turn-in directory : ex04/
Files to turn in: ft_point.h
Allowed functions: None

- Create a file **ft_point.h** so that the following compiles:

```
#include "ft_point.h"

void        set_point(t_point *point)
{
        point->x = 42;
        point->y = 21;
}

int                 main(void)
{
        t_point                 point;

        set_point(&point);
        return (0);
}
```

Chapter 9

Exercise 05 : ft_param_to_tab

Turn-in directory : ex05/
Files to turn in: ft_param_to_tab.c, ft_stock_par.h
Allowed functions: ft_split_whitespaces, ft_show_tab, malloc

- Create a function that stores the program's arguments within an array of structures and that returns the address of that array's first element.
- All arguments must be processed, including **av[0]**.
- Here's how it should be prototyped :

```
struct s_stock_par *ft_param_to_tab(int ac, char **av);
```

- The array should be allocated and its last element shall contain 0 in its **str** field to mark the end of the array.

- The structure is defined in the **ft_stock_par.h** file, like this :

```
typedef struct s_stock_par
{
  int  size_param;
  char *str;
  char *copy;
  char **tab;
}            t_stock_par;
```

  - **size_param** being the length of the argument ;
  - **str** being the address of the argument ;
  - **copy** being a copy of the argument ;
  - **tab** being the array returned by **ft_split_whitespaces**.

- We'll test your function with our **ft_split_whitespaces** and our **ft_show_tab** (next exercise). Take the appropriate measures for this to work !

Chapter 10

Exercise 06 : ft_show_tab

Turn-in directory : `ex06/`

Files to turn in: `ft_show_tab.c, ft_stock_par.h`

Allowed functions: `ft_putchar`

- Create a function that displays the content of the array created by the previous function.
- Here's how it should be prototyped :

```
void ft_show_tab(struct s_stock_par *par);
```

- The structure is defined in the **ft_stock_par.h** file, like this :

```
typedef struct s_stock_par
{
  int  size_param;
  char *str;
  char *copy;
  char **tab;
}            t_stock_par;
```

- For each element, we'll display (one element per line):
  - the argument
  - the size
  - each word (one per line)
- We'll test your function with our **ft_param_to_tab** (previous exercise). Take the appropriate measures for this to work !