C Piscine

Day 05

Staff 42 pedago@42.fr

Summary: This document is the subject for Day05 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.

norminette must be launched with the -R CheckForbiddenSourceHeader flag. Moulinette will use it too.

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

Chapter 2

Topics

Today, you will have to learn about:

- The ASCII encoding
- Positional number systems

Chapter 3

Foreword

Here is a discuss extract from the **Silicon Valley** serie:

- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- Oh, God, help us! Okay, uh you know what? I just don't think this is going to work. I'm so sorry. Uh, I mean like, what, we're going to bring kids into this world with that over their heads? That's not really fair to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SLAMS) - (BANGING)

. . .

(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.

Hopefully, you are not forced to use emacs and your space bar to complete the following exercices.

Chapter 4

Exercise 00 : ft_putstr

Turn-in directory : `ex00/`
Files to turn in: `ft_putstr.c`
Allowed functions: `ft_putchar`

---

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 displays a string of characters on the standard output.
- Here's how it should be prototyped :

```
void    ft_putstr(char *str);
```

Chapter 5

Exercise 01 : ft_putnbr

Turn-in directory : `ex01/`
Files to turn in: `ft_putnbr.c`
Allowed functions: `ft_putchar`

---

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 displays the number entered as a parameter. The function has to be able to display all possible values within an **int** type variable.
- Here's how it should be prototyped :

```
void ft_putnbr(int nb);
```

- For example:
    - **ft_putnbr(42)** displays "**42**".

Chapter 6

Exercise 02 : ft_atoi

Turn-in directory : `ex02/`
Files to turn in: `ft_atoi.c`
Allowed functions: None

---

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.

- Reproduce the behavior of the function **atoi** (man atoi).
- Here's how it should be prototyped :

```
int     ft_atoi(char *str);
```

Chapter 7

Exercise 03 : ft_strcpy

Turn-in directory : ex03/
Files to turn in: ft_strcpy.c
Allowed functions: None

- Reproduce the behavior of the function **strcpy** (man strcpy).
- Here's how it should be prototyped :

```
char        *ft_strcpy(char *dest, char *src);
```

Chapter 8

Exercise 04 : ft_strncpy

Turn-in directory : ex04/
Files to turn in: ft_strncpy.c
Allowed functions: None

- Reproduce the behavior of the function **strncpy** (man strncpy).
- Here's how it should be prototyped :

```
char        *ft_strncpy(char *dest, char *src, unsigned int n);
```

Chapter 9

Exercise 05 : ft_strstr

Turn-in directory : ex05/
Files to turn in: ft_strstr.c
Allowed functions: None

- Reproduce the behavior of the function **strstr** (man strstr).
- Here's how it should be prototyped :

```
char        *ft_strstr(char *str, char *to_find);
```

Chapter 10

Exercise 06 : ft_strcmp

Turn-in directory : ex06/
Files to turn in: ft_strcmp.c
Allowed functions: None

- Reproduce the behavior of the function **strcmp** (man strcmp).
- Here's how it should be prototyped :

```
int        ft_strcmp(char *s1, char *s2);
```

Chapter 11

Exercise 07 : ft_strncmp

Turn-in directory : ex07/
Files to turn in: ft_strncmp.c
Allowed functions: None

- Reproduce the behavior of the function **strncmp** (man strncmp).
- Here's how it should be prototyped :

```
int        ft_strncmp(char *s1, char *s2, unsigned int n);
```

Chapter 12

Exercise 08 : ft_strupcase

Turn-in directory : ex08/
Files to turn in: ft_strupcase.c
Allowed functions: None

- Create a function that transforms every letter of every word to uppercase.
- Here's how it should be prototyped :

```
char	*ft_strupcase(char *str);
```

- It should return **str**.

Chapter 13

Exercise 09 : ft_strlowcase

Turn-in directory : `ex09/`
Files to turn in: `ft_strlowcase.c`
Allowed functions: `None`

- Create a function that transforms every letter of every word to lowercase.
- Here's how it should be prototyped :

```
char        *ft_strlowcase(char *str);
```

- It should return **str**.

Chapter 14

Exercise 10 : ft_strcapitalize

Turn-in directory : ex10/
Files to turn in: ft_strcapitalize.c
Allowed functions: None

- Create a function that capitalizes the first letter of each word and transforms all other letters to lowercase.
- A word is a string of alphanumeric characters.
- Here's how it should be prototyped :

```
char        *ft_strcapitalize(char *str);
```

- It should return **str**.
- For example:

```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

- Becomes:

```
Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un
```

Chapter 15

Exercise 11 : ft_str_is_alpha

Turn-in directory : `ex11/`
Files to turn in: `ft_str_is_alpha.c`
Allowed functions: `None`

- Create a function that returns 1 if the string given as a parameter contains only alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int        ft_str_is_alpha(char *str);
```

- It should return 1 if **str** is empty.

Chapter 16

Exercise 12 : ft_str_is_numeric

Turn-in directory : `ex12/`
Files to turn in: `ft_str_is_numeric.c`
Allowed functions: `None`

- Create a function that returns 1 if the string given as a parameter contains only digits, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int		ft_str_is_numeric(char *str);
```

- It should return 1 if **str** is empty.

Chapter 17

Exercise 13 : ft_str_is_lowercase

Turn-in directory : `ex13/`
Files to turn in: `ft_str_is_lowercase.c`
Allowed functions: `None`

- Create a function that returns 1 if the string given as a parameter contains only lowercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int        ft_str_is_lowercase(char *str);
```

- It should return 1 if **str** is empty.

Chapter 18

Exercise 14 : ft_str_is_uppercase

Turn-in directory : ex14/
Files to turn in: ft_str_is_uppercase.c
Allowed functions: None

- Create a function that returns 1 if the string given as a parameter contains only uppercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int        ft_str_is_uppercase(char *str);
```

- It should return 1 if **str** is empty.

Chapter 19

Exercise 15 : ft_str_is_printable

Turn-in directory : `ex15/`
Files to turn in: `ft_str_is_printable.c`
Allowed functions: `None`

- Create a function that returns 1 if the string given as a parameter contains only printable characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int        ft_str_is_printable(char *str);
```

- It should return 1 if **str** is empty.

Chapter 20

Exercise 16 : ft_strcat

Turn-in directory : `ex16/`
Files to turn in: `ft_strcat.c`
Allowed functions: None

- Reproduce the behavior of the function **strcat** (man strcat).
- Here's how it should be prototyped :

```
char *ft_strcat(char *dest, char *src);
```

Chapter 21

Exercise 17 : ft_strncat

Turn-in directory : ex17/
Files to turn in: ft_strncat.c
Allowed functions: None

- Reproduce the behavior of the function **strncat** (man strncat).
- Here's how it should be prototyped :

```
char *ft_strncat(char *dest, char *src, int nb);
```

Chapter 22

Exercise 18 : ft_strlcat

Turn-in directory : ex18/
Files to turn in: ft_strlcat.c
Allowed functions: None

- Reproduce the behavior of the function **strlcat** (man strlcat).
- Here's how it should be prototyped :

```
unsigned int ft_strlcat(char *dest, char *src, unsigned int size);
```

Chapter 23

Exercise 19 : ft_strlcpy

Turn-in directory : ex19/
Files to turn in: ft_strlcpy.c
Allowed functions: None

- Reproduce the behavior of the function **strlcpy** (man strlcpy).
- Here's how it should be prototyped :

```
unsigned int ft_strlcpy(char *dest, char *src, unsigned int size);
```

Chapter 24

Exercise 20 : ft_putnbr_base

Turn-in directory : `ex20/`
Files to turn in: `ft_putnbr_base.c`
Allowed functions: `ft_putchar`

- Create a function that displays a number in a base system onscreen.
- This number is given in the shape of an **int**, and the radix in the shape of a **string of characters**.
- The base-system contains all useable symbols to display that number :
    - **0123456789** is the commonly used base system to represent decimal numbers ;
    - **01** is a binary base system ;
    - **0123456789ABCDEF** an hexadecimal base system ;
    - **poneyvif** is an octal base system.
- The function must handle negative numbers.
- If there's an invalid argument, nothing should be displayed. Examples of invalid arguments :
    - base is empty or size of 1;
    - base contains the same character twice ;
    - base contains + or - ;
    - etc.
- Here's how it should be prototyped :

```
void        ft_putnbr_base(int nbr, char *base);
```

Chapter 25

Exercise 21 : ft_atoi_base

Turn-in directory : `ex21/`
Files to turn in: `ft_atoi_base.c`
Allowed functions: `None`

- Create a function that returns a number. This number is shaped as a **string of characters**.
- The string of characters reveals the number in a specific base, given as a second parameter.
- The function must handle negative numbers.
- The function must handle signs like **man atoi**.
- If there's an invalid argument, the function should return 0. Examples of invalid arguments :
    - str is an empty string ;
    - the base is empty or size of 1;
    - str contains characters that aren't part of the base, or aren't + nor - ;
    - the base contains the same character twice ;
    - the base contains + or - ;
    - etc.
- Here's how it should be prototyped :

```
int        ft_atoi_base(char *str, char *base);
```

Chapter 26

Exercise 22 : ft_putstr_non_printable

Turn-in directory : ex22/
Files to turn in: ft_putstr_non_printable.c
Allowed functions: ft_putchar

- Create a function that displays a string of characters onscreen. If this string contains characters that aren't printable, they'll have to be displayed in the shape of hexadecimals (lowercase), preceeded by a "backslash".
- For example :

```
Coucou\ntu vas bien ?
```

- The function should display :

```
Coucou\0atu vas bien ?
```

- Here's how it should be prototyped :

```
void        ft_putstr_non_printable(char *str);
```

Chapter 27

Exercise 23 : ft_print_memory

Turn-in directory : `ex23/`
Files to turn in: `ft_print_memory.c`
Allowed functions: `ft_putchar`

- Create a function that displays the memory area onscreen.
- The display of this memory area should be split into three columns :
  - The hexadecimal address of the first line's first character ;
  - The content in hexadecimal ;
  - The content in printable characters.
- If a character is non-printable, it'll be replaced by a dot.
- Each line should handle sixteen characters.
- If **size** equals to 0, nothing should be displayed.

- Example:

```
guilla_i@seattle $> ./ft_print_memory
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368  Salut les aminch
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368  es c'est cool sh
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064  ow mem on fait d
00000030: 6520 7472 7563 2074 6572 7269 626c 6500  e truc terrible.
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............
guilla_i@seattle $> ./ft_print_memory | cat -te
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368  Salut les aminch$
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368  es c'est cool sh$
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064  ow mem on fait d$
00000030: 6520 7472 7563 2074 6572 7269 626c 6500  e truc terrible.$
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............$
guilla_i@seattle $>
```

- Here's how it should be prototyped :

```
void        *ft_print_memory(void *addr, unsigned int size);
```

- It should return **addr**.