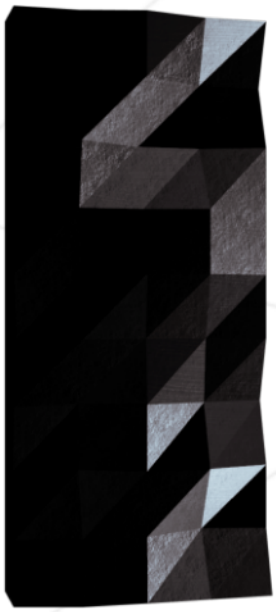

Piscine

02

Summ ry: This document is the subject for the C 02 module of the C Piscine @ 42.

Version: 4.2

Contents

Ch pter I

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- 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. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get `-42`, and this grade is non-negotiable.
- 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, you'll get `0`.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? sk your peer on the right. Otherwise, try your peer on the left.

- Your reference guide is called `Google / man / the Internet / ...`.
- Check out the "C Piscine" part of the forum on the intranet, or the slack Piscine.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor ! Use your brain !!!

`Norminette must be l unched with the -R CheckForbiddenSourceHe der fl g. Moulinette will use it too.`

Ch pter II

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.
-  nd 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 SP CE B R M NY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SL MS) - (B NGING)

. . .

(RICH RD MO NS)

- 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.

Ch pter III

Exercise 00 : ft_strcpy

	Exercise 00
ft_strcpy	
Turn-in directory : $ex00$	
Files to turn in : `ft_strcpy.c`	
llowed functions : `None`	

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

```
ch r        *ft_strcpy(ch r *dest, ch r *src);
```

Ch pter IV

Exercise 01 : ft_strncpy

	Exercise 01
ft_strncpy	
Turn-in directory : $ex01$	
Files to turn in : `ft_strncpy.c`	
llowed functions : `None`	

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

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

Ch pter V

Exercise 02 : ft_str_is_ lph

	Exercise 02
ft_str_is_alpha	
Turn-in directory : *ex*02	
Files to turn in : `ft_str_is_ lph .c`	
llowed 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_ lph (ch r *str);
```

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

Ch pter VI

Exercise 03 : ft_str_is_numeric

	Exercise 03
ft_str_is_numeric	
Turn-in directory : $ex03$	
Files to turn in : `ft_str_is_numeric.c`	
llowed 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(ch r *str);
```

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

Ch pter VII

Exercise 04 : ft_str_is_lowerc se

	Exercise 04
ft_str_is_lowercase	
Turn-in directory : $ex04$	
Files to turn in : `ft_str_is_lowerc se.c`	
llowed 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_lowerc se(ch r *str);
```

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

Ch pter VIII

Exercise 05 : ft_str_is_upperc se

	Exercise 05
ft_str_is_uppercase	
Turn-in directory : $ex05$	
Files to turn in : `ft_str_is_upperc se.c`	
llowed 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_upperc se(ch r *str);
```

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

Ch pter IX

Exercise 06 : ft_str_is_print ble

	Exercise 06
ft_str_is_printable	
Turn-in directory : *ex*06	
Files to turn in : `ft_str_is_print ble.c`	
llowed 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_print ble(ch r *str);
```

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

Ch pter X

Exercise 07 : ft_strupc se

	Exercise 07
ft_strupcase	
Turn-in directory : *ex*07	
Files to turn in : `ft_strupc se.c`	
llowed functions : `None`	

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

```
ch r        *ft_strupc se(ch r *str);
```

- It should return `str`.

Ch pter XI

Exercise 08 : ft_strlowc se

	Exercise 08
ft_strlowcase	
Turn-in directory : $ex08$	
Files to turn in : `ft_strlowc se.c`	
llowed functions : `None`	

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

```
ch r        *ft_strlowc se(ch r *str);
```

- It should return `str`.

Ch pter XII

Exercise 09 : ft_strc pit lize

	Exercise 09
ft_strcapitalize	
Turn-in directory : *ex*09	
Files to turn in : `ft_strc pit lize.c`	
llowed functions : `None`	

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

```
ch r        *ft_strc pit lize(ch r *str);
```

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

```
s lut, comment tu v s ? 42mots qu r nte-deux; cinqu nte+et+un
```

- Becomes:

```
S lut, Comment Tu V s ? 42mots Qu r nte-Deux; Cinqu nte+Et+Un
```

Ch pter XIII

Exercise 10 : ft_strlcpy

	Exercise 10
ft_strlcpy	
Turn-in directory : $ex10$	
Files to turn in : `ft_strlcpy.c`	
llowed functions : `None`	

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

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

Ch pter XIV

Exercise 11 : ft_putstr_non_print ble

	Exercise 11
ft_putstr_with_non_printable	
Turn-in directory : $ex11$	
Files to turn in : `ft_putstr_non_print ble.c`	
llowed functions : `write`	

- 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 v s bien ?
```

- The function should display :

```
Coucou\0 tu v s bien ?
```

- Here's how it should be prototyped :

```
void        ft_putstr_non_print ble(ch r *str);
```

Ch pter XV

Exercise 12 : ft_print_memory

	Exercise 12
ft_print_memory	
Turn-in directory : $ex12$	
Files to turn in : `ft_print_memory.c`	
llowed functions : `write`	

- Create a function that displays the memory area onscreen.
- The display of this memory area should be split into three "columns" separated by a space:

  The hexadecimal address of the first line's first character followed by a ':'.

  The content in hexadecimal with a space each 2 characters and should be padded with spaces if needed (see the example below).

  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` is equal to 0, nothing should be displayed.

- Example:

```
$> ./ft_print_memory
000000010 161f40: 426f 6e6  6f75 7220 6c65 7320 616d 696e Bonjour les  min
000000010 161f50: 6368 6573 090  0963 0720 6573 7420 666f ches...c. est fo
000000010 161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010 161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut f ire  vec.
000000010 161f80: 0 09 7072 696e 745f 6d65 6d6f 7279 0 0  ..print_memory..
000000010 161f90: 0 09 6c6f 6c2e 6c6f 6c0  2000           ..lol.lol. .
$> ./ft_print_memory | c t -te
0000000107ff9f40: 426f 6e6  6f75 7220 6c65 7320 616d 696e Bonjour les  min$
0000000107ff9f50: 6368 6573 090  0963 0720 6573 7420 666f ches...c. est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut f ire  vec.$
0000000107ff9f80: 0 09 7072 696e 745f 6d65 6d6f 7279 0 0  ..print_memory..$
0000000107ff9f90: 0 09 6c6f 6c2e 6c6f 6c0  2000           ..lol.lol. .$
$>
```

- Here's how it should be prototyped:

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

- It should return `addr`.