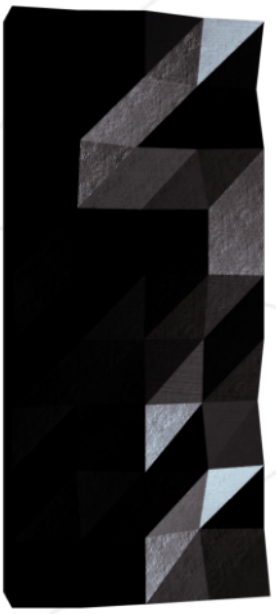

C Piscine

C 06

Staff 42 pedago@42.fr

Summary: This document is the subject for the module C 06 of the C Piscine @ 42.

Version: 7.3

Contents

I	**Instructions**	**2**
II	**Foreword**	**4**
III	**Exercise 00 : ft_print_program_name**	**5**
IV	**Exercise 01 : ft_print_params**	**6**
V	**Exercise 02 : ft_rev_params**	**7**
VI	**Exercise 03 : ft_sort_params**	**8**
VII	**Submission and peer-evaluation**	**9**

Chapter I

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change 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 `cc`.
- 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? Ask 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 launched with the -R CheckForbiddenSourceHeader flag. Moulinette will use it too.`

Chapter II

Foreword

Dialog from the movie `The Big Lebowski`:

```
The Dude: Walter, ya know, it's Smokey, so his toe slipped over the line a little,
  big deal. It's just a game, man.
Walter Sobchak: Dude, this is a league game, this determines who enters the next
  round robin. Am I wrong? Am I wrong?
Smokey: Yeah, but I wasn't over. Gimme the marker Dude, I'm marking it 8.
Walter Sobchak: [pulls out a gun] Smokey, my friend, you are entering a world
  of pain.
The Dude: Walter...
Walter Sobchak: You mark that frame an 8, and you're entering a world of pain.
Smokey: I'm not...
Walter Sobchak: A world of pain.
Smokey: Dude, he's your partner...
Walter Sobchak: [shouting] Has the whole world gone crazy? Am I the only one
  around here who gives a shit about the rules? Mark it zero!
The Dude: They're calling the cops, put the piece away.
Walter Sobchak: Mark it zero!
[points gun in Smokey's face]
The Dude: Walter...
Walter Sobchak: [shouting] You think I'm fucking around here? Mark it zero!
Smokey: All right, it's fucking zero. Are you happy, you crazy fuck?
Walter Sobchak: ...It's a league game, Smokey.
```

Chapter III

Exercise 00 : ft_print_program_name

	Exercise 00
ft_print_program_name	
Turn-in directory : *ex*00/	
Files to turn in : `ft_print_program_name.c`	
Allowed functions : `write`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its own name followed by a new line.
- Example :

```
$>./a.out | cat -e
./a.out$
$>
```

Chapter IV

Exercise 01 : ft_print_params

	Exercise 01
	ft_print_params
Turn-in directory : $ex01/$	
Files to turn in : `ft_print_params.c`	
Allowed functions : `write`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments.
- One per line, in the same order as in the command line.
- It should display all arguments, except for `argv[0]`.
- Example :

```
$>./a.out test1 test2 test3 | cat -e
test1$
test2$
test3$
$>
```

Chapter V

Exercise 02 : ft_rev_params

	Exercise 02
ft_rev_params	
Turn-in directory : $ex02/$	
Files to turn in : `ft_rev_params.c`	
Allowed functions : `write`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments.
- One per line, in the reverse order of the command line.
- It should display all arguments, except for `argv[0]`.

Chapter VI

Exercise 03 : ft_sort_params

	Exercise 03
ft_sort_params	
Turn-in directory : $ex03/$	
Files to turn in : `ft_sort_params.c`	
Allowed functions : `write`	

- We're dealing with a program here, you should therefore have a function `main` in your `.c` file.
- Create a program that displays its given arguments sorted by ascii order.
- It should display all arguments, except for `argv[0]`.
- One argument per line.

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

`You need to return only the files requested by the subject of this project.`