

C Piscine C 09

Summary: This document is the subject for the module C 09 of the C Piscine @ 42.

Version: 4

Contents

Ι	Instructions	2
II	Foreword	4
III	Exercise 00 : libft	5
IV	Exercise 01 : Makefile	6
V	Exercise 02 : ft_split	8
\mathbf{VI}	Submission and peer-evaluation	g

Chapter I

Instructions

- Only this page serves as your reference, do not trust rumors.
- Watch out! This document may change before submission.
- Ensure you have the appropriate permissions on your files and directories.
- You must follow the **submission procedures** for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be evaluated by a program called **Moulinette**.
- Moulinette is meticulous and strict in its assessment. It is fully automated, and there is no way to negotiate with it. To avoid unpleasant surprises, be as thorough as possible.
- Moulinette is not open-minded. If your code does not adhere to the Norm, it won't attempt to understand it. Moulinette relies on a program called norminette to check if your files comply with the Norm. TL;DR: Submitting work that doesn't pass norminette's check makes no sense.
- These exercises are arranged in order of difficulty, from easiest to hardest. We will not consider a successfully completed harder exercise if an easier one is not fully functional.
- Using a forbidden function is considered cheating. Cheaters receive a grade of **-42**, which is non-negotiable.
- You only need to submit a main() function if we specifically ask for a program.
- Moulinette compiles with the following flags: -Wall -Wextra -Werror, using cc.
- If your program does not compile, you will receive a grade of **0**.
- You **cannot** leave **any** additional file in your directory beyond those specified in the assignment.
- Have a question? Ask the peer on your right. If not, try the peer on your left.

- \bullet Your reference guide is called **Google / man / the Internet / ...**
- Check the "C Piscine" section of the forum on the intranet or the Piscine on Slack.
- Carefully examine the examples. They may contain crucial details that are not explicitly stated in the assignment...
- 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 ...

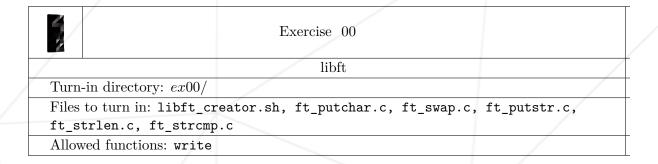
Walter Sobchak: [shouting] You think I'm fucking around here? Mark it zero!

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: libft



- Create your ft library. It will be called libft.a.
- A shell script called libft_creator.sh will compile the source files appropriately and will create your library.
- This library should contain <u>all</u> of the following functions :

```
void ft_putchar(char c);
void ft_swap(int *a, int *b);
void ft_putstr(char *str);
int ft_strlen(char *str);
int ft_strcmp(char *s1, char *s2);
```

• We'll launch the following command-line :

sh libft_creator.sh

Chapter IV

Exercise 01: Makefile

	Exercise 01	
/	Makefile	
Turn-in directory: $ex01/$		
Files to turn in: Makefile		
Allowed functions: None		

- Create the Makefile that will compile a library libft.a.
- Your Makefile should print all the commands it's running.
- Your Makefile should not run any unnecessary commands.
- The Makefile will get its source files from the "srcs" directory.
- These files will be: ft_putchar.c, ft_swap.c, ft_putstr.c, ft_strlen.c, ft_strcmp.c.
- The Makefile will get its header files from the "includes" directory.
- These files will be: ft.h.
- It should compile the .c files with cc and with -Wall -Wextra -Werror flags in that order.
- The lib should be at the root of the exercise.
- .o files should be near their corresponding .c files.
- The Makefile should also implement the following rules: clean, fclean, re, all, and of course libft.a.
- Running just make should be equivalent to make all.
- The all rule should be equivalent to make libft.a.
- The clean rule should remove all the temporary generated files.

C Piscine

C 09

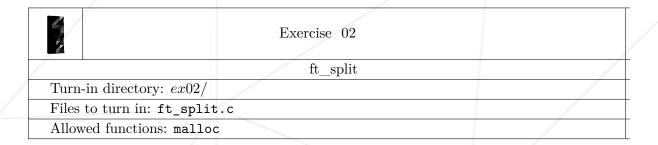
- The fclean rule should be like a make clean, plus removing all the binaries generated with make all.
- The re rule should be like a make fclean followed by make all.
- Your Makefile should not compile any file unnecessarily.
- We'll only fetch your Makefile and test it with our files.



Watch out for wildcards!

Chapter V

Exercise 02: ft_split



- Create a function that splits a string, using each character from the **charset** string as a separator.
- You will need to use each character from the **charset** string individually as a separator.
- The function should return an array where each element contains the address of a substring wrapped between two separators. The last element of the array should be NULL to indicate the end of the array.
- There should be no empty strings in your array. Draw your conclusions accordingly.
- The string provided as an argument cannot be modified.
- Here's the function prototype:

char **ft_split(char *str, char *charset);

Chapter VI

Submission and peer-evaluation

Submit your assignment to your Git repository as usual. Only the work inside your repository will be evaluated during the defense. Make sure to double-check the filenames to ensure they are correct.



You must submit only the files required by the project instructions.