



Piscine Reloaded

It's good to be back

Summary:

The Piscine was good but the time has past. This serie of exercises will help you to remind all the basics you've learned during the piscine. Functions, loops, pointers, structures, let's remind together the syntactic and semantic bases of the C

Version:

Contents

I	Foreword	3
II	Introduction	4
III	General rules	5
IV	Exercise 00: Z	7
V	Exercise 01: Oh yeah, mooore...	8
VI	Exercise 02: clean	9
VII	Exercise 03: find_sh	10
VIII	Exercise 04: midLS	11
IX	Exercise 05: GiT commit	12
X	Exercise 06: ft_print_alphabet	13
XI	Exercise 07: ft_print_numbers	14
XII	Exercise 08: ft_is_negative	15
XIII	Exercise 09: ft_ft	16
XIV	Exercise 10: ft_swap	17
XV	Exercise 11: ft_div_mod	18
XVI	Exercise 12: ft_iterative_factorial	19
XVII	Exercise 13: ft_recursive_factorial	20
XVIII	Exercise 14: ft_sqrt	21
XIX	Exercise 15: ft_putstr	22
XX	Exercise 16: ft_strlen	23
XXI	Exercise 17: ft_strcmp	24
XXII	Exercise 18: ft_print_params	25
XXIII	Exercise 19: ft_sort_params	26

XXIV Exercise 20: ft_strdup	27
XXV Exercise 21: ft_range	28
XXVI Exercise 22: ft_abs.h	29
XXVII Exercise 23: ft_point.h	30
XXVIII Exercise 24: Makefile	31
XXIX Exercise 25: ft_FOREACH	32
XXX Exercise 26: ft_count_if	33
XXXI Exercise 27: display_file	34
XXXII Submission and peer-evaluation	35

Chapter I

Foreword

Edward Joseph Snowden (born June 21, 1983) is an American computer professional, former Central Intelligence Agency (CIA) employee, and former contractor for the United States government who copied and leaked classified information from the National Security Agency (NSA) in 2013 without authorization. His disclosures revealed numerous global surveillance programs, many run by the NSA and the Five Eyes Intelligence Alliance with the cooperation of telecommunication companies and European governments.

In 2013, Snowden was hired by an NSA contractor, Booz Allen Hamilton, after previous employment with Dell and the CIA. On May 20, 2013, Snowden flew to Hong Kong after leaving his job at an NSA facility in Hawaii, and in early June he revealed thousands of classified NSA documents to journalists Glenn Greenwald, Laura Poitras, and Ewen MacAskill. Snowden came to international attention after stories based on the material appeared in The Guardian and The Washington Post. Further disclosures were made by other publications including Der Spiegel and The New York Times.

On June 21, 2013, the U.S. Department of Justice unsealed charges against Snowden of two counts of violating the Espionage Act of 1917 and theft of government property. Two days later, he flew into Moscow's Sheremetyevo Airport, but Russian authorities noted that his U.S. passport had been cancelled and he was restricted to the airport terminal for over one month. Russia ultimately granted him right of asylum for one year, and repeated extensions have permitted him to stay at least until 2020. He reportedly lives in an undisclosed location in Moscow, and continues to seek asylum elsewhere in the world.

A subject of controversy, Snowden has been variously called a hero, a whistleblower, a dissident, a traitor and a patriot. His disclosures have fueled debates over mass surveillance, government secrecy, and the balance between national security and information privacy.

There is a very good documentary on HBO [here](#).

Chapter II

Introduction

The **Piscine Reloaded** is a best-of of the exercises you did during the **Piscine C** to remind you all the basics of the C programming language. All the exercises has to be done entierly to unlock the next project.

If you have already done some of these exercises during the Piscine C, we highly recommend not be tempted to retrieve your old code. The learning of programming involves practice and making an existing code has no interest.

Chapter III

General rules

- Only this page will serve as reference; do not trust rumors.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the turn-in procedures for every exercise.
- 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.
- Exercises in Shell must be executable with /bin/sh.
- 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 forum on the intranet and Slack.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- 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 Norminator to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass Norminator's check.
- 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 ft_putchar() is an authorized function, we will compile your code with our ft_putchar.c.

- If your program doesn't compile, you'll get 0.
- By Odin, by Thor! Use your brain!!!

Chapter IV

Exercise 00: Z

	Exercise 00
	Only the best know how to display Z
	Turn-in directory : <i>ex00/</i>
	Files to turn in : z
	Allowed functions : None

- Create a file called **z** that returns "Z", followed by a new line, whenever the command **cat** is used on it.

```
?>cat z  
Z  
?>
```

Chapter V

Exercise 01: Oh yeah, mooore...

	Exercise 01
	Oh yeah, mooore...
Turn-in directory :	<i>ex01/</i>
Files to turn in :	exo.tar
Allowed functions :	None

- Create the following files and directories. Do what's necessary so that when you use the `ls -l` command in your directory, the output will looks like this :

```
$> ls -l
total 42
drwx--xr-x 2 XX XX  XX Jun 1 20:47 test0
-rwx--xr-- 1 XX XX   4 Jun 1 21:46 test1
dr-x----r-- 2 XX XX  XX Jun 1 22:45 test2
-r-----r-- 2 XX XX   1 Jun 1 23:44 test3
-rw-r----x 1 XX XX   2 Jun 1 23:43 test4
-r-----r-- 2 XX XX   1 Jun 1 23:44 test5
lrwxr--xr-x 1 XX XX   5 Jun 1 22:20 test6 -> test0
$>
```

- About the hours, it will be accepted if the year is displayed in the case of the exercise's date (1 Jun) is outdated by six month or more.
- Once you've done that, run `tar -cf exo.tar *` to create the file to be submitted.



Don't worry about what you've got instead of "XX".

Chapter VI

Exercise 02: clean

	Exercise 02
	Turn-in directory : <i>ex02/</i>
	Files to turn in : clean
	Allowed functions : None

- In a file called **clean** place the command line that will search for all files - in the current directory as well as in its sub-directories - with a name ending by `~`, or with a name that start and end by `#`
- The command line will show and erase all files found.
- Only one command is allowed: no `;` or `&&` or other shenanigans.



`man find`

Chapter VII

Exercise 03: find_sh

	Exercise 03
	find_sh.sh
Turn-in directory :	<i>ex03/</i>
Files to turn in :	find_sh.sh
Allowed functions :	None

- Write a command line that searches for all file names that end with ".sh" (without quotation marks) in the current directory and all its sub-directories. It should display only the file names without the .sh.
- Example of output :

```
$>./find_sh.sh | cat -e
find_sh$  
file1$  
file2$  
file3$  
$>
```

Chapter VIII

Exercise 04: midLS

	Exercise 04
	midLS
Turn-in directory :	<i>ex04/</i>
Files to turn in :	midLS
Allowed functions :	None

- In a file named midLS, write the command that lists all files and directories in your current directory (excluding hidden files or any file starting with a dot, including double dots).
- The output should be sorted by modification date, with entries separated by a comma and a space.
- Directory names should end with a slash (/).



Do only what is asked, nothing more!



- RTFM!
- Git push regularly.

Chapter IX

Exercise 05: GiT commit

	Exercise 05
	GiT commit?
	Turn-in directory : <i>ex05/</i>
	Files to turn in : <i>git_commit.sh</i>
	Allowed functions : None

- Create a shell script that displays the ids of the last 5 commits in your git repository.

```
%> bash git_commit.sh | cat -e  
baa23b54f0adb7bf42623d6d0a6ed4587e11412a$  
2f52d74b1387fa80ea844969e8dc5483b53iac1$  
905f53d98656771334f53f59bb984fc29774701f$  
5ddc8474f4f15b3fc72d08fc333e19c3a27078$  
e94d0b448c03ec633f16d84d63beaef9ae7e7be8$  
%>
```

- Your script will be tested in our own environment.



- RTFM!
- The first retry delay is short, trigger an intermediate evaluation to track your progress!

Chapter X

Exercise 06: ft_print_alphabet

	Exercise 06
	ft_print_alphabet
Turn-in directory :	<i>ex06/</i>
Files to turn in :	<code>ft_print_alphabet.c</code>
Allowed functions :	<code>ft_putchar</code>

- Create a function that displays the alphabet in lowercase, on a single line, by ascending order, starting from the letter 'a'.
- Here's how it should be prototyped :

```
void ft_print_alphabet(void);
```

Chapter XI

Exercise 07: ft_print_numbers

	Exercise 07
	ft_print_numbers
	Turn-in directory : <i>ex07/</i>
	Files to turn in : ft_print_numbers.c
	Allowed functions : ft_putchar

- Create a function that displays all digits, on a single line, by ascending order.
- Here's how it should be prototyped :

```
void ft_print_numbers(void);
```

Chapter XII

Exercise 08: ft_is_negative

	Exercise 08
	ft_is_negative
Turn-in directory :	<i>ex08/</i>
Files to turn in :	<code>ft_is_negative.c</code>
Allowed functions :	<code>ft_putchar</code>

- Create a function that displays 'N' or 'P' depending on the integer's sign entered as a parameter. If `n` is negative, display 'N'. If `n` is positive or null, display 'P'.
- Here's how it should be prototyped :

```
void ft_is_negative(int n);
```

Chapter XIII

Exercise 09: ft_ft

	Exercise 09
	ft_ft
Turn-in directory :	<i>ex09/</i>
Files to turn in :	ft_ft.c
Allowed functions :	None

- Create a function that takes a pointer to int as a parameter, and sets the value "42" to that int.
- Here's how it should be prototyped :

```
void ft_ft(int *nbr);
```

Chapter XIV

Exercise 10: ft_swap

	Exercise 10
	ft_swap
Turn-in directory :	<i>ex10/</i>
Files to turn in :	ft_swap.c
Allowed functions :	None

- Create a function that swaps the value of two integers whose addresses are entered as parameters.
- Here's how it should be prototyped :

```
void    ft_swap(int *a, int *b);
```

Chapter XV

Exercise 11: ft_div_mod

	Exercise 11
	ft_div_mod
	Turn-in directory : <i>ex11/</i>
	Files to turn in : ft_div_mod.c
	Allowed functions : None

- Create a function **ft_div_mod** prototyped like this :

```
void    ft_div_mod(int a, int b, int *div, int *mod);
```

- This function divides parameters **a** by **b** and stores the result in the int pointed by **div**. It also stores the remainder of the division of **a** by **b** in the int pointed by **mod**.

Chapter XVI

Exercise 12: ft_iterative_factorial

	Exercise 12
	ft_iterative_factorial
	Turn-in directory : <i>ex12/</i>
	Files to turn in : ft_iterative_factorial.c
	Allowed functions : None

- Create an iterated function that returns a number. This number is the result of a factorial operation based on the number given as a parameter.
- If there's an error, the function should return 0.
- Here's how it should be prototyped :

```
int ft_iterative_factorial(int nb);
```
- Your function must return its result in less than two seconds.

Chapter XVII

Exercise 13: ft_recursive_factorial

	Exercise 13
	ft_recursive_factorial
	Turn-in directory : <i>ex13/</i>
	Files to turn in : ft_recursive_factorial.c
	Allowed functions : None

- Create a recursive function that returns the factorial of the number given as a parameter.
- If there's an error, the function should return 0.
- Here's how it should be prototyped :

```
int ft_recursive_factorial(int nb);
```

Chapter XVIII

Exercise 14: ft_sqrt

	Exercise 14
	ft_sqrt
Turn-in directory :	<i>ex14/</i>
Files to turn in :	ft_sqrt.c
Allowed functions :	None

- Create a function that returns the square root of a number (if it exists), or 0 if the square root is an irrational number.
- Here's how it should be prototyped :

```
int ft_sqrt(int nb);
```

- Your function must return its result in less than two seconds.

Chapter XIX

Exercise 15: ft_putstr

	Exercise 15
	ft_putstr
Turn-in directory :	<i>ex15/</i>
Files to turn in :	ft_putstr.c
Allowed functions :	ft_putchar

- 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 XX

Exercise 16: ft_strlen

	Exercise 16
	ft_strlen
Turn-in directory :	<i>ex16/</i>
Files to turn in :	ft_strlen.c
Allowed functions :	None

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

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

Chapter XXI

Exercise 17: ft_strcmp

	Exercise 17
	ft_strcmp
Turn-in directory :	<i>ex17/</i>
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 XXII

Exercise 18: ft_print_params

	Exercise 18
	ft_print_params
	Turn-in directory : <i>ex18/</i>
	Files to turn in : <code>ft_print_params.c</code>
	Allowed functions : <code>ft_putchar</code>

- 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.
- Example :

```
$>./a.out test1 test2 test3
test1
test2
test3
$>
```

Chapter XXIII

Exercise 19: ft_sort_params

	Exercise 19
	ft_sort_params
Turn-in directory :	<i>ex19/</i>
Files to turn in :	<code>ft_sort_params.c</code>
Allowed functions :	<code>ft_putchar</code>

- 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]`.
- All arguments have to have their own line.

Chapter XXIV

Exercise 20: ft_strdup

	Exercise 20
	ft_strdup
Turn-in directory :	<i>ex20/</i>
Files to turn in :	<code>ft_strdup.c</code>
Allowed functions :	<code>malloc</code>

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

```
char *ft_strdup(char *src);
```

Chapter XXV

Exercise 21: ft_range

	Exercise 21
	ft_range
Turn-in directory :	<i>ex21/</i>
Files to turn in :	ft_range.c
Allowed functions :	malloc

- Create a function `ft_range` which returns an array of `int`s. This `int` array should contain all values between `min` and `max`.
- `Min included - max excluded.`
- Here's how it should be prototyped :

```
int *ft_range(int min, int max);
```

- If `min`'s value is greater or equal to `max`'s value, a null pointer should be returned.

Chapter XXVI

Exercise 22: ft_abs.h

	Exercise 22
	ft_abs.h
Turn-in directory :	<i>ex22/</i>
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)
```



You are asked to do something that is normally banned by the Norm,
that will be the only time we authorize it.

Chapter XXVII

Exercise 23: ft_point.h

	Exercise 23
	ft_point.h
Turn-in directory :	<i>ex23/</i>
Files to turn in :	ft_point.h
Allowed functions :	None

- Create a file **ft_point.h** that'll compile the following main :

```
#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 XXVIII

Exercise 24: Makefile

	Exercise 24
	Makefile
Turn-in directory : <i>ex24/</i>	
Files to turn in : Makefile	
Allowed functions : None	

- Create the **Makefile** that'll compile your **libft.a**.
- The **Makefile** will get its source files from the "srcs" directory.
- The **Makefile** will get its header files from the "includes" directory.
- The lib will be at the root of the exercise.
- The **Makefile** should also implement the following rules: **clean**, **fclean** and **re** as well as **all**.
- **fclean** does the equivalent of a make clean and also erases the binary created during the make. **re** does the equivalent of a make fclean followed by a make.
- We'll only fetch your Makefile and test it with our files. For this exercise, only the following 5 mandatory functions of your lib have to be handled : (**ft_putchar**, **ft_putstr**, **ft_strcmp**, **ft_strlen** and **ft_swap**).



Watch out for wildcards!

Chapter XXIX

Exercise 25: ft_foreach

	Exercise 25
	ft_foreach
Turn-in directory :	<i>ex25/</i>
Files to turn in :	ft_foreach.c
Allowed functions :	None

- Create the function **ft_foreach** which, for a given ints array, applies a function on all elements of the array. This function will be applied following the array's order.
- Here's how the function should be prototyped :

```
void ft_foreach(int *tab, int length, void(*f)(int));
```

- For example, the function **ft_foreach** could be called as follows in order to display all ints of the array :

```
ft_foreach(tab, 1337, &ft_putstr);
```

Chapter XXX

Exercise 26: ft_count_if

	Exercise 26
	ft_count_if
Turn-in directory :	<i>ex26/</i>
Files to turn in :	ft_count_if.c
Allowed functions :	None

- Create a function `ft_count_if` which will return the number of elements of the array that return 1, passed to the function `f`.
- Here's how the function should be prototyped :

```
int ft_count_if(char **tab, int(*f)(char*));
```

- The array will be delimited by 0.

Chapter XXXI

Exercise 27: display_file

	Exercise 27
	display_file
	Turn-in directory : <i>ex27/</i>
	Files to turn in : Makefile , and files needed for your program
	Allowed functions : close , open , read , write

- Create a program called **ft_display_file** that displays, on the standard output, only the content of the file given as argument.
- The submission directory should have a **Makefile** with the following rules : **all**, **clean**, **fclean**. The binary will be called **ft_display_file**.
- The **malloc** function is forbidden. You can only do this exercise by declaring a fixed-sized array.
- All files given as arguments will be valid.
- Error messages have to be displayed on their reserved output.

```
$> ./ft_display_file  
File name missing.  
$> ./ft_display_file Makefile  
*content of file Makefile*  
$> ./ft_display_file Makefile display_file.c  
Too many arguments.  
$>
```

Chapter XXXII

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 explicitly required by the project instructions.