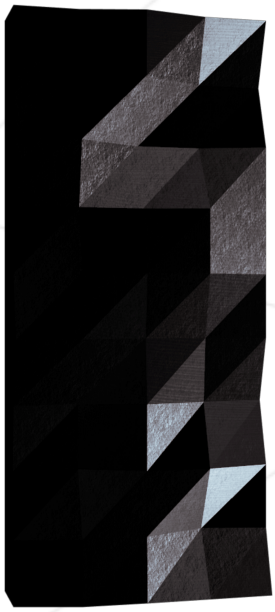

C Piscine

C 10

Summary: This document is the subject for the module C 10 of the C Piscine @ 42.

Version: 6

Contents

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.

- 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!!!

Chapter II

Foreword

Body Count is an American heavy metal band formed in Los Angeles, California, in 1990. The group is fronted by Ice-T, who co-founded the group with lead guitarist Ernie C out of their interest in heavy metal music. Ice-T took on the role of vocalist and writing the lyrics for most of the group's songs. Lead guitarist Ernie C has been responsible for writing the group's music. Their controversial self-titled debut album was released on Sire Records in 1992.

The song "Cop Killer" was the subject of much controversy. Although Sire Records' parent company, Warner Bros. Records, defended the single, Ice-T chose to remove the track from the album because he felt that the controversy had eclipsed the music itself. The group left Sire the following year. Since then, they have released three further albums on different labels, none of which have been received as commercially or critically well as their debut album.

Three out of the band's original six members are deceased: D-Roc died from lymphoma, Beatmaster V from leukemia and Mooseman in a drive-by shooting.

Click here, start it, and work... Right Now !

Chapter III

Exercise 00 : display_file

	Exercise 00
display_file	
Turn-in directory : $ex00/$	
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 an argument.
- The submission directory should contain a `Makefile` with the following rules: `all`, `clean`, `fclean`. The binary must be named `ft_display_file`.
- The `malloc` function is forbidden. You must complete this exercise by declaring a fixed-size array.
- All files given as arguments will be valid.
- Error messages must be displayed on their designated output, followed by a new line.
- If no argument is given, it should display:

```
File name missing.
```

- If more than one argument is given, it should display:

```
Too many arguments.
```

- If the file cannot be read, it should display:

```
Cannot read file.
```

Chapter IV

Exercise 01 : cat

	Exercise 01
cat	
Turn-in directory : $ex01/$	
Files to turn in : `Makefile, and files needed for your program`	
Allowed functions : `close, open, read, write, strerror, basename`	

- Create a program called `ft_cat`, which performs the same function as the system's `cat` command-line tool.
- You don't need to handle options.
- The submission directory should contain a `Makefile` with the following rules: `all`, `clean`, `fclean`.
- You may use the variable `errno` (refer to the `man` page for `errno`).
- You should read the manual pages of all the authorized functions.
- You must complete this exercise by declaring a fixed-size array. This array should have a size limited to slightly less than `30 ko`. To test this size limit, use the `ulimit` command in your shell.

Chapter V

Exercise 02 : tail

	Exercise 02
tail	
Turn-in directory : $ex02/$	
Files to turn in : `Makefile, and files needed for your program`	
Allowed functions : `close, open, read, write, malloc, free, strerror, basename`	

- Create a program called `ft_tail`, which performs the same function as the system command `tail`.
- The only option you need to handle is `-c`, but you don't need to handle the '+' or '-' signs.
- All tests will be conducted using the `-c` option.
- The submission directory should contain a `Makefile` with the following rules: `all`, `clean`, `fclean`.
- You may use the variable `errno`.

Chapter VI

Exercise 03 : hexdump

	Exercise 03
hexdump	
Turn-in directory : $ex03/$	
Files to turn in : `Makefile, and files needed for your program`	
Allowed functions : `close, open, read, write, malloc, free, strerror, basename`	

- Create a program called `ft_hexdump`, which performs the same function as the system's `hexdump` command-line tool, without redirection.
- The only option you need to handle is `-C`.
- The submission directory should contain a `Makefile` with the following rules: `all`, `clean`, `fclean`.
- You may use the variable `errno`.

Chapter VII

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.`