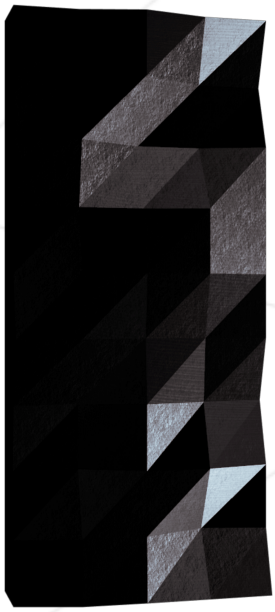

C Piscine

C 10

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

Version: 5

Contents

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 `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? 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 !!!

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 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 followed by a new line.
- If no argument is given, it should display

```
File name missing.
```

- If there is more than one argument, 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 does the same thing as the system's `cat` command-line.
- You don't have to handle options.
- The submission directory should have a `Makefile` with the following rules : `all`, `clean`, `fclean`.
- You may use the variable `errno` (check the `man` for `Errno`).
- You should read the man of all the authorized functions
- You can only do this exercise by declaring a fixed-sized array. This array will have a size limited to a little less than `30 ko`. In order to test that size-limit, use the `ulimit` command-line 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 does the same thing as the system command `tail`.
- The only option you have to handle is `-c`, but you don't need to handle '+' or '-' signs.
- all the test will be done with the `-c` option.
- The submission directory should have 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 does the same thing as the system's `hexdump` command-line without redirection.
- The only option you have to handle is `-C`.
- The submission directory should have a `Makefile` with the following rules : `all`, `clean`, `fclean`.
- You may use the variable `errno`.

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.

As these assignments are not verified by a program, feel free to organize your files as you wish, as long as you turn in the mandatory files and comply with the requirements.

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