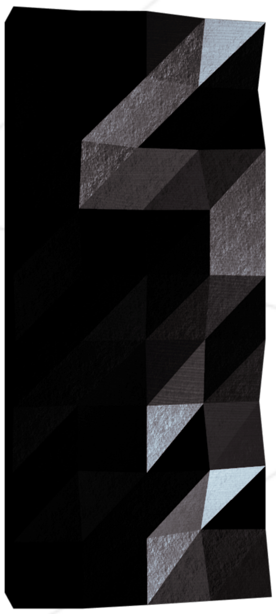

C Piscine - Dash 02

Reading files

Summary: this document is the subject for the Dash 02 module of the C Piscine @ 42Tokyo.

Contents

Chapter I

Foreword

Many programs read files as one of their very first processes.
Whether it’s to read configuration a file or load some data for initialization,
understanding and utilizing read/write is a skill you cannot go without.
In this project, implement a basic function that handles reading files.

Chapter II

Instructions

- You have exactly 3 hours to complete and submit this project. No retries.
- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up 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 for this dash will only be checked and graded by Moulinette. `NO PEER EVALUATION`.
- 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.
- 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`.
- Your program will be compiled on 42 Tokyo's iMac.
- You cannot leave any additional file in your directory than those specified in the subject.

Chapter III

Exercice 00 : read_file

	Exercise 00
read_file	
Turn-in directory : *ex*00/	
Files to turn in : `read_file.c`	
Allowed functions : `open, close, read, malloc`	

Write a function that does the following.

- Takes in a filepath as an argument.
- Opens and reads the entire file, dynamically allocating and storing the contents in an string of characters.
- Returns a pointer to the allocated char string.
- In the case of error your function should return a null pointer.
- Your function should be prototyped as below.

```
char        *read_file(char *filepath);
```