C Piscine - Dash 00

Command-line arguments

Summary: this document is the subject for the Dash 00 module of the C Piscine @ 42Tokyo.

Contents

Chapter I

Foreword

Ever wonder how options work for commands like ls?
Using command-line arguments, you can make your programs more powerful and interactive
Learn and harness the power of command-line arguments!

Chapter II

Instructions

- You have exactly 1 hour 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 : sum_args

Exercise 00
sum_args
Turn-in directory : *ex*00/
Files to turn in : `sum_args.c`
Allowed functions : `*`

- Create a program that accepts two arguments from the command line, and prints out their sum followed by a newline to standard output.
- Both arguments will be integers N where `-10000 < N < 10000`.
- In the case of unexpected arguments (greater than or less than 2 arguments, non-integer arguments, integers beyond specified range, etc), print "Error" followed by a newline to standard output.
- All standard C library functions are allowed.
- Here's an example of the program's output when executed:

```
$>./a.out 10 32
42
$>./a.out 13 -55
-42
$>./a.out
Error
$>./a.out hello world
Error
```