C - Pool - Tek1

Subject Day 12

C Pool Managers
looneytunes@epitech.eu

Contents

Instructions

- The subject may change until one hour before turn-in.
- Respect the norm takes time, but is good for you. This way your code will respect the norm since the first written line.
- We will compile with your library and your includes.
- Turn-in directory:
  `Piscine_C_J12`

💡 *Hints* `Remember it is always better to create your repository at the beginning of the day and to turn-in your work on a regular basis`

💡 *Hints* `On the instructions of each exercises, this directory is specified for every turn-in path`

Unit Tests

- It is highly recommended to test your functions when you are developing them.
- Usually, it is common to create a function named “`main`” (and a dedicated file to host it) to check the functions separately.
- Create a directory named “`tests`”.
- Create a function “`int main()`” in a file named “`tests-exercise_name.c`”, stored inside the directory “`tests`” previously created.
- According to you, this function must contains all the necessary call to “`exercise_name`” to cover all possible cases (special or regular) of the function.

Indices `The tests are not applicable on binaries.`

Exercise 1 - cat

- Write a program called `cat` which realizes the same work as the command `cat` of your system.
- You don't have to handle the options.
- The number of files given as parameters is unlimited.
- `cat` without parameters must be supported.
- This directory must have a `Makefile` with the rules `all, clean, fclean, re` and must not relink. The binary's name will be `cat`.
- You can use the errno variable. (see `man errno`).
- The `perror()` function is forbidden.
- The `malloc` function is also forbidden.
- You can only do this exercise by declaring a fixed size array. This array will have a limited size of about `30 ko`. To test this limitation, use the command `limit` in your shell.

```
$> limit stacksize 32
$> limit stacksize
stacksize 32 kbytes
$>
```

Indices `limit is an internal feature of a specific shell. Find the good one :)`

- All the files of your program and the Makefile must be into : `Piscine_C_J12/cat/`

Hints `man cat`

Exercise 2 - grep

- Write a program called `grep` which realizes the same work as the command `grep` of your system.
- You don't have to handle the options.
- The number of files given as parameters is unlimited.
- This directory must have a `Makefile` with the rules `all, clean, fclean, re` and must not relink. The binary's name will be `grep`.
- You can use the errno variable. (see `man errno`).
- The `perror()` function is forbidden.
- All the files of your program and the Makefile must be into : `Piscine_C_J12/grep/`

Hints You don't have to handle regex. We just ask a simple matching system.

```

$./grep looneytunes /etc/passwd
looneytunes:x:1000:100:looney tunes:/home/looneytunes:/bin/bash

$./grep http /etc/services
http 80/tcp
...

$./grep "application/pdf" /usr/share/misc/magic
!:mime application/pdf

$./grep http /doesnt_exist
grep: /doesnt_exist: No such file or directory

$./grep http /root
grep: /root: Permission denied
```

Hints `man grep`

That's all Folks!