B1- Unix and C Lab Seminar

B-CPE-100

Day 12

File descriptors

v2.0

{EPITECH.}

Day 12

File descriptors

binary name:	no binary
repository name:	CPool_Day12_$ACADEMICYEAR
repository rights:	ramassage-tek
language:	C
group size:	1
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

If you use your libmy for the following tasks, it must be built (if required) from the task's Makefile.
The prefered way to do this is to call the lib's Makefile from the project's Makefile.

Allowed functions: open, read, write, close

Create your repository at the beginning of the day and submit your work on a regular basis!
The delivery directory is specified within the instructions for each task.
In order to keep your repository clean, pay attention to `gitignore`.

Task 01

Cat

Write a program called **cat**, which executes the same tasks as your system's **cat** command. You do not have to handle options.
There is an unlimited number of files given as parameter.
cat without parameters must be supported.

You must deliver a **Makefile** with the following rules: **all**, **clean**, **fclean** and **re**, and it must not relink. The binary's name must be **cat**.
You may use the **errno** variable (refer to **man errno**), but **perror()** and **malloc** functions are prohibited.
This task can only be performed by declaring a fixed-size array. It will have a limited size of approximately **30 ko**. In order to test the limitation, use the command limit in your shell:

```
Terminal
~/B-CPE-100> limit stacksize 32
~/B-CPE-100> limit stacksize
stacksize 32 kbytes
```

The read of size 1 is forbidden

Delivery: CPool_Day12_$ACADEMICYEAR/cat/

Limit is an internal feature of a specific shell. Find the good one :)
man cat

Task 02

Testing cat

You must now write unit tests for all the code composing your "cat" program. We expect that most of your functions will be tested (so don't only test the final results).
Your tests will be built and executed with a rule **tests_run** that you have to add to your previous Makefile. But your tests files need to be in a directory called "tests" **at the root** of your repository. See how_to_write_unit_tests.pdf for more informations.

If you need to open a file (to obtain a file descriptor) for multiple tests, we encourage you to use fixtures to avoid code duplication.

Task 03

Grep

Write a program called **grep**, which executes the same tasks as your system's **grep** command. You do not have to handle options.
There is an unlimited number of files given as parameter.

You must deliver a **Makefile** with the following rules: **all**, **clean**, **fclean** and **re**, and it must not relink. The binary's name must be **grep**.
You may use the errno variable (refer to **man errno**), but the **perror()** function is prohibited.

For this task, malloc and free are allowed.

Delivery: CPool_Day12_$ACADEMICYEAR/grep/

You don't have to handle regex. We are only asking for a simple matching system.
man grep

The read of size 1 is forbidden

Terminal

```
~/B-CPE-100> ./grep looneytunes /etc/passwd
looneytunes:x:1000:100:looney tunes:/home/looneytunes:/bin/bash
```

Terminal

```
~/B-CPE-100> ./grep http /etc/services
http 80/tcp
...
```

Terminal

```
~/B-CPE-100> ./grep "application/pdf" /usr/share/misc/magic
!:mine application/pdf
```

Terminal

```
~/B-CPE-100> ./grep http /doesnt_exist
grep:  /doesnt_exist:  No such file or directory
```

Terminal

```
~/B-CPE-100> ./grep http /root
grep:  /root:  Permission denied
```

Task 04

Testing grep

You must now write unit tests for all the code composing your "grep" program. We expect that most of your functions will be tested (so don't only test the final results).
Your tests will be built and executed with a rule **tests_run** that you have to add to your previous Makefile. But your tests files need to be in a directory called "tests" **at the root** of your repository. See how_to_write_unit_tests.pdf for more informations.

If you need to open a file (to obtain a file descriptor) for multiple tests, we encourage you to use fixtures to avoid code duplication.