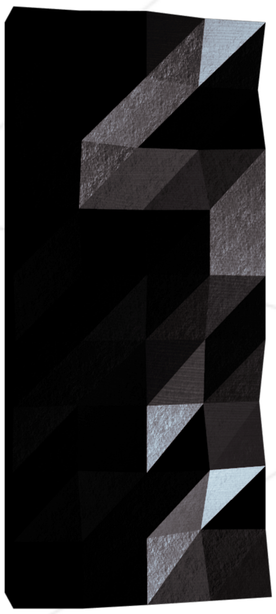

Pipex

Rush of UNIX module

Pedago Team pedago@42.fr

Summary: This project is discovery in detail and by programming a UNIX mechanism that you already know

Contents

Chapter I

Foreword

A tobacco pipe, often called simply a pipe, is a device specifically made to smoke tobacco. It comprises a chamber (the bowl) for the tobacco from which a thin hollow stem (shank) emerges, ending in a mouthpiece (the bit). Pipes can range from very simple machine-made briar models to highly prized hand-made artisanal implements made by renowned pipemakers, which are often very expensive collector's items. Pipe smoking is the oldest known traditional form of tobacco smoking.

The bowls of tobacco pipes are commonly made of briar wood, meerschaum, corncob or clay. Less common are other dense-grained woods such as cherry, olive, maple, mesquite, oak, and bog-wood. Minerals such as catlinite and soapstone have also been used. Pipe bowls are sometimes decorated by carving, and moulded clay pipes often had simple decoration in the mould.

Unusual, but still noteworthy pipe materials include gourds, as in the famous calabash pipe, and pyrolytic graphite. Metal and glass are uncommon materials for tobacco pipes, but are common for pipes intended for other substances, such as cannabis.

Now, It's your turn to make your own `pipe`..

Chapter II

Instructions

- This project will be corrected by humans only. So, feel free to organize and name your files as you wish, but within the constraints listed here.
- Your executables have to named `pipex`.
- You must render a Makefile.
- Your Makefile will compil the project, and will hold the usuals rules. He must recompil the program only if necessary. He must obviously compil your both executables.
- If you are clever, you will use your library for your project. Submit also your `libft` folder, including its own `Makefile` at the root of your repository. Your `Makefile` will have to compile the library, and then compile your project.
- Your project must be written in accordance with the Norm.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- You'll have to submit a file called `author` containing your usernames followed by a '\n' at the root of your repository:

```
$>cat -e author
xlogin$
ylogin$
$>
```

- In the mandatory part, you are allowed to use the following functions:

  - malloc.
  - free.
  - All libc syscall. Warning, not the libc functions!

- You can ask your questions on the forum, jabber, IRC, Slack, ...

Chapter III

Subject - Mandatory part

You must realize the pipex program. This program works as follows :

```
$> ./pipex file1 cmd1 cmd2 file2
```

`file1` and `file2` are the files names.
`cmd1` and `cmd2` are command shell with his parameter.

The execution of pipex program have same effect that this command shell:

```
$> < file1 cmd1 | cmd2 > file2
```

You must realize same usage which is setup by the shell (We don't want to see transitional step or cutting, all must works at same time) and .. that all ! You must make an error management, she's tested during the defense

Some examples :

- The command :

  ```
  $> ./pipex infile ``ls -l'' ``wc -l'' outfile
  ```

  Execute same thing that this command shell :

  ```
  $> < infile ls -l | wc -l > outfile
  ```

- The command :

  ```
  $> ./pipex infile ``grep a1'' ``wc -w'' outfile
  ```

  Execute same thing that this command shell :

  ```
  $> < infile grep a1 | wc -w > outfile'' en sh
  ```

Chapter IV

Subject - Bonus part

`The bonus is graduated only if your mandatory part is PERFECT. Which means it is perfectly realized, that your error management is OK, even tricky case, or bad utilisation cases. If your mandatory part isn't perfect, your bonus will be completely IGNORE.`

The only bonus accepted on this project is manage a lot of pipes at the same time.

Chapter V

Submission and peer correction

Submit your work on your `GiT` repository as usual. Only the work on your repository will be graded.