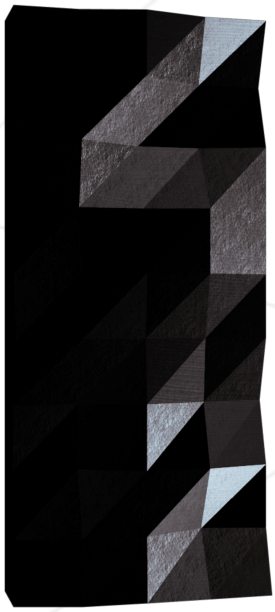

Rush UNIX Environment I

minitalk

Pedago Team pedago@42.fr

Summary: This rush is meant to make you realize a little swap program of datas using UNIX signals.

Contents

Chapter I

Préambule

here is generique lyrics of `Firefly`:

```
Take my love, take my land
Take me where I cannot stand
I don't care, I'm still free
You can't take the sky from me.

Take me out to the black
Tell them I ain't comin' back
Burn the land and boil the sea
You can't take the sky from me.

Leave the men where they lay
They'll never see another day
Lost my soul, lost my dream
You can't take the sky from me.

I feel the black reaching out
I hear its song without a doubt
I still hear and I still see
That you can't take the sky from me.

Lost my love, lost my land
Lost the last place I could stand
There's no place I can be
Since I've found Serenity

And you can't take the sky from me.
```

This rush if easier if you watched `Firefly`

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 `client` and `server`
- 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 smart and that you use your library `libft` for your project, You must copy the sources and the `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépôt de rendu. Votre `Makefile` devra compiler la librairie, en appelant son `Makefile`, puis compiler votre projet.
- 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 username 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:
  - signal
  - kill
  - getpid

  - malloc
  - free
  - pause
  - sleep
  - usleep
  - exit

- In your bonus part, you have more the right to use the following functions:

  - sigaction

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

Chapter III

Subject - Mandatory part

- You must realize a communication programe in the form of a client and a server.
- The server must be launch at first, and must show his PID after the launch.
- The client will take in parameter :
    - The server PID
    - A string to deliver
- The client must communicate to the server the string passed in parameter. Once the string is fully received, the server should display it.
- The communication between programs must be ONLY a signal help UNIX.
- The server must be able to show the string quickly. By quickly, If you think is too long, so it's certainly too long (hint : 1 second for 100 characters, it's COLOSSAL).
- Your server must be able to receive strings from multiple clients one after the other, without needing to be restart.
- You have the right to use one (and only one !) global variable by program, that you have to rigorously justify at the correction.
- You can use only two signals `SIGUSR1` and `SIGUSR2`
- The volume of data transmitted must be reasonable (hint: more than 8 signals for a single character is too much!)

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.

Below are a few interesting ideas of bonuses for you to either create or use. You can of course add your own bonuses, which will then be evaluated directly by your correctors.

- Error management of transmission and re-emission segments failures.
- Management of multi clients AT SAME TIME.
- Datas compresions
- Confirm receipt ("ping-pong")
- Optimization of CPU / memory usage.

Chapter V

Submission and peer correction

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