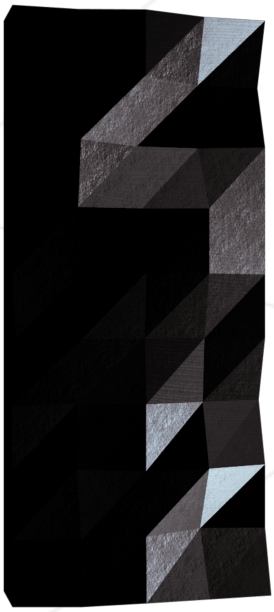

Rush Environnement UNIX I

minitalk

Pedago Team pedago@42.fr

Résumé: Ce rush a pour but de vous faire réaliser un petit programme d'échange de données utilisant les signaux UNIX.

Table des matières

Chapitre I

Préambule

Voici les paroles du générique de `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.
```

Ce rush est plus facile si vous avez regardé l'intégralité de `Firefly`

Chapitre II

Consignes

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- Vos éxécutables doivent s'appeler `client` et `server`
- Vous devez rendre un Makefile.
- Votre Makefile devra compiler le projet, et doit contenir les règles habituelles. Il ne doit recompiler le programme qu'en cas de nécessité. Il doit bien évidemment compiler vos DEUX éxécutables.
- Si vous êtes malin et que vous utilisez votre biliothèque `libft` pour votre projet, vous devez en copier les sources et le `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.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...).
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant vos logins suivi d'un '\n' :

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

- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - signal
  - kill
  - getpid
  - malloc
  - free

  - pause
  - sleep
  - usleep
  - exit

- Dans le cadre de votre partie bonus, vous avez EN PLUS le droit d'utiliser les fonctions suivantes :

  - sigaction

- Vous pouvez poser vos questions sur le forum, sur jabber, IRC, ...

Chapitre III

Sujet - Partie obligatoire

- Vous devez réaliser un programme de communication sous la forme d'un client et d'un serveur.
- Le serveur doit être lancé en premier, et doit après le lancement afficher son PID.
- Le client prendra en paramètre :
  - Le PID du serveur
  - Une chaîne de caractères à transmettre
- Le client doit communiquer au serveur la chaîne passée en paramètre. Une fois la chaîne entièrement reçue, le serveur doit l'afficher.
- La communication entre vos programmes doit se faire UNIQUEMENT à l'aide de signaux UNIX.
- Le serveur doit être capable d'afficher la chaîne rapidement. Par rapidement, on entend que si vous pensez que c'est trop long, alors c'est sûrement trop long (hint : 1 seconde pour 100 caractères, c'est COLOSSAL)
- Votre serveur doit pouvoir recevoir des chaînes de plusieurs clients à la suite, sans avoir besoin d'être relancé.
- Vous avez le droit à une (et une seule !) variable globale par programme, que vous devrez rigoureusement justifier à la correction.
- Vous ne pouvez utiliser que les deux signaux `SIGUSR1` et `SIGUSR2`
- Le volume de données transmises doit être raisonnable (hint : plus de 8 signaux pour un seul caractère, c'est trop !)

Chapitre IV

Sujet - Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est PARFAITE. On entend par là qu'elle est entièrement réalisée, que votre gestion d'erreur est au point, même dans des cas vicieux, ou des cas de mauvaise utilisation. Concrètement, si votre partie obligatoiren'est pas parfaite, vos bonus seront intégralement IGNORÉS.

Voici quelques idées de bonus intéressants à réaliser, voire même utiles. Vous pouvez évidemment ajouter des bonus de votre invention, qui seront évalués à la discrétion de vos correcteurs.

- Gestion des erreurs de transmission et ré-émission des segments ratés
- Gestion de plusieurs clients SIMULTANÉS
- Compression de données
- Accusés de réception ("ping-pong")
- Optimisation de l'usage CPU / usage mémoire.

Chapitre V

Rendu et peer-évaluation

Rendez votre travail sur votre dépot GiT comme d'habitude. Seul le travail présent dessus sera évalué en soutenance.