Pip x

Résumé:
Ce projet est l découverte en dét ils et p r l progr mm tion d'un méc nisme UNIX que vous conn issez déjà.

Version: 3

Table des matières

Chapitre I

Préambule

Cristina : `" llez danser la salsa quelque part :)"`

Chapitre II

Règles communes

Votre projet doit être écrit en C.

Votre projet doit être codé à la Norme. Si vous avez des fichiers ou fonctions bonus, celles-ci seront inclues dans la vérification de la norme et vous aurez 0 au projet en cas de faute de norme.

Vos fonctions ne doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.

Toute mémoire allouée sur la heap doit être libéré lorsque c'est nécessaire. ucun leak ne sera toléré.

Si le projet le demande, vous devez rendre un Makefile qui compilera vos sources pour créer la sortie demandée, en utilisant les flags `-Wall`, `-Wextra` et `-Werror`. Votre Makefile ne doit pas relink.

Si le projet demande un Makefile, votre Makefile doit au minimum contenir les règles `$(N ME)`, `all`, `clean`, `fclean` et `re`.

Pour rendre des bonus, vous devez inclure une règle `bonus` à votre Makefile qui ajoutera les divers headers, librairies ou fonctions qui ne sont pas autorisées dans la partie principale du projet. Les bonus doivent être dans un fichier différent : `_bonus.{c/h}`. L'évaluation de la partie obligatoire et de la partie bonus sont faites séparément.

Si le projet autorise votre `libft`, vous devez copier ses sources et son Makefile associé dans un dossier libft contenu à la racine. Le Makefile de votre projet doit compiler la librairie à l'aide de son Makefile, puis compiler le projet.

Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.

Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre III

Partie obligatoire

Nom du programme	`pipex`
Fichiers de rendu	`Makefile, *.h, *.c`
Makefile	`N ME, all, clean, fclean, re`
rguments	`file1 cmd1 cmd2 file2`

Fonctions externes autorisées	`open, close, read, write,`
	`malloc, free, perror,`
	`strerror, access, dup, dup2,`
	`execve, exit, fork, pipe,`
	`unlink, wait, waitpid`
	`ft_printf et tout équivalent`
	`que VOUS avez codé`

Libft autorisée	`Oui`
Description	`Ce projet consiste à gérer des pipes.`

Votre programme sera exécuté comme suit :

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

Il doit prendre 4 arguments :

`file1` et `file2` sont des **noms de fichier**.
`cmd1` et `cmd2` sont des **commandes shell** avec leurs paramètres.

Votre programme doit se comporter exactement comme la commande shell suivante :

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

III.1 Exemples

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

Devrait être identique à `< infile ls -l | wc -l > outfile`

```
$> ./pipex infile "grep  1" "wc -w" outfile
```

Devrait être identique à `< infile grep a1 | wc -w > outfile`

III.2 Prérequis

Votre projet doit respecter les règles suivantes :

Vous devez rendre un `Makefile` qui compilera vos fichiers sources. Il ne doit pas `relink`.

Vous devez gérer les erreurs avec du bon sens. En aucun cas votre programme ne doit quitter de manière inattendue (erreur de segmentation, erreur de bus, double free, etc.).

Votre programme ne doit pas avoir de **fuites de mémoire**.

Si vous avez le moindre doute, référez-vous à la commande shell :
`< file1 cmd1 | cmd2 > file2`

Chapitre IV

Partie bonus

Vous aurez des points supplémentaires si vous :

Gérez plusieurs *pipes*.

Ceci :

```
$> ./pipex file1 cmd1 cmd2 cmd3 ... cmdn file2
```

Devrait être identique à :

```
< file1 cmd1 | cmd2 | cmd3 ... | cmdn > file2
```

Gérez « et » quand le premier paramètre est "here_doc".

Ceci :

```
$> ./pipex here_doc LIMITER cmd cmd1 file
```

Devrait être identique à :

```
cmd << LIMITER | cmd1 >> file
```

Les bonus ne seront évalués que si la partie obligatoire est P RF ITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre V

Rendu et peer-evaluation

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

```
file.bfe:V CsSfsWN1cy33ROe SsmsgnY0o0sDMJev7zFHhw
QS8mvM8V5xQQpLc6cDCFXDWTiFzZ2H9skYkiJ/DpQtnM/uZ0
```