Mini-shell - Contre rendu

Réalisé par Quentin Carpentier & Paul-joseph Krogulec

Projet du première semestre de Master 1 Informatique dans le cadre du cours Systèmes d'Exploitation Centralisés.

1. Le projet

L'objectif du projet était de réaliser un programme C qui modélise un shell capable d'exécuter des programmes. Le projet se compose d'un dossier `src` contenant l'ensemble du code source avec des fichiers *.c* et *.h*. Au sein de ce dossier, se trouve également un sous-dossier `commands` regroupant le code pour les commandes myps, myls et mycd.

Pour une meilleure organisation dans le code, nous avons divisé celui-ci en plusieurs fichiers :

- **main.c** : le fichier qui lance le programme principal.
- **myshell.c** : le fichier qui regroupe les principales méthodes au bon déroulement du programme shell (exécution des commandes, vérifications, ...).
- **utils.c** : regroupant les fonctions utilitaires pouvant être utilisé dans plusieurs fichiers.
- **variable.c** : le fichier qui gère les variables locales du shell avec ses méthodes set/echo/unset.

Chacun de ces fichiers à une librairie locale qui lui est associée contenant les includes qui vont bien et les déclarations des fonctions.

Liens utiles

- Projet GitLab

Part de travail de chaque membre du groupe

Dans l'ensemble, nous avons bien géré la répartition de la charge de travail entre nous deux. En début de projet, nous avions listé dans notre projet Git les différentes issues possibles qui représentent la plupart des fonctionnalités. Certaines fonctionnalités comme les commandes myls/myps ou les redirections prenaient plus de temps que d'autre comme le mycd par exemple. Ce qui a parfois donné à l'un de nous deux de faire plus de fonctionnalités que l'autre.

En pourcentage, nous dirons que le part de chaque membre équivaut à : **Quentin : 55% Paul-joseph : 45%**

Compiler et lancer le projet

Pour compiler le programme, rendez dans vôtre terminal et exécuter la compilation grâce au Makefile.

```
$ make
```

Une fois le programme compiler et l'exécutable généré à la racine du projet, il suffit de lancer la commande suivante.

```
$ ./mysh
```

Note: pour nettoyer le projet, lancez la commande: `make clean`

2. Les fonctionnalités

Ci-dessous vous trouverez un tableau qui récapitule l'ensemble des fonctionnalités implémentées (ou non) dans nôtre projet.

Fonctionnalités	**Implémentée?**	**Note**	**Par qui?**
Lancement de commandes			
Séquencement	✔	le séparateur `;` doit être espacé entre deux séquences (comme dans l'exemple du sujet).	PJ
Wildcards	✔	-	Quentin
Commandes			
Changement de répertoire	✔	-	PJ
Sortie du Shell (et propagation du Ctrl-C)	—	Le Ctrl+C ne kill pas un processus en cours.	Quentin
Code de retour d'un processus	✔	-	Quentin
Lister le contenu d'un répertoire	✔	-	Quentin
Afficher l'état des processus en cours	✔	-	Quentin
Les redirections			
Les tubes	✘	trop de bugs, pas implémenté mais dispo dans une branch du Git	PJ
Redirections vers ou depuis un fichier	✔	-	PJ
Premier et arrière plans		Bcp de bugs, erreur valgrind donc pas implémentée.	PJ & Quentin
Commande myjobs	✔	implémenté mais ne sert à rien car il n'y a pas le reste.	Quentin

Fonctionnalités	Implémentée?	Note	Par qui?
Passer une commande de foreground à background et inversement	✗	pas implémentée	
Les variables			
Les variables d'environnment	✗	pas de variables d'environnement. (pas réussi)	
Les variables locales	✔	-	Quentin
Utilisation des variables	—	la partie utilisation avec les variables locales est fonctionnel mais pas avec ceux d'environnement.	Quentin

3. Les bugs

Au cours du projet, nous avons rencontré pas mal de bugs que nous avons pu résoudre. Cependant, ils restent certains bugs que nous n'avons pas pu traiter et donc qui n'ont pas été implémenté. Notamment la partie `Premier et arrière plans` que nous avons tentés d'implémenter, mais sans succès puisque cela posait beaucoup de bugs sur une large partie du projet.

Les pipes ne sont pas implémentées dans le programme principal. Après plusieurs tentatives, nous avons réussi l'implémentation des commandes avec les pipes (sans pipeline infini) mais avec beaucoup trop d'erreurs valgrind et de bugs. C'est pourquoi, nous avons décidé de ne pas les implémenter dans le programme main. Cependant, le code est disponible dans une branch à part ici.

Il en va de même pour les variables d'environnement que nous n'avons pas réussi, car nous n'arrivions pas à résoudre le problème de mémoire partagé. Mise à part ces bugs rencontrés et non implémentés donc, nous avons, durant nos tests, résolus l'ensemble des bugs possible que nous avons trouvé.

4. Petite blagounette

Où partent les étudiants de M1 pendant les vacances de Noël?

Réponse: au C-Shell. si vous rigolez, pourrions-nous avoir un point bonus?