SEC

Rapport du projet minishell

Maxime Moshfeghi

ENSEEIHT - 1SN - groupe J - 2022-2023

Table des matières

Introduction

Remarque préliminaire : Pour des raisons que j'ignore, certaines fonctionnalités sont indisponible (je n'ai jamais su résoudre l'erreur). Je reviendrais dessus en **question 6**.

ATTENTION : Pour que le minishell fonctionne, il ne faut pas utiliser le `Makefile` mais la commande :

```
gcc -Wall minishell.c readcmd.c proc.c -o minishell
```

Avec le `Makefile`, le code ne compile pas car celui si ne connait pas `action` ou encore `WCONTINUED` lors de l'utilisation des handlers.

Déroulé

Questions 1, 2 et 3

Les questions 1, 2 et 3 reprennent globalement l'implantation du minishell effectuée en début de semestre. La majeure différence est que l'on va cette fois-ci appeler la primitive `execvp`, et utiliser à la place d'un buffer une structure `cmdline` présente dans le module `readcmd.c` proposé par l'équipe pédagogique.

Question 4

La question 4 nous demande d'implémenter deux fonctions internes : `cd` et `exit`. J'ai alors décidé de créer deux sous-programmes avant le `main` : `exec_commande_interne` et `exec_commande_externe` qui me serviront respectivement à exécuter des commandes externes (avec `execvp`) et interne (comme `cd` et `exit`).

Question 5

La question 5 nous demande de gérer la mise en arrière plan d'un processus. J'avais initialement un `waitpid` dans le sous-programme `exec_commande_externe`, et cela fonctionnait correctement lorsqu'on demandait si la ligne de commande contenait une commande à exécuter en arrière plan. Cependant, en utilisant un handler pour le signal `SIGCHLD`, la mise en arrière plan fonctionnait toujours, mais la terminaison d'un processus en arrière plan provoquait un `segmentation fault` que je n'ai pas su résoudre.

Question 6

Dans le module que j'ai appelé `proc.c` (ayant une spécification contenue dans `proc.h`), j'ai implémenté une fonction qui me permettait d'afficher itérativement l'ensemble des processus lancés par le minishell. Cette procédure est exécuté en tant que commande interne avec la commande `lj`.
Cependant, le non fonctionnement partiel de la question 5 ont rendu difficile l'implémentation des autres fonctions `sj`, `bg` et `fg` qui n'ont donc pas pu être implantées.

Question 9

La redirection est fonctionnelle. Celle-ci a nécessitée la redirection de l'entrée standard et/ou de la sortie standard lorsque les attributs `in` et/ou `out` de la ligne de commande était non vide.

Conclusion

Bilan

D'un point de vue personnel, j'ai trouvé ce projet très intéressant puisqu'il permet d'appliquer rigoureusement les notions enseignées durant l'ensemble du semestre de SEC. Malheureusement, je n'ai pas pu terminé certains points du sujet correctement, par manque de recul personnel vis-à-vis des TP, et peut-être aussi par manque de pratique.

Pour la suite

Je me permets d'écrire un court paragraphe sur ce que j'étais en train de faire avant le rendu. Une fois que j'aurais résolu mon problème avec le `segmentation fault` (qui est selon moi du au `readline` après avoir posé des `printf` un peu partout pour tenter de débugger le code), j'aurais défini un handler pour les signaux `SIGINT` et `SIGTSTP` pour me permettre de tuer ou d'interrompre un processus en avant plan du mini shell (en définissant dans un premier temps le handler `sigign`). J'aurais par ailleurs aussi pu implémenté et tester l'ensemble des commandes de la question 6.

Bibliographie

[1] Equipe pédagogique, *Sujet du projet minishell*