Projet de SEC
Minishell
Rapport final

YANGOUR Cylia

Département Sciences du Numérique - Première année
2022-2023

Table des matières

1 Introduction

Ce projet a pour objectif de mettre en pratique les notions de base vues en cours, TD et TP, autour de la gestion des processus, des signaux et des E/S, dans un contexte concret. Le but final étant de développer un interpréteur de commande simplifié, offrant les services de base des shells tels que Unix.

2 Principaux choix de conception

Au cours de l'implémentation de mon minishell, j'ai du prendre certaines décisions pour répondre au mieux à la problématique initiale.
Premièrement, j'utilise une boucle do...while(1) afin d'obtenir une boucle infinie qui permet au shell de traiter les commandes sans s'arrêter (sauf interruption par exit, ctrlC, etc). J'ai également choisi d'implémenter la liste de processus sous forme d'un tableau d'enregistrement de capacité 20 (subjectif), ce qui m'a énormément simplifié, par la suite, l'implémentation des fonctions de base (Taille, CreerJob, etc).
J'ai fais le choix délicat de les implémenter dans le fichier minishell.c directement (ce qui était plus "rapide" pour moi), mais il serait préférable de créer un module.

Mon code ne comporte pas toujours les tests de code de retour des primitives telles que open ou pipe (par soucis de "clarté" et de "simplification" pour moi), mais il est évidemment très important de tester chaque appel pour assurer la cohérence du shell, et il aurait été beaucoup plus judicieux de ma part de créer une fonction qui teste automatiquement les appels, et appeler cette fonction à chaque appel de primitives Unix.

3 Avancement et difficultés

Les questions 1 à 5 ont été traitées entièrement(cf fichier minishellq5.c).
La question 6 a été partiellement traitée, (cf fichier minishell.c) les commandes lj, sj, bg fonctionnent correctement. La difficulté vient au niveau de la commande fg. Elle fonctionne pour reprendre en avant plan un processus "running", mais fonctionne mal pour reprendre un processus suspendu, et je n'arrive pas à trouver d'ou peut venir cette erreur...
Les questions 7, 8, 9, 10 et 11 ont été entièrement traitées.

4 Méthodologie de tests

4.1 Question 7 et 8

Dans un premier temps, je m'étais orientée vers la création de deux handler (handlerSIGTSTP et handlerSIGINT) et l'utilisation du masquage/démasquage de ces deux signaux pour la bonne gestion des ctrlC et ctrlZ.

Cela s'est révélé assez compliqué pour moi avec plusieurs erreurs que j'avais du mal à résoudre, j'ai donc opté pour une solution plus simple ; utiliser les handler par défaut de SIGIGN (pour le shell) et SIGDFL (pour les fils). En effet, en utilisant le handler par défaut de SIGIGN dans la boucle principale, cela permet d'éviter au shell d'être suspendu/terminé par ces signaux, en les évitant simplement.

Pour les tests, je lance des processus en avant-plan avec "sleep 30" par exemple, puis je tappe un ctrlC ou ctrlZ pour voir si l'invite de commande revient bien.
Je teste aussi directement sur le minishell ces deux combinaisons de touche, pour s'assurer qu'elles ne suspendent pas le minishell.

Pour la commande "susp", je tape "susp" dans mon shell ce qui suspend ce dernier. Puis à l'aide d'un kill -CONT pidShell, je m'assure qu'il est bien "repris".

4.2 Question 9 et 10

En utilisant les notions vues en TD et notamment en TP, la résolution de ces deux questions ne m'a pas posé de difficultés.

J'ai fais le choix de considérer que le tube se crée entre le fils et le "fils du fils" en faisant un nouveau fork dans le fils.
Pour les tests, j'ai effectué notamment les commandes du sujet et verifié dans les fichiers si les commandes s'étaient exécutées correctement.

4.3 Question 11 - fichier minishellq11.c

Cette question a été assez complexe à gérer. J'avais fais le choix pour la question 10 de créer les tubes entre le fils et le fils du fils. Or pour la question 11, qui nécessite la création de n tubes, il aurait fallu utiliser une fonction/stratégie récursive, que j'avais du mal à "visualiser". J'ai donc repris le code de la question 10 pour le modifier de telle sorte que je créeais les tubes entre plusieurs fils du même père initial.
Cette solution a été "plus simple" à implémenter pour moi, en utilisant une boucle for pour la création des n fils et des n tubes.

Pour les tests, j'ai utilisé la commande du sujet notamment, en verifiant dans les fichiers cités que la commande s'était exécutée correctement.