CAILHE Guillaume

LEFEBVRE Fabien

MINI SHELL

Réalisations effectuées

Réalisation du shell

- 2 : commandes pour réaliser le shell.
- 3 : Interprétation de commande simple : réalisée avec la commande execvp et la structure readcmd fournie.
- 4 : Redirection d'entrée sortie : nous redirigeons la commande lue vers la sortie ou l'entrée avec dup2.
- 5 : Gestion des erreurs : nous avons simplement utilisé les retours des fonctions exec et open afin de gérer les erreurs en conséquence.
- 6 et 7 : Gestion des tubes : l'étape qui nous a pris le plus temps et nous a causé le plus de problèmes. Nous avons en premier lieu réalisé un tube simple qui permettait d'exécuter 2 commandes l'une à la suite (question 6). Nous avons ensuite utilisé pour la question 7 un tableau de tubes qui contient autant de tubes qu'il y a de commandes. Chaque processus fils qui exécute une commande communique donc avec le fils « précédent » et le fils « suivant ».
- 8 : Exécution de commandes en arrière-plan : nous avons modifié la structure readcmd.
- 9 : Gestion des zombies : nous avons utilisé un traitant pour le signal SIGCHLD.

Extensions

- 1 : Gestion des signaux SIGKILL et SIGCHLD : nous avons utilisé des handlers pour ces deux signaux.

Ce qu'il reste à faire

Les fonctionnalités essentielles du minishell sont implémentées. Seules les extensions 2/3/4 restent à faire, soit, respectivement : la gestion des jobs, agir sur les commandes en arrière-plan (fg, bg), ajouter la possibilité d'utiliser le tilde, l'étoile et les variables d'environnement.

Nous n'avons pas réalisé ces dernières fonctionnalités par manque de temps. Nous avons cependant réfléchi au problème. Voici comment nous les implémenterions :

- Jobs : nous savons déjà comment exécuter des commandes en arrière-plan (question 8). Nous pourrions stocker le pid des commandes mises en arrière-plan dans un tableau. La commande jobs ne serait qu'un simple parcours de ce tableau.
- Agir sur les commandes en arrière-plan : suite logique du travail sur les jobs, ces commandes sont relativement simples à implémenter. Il faudrait utiliser le tableau des jobs précédemment alloué, et utiliser les ids des jobs (%1, %2…) en paramètre de ces nouvelles commandes. Les commandes agissant sur l'arrière-plan ou non, elles ne font qu'utiliser ce qui a été fait à la question 8.

- Environnement : tout cela peut être réalisé à l'aide des variables d'environnement et des fonctions permettant d'agir sur elles (getenv en particulier). Utiliser les bonnes variables d'environnement permet de récupérer ce qui nous intéresse (la tilde est liée à la variable home). Il faut utiliser wordexp et wordfree pour les parser efficacement.

Jeu de tests

Les jeux de tests sont tous assez clairs quant à leur utilisation : un fichier d'entêtre l'explique pour chaque fichier. Voici tout de même leurs descriptions :

- Test 1 à 4 : les tests qui nous étaient fournis. Ils testent des commandes simples et la gestion des zombies.
- Test 5 : test des redirections entrées/sorties à l'aide des commandes echo et cat.
- Test 6 : vérification de la gestion d'erreur (accès à un fichier non autorisé, tentative d'exécution d'une commande non existante).
- Test 7 : gestion des tubes. On compte avec ls, wc et grep un nombre de fichiers.
- Test 8 : exécution de top en arrière-plan.