Compte-rendu : TP Shell

Benoit PEGAZ-NOIN - Rémy DEL-GROSSO (G4)

1 Introduction

Ce TP portait sur un mini-shell à réaliser. Nous allons décrire les fonctionnalités que nous avons pu développer ou dont nous avons commencé l'implémentation. Nous expliquerons aussi les tests à effectuer pour les valider.

2 Fonctionnalités et tests

Le projet se décompose en plusieurs fichiers. Nous avons rajouté un fichier util regroupant les fonctions de comptage de pipes, de redirections d'entrée d'entrée et de sortie, etc... Le fichier handler comporte tous les traitant de signaux. Le fichier infiniteloop effectue une boucle infinie et sera utile pour nos tests d'arrière-plan.

Fonctionnalité	Code	Tests	Validation
Commande pour terminer le shell	Rajout d'une vérification en amont si la séquence[0][0] correspond à "quit"	test01.txt effectuant un quit	Validée
Interprétation de commande simple	Il y a un parcours sur chaque commande et une création d'un processus fils pour chacune d'entre elle. La commande est effectuée à l'aide de la fonction execvp.	test02.txt effectuant un ls sans argument test03.txt effectuant une succession de ls	Validée
Commande avec redirection d'entrée ou de sortie	Avant l'éxécution des commandes, il y a une vérificatioon d'une présence de redirection. Les redirections sont effectuées à l'aide de dup2 dans un descripteur de fichiers.	test05.txt effectuant un ls -l > toto et un cat < toto	Validée => Le contenu de toto est écrasé avec le résultat de ls -l et il s'affiche à l'écran
Gestion des erreurs	Les erreurs sont gérées à divers niveaux (tentative de redirection, exécution de commandes,...). La fonction perror est utilisée.	test06.txt faisant des redirections d'entrée et de sortie sur un fichier sans droit test09.txt effectuant une commande inconnue Les différentes erreurs de syntaxe (<,>,|,&)	Validée => La bonne erreur est levée pour chaque cas

Table 1: Tableau des fonctionnalités et tests

Fonctionnalité	Code	Tests	Validation
Séquence de commandes composée de deux commandes reliées par un tube	Il y a d'abord un comptage du nombre de pieps dans la séquence en amont. Puis, le bon nombre de pipes est créé. Les redirections de pipes sont effectuées dans la boucle de parcours des commandes en vérifiant si la commande actuelle est la première ou la dernière. Les pipes sont ensuite fermés avant d'exécuter la commande. Les pipes sont aussi fermés au niveau du père.	test07.txt effectuant ls | wc	Validée
Séquence de commandes composée de plusieurs commandes et plusieurs tubes	Idem que le point précédent	test08.txt effectuant cat sample | grep -v a | sort -r	Validée
Exécution de commandes en arrière-plan	Une gestion du caractère & a été rajoutée dans la fonction readcmd. Un champ bg signifiant si la commande est à effectuer en arrière-plan ou non a été rajoutée à la structure cmdline. Dans le cas où la commande est à effectuer en arrière-plan, le père n'attend pas la mort de ses fils sauf si SIGCHLD est émis.	Lancement de ./infiniteloop & dans le shell et on teste des commandes pour vérifier si elles peuvent d'exécuter	Non validée => Il semblerait que infiniteloop soit bien lancée en arrière-plan mais on arrive seulement à exécuter une commande
Gestion des zombies	Il y aune boucle d'attente de la mort des fils avant la fin.	test04.txt	Validée => Il semblerait que il n'y ait pas de processus zombies à la fin de nos exécutions

Table 2: Tableau des fonctionnalités et tests

Fonctionnalité	Tests	Validation
Changer l'état du processus en premier plan	Pas de test	Non débutée
Commande intégrée jobs	Pas de test	Non débutée
Agir sur les commandes en arrière-plan	Pas de test	Non débutée
Ajouter la possibilité d'utiliser le tilde, l'étoile et les variables d'environnement	Pas de test	Non débutée

Table 3: Tableau des fonctionnalités non implémentées et tests