Systèmes d'exploitation centralisés

Héloïse Lafargue

Département Sciences du Numérique - Première année
2021-2021

1 Avancement

Questions complètement traitées : 1à 8.
Questions partiellement traitées : 9.
Questions non traitées : 10 à 11.

2 Explications des différentes questions traitées

1. Il fallait distinguer différents cas pour les processus fils après leur création. En effet, si le PID est négatif cela signifie que le fork a échoué et donc qu'aucun processus fils n'a été créé.

2. On exécute la question 1 puis on ajoute la commande « ls », nous remarquons la présence de « sh-3.2$ » devant les différents sous-fichiers.

La ligne de commande s'affiche avant la fin de l'exécution de l'ancienne commande par le fils. En ajoutant un wait, le père attend la terminaison de tous les fils avant de re faire un tour de boucle et demande à l'utilisateur de nouveau de rentrer une commande. On obtient l'affichage correct suivant :
hlafargu@n7-ens-lnx003 : /Annee-1/s6/sec/projet ls
cmdint.c LisezMoi.md minishell6.c processus.c Q2.pdf rapport-LAFARGUE.pdf
cmdint.h minishell5.c minishell6.o processus.h Q3.c readcmd.c LisezMoi.html minishell5.o minishell.c Q1.c Q4.c readcmd.h

3. Pour régler le problème de la question 1 il fallait que le code attende la fin de la dernière commande lancée avant de passer à la lecture de la ligne suivante. J'ai donc utilisé la commande « wait » (ligne 43).

4. On ajoute les deux commandes internes « exit » et « cd ».
« cd » correspond à un changement de répertoire. Deux cas sont alors possibles : soit l'utilisateur indique dans quel répertoire il souhaite se rendre, soit il n'indique rien et retourne alors à « HOME ».
« exit » permet d'arrêter le shell.

5. Pour traiter une commande en tâche de fond, on lit d'après la documentation de « readcmd », il faut utiliser la commande « backgrounded ».

6. Pour pouvoir utiliser ces différentes commandes (« lj », « sj », « bg » et « fg »), il est nécessaire de connaitre à chaque instant l'état du processus, son PID et la commande qui lui est associée. Il faut donc créer différents types pour répertorier cela ainsi que les procédures utilisées. De plus, pour traiter la terminaison des fils au niveau du traitant, j'utilise waitpid, et non wait au niveau de la boucle principale.

7. J'ajoute une comande ctrlZ pour qu'un processus lancé depuis le minishell puisse être suspendu par la frappe de CTR-Z sur le clavier. Pour cela j'utilise SIGSTOP sur le pid en avant plan.

8. J'ai ajouté la commance ctrC pour réaliser la terminaison du processus en avant plan sans provoquer la terminaison du shell. Pour cela, j'utilise SIGINT sur le pid en avant plan pour ne pas arrêter le processus père.

9. J'ai essayé d'utiliser les tubes.

3 Architecture de l'application

Pour le minishell (minishell.c) on utilise le module "cmdint" pour l'exécution des commandes internes (lj,sj,fg,bg) et le module "processus" pour gérer les processus lancés.
Les test ont été fait avec des commandes comme ls, bg, fg...

4 Principales difficultés rencontrées

Compréhension et utilisation de ≪ readcmd ≫.
Utilisation du wait (mais compris avec les indications du terminal après compilation de mon fichier).
Gérer les changements d'états des fils.
Utilisation des tubes et traitement des redirections.