



Systèmes d'exploitation centralisée.

Rapport.

1ère Année, Département Sciences du Numérique

Chaimaa Lotfi

Mai 2020

Les traitements, de cinq premières questions étaient faciles, car j'ai déjà traité ça dans le TP1. J'ai réussi à faire un programme infini qui exécute une commande donner par l'utilisateur.

- La question 2, on remarque que le processus père n'attend pas la terminaison de l'exécution de processus fils.

- La question 3, j'ai utilisé la fonction `wait`, qui fait attendre le père jusqu'à la terminaison de l'exécution de son fils.

- La question 4, j'ai ajouté les 2 traitements, `cd`, et `exit`, en utilisant les deux fonction, `chdir` et `exit`.

- La question 5, j'ai utilisé le signal `SIGCHLD`, et après j'ai traité si le background est nul "c'est-à-dire" si la commande se finit par `&` ou pas, pour savoir si je dois faire attendre le processus père jusqu'à la terminaison de l'exécution de processus fils.

- La question 6 :

- 1— pour donner la liste des processus lancés depuis le shell et non encore terminés, avec leur identifiant propre au minishell, leur pid, leur état (actif/suspendu) et la ligne de commande lancée, il était plus simple d'utiliser un module liste, qui contient des fonctionnalités, qui peuvent servir à faciliter le travail, comme ajouter un processus, le supprimer, afficher sa position, En fin la fonction la plus importante c'est afficher la liste des processus.

- 2— Pour réaliser le traitement de la commande `stop`, j'ai utilisé le signal `SIGTSTP`. Lorsqu'on tape la commande "stop x" avec x le pid de processus qu'on veut le suspendre, on envoie un signal `SIGSTOP`, on change l'état de processus dans la liste de actif à suspendu.

- 3— Pour `fg` et `bg`, j'ai utilisé le signal `SIGCONT`, avec l'utilisation de signal `SIGCHLD`, et la fonction `waitpid`, tel que le status de cette dernier nous permette de savoir quel signal est envoyé 'soit `SIGINT`,`SIGSTOP`' et savoir la terminaison de processus fils.

- La question 7, `ctrlc`, j'ai utilisé le signal `SIGINT`, tel que si j'ai un processus en cours de l'exécution je l'arrête sans provoquer la terminaison de mon shell, et s'il n'y a aucun processus lancé je demande au utilisateur d'utiliser `exit` pour quitter le shell, sinon il entre la commande suivante.

- La question 8, j'ai ajouté les traitements associés à l'entrée standard ou la sortie standard d'une commande à un fichier, j'ai séparé les cas en traitant si on a la `'in'` ou `'out'` de commande est non nul ou les deux.

- La question 9, pour traiter deux commandes simultanément, j'ai utiliser un seul pipe, avec le fils exécute la 2ème commande, et le petit fils 'le fils de ce fils' exécute la première commande.

- La question 10, On doit exécuter 3 commandes simultanément, donc j'ai utilisé 2 pipe, et un 2 petits-fils de fils, tel que le petit fils 1 exécute la 1ère commande, et le petit fils 2 exécute la 2ème, et le fils exécute la dernière.

Aussi, j'ai essayé d'afficher le répertoire courant pour chaque commande en utilisant la fonction `getcwd`.

J'ai essayé de traiter tout les cas particuliers, comme le cas de taper entrer, donc une commande vide, ou entre une commande invalide, ou bien d'essayer d'utiliser les fonctions `stop`, `fg`, `bg` sans `PID`, ou avec un `PID` qui n'exécute par dans la liste des processus.

Une petite démonstration de fonctionnement de mon MiniShell.

Testant quelques commandes usuelles.

```
clotfi@titane:~/Annee_1/SEC/untitled$ ./main
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls
fl.c f3      hello.c liste.h main  main2 mainhello q4.c q.c      readcmd.h shsh.c va.c
f2  hello liste.c liste.o mainl main.c main.o  Q6.c readcmd.c readcmd.o test.c
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ps
  PID TTY          TIME CMD
 8844 pts/2    00:00:00 bash
18709 pts/2    00:00:00 main
18719 pts/2    00:00:00 ps
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ pwd
/home/clotfi/Annee_1/SEC/untitled
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ sec
La commande saisi est invalide.
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls -a
.  fl.c f3      hello.c liste.c liste.o mainl main.c  main.o      .nfs00000000069634cd000000001 q4.c q.c
.. f2  hello .idea liste.h main  main2 mainhello .nfs00000000069634c000000003 .nfs00000000069648c500000002 Q6.c readcmd.c
~/home/clotfi/Annee_1/SEC/untitled Monminishell$
```

Un test pour les fonctionnes bg,fg,stop, et jobs 'ou list'.

```
titane.enseiht.fr - PuTTY
f2  hello  liste.c  liste.o  main1  main.c  main.o      Q6.c  readcmd.c  readcmd.o  test.c
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ps
  PID TTY          TIME CMD
 8844 pts/2    00:00:00 bash
13449 pts/2    00:00:00 main
13463 pts/2    00:00:00 ps
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ sleep 40 &
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13492    ACTIF     sleep 40
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ stop 13492
le processus de Pid 13492 a été suspendu
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13492    SUSPENDU  sleep 40
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ fg 5
Le processus de PID 5 est introuvable.
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ fg
Vous devez entrer le PID de processus
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ fg 13492
On reprend le processus de Pid 13492 en avant plan .
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ sleep 50 &
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13587    ACTIF     sleep 50
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ stop 13587
le processus de Pid 13587 a été suspendu
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13587    SUSPENDU  sleep 50
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ bg 13587
On reprend le processus de Pid 13587 en arrière plan .
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13587    ACTIF     sleep 50
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
1       13587    ACTIF     sleep 50
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ jobs
Id      PID      STAT      COMMAND
~/home/clotfi/Annee_1/SEC/untitled Monminishell$
```

J'ai traité tout les cas exceptionnels où l'utilisateur entre une commande(stop,fg,bg) sans PID de processus, ou bien avec un PID qui n'existe pas dans la liste des processus.

Durant ce test on fait une frappe de ctrl-Z et ctrl-C au clavier.

titane.enseiht.fr - PuTTY

```
clotfi@titane:~/Annee_1/SEC/untitled$ ./main
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ sleep 50
^Z~/home/clotfi/Annee_1/SEC/untitled Monminishell$ list
Id      PID      STAT      COMMAND
1       14292     SUSPENDU  sleep 50
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ sleep 50
^C~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ^Z
Pour quitter utilisez 'exit' sinon tapez la commande que vous voulez faire.
^C
Pour quitter utilisez 'exit' sinon tapez la commande que vous voulez faire.
exit
clotfi@titane:~/Annee_1/SEC/untitled$
```

On remarque que la frappe de ctrl-C au clavier se traduit par l'envoi à mon minishell du signal SIGINT. La réception de ce signal ne provoque pas la terminaison de mon minishell, ni celle de mes processus en arrière-plan, mais il amène la terminaison du processus en avant-plan (processus courant).

Testant le cas où on compose des commandes en les reliant par un tube. D'autre part, on étend la fonctionnalité précédent en offrant la possibilité d'enchaîner une séquence de filtres liés par des tubes, de sorte à obtenir un traitement en pipeline.

```
clotfi@titane:~/Annee_1/SEC/untitled$ ./main
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls
f1.c f3      hello.c liste.h main  main2  mainhello q4.c q.c      readcmd.h shsh.c va.c
f2     hello liste.c liste.o main1  main.c main.o   Q6.c readcmd.c readcmd.o test.c
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls | wc -l
23
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls | grep main
main
main1
main2
main.c
mainhello
main.o
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls | grep main | wc -l
6
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ ls | grep f | wc -l
3
~/home/clotfi/Annee_1/SEC/untitled Monminishell$ █
```

On remarque le bon fonctionnement de ces commandes.

Conclusion : Il était très amusant de reconstruire ce Minishell.