



Projet SEC - Mini shell

Bonetto Tom

Département Sciences du Numérique - Première année
2021-2022

Question 1 : Question entièrement traitée, pour ce qui est du code j'ai décidé de faire des fonctions pour lire la commande, exécuter la commande ou encore afficher le prompt. Ce qui permet d'avoir un Main plus lisible avec seulement quelques appels de fonctions.

```
/* Exécution du mini shell */
int main() {
    initialiser_shell();
    while(1) {
        affichage_terminal();
        lire_commande();
        executer_commande();
    }
    return EXIT_SUCCESS;
}
```

Ne connaissant pas à l'avance le nombre d'arguments de la commande entrée par l'utilisateur, j'ai décidé d'utiliser `execvp` plutôt que la primitive `execlp` qui est moins pratique dans notre cas

Question 2 : Question entièrement traitée, voir Q2.pdf

Question 3 : Question entièrement traitée, pour que le père attende que l'exécution de son fils soit terminée avant de continuer sa propre exécution, j'ai utilisé la primitive `waitpid` plutôt que `wait` car celle-ci est plus riche et peut-être utilisée de façon non bloquante.

Question 4 : Question entièrement traitée, peu de chose à dire là-dessus si ce n'est l'utilisation de la primitive `chdir` pour que le `cd` soit fonctionnel, là où `execvp` n'effectue pas réellement ce qui est attendu de la commande `cd`. Pour tester, j'ai simplement parcouru les différents répertoires :

```
Terminal
Fichier  Editer  Affichage  Rechercher  Terminal  Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ cd ..
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet]$ cd ../../..
[tbonetto@mini-shell ~/home/tbonetto]$ cd Annee_1
[tbonetto@mini-shell ~/home/tbonetto/Annee_1]$ cd SEC/projet
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet]$ cd fournitures
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ exit
[tbonetto@n7-ens-lnx082 ~/Annee_1/SEC/projet/fournitures]$
```

Question 5 : Question entièrement traitée, j'ai ajouté au père une condition qui lui permet de ne pas attendre son fils quand celui-ci exécute une commande en fond de tâche. Pour tester je lance une commande en fond de tâche puis lance une commande en avant-plan pour regarder si elle s'exécute :

```
Terminal
Fichier  Editer  Affichage  Rechercher  Terminal  Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ &sleep 10000
=== Commande en tache de fond
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ date
mer. 25 mai 2022 21:56:51 CEST
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```

Question 6 : Question entièrement traitée, pour avoir un suivi des processus n'ont fini, il a fallu créer une structure de données qui permet de les lister. Encore une fois, pour une question de lisibilité j'ai préféré encapsuler cette structure dans un fichier à part (listeProcessus.c) et importer uniquement l'interface dans le code du minishell. Pour ce qui est de la structure en elle-même je me suis grandement inspiré de ce que l'on faisait en PIM au semestre 5 pour la création de tableaux ou listes chaînées.

En ce qui concerne la mise en pratique des commandes : lj, sj, bg, fg, j'ai associé au signal SIGCHLD un handler qui s'occupe de modifier les états des processus et de les retirer de la liste quand ceux-ci sont terminés.

J'ai finalement réussi à corriger le problème que j'avais mentionné lors du rendu intermédiaire, la commande fg fonctionne bien même sur un processus qui a été suspendu avec Ctrl-Z.

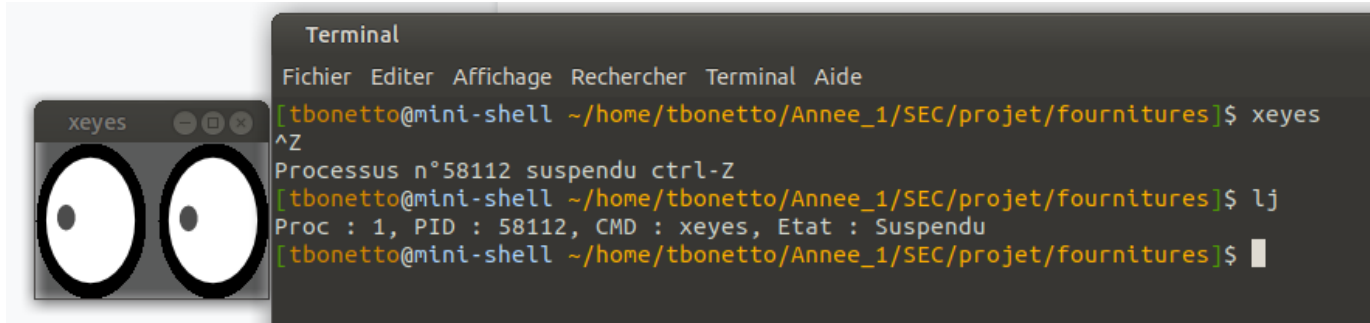
N'étant pas très à l'aise avec la notion de pointeurs en C, j'ai rencontré des difficultés pour la création des fonctions de ma liste de processus, notamment des erreurs de segmentation liées à des problèmes de pointage. Mais avec un peu d'effort on arrive à obtenir ce que l'on veut.

Pour tester j'ai lancé des processus et j'ai essayé les différentes commandes : lj, fg ,bg, sj.

```
Terminal
Fichier  Editor  Affichage  Rechercher  Terminal  Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ &xeyes
=== Commande en tache de fond
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 72185, CMD : xeyes, Etat : Actif
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ &sleep 1000
=== Commande en tache de fond
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 72185, CMD : xeyes, Etat : Actif
Proc : 2, PID : 72202, CMD : sleep, Etat : Actif
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ sj 1
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 72185, CMD : xeyes, Etat : Suspendu
Proc : 2, PID : 72202, CMD : sleep, Etat : Actif
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ sj 2
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ bg 1
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 72185, CMD : xeyes, Etat : Actif
Proc : 2, PID : 72202, CMD : sleep, Etat : Suspendu
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ fg 2
^Z
Processus n°72202 suspendu ctrl-Z
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 72185, CMD : xeyes, Etat : Suspendu
Proc : 2, PID : 72202, CMD : fg, Etat : Suspendu
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ □
```

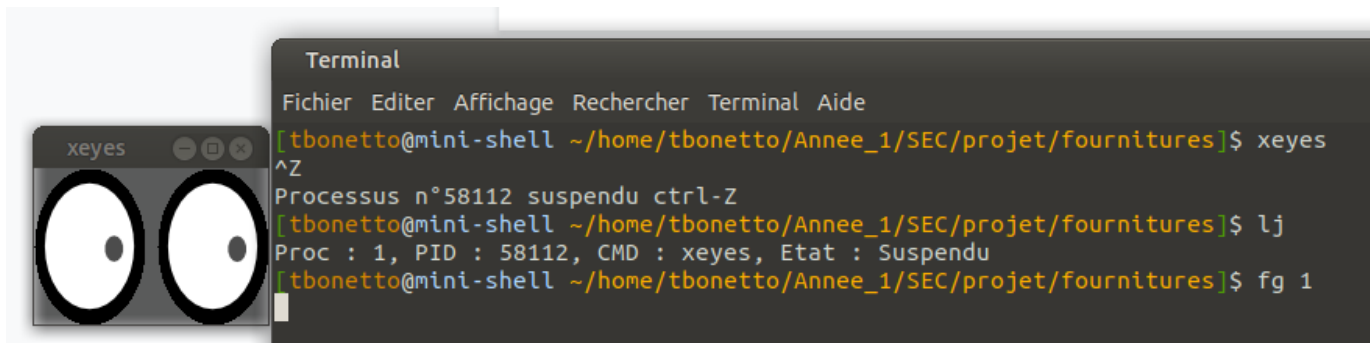
Question 7 : Question entièrement traitée, j'ai décidé de redéfinir un handler pour le Ctrl-Z car celui par défaut n'effectue pas du tout le résultat que j'attendais. Pour tester, je lance un processus avec la commande xeyes par exemple puis j'effectue un Ctrl-Z, je vérifie que le processus a été ajouté dans la liste et qu'il est bien suspendu (dans le cas de xeyes, suspendu veut dire que les yeux ne suivent plus le curseur de la souris).

Avant de suspendre xyes, j'ai orienté les yeux vers la gauche. Au moment de ma capture d'écran ma souris se trouvait à droite de la fenêtre xeyes, mais les yeux restent fixés à gauche, selon montre que le processus a bien été mis en suspend.



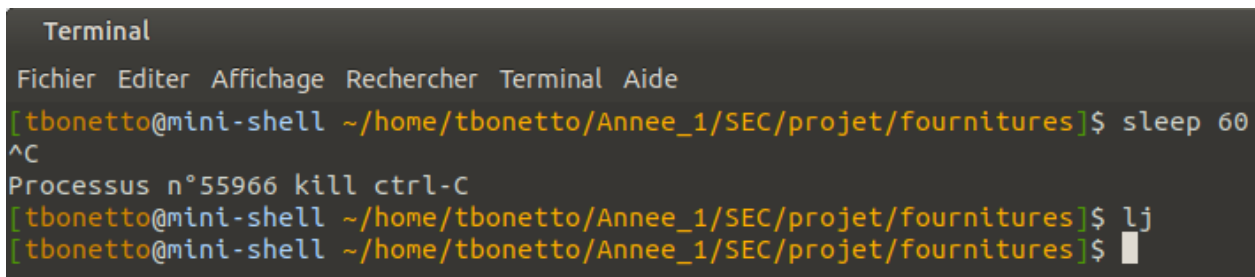
```
Terminal
Fichier Editer Affichage Rechercher Terminal Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ xeyes
^Z
Processus n°58112 suspendu ctrl-Z
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ l j
Proc : 1, PID : 58112, CMD : xeyes, Etat : Suspendu
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```

Je vérifie également qu'après avoir ramené le processus en avant plan avec fg il fonctionne correctement.



```
Terminal
Fichier Editer Affichage Rechercher Terminal Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ xeyes
^Z
Processus n°58112 suspendu ctrl-Z
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ l j
Proc : 1, PID : 58112, CMD : xeyes, Etat : Suspendu
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ fg 1
```

Question 8 : Question entièrement traitée, rien de très intéressant à dire sur cette question, j'ai également procédé par la réalisation d'un handler qui effectue le comportement attendu d'un Ctrl-C. Pour tester, je lance une commande sleep et effectue un Ctrl-C avant la fin de son exécution, j'affiche ensuite la liste des processus pour vérifier que celui-ci n'est pas dedans.



```
Terminal
Fichier Editer Affichage Rechercher Terminal Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ sleep 60
^C
Processus n°55966 kill ctrl-C
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ l j
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```

Pour tester le Ctrl-Z et le Ctrl-C j'ai également utilisée la commande ping et j'effectue un ping loopback (je me ping moi même pour voir les message affichés au fur et mesure), on voit que suite au Ctrl-Z les messages du ping s'arrêtent et reprennent uniquement après le fg. Une fois Ctrl-Z effectuer il n'y a plus le processus.

```
Terminal
Fichier Editor Affichage Rechercher Terminal Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 octets de 127.0.0.1 : icmp_seq=1 ttl=64 temps=0.012 ms
64 octets de 127.0.0.1 : icmp_seq=2 ttl=64 temps=0.029 ms
64 octets de 127.0.0.1 : icmp_seq=3 ttl=64 temps=0.021 ms
^Z
Processus n°61535 suspendu ctrl-Z
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
Proc : 1, PID : 61535, CMD : ping, Etat : Suspendu
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ fg 1
64 octets de 127.0.0.1 : icmp_seq=4 ttl=64 temps=0.016 ms
64 octets de 127.0.0.1 : icmp_seq=5 ttl=64 temps=0.025 ms
64 octets de 127.0.0.1 : icmp_seq=6 ttl=64 temps=0.029 ms
64 octets de 127.0.0.1 : icmp_seq=7 ttl=64 temps=0.019 ms
^C
Processus n°61535 kill ctrl-C

--- statistiques ping 127.0.0.1 ---
7 paquets transmis, 7 reçus, 0 % paquets perdus, temps 12260 ms
rtt min/moy/max/mdev = 0,012/0,021/0,029/0,006 ms
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ lj
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```

Question 9 : Question entièrement traitée, pour la redirection d'entrée, j'ouvre le fichier en lecture et pour la redirection de sortie j'ouvre le fichier en écriture avec l'option O_CREAT qui génère le fichier si celui-ci n'existe pas. Pour tester la redirection de sortie, j'exécute un cat sur un fichier source et écris le résultat dans un fichier destination, le contenu de source est alors écrit dans le fichier destination. Pour la redirection d'entrée, j'utilise la commande *sort* qui va s'exécuter sur le contenu du fichier contenant le résultat du cat.

Contenu du fichier source :

```
src x
1 "World"
2 "Hello"
```

Contenu du fichier destination après exécution du cat :

```
dest x
1 "World"
2 "Hello"
```

Affichage du sort dans la console :

```
Terminal
Fichier  Editor  Affichage  Rechercher  Terminal  Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ cat src > dest
=== Redirection de la sortie : dest
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ sort < dest
=== Redirection de l'entrée : dest
"Hello"
"World"
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```

Question 10 et 11 : Questions entièrement traitées, je réunis ces deux questions ensemble car j'ai directement traité la question 11 qui englobe la question 10. Rien à ajouter si ce n'est qu'au vu des attentes il me semblait logique de procéder par programmation récursive pour répondre à ce problème. Je me suis inspiré du TP4 pour la réalisation de la fonction qui permet d'avoir un traitement en pipeline et j'ai également testé son fonctionnement avec la commande `who | grep nom_utilisateur | wc -l`.

```
Terminal
Fichier  Editor  Affichage  Rechercher  Terminal  Aide
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ who | grep tbonetto
tbonetto          2022-05-25 17:56 (172.20.16.136:100)
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$ who | grep tbonetto | wc -l
1
[tbonetto@mini-shell ~/home/tbonetto/Annee_1/SEC/projet/fournitures]$
```