Yael Gras 1SN-N

Rapport final
Minishell

I - Questions traitées

Dans ce minishell, toutes les questions ont été traitées.

II - Architecture

L'architecture de l'application se compose en 2 composants : le module readcmd et le fichier principal minishell.c. Ensuite toutes les commandes particulières sont gérées grâce à des fonctions dans le fichier principal, en particulier les fonctions de gestion des processus en arrière plan. De plus, les traitants des signaux sont aussi dans le fichier principal.

III - Choix de conception

Comme dit dans la note du rendu intermédiaire, j'ai pris le choix de faire une liste statique pour enregistrer les processus car je n'ai pas réussi à les enregistrer dans une liste dynamique au moment de la conception de la gestion des processus. Ensuite pour la gestion des pipes j'ai choisi de faire une boucle qui itère sur les différentes commandes pour créer des pipes qui les relient. Pour finir le choix de tout faire dans un seul fichier est dû à la taille du projet, il ne m'a pas semblé nécessaire de faire plusieurs modules car c'est selon moi encore clair dans un seul fichier.

IV - Tests

La méthodologie de test adoptée est de tester à chaque question, chaque ajout de fonction avant de passer à la suite. J'ai donc commencé par des tests simples puis après je testais un enchaînement de commande qui pouvait poser un problème dans la conception choisie pour tester qu'avec les modifications faites ne causait pas de problèmes sur les parties faites précédemment. Vous trouverez des exemples de tests sur les pages suivantes.

```
minishell >>> sleep 2 &
[2] : 203875

minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE
 [1] 203839   ACTIF    ./sleep 10
 [2] 203875   ACTIF    ./sleep 2

minishell >>>
[2] Fini : sleep 2

minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE
 [1] 203839   ACTIF    ./sleep 10

minishell >>>
[1] Fini : sleep 10

minishell >>> sleep 10 &
[1] : 203982

minishell >>> sleep 20 &
[2] : 203996

minishell >>> sleep 5 &
[3] : 204021

minishell >>> sleep 30
[1] Fini : sleep 10

minishell >>> &
[1] : 204072

minishell >>>
[3] Fini : sleep 5

minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE
 [1] 204072   ACTIF    ./sleep 30
 [2] 203996   ACTIF    ./sleep 20

minishell >>>
[2] Fini : sleep 20

minishell >>>
[1] Fini : sleep 30
```

Test de mise de plusieurs processus en arrière plan

```
minishell >>> sleep 10 &
[1] : 204869

minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE
 [1] 204869   ACTIF    ./sleep 10

minishell >>> sj 1
[1] : 204869 Suspendu

minishell >>>
minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE
 [1] 204869  SUSPENDU  ./sleep 10

minishell >>> bg 1
[1] Reprise en background : sleep 10

minishell >>>
minishell >>>
[1] Fini : sleep 10

minishell >>> lj
-ID- --PID-- -STATUS-   COMMANDE

minishell >>>
```

Test de suspension et de remise en marche d'un processus en arrière-plan (sleep se termine à la relance du processus mais c'est dû à la commande sleep)

```
minishell >>> cat < ficout.txt
a.out
ficout.txt
fournitures
main
minishell
minishell5.c
minishell6.c
minishell8.c
minishell9.c
minishellfin.c
minishell_sans_wait.c
readcmd.c
readcmd.h
Rendu projet

minishell >>> ficout.txt | grep *.c

minishell >>> ficout.txt | wc -l
0

minishell >>> cat ficout.txt | wc -l
14

minishell >>> cat < ficout.txt | grep *.c | wc -l
0

minishell >>> cat < ficout.txt | grep .c
ficout.txt
minishell5.c
minishell6.c
minishell8.c
minishell9.c
minishellfin.c
minishell_sans_wait.c
readcmd.c
readcmd.h

minishell >>> ls
 a.out   ficout.txt   fournitures   main   minishell   minishell5.c   minishell6.c   minishell8.c   minishell9.c   minishellfin.c   minishell_sans_wait.c   readcmd.c   readcmd.h  'Rendu projet'

minishell >>>
```

Test de la gestion des pipes et de la redirection en sortie, la redirection de l'entrée a été utilisé pour remplir le fichier ficout.txt par la sortie de la commande ls