Rapport minishell

MARTIN Nolann

Projet de première année de SN

Table des matières

TP 1

Dans ce premier TP nous avons fais la découverte de l'interface qui nous a été fourni affin de nous familiariser et commencer à coder ce minishell en C, au début de la première séance, quand nous tapions une commande, celle-ci afficher simplement la dites commande sans réelement l'éxecuter. Ensuite nous avons donc fais pour que la commande se réalise et pas simplement s'écrive, en voici un exemple avec la commande « ls » :

```
> ls
commande : ls
Copier.c  Makefile   minishell.c  readcmd.c  readcmd.o   test_readcmd.c
Copier.o  minishell  minishell.o  readcmd.h  source.txt
```

Nous pouvons aussi le voir avec la commande « pwd » :

```
> pwd
commande : pwd
/home/martin/Ecole/N7/1A/S6/SEC/Projet/minishell
```

Nous nous sommes rendu compte que le minishell n'attendais pas la fin de la commande pour afficher le prompt, nous avons donc modifié le code pour que le prompt ne s'affiche qu'une fois la commande terminée. Ce qui fait que si nous faisons un

```
sleep 10
```

la lecture de ce qui sera écrit dans le terminal sera lu qu'aprés les 10 secondes. En voici un exemple avec la commande « sleep 10 » et la commande « ls » :

```
> sleep 10
commande : sleep 10
ls

processus 9770 terminé
> commande : ls
Copier.c  Makefile   minishell.c  readcmd.c  readcmd.o   test_readcmd.c
Copier.o  minishell  minishell.o  readcmd.h  source.txt
```

Nous avons aussi par la suite ajouter la possibilité de mettre la commande en arrière plan en ajoutant un « & » à la fin de la commande, en voici un exemple avec la commande « sleep 10 & » :

```
> sleep 10 &
commande : sleep 10
[1] 10174
>
```

Nous avons directement récupérer la main et nous pouvons donc tapper d'autres commandes sans se soucier de l'attente de la fin de la commande en arrière plan. en voici un exemple avec la commande « ls » :

```
> sleep 10 &
commande : sleep 10
[1] 10346
> ls
commande : ls
Copier.c  Makefile   minishell.c  readcmd.c  readcmd.o   test_readcmd.c
Copier.o  minishell  minishell.o  readcmd.h  source.txt

processus 10347 terminé
>
processus 10346 terminé
```

et la commande `sleep 10` ayant le PID 10346 se termine bien aprés la commande ls.

TP 2

Dans ce tp nous avons vu comment manipuler plus en détail les PIDet nous avons mis des messages quand un fils se termine, est tué ou est suspendu, en voici un exemple avec la commande « sleep 10 » :

```
> sleep 10
commande : sleep 10

processus 13316 terminé normalement
>
>
> sleep 10
commande : sleep 10
^Z
processus 13320 mis en pause
>
> sleep 10
commande : sleep 10
^C
processus 13324 tué
>
```

Où nous voyons bien que le processus 13316 se termine normalement, le processus 13320 est mis en pause et le processus 13324 est tué.

TP 3

Dans ce tp nous nous sommes attardés sur le masquage des signaux, c'est à dire que nous ne faisons d'action sur un signal que si il existe et qu'il et en avant plan, sinon nous ne faisons rien, voici un démonsration avec la commande « sleep 10 » :

```
> sleep 10
commande : sleep 10
^C
processus 14494 tué
> ^C
Aucun processus en avant plan
```

Nous voyons bien que le processus 14494 est tué et que si nous faisons un ^C sans qu'il y ai de processus en avant plan, rien ne se passe. Nous avons seulement afficher un message d'erreur.

TP 4

Dans ce tp, nous avons vu comment rediriger les entrées et sorties standard, en voici un exemple avec la commande « ls > test.txt », nous lirons le contenu du fichier ainsi créer avec la commande « cat test.txt »:

```
> ls > test.txt
commande : ls

processus 14783 terminé normalement
> cat test.txt
commande : cat test.txt
Copier.c
Copier.o
Makefile
minishell
minishell.c
minishell.o
readcmd.c
readcmd.h
readcmd.o
source.txt
test_readcmd.c
test.txt

processus 14785 terminé normalement
>
```

et nous pouvons mettre un nouveau fichier en entrée standard :

```
> wc -l < test.txt
commande : wc -l
12

processus 14852 terminé normalement
```

nous voyons bien que le fichier test.txt est bien pris en entrée standard de la commande wc -l qui compte le nombre de ligne du fichier.

TP 5

Dans ce tp nous avons vu comment gérer les pipes, en voici un exemple avec la commande « ls | wc -l » :

```
> ls | wc -l
commande : ls

processus 15073 terminé normalement
commande : wc -l
12

processus 15074 terminé normalement
>
```

Nous avons les 12 lignes du ls qui sont bien compté par wc -l de la sortie de ls.

Nous pouvons aussi tester de mettre plusieurs pipes, en voici un exemple avec la commande « ls | wc -l | wc -l » :

```
> ls | wc -l | wc -l
commande : ls

processus 15210 terminé normalement
commande : wc -l

processus 15211 terminé normalement
commande : wc -l
1

processus 15212 terminé normalement
```

Nous pouvons aussi mettre un fichier en entrée standard, en voici un exemple avec le fichier « minishell.c » et wc -l :

```
> wc -l | wc -l < minishell.c
commande : wc -l

processus 15252 terminé normalement
commande : wc -l
1

processus 15253 terminé normalement
```

Bilan

Nous avons donc vu comment gérer les commandes simples, les commandes en arrière plan, les signaux, les redirections et les pipes. Nous avons donc un minishell qui est fonctionnel et qui peut gérer des commandes simples et des commandes plus complexes.