Rapport projet Minishell

Maxime Laurent

May 2024

Table des matières

1 Introduction

L'objectif de ce projet est de développer un shell simplifié qui démontre quelques concepts fondamentaux tels que la gestion des processus, la gestion des signaux et les opérations d'E/S ainsi que la gestion des tubes et des redirections. Ce rapport décrit l'implémentation, les choix de conception et les fonctionnalités du minishell.

2 Lancement commande élémentaire

2.1 Question 1

Pour exécuter une commande élémentaire on utilise la primitive *execvp*.

```
mlt6297@maupassant:~/Documents/Annee_1/SEC/Projet/minishell$ ./minishell
> ls
 Makefile   minishell   minishell.c   minishell.o   readcmd.c   readcmd.h   readcmd.o   test_readcmd.c  'TP 5'
> echo toto
toto
>
```

Figure 1 –

3 Synchronisation entre le shell et ses fils

3.1 Question 2

On pourrait appelé la fonction *cat* et directement appelé une nouvelle fois *cat*, le résultat du premier appel se mélengera avec le deuxième appel de *cat*. Il y a donc un chevauchement des sorties.

3.2 Question 3

Il nous suffit d'attendre que l'enfant envoie un signal de fin. On utilise la primitive *pause* pour bloquer le minishell en attendant que le fils ait fini.

3.3 Question 4

On utilise la primitive *pause* uniquement si la commande est en avant plan.

```
> sleep 50 &
commande en tâche de fond
> sleep 10
```

FIGURE 2 –

4 Gestion des processus lancés depuis le shell

4.1 Question 5

Je n'ai pas eu le temps de traiter cette question.

5 Contrôle des processus aen avant-plan via le terminal

5.1 Question 6 et 7

Le shell gère $SIGINT$ (Ctrl+C) et $SIGTSTP$ (Ctrl+Z) pour gérer les processus en avant-plan sans terminer le shell. Pour cela, on définit de nouveaux traitants de signaux pour $SIGCHLD$, $SIGINT$ et $SIGTSTP$ pour gérer le contrôle des processus de manière appropriée.

```
> sleep 10
^C>
>
> sleep 50
^Z>
>
```

FIGURE 3 –

6 Gestion des redirections

6.1 Question 8

Le minishell supporte les redirections d'entrée et de sortie en utilisant $<$ et $>$. Pour les redirections je me suis servi de la primitive *dup*2 qui permet de changer l'entrée ou la sortie de nos commandes.

```
mlt6297@maupassant:~/Documents/Annee_1/SEC/Projet/minishell$ ./minishell
> touch toto
> echo Hello
Hello
> echo Hello > toto
> cat toto
Hello
> touch tata
> cat < toto
Hello
>
```

FIGURE 4 –

7 Tubes

7.1 Tubes simples et Pipelines

Pour l'implémentation des tubes et du pipelines on utilise la primitive *dup*2 permettant de changer l'entrée et la sortie de nos commandes. On va traiter de manière séparée la première et la dernière commande du pipeline et pour celles du "milieu" on utilise *dup*2 ainsi que des tubes simples pour que la sortie de la commande n devienne l'entrée de la commance $n+1$ et ainsi créer un pipeline.

```
mlt6297@maupassant:~/Documents/Annee_1/SEC/Projet/minishell$ ./minishell
> ls | wc -l
11
> cat minishell.c | grep main | wc -l
1
>
```

FIGURE 5 –