



Rapport du projet Systèmes d'Exploitation Centralisés

par Félix Foucher de Brandois

Département Sciences du Numérique - Première année
2022-2023

Table des matières

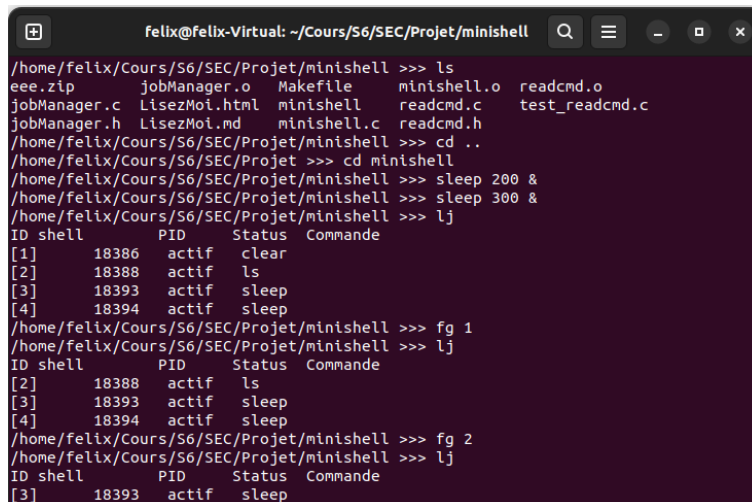
1	Introduction	3
2	Questions traitées	3
3	Questions partiellement traitées ou non traitées	3
4	Architecture de l'application	4
5	Choix et spécificités de conception	4
6	Méthodologie de tests	4

1 Introduction

Le projet "Minishell" consiste en la création d'un interpréteur de commandes minimaliste en utilisant le langage de programmation C. L'objectif est de développer un programme qui permet d'exécuter des commandes et de gérer des processus en offrant des fonctionnalités telles que la gestion des jobs, les redirections, les tubes et les pipelines.

2 Questions traitées


Les questions de 1 à 6 sont complètement traitées. Le minishell conçu permet de prendre en charge des commandes simples mais également des commandes internes comme le changement de répertoire, et la gestion de jobs (lj, sj, bg, fg, susp).



```
felix@felix-Virtual: ~/Cours/S6/SEC/Projet/minishell
/home/felix/Cours/S6/SEC/Projet/minishell >>> ls
eee.zip      jobManager.o  Makefile      minishell.o   readcmd.o
jobManager.c LisezMoi.html minishell      readcmd.c     test_readcmd.c
jobManager.h LisezMoi.md   minishell.c   readcmd.h
/home/felix/Cours/S6/SEC/Projet/minishell >>> cd ..
/home/felix/Cours/S6/SEC/Projet/minishell >>> cd minishell
/home/felix/Cours/S6/SEC/Projet/minishell >>> sleep 200 &
/home/felix/Cours/S6/SEC/Projet/minishell >>> sleep 300 &
/home/felix/Cours/S6/SEC/Projet/minishell >>> lj
ID shell      PID      Status  Commande
[1]          18386    actif   clear
[2]          18388    actif   ls
[3]          18393    actif   sleep
[4]          18394    actif   sleep
/home/felix/Cours/S6/SEC/Projet/minishell >>> fg 1
/home/felix/Cours/S6/SEC/Projet/minishell >>> lj
ID shell      PID      Status  Commande
[2]          18388    actif   ls
[3]          18393    actif   sleep
[4]          18394    actif   sleep
/home/felix/Cours/S6/SEC/Projet/minishell >>> fg 2
/home/felix/Cours/S6/SEC/Projet/minishell >>> lj
ID shell      PID      Status  Commande
[3]          18393    actif   sleep
```

3 Questions partiellement traitées ou non traitées

A partir de la question 6, j'ai eu une erreur dans la compilation :



```
felix@felix-Virtual: ~/Cours/S6/SEC/Projet/minishell$ make
gcc -Wall -Wextra -std=c99 -c -o minishell.o minishell.c
In file included from minishell.c:9:
/usr/include/x86_64-linux-gnu/bits/sigaction.h:46:5: error: unknown type name '_sigset_t'
   46 |     __sigset_t sa_mask;
      |
```

Malgré mes nombreuses recherches sur internet, je n'ai pas trouvé de moyens pour la contourner. J'ai donc utilisé un compilateur différent de gcc. Le code pour les questions de 7 à 11 compile sans warnings avec l'option -Wall.

Les questions 7, 8 et 9 ont été traitées avec ce nouveau compilateur et fonctionnent bien.

4 Architecture de l'application

Pour ce Minishell, afin de pouvoir tester séparément la gestion de la liste des jobs du programme principal, j'ai créé le module jobManager composé de sa spécification jobManager.h et du corps du module jobManager.c. Dans ce module, la liste des jobs est un tableau. C'est aussi dans ce module que les commandes lj, sj, fg et bg sont traitées, respectivement grâce aux procédures listJobs, stopJob, backgroundJob et foregroundJob. Certaines de mes méthodes sont liées à ces procédures, c'est donc également dans ce module jobManager que j'ai implanté les méthodes addJob et removeJob et getPid qui permettent d'ajouter et de supprimer un processus. Ainsi, le fichier minishell.c utilise les modules readcmd et jobManager.

5 Choix et spécificités de conception

Le code utilise des structures de données appropriées pour stocker les informations sur les commandes, les jobs et les processus. Il utilise des fonctions modulaires pour effectuer des opérations spécifiques telles que l'exécution des commandes internes, la gestion des jobs, la gestion des signaux, etc. Le code utilise également les appels système appropriés pour effectuer des opérations d'entrée/sortie, la gestion des processus, la manipulation des signaux, etc.

6 Méthodologie de tests

Voici les tests réalisés afin de justifier le bon fonctionnement du Minishell.

1. Commandes de base

J'ai testé des commandes simples sur le Minishell.

```
/home >>> ls
a.out  jobManager.c  jobManager.h  main.c  readcmd.c  readcmd.h  test
/home >>>
```

On observe bien que les commandes ont été effectuées.

2. Commandes internes

On teste les commandes cd et exit :

```
/home >>> ls
a.out  jobManager.c  jobManager.h  main.c  readcmd.c  readcmd.h  test
/home >>> cd ..
/ >>> ls
bin  dev  home  lib32  libx32  mnt  proc  run  script  sys  usr
boot  etc  lib  lib64  media  opt  root  sbin  srv  tmp  var
/ >>> cd home
/home >>> ls
a.out  jobManager.c  jobManager.h  main.c  readcmd.c  readcmd.h  test
/home >>> exit
Sortie du shell...
```

Le répertoire courant a bien été changé et exit a bien permis de quitter le Minishell.

3. Gestion des processus

J'ai testé les commandes lj, sj, fg et bg.

```
/home >>> sleep 100 &
/home >>> sleep 200 &
/home >>> lj
ID shell      PID      Status  Commande
[1]          339      actif   sleep
[2]          340      actif   sleep
/home >>> sj 1
/home >>> lj
ID shell      PID      Status  Commande
[1]          339      suspendu sleep
[2]          340      actif   sleep
/home >>> bg 1
/home >>> ls
a.out  jobManager.c  jobManager.h  main.c  readcmd.c  readcmd.h  test
/home >>> lj
ID shell      PID      Status  Commande
[1]          339      actif   sleep
[2]          340      actif   sleep
[3]          341      actif   ls
/home >>> fg 1
```

4. Contrôle des processus

On observe le résultat à l'appui de ctrl-C et de ctrl-Z.

```
/home >>> sleep 100
^Z
Suspension du processus.
/home >>> sleep 100
^C
Terminaison du processus foreground.
/home >>>
```

5. Redirections Si on considère deux fichiers test1, avec écrit « Redirection », et test2, un fichier vide, alors la commande « cat < test1 > test2 » permettrait d'écrire dans le fichier test2 la ligne du fichier test1. Lorsqu'on réalise ce test dans le Minishell, c'est effectivement ce qui se produit.

```
/home >>> cat test1
Redirection
/home >>> cat test2

/home >>> cat <test1> test2

/home >>> cat test2
Redirection
/home >>>
```