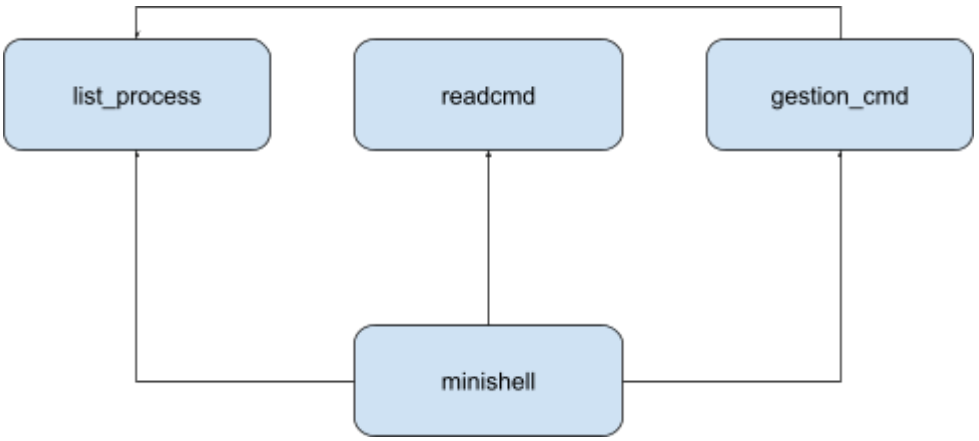
Rapport Final Projet SEC

Bilal Azdad

Architecture du code:



list_process : fournit les fonctions élementaires pour gérer la liste des processus

gestion_cmd : contient le traitement des commandes internes demandées ainsi le traitement des redirections/pipelines

minishell : est le main.

Pour lancer le programme il faut taper “make” puis “./minishell” depuis le dossier

Prise en compte des remarques de l’étape intermédiaire :

-Tout a été lu et traité hormis l’utilisation de sigaction, son utilisation provoque l’erreur segmentation fault lorsqu’on tente de lancer des commandes, ayant coder tout le projet avec la primitive signal, j’ai voulu à la fin remplacer l’utilisation de signal par l’utilisation de sigaction mais ça n’a pas marché, je fournis la version avec sigaction dans le rendu sous le nom de **minishell_non_fonctionnel.c.**

Question 1 :

Le projet consistait à développer un minishell robuste, simple et efficace. Pour cela, une boucle infinie a été utilisée pour lire chaque ligne sur l'entrée standard et créer un processus fils avec la fonction fork(), puis l'interpréter avec execvp() (sauf si cétait une commande que nous devions nous même coder).

Question 2:

```
bilalazdad@Air-de-Bilal minishell % ./minishell
bilal_Azdad_shell$ ls
bilal_Azdad_shell$ LisezMoi.html        gestion_cmd.h   list_process.o  readcmd.c
LisezMoi.md     gestion_cmd.o   minishell       readcmd.h
Makefile        list_process.c  minishell.c     readcmd.o
gestion_cmd.c   list_process.h  minishell.o     test_readcmd.c
```

On voit ici un exemple de désynchronisation du shell, on attend une commande avec que le fils ne se termine, ainsi l'affichage du prompteur gérée par le père précède le résultat de l'execution du fils.

Question 3:

Le problème de synchronisation a été résolu en ajoutant une attente de la fin de l'exécution du processus fils avec "`waitpid(pid, &status, WUNTRACED)`" dans le traitement du processus parent, en exploitant l'attribut backgrounded du type struct cmdline* qui permet d'identifer ce qui est lancer au premier plan et donc ce qui nécessite que le père attende.

```
bilal_Azdad_shell$ ls
LisezMoi.html   gestion_cmd.c   list_process.c  minishell       readcmd.c       test_readcmd.c
LisezMoi.md     gestion_cmd.h   list_process.h  minishell.c     readcmd.h
Makefile        gestion_cmd.o   list_process.o  minishell.o     readcmd.o
bilal_Azdad_shell$
```

le problème a été résolu.

Pour ces questions, le code se trouve dans **gestion_cmd**

Question 4:

Tout d'abord deux commandes internes ont été implémentées, cd en utilisant la fonction native chdir() et la commande exit en utilisant la fonction exit(0) qui quitte le programme.

Question 5 :

La possibilité de lancer des commandes en tâche de fond a également été implémentée en exploitant l'attribut backgrounded de la structure cmdline*. Lancer dans le background signifie que le père n'attent pas, on a alors deux fonction **execute_foreground** qui s'execute lorsque l'attribut background est faux sinon c'est **execute_background** dans la quelle le père n'attend pas?

Question 6:

"lj" : La commande "lj" a été implémentée en utlisant une liste chaînée pour permettre une flexibilité de gestion des processus de processus, la structure offre la possibilté d'ajouter, de supprimer des processus, chaque processus a un identifiant et un état et sont associé à la commande qui les a lancé. La structure est classique mais elle est difficile à mettre en place dans le main car beaucoup de cas de figure a traité.

"sj" : La commande "sj" a été implémentée avec la fonction kill(pid, SIGSTOP) pour suspendre le processus.

"bg" : La commande "bg" a été implémentée avec la fonction kill(pid, SIGCONT) pour mettre le processus en tâche de fond.

"fg" : La commande "fg" a été implémentée avec la fonction "`waitpid(pid, &status, WUNTRACED)`" et kill(pid, SIGCONT).

Question 7-8:

La gestion de la frappe de ctrl C et du suspension d'un processus (Ex : ctrl-Z) a été implantée dans le main **minishell** qui contient un traitant gérant les signaux SIGINT et SIGTSTP.

La conception a été réalisée de la manière suivante :

-On localise les processus en cours d'exécution en avant-plan grâce à la fonction **parcours_list_fg** de **list_process** qui retourner le process en avant plan dans les handlers, ensuite on fait soit kill(pid, SIGINT), soit kill(pid, SIGSTOP).

-De plus on masque les signaux SIGTSTP et SIGINT pour les processus dans le background et ne rien faire pour le processus du minishell (voir procédure **execute_background** dans le code du fils)

Question 9:

Les redirections ont été implantées dans le le fichier **gestion_cmd** dans la procédure **execute_redirections**, pour associer l'entrée standard à un fichier, on ouvre ce fichier en mode lecture, et on utilise la fonction **dup2** pour dupliquer le descripteur de ce fichier dans l'entrée, du même, pour l'association de la sortie standard à un fichier où on ouvre le fichier en mode écriture et on utilise de même la fonction **dup2**. Avant cela on vérifie qu'il y a bien une demande de redirection.

Question 10-11:

La réalisation de ces deux questions se trouve dans la procédure **execute_piplines** dans le fichier **gestion_cmd**, le principe est d'utiliser la récursivité et de parcourir chaque commande, si la commande existe on créé une pipe et on lance un fils, on redirige l'entrée vers celle de la commande précédente et la sortie vers l'entrée de la suivante si elle existe et ainsi de suite, on incrémente i à chaque passage.

Méthodologie de Test

Les tests ont été fait en imaginant des scénarios et en comparant les résultats avec le vrai shell.

Voici quelques tests significatifs :

Question 4,5 :

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ cd ..
bilal /Users/bilalazdad/Documents $ exit
 à bientot ! %
bilalazdad@MacBook-Air-de-Bilal minishell_AZDAD_BILAl %
```

Question 6 :

l'utilisation de **"ps m"** en parallèle de **lj** permet verifier que **lj** fonctionne ainsi que **sj et fg.**

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ sleep 20&
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ lj
ID              PID             ETAT            COMMANDE
1               80559           ACTIF           minsihell
2               80562           ACTIF           sleep
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ sj 2

 le processus 2 est suspendu
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ lj
ID              PID             ETAT            COMMANDE
1               80559           ACTIF           minsihell
2               80562           SUSPENDU                sleep
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ps m
  PID   TT  STAT      TIME COMMAND
79551 s000  S      0:00.12 -zsh
80562 s000  T+     0:00.01 sleep 20
80559 s000  S+     0:00.03 ./minishell
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ fg 2

 le processus 2 est mis en avant plan
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ lj
ID              PID             ETAT            COMMANDE
1               80559           ACTIF           minsihell
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ps m
  PID   TT  STAT      TIME COMMAND
79551 s000  S      0:00.12 -zsh
80559 s000  S+     0:00.04 ./minishell
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $
```

Question 7-8:

Test lorsqu'il n'y a aucun processus dans le foreground + commande susp

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ^C
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ^Z
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ^C
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ^Z
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ susp
zsh: suspended (signal)  ./minishell
bilalazdad@MacBook-Air-de-Bilal minishell_AZDAD_BILAl %
```

Test complet lorsqu'il y a un process en avant plan (en l'occurence ping)

-lorsque controle Z est lancé le process est bien suspendu,
-on reprend le process
-lorsque controle C est lancé le process se termine
-le process disparait de la liste des process

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ping www.google.com
PING www.google.com (142.250.179.100): 56 data bytes
64 bytes from 142.250.179.100: icmp_seq=0 ttl=119 time=24.272 ms
64 bytes from 142.250.179.100: icmp_seq=1 ttl=119 time=26.171 ms
64 bytes from 142.250.179.100: icmp_seq=2 ttl=119 time=25.292 ms
^Z
 le processus 3 est suspendu

bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ lj
ID              PID             ETAT            COMMANDE
1               80719           ACTIF           minsihell
3               80721           SUSPENDU                ping
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ps m
  PID   TT  STAT      TIME COMMAND
80696 s000  S      0:00.03 -zsh
80721 s000  T+     0:00.01 ping www.google.com
80719 s000  S+     0:00.04 ./minishell
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ fg 3

 le processus 3 est mis en avant plan

64 bytes from 142.250.179.100: icmp_seq=3 ttl=119 time=17.096 ms
64 bytes from 142.250.179.100: icmp_seq=4 ttl=119 time=24.962 ms
64 bytes from 142.250.179.100: icmp_seq=5 ttl=119 time=24.811 ms
^C
--- www.google.com ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 17.096/23.767/26.171/3.038 ms
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ lj
ID              PID             ETAT            COMMANDE
1               80719           ACTIF           minsihell
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ps m
  PID   TT  STAT      TIME COMMAND
80696 s000  S      0:00.03 -zsh
80719 s000  S+     0:00.05 ./minishell
```

Question 9

La redirection entre fichiers, vers sortie standard, depuis entree standard fonctionne

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ echo bonjour > Question9_1.txt
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ cat <Question9_1.txt> Question9_2.txt
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ cat < Question9_1.txt
bonjour
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $
```

Question 10-11

```
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $ ls | wc -l
      25
      25
bilal /Users/bilalazdad/Documents/minishell_AZDAD_BILAl $
```