
# Rapport sur le projet Minishell

ZHANI Reda

Département Sciences du Numérique - Première année 2022-2023

# 1 Choix de conception :

## 1.1 Question 1 :

L'objectif pour cette question est de réaliser la boucle de base de l'interpréteur, en se limitant à des commandes simples. Pour cela, j'ai créé une boucle infinie (avec *while*(1)), où je traite la commande entrée par l'utilisateur et je l'exécute en utilisant *execvp* dans le processus fils.

## 1.2 Question 3 :

Pour cette question, j'ai utilisé la fonction "*waitpid*" pour que le processus parent attende la fin de l'exécution du processus fils. Une fois le processus fils a terminé son exécution, le processus parent continue à lire la prochaine commande.

## 1.3 Question 4 :

Ici, j'ai traité les commandes internes **cd** et **exit** directement sans lancer de processus fils.

## 1.4 Question 5 :

Pour cette question, j'ai vérifié si "&" se trouve à la fin de la ligne et si c'est le cas, j'exécute la commande sans attendre la fin de l'exécution du processus fils.

## 1.5 Question 6 :

Pour répondre à cette question, j'ai commencé par créer la structure Process pour stocker les informations des processus lancés. Cette structure contient les informations suivantes :

- ident : c'est un identifiant propre à chaque processus que j'initialise à 1.
- pid : c'est le PID du processus.
- etat : l'état du processus (actif ou suspendu).
- cmd : ligne de commande qui a lancé ce processus.
- taille_cmd : taille de la commande.
- suiv : un pointeur vers la structure Process suivante.

Pour traiter la commande **lj**, j'ai créé la procédure listjobs qui parcourt la structure Process et affiche les informations de chaque processus.

Pour la commande **sj**, j'ai créé la procédure stopjob qui utilise l'identifiant pour trouver le processus correspondant dans la liste, et envoie le signal SIGSTOP.

De même pour la commande **bg**, j'ai créé la procédure background qui utilise l'identifiant pour trouver le processus correspondant dans la liste, et envoie le signal SIGCONT.

Pour la commande **fg**, j'ai créé la procédure foreground qui utilise l'identifiant pour trouver le processus correspondant dans la liste, envoie le signal SIGCONT et utilise ensuite waitpid pour attendre la fin de l'exécution du processus.

- **Le point de blocage** que j'ai rencontré pour cette question concernait la compilation avec c99. Lorsque j'essaie de compiler, l'option WCONTINUED n'est pas reconnue. J'ai trouvé sur internet qu'il faut ajouter "#define _XOPEN_SOURCE 700", cela a fonctionné mais je ne sais pas pourquoi.

## 2 Questions 7 et 8 :

Pour les questions 7 et 8, le problème était comment suspendre ou terminer uniquement le processus en avant-plan sans affecter le shell lui-même ni les processus en arrière-plan.

Pour résoudre ce problème, j'ai d'abord utilisé *pid_avant_plan*, qui contient le PID du processus en avant-plan. Et si aucun processus n'est en avant-plan, sa valeur sera -1.

J'ai initialisé cette variable à -1, et je modifie sa valeur dans la procédure foreground (qui permet de mettre un processus en avant-plan) et dans le processus parent dans le cas où il n'y a pas "&" dans la commande.

j'ai ensuite utiliser la fonction "signal()" pour associer le signal SIGTSTP et SIGINT à deux gestionnaires de signaux(handler_sigtstp et handler_sigint) et j'ai ignoré ces deux signaux dans le processus fils. Dans chaque gestionnaire, j'envoye soit le signal SIGSTOP soit SIGKILL au processus en avant-plan.

## 3 Question 9 :

Pour cette question, j'ai utilisé in et out de readcmd pour vérifier la présence de ">" et "<". J'ai ensuite utilisé dup2 pour effectuer la redirection.

## 4 Question 10 et 11 :

Pour ces deux questions,j'ai commencé par vérifier s'il existe une deuxième commande dans la séquence de commandes et si c'est la cas je compte le nombre total de commandes. J'ai ensuite créé le nombre de tubes nécessairesqui est égal au nombre de commandes moins un. Ensuite, pour chaque commande, j'ai créé un nouveau processus fils. Chaque processus fils est responsable de l'exécution d'une commande.et dans chaque processus fils j'ai redirigé l'entrée standard et la sortie standard vers les extrémités appropriées des tubes en utilisant la fonction dup2.