

Rapport global : Nihal BELMAKHFI

Les questions traitées complètement :

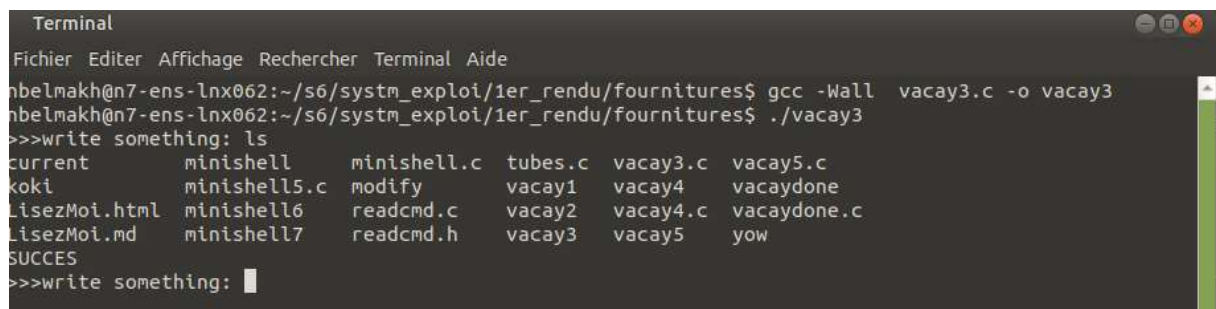
Sont les questions de 1 à 8 mises dans le fichier minishell8.c

Les questions traitées partiellement :

Après ajout du traitement des questions 9, 10 et 11, les questions 6 et 8 ne marchent plus totalement, plutôt à moitié, or les questions de 1 à 5, 7 et 9 marchent bien

Choix de conception :

Pour les questions de 1 à 2 : Je me suis basée sur une boucle 'Répéter' qui permet à chaque fois de demander à l'utilisateur d'écrire une commande (>>>write something :) qu'il exécute. Par exemple si on écrit ls, on remarque l'affichage de tous les fichiers et les dossiers dans le répertoire.



```
Terminal
Fichier Editor Affichage Rechercher Terminal Aide
nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ gcc -Wall vacay3.c -o vacay3
nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ ./vacay3
>>>write something: ls
current      minishell      minishell.c    tubes.c        vacay3.c       vacay5.c
koki         minishell5.c   modify         vacay1         vacay4         vacaydone
LisezMoi.html minishell6     readcmd.c      vacay2         vacay4.c       vacaydone.c
LisezMoi.md  minishell7     readcmd.h      vacay3         vacay5         yow
SUCCES
>>>write something: 
```

Pour la question 3 : j'ai utiliser la commande 'wait' pour forcer le processus père à attendre le processus fils

Pour la question 4, pour inclure la commande 'cd' dans le minishell, qui permet de revenir au répertoire d'accueil, j'ai utilisé la méthode 'chdir(HOME)' ainsi que 'getenv(HOME)', sachant que la variable HOME contient le chemin du répertoire d'accueil de chaque utilisateur. Et pour inclure la commande 'exit' dans le minishell, j'ai mis la variable boucler à 0, sachant que la condition de répétition de ma boucle est boucler = 1.

Pour la question 5, pour permettre de lancer les commandes en tache de fond, on utilise 'execvp', par contre dans ce cas, le père n'attend pas le fils. C'est pour cela que j'ai ajouté dans le cas (s->background == NULL), un wait .

Pour répondre à la question 6, j'ai défini le type job qui me permet d'enregistrer les informations concernant chaque processus fils : identifiant, pid, état et commande. Après j'ai créé une liste tab_affichage de type tab_job dont les éléments sont de type job. Le signal SIGCHLD que recoit le processus père modifie tabaffichage grace au hadler. J'ai aussi utilisé la fonction commande_interne dans laquelle, j'ai rassemblé toutes les commandes internes et je les ai traité. J'appelle commande_interne dans le main.

Pour tester lj, sj et fg :

```

nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ ./vacay3
>>>write something: sleep 100&
SUCCES
>>>write something: lj
identifiant: 1      pid: 43775      actif(background)      ligne de commande: sleep
>>>write something: sj 1
>>>write something: lj
identifiant: 1      pid: 43775      suspendu      ligne de commande: sleep
>>>write something: fg 1

```

Et pour tester bg :

```

nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ gcc -Wall vacay3.c -o vacay3
nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ ./vacay3
>>>write something: sleep 100&
SUCCES
>>>write something: lj
identifiant: 1      pid: 45571      actif(background)      ligne de commande: sleep
>>>write something: sj 1
>>>write something: lj
identifiant: 1      pid: 45571      suspendu      ligne de commande: sleep
>>>write something: bg 1
>>>write something: lj
identifiant: 1      pid: 45571      actif(background)      ligne de commande: sleep
>>>write something:

```

Pour répondre aux questions 7 et 8, pour traiter le ctrl Z, j'ai utilisé le handlerSIGTSTP qui envoie le signal SIGSTOP au dernier processus lancé actif en foreground. Or pour traiter le ctrl C, j'ai utilisé le handlerSIGINT qui envoie le signal SIGKILL au dernier processus lancé actif en foreground.

Pour tester ctrl Z, on fait ctrl Z juste après le fg

```

>>>write something: sleep 100&
SUCCES
>>>write something: lj
identifiant: 1      pid: 46103      actif(background)      ligne de commande: sleep
>>>write something: sj 1
>>>write something: fg 1
^Z>>>write something: lj
identifiant: 1      pid: 46103      suspendu      ligne de commande: sleep
>>>write something:

```

Et pour tester le ctrl C, on le fait après le fg également

```

nbelmakh@n7-ens-lnx062:~/s6/systm_exploi/1er_rendu/fournitures$ ./vacay3
>>>write something: sleep 100&
SUCCES
>>>write something: lj
identifiant: 1      pid: 46333      actif(background)      ligne de commande: sleep
>>>write something: fg 1
^C>>>write something:

```

Pour répondre à la question 9, J'ai essayé de rediriger la sortie/ entrée standards vers les fichiers : s->out et s->in.

Pour tester cette question j'ai écrit sur le minishell la commande ls > f.c

```

nbelmakh@n7-ens-lnx062:~/s6/system_exploi/1er_rendu/fournitures$ gcc -Wall minishell.c -o minishell
nbelmakh@n7-ens-lnx062:~/s6/system_exploi/1er_rendu/fournitures$ ./minishell
>>>write something: ls
current  LisezMoi.html  minishell5.c  minishell.c  readcmd.h  vacay2  vacay4  vacay5.c  yow
f1.c     LisezMoi.md    minishell6    modify       tubes.c    vacay3  vacay4.c  vacaydone
koki     minishell      minishell7    readcmd.c    vacay1     vacay3.c  vacay5    vacaydone.c
SUCCES
>>>write something: ls > f.c
SUCCES
>>>write something: 

```

Cela m'a généré le fichier f.c qui contient les fichiers dans le répertoire



```

1 current
2 f1.c
3 f.c
4 koki
5 LisezMoi.html
6 LisezMoi.md
7 minishell
8 minishell5.c
9 minishell6
10 minishell7
11 minishell.c
12 modify
13 readcmd.c
14 readcmd.h
15 tubes.c
16 vacay1
17 vacay2
18 vacay3
19 vacay3.c
20 vacay4
21 vacay4.c

```

Pour répondre aux questions 10 et 11, j'ai calculé tout d'abord le nombre de tubes, j'ai créé ensuite tous les tubes nécessaires, et j'ai mis dans le cas des tubes aussi la redirection de l'entrée et la sortie.

Point de blocage :

Au début, dans la question 6 j'avais des erreurs de ségmentation à cause de la mauvaise utilisation des malloc. Après ajout des traitements des question 9, 10 et 11, j'ai eu plusieurs problèmes que j'arrive pas à corriger pour le moment dont les erreurs de segmentations .