Processus en arriÃ¨re plan et commande cd:

arriere-plan:
Pour cela , je verifie avec la commande strcmp() si le dernier mot entrer est un '& ' , si c'est bien le cas je retire le "&" avant de passer au processus enfant et je fait c ontinue la boucle du for initial pour eviter d'attendre la fin du processus enfant et donc avoir directement la main.

commande cd:
je verifie avec la commande strcmp() si le premier mot entrer est "cd" , si c'est b ien le cas j'utilise donc la fonction chdir() sur le mot entrer suivant le cd pour y accede r(on continue la boucle du for initial une fois cette action achever pour eviter qu'il y ai t un fork inutile).

Passage de processus entre background et foreground:

stoper le signal ctrl-z:
Pour cela , j'utilise la fonction signal(), elle permet de detecter un signal ,igno rer son execution par default et executer une fonction qu'on peut lui passer en parametre , ou un autre signal a la place , dans ce cas precis le signal pour reconnaitre controle-Z e st
SIGTSTP et je l'ignore pour pas qu'il arrete mon minishelle avec SIG_IGN.
cela va ignorer tous les controle z , le probleme c'est que du coup je n'ai pas tro uver comment arreter des processus dans mon minishelle (les mettre en pause).

fg:
j'utilise la fonction strcmp pour reconnaitre si le premier mot est fg , ensuite j' utilise la fonction atoi() qui va convertir le pid donner en argument en int :il est en str ing dans le tableau mot (si il n'y a pas de pid donner en argument je passe donc au premie r plan le dernier processus passer en arriere plan).
je poursuis ensuite l'execution de ce processus normalement.

jobs:
j'utilise la fonction strcmp pour reconnaitre si le premier mot passer en commande est jobs , si c'est le cas je vais parcourir un tableau de int contenant les pid de mes pro cessus qui fonctionne de la maniere suivante :
- losrque on arrive au fork , j'appele ma fonction trouver_libre qui va renvoyer l' emplacement de la premier case libre de mon tableau de pid
-je vais ensuite ecrire le pid de mon processus dans ce tableau pour l'enregistrer
-lors de l'appele de la comande jobs , je vais donc parcourir ce meme tableau et ve rifier a chaque fois si la case i correspond a un pid ou non , si c'est le cas je l'affiche car cela signifie que c'est un processus qui n'est pas fini
-pour eviter que des pid de processus deja fini ce retrouve dans le tableau je les efface lorsque le processus enfant leur etant dedier ce termine (je sais qu'il sont termin er grace a la fonction waitpid() , qui renvoie le pid du processus si celui si est bien ter miner.

Redirections d'entrÃ©es-sorties:

j'ai proceder de la maniere suivante :

-je parcour le tableau mot pour savoir si j'ai une redirection de sortie "> " ou d'entrer "<" dans les arguments passer en commande
-une fois cela fait j'efface du tableau mot le ">" ou "<" ainsi que l'argum ent qui le suit qui correspond donc au nom de la destination de la redirection (ou de l'em placement de l'entrer)
-j'ouvre la detination de sortie ou l'entree avec open (en mode O_RDONLY p our l'entrer c'est a dire l'ecture et O_CREAT|O_WRONLY,0777 pour la sortie c'est a dire ecr iture , lui permettre de cree le fichier si celui ci n'existe pas et d'y avoir tous les dro it.
-je decrement nbmot (qui correspond a nombre de mot dans le tableau mot) d e 2 car je vien d'effacer 2 mot du tableau.
-je change ensuite l'entree/sortie avec dup2() (0 si redirection d'entrer , 1 si redirection de sortie).

-j’ai ensuite cree une fonction decaler() qui va decaler tous les argument du tableau mot pour ne pas qu’il y ait des case contenant NULL en plein milieu de mon table au mot suite a l’effacement des 2 case precedente

-je continue a parcourir le tableau mot pour voir si il n’y aurait pas d’au tre redirection d’entrer ou de sortie (si c’est le cas je repete les meme instruction pour chaque redirection trouver).