MINISHELL

Ce projet consiste en deux parties : celle de l'analyse (parce: où vous traitez les entrées utilisateur) et l'exécution(oú vous exécutez ce qui a été analysé).

- Dans notre mignon shell moi je vais m'occuper de la partie de l'exécution.
- Nous devons créer un mini shell :

**** Le shell fonctionnera uniquement en mode interactif (pas de scripts, c'est à dire que l'exécutable ne prend aucun argument).**

**** Exécutez des commandes simples avec un chemin absolu ou relatif (par exemple `/bin/ls`, `../bin/ls`).**

**** Exécutez des commandes simples sans chemin (par exemple les, `cat`, `grep`, etc...).**

**** Avoir un historique de travail (vous pouvez naviguer dans les commandes avec les flèches haut/bas).**

**** Mettre en œuvre des tuyaux (`|`).**

**** Implement redirections (`<`, `>`, `>>`).**

**** Implémenter le here-doc (`<<`).**

**** Gérez les guillemets doubles (`""`) et les guillemets simples (`''`), qui doivent échapper aux caractères spéciaux, en plus `$` des guillemets doubles.**

**** Gérer les variables d'environnement (`suivies` d'une séquence de caractères).**

**** Gérer les signaux comme dans bash (`ctrl + C`, `ctrl + \`, `ctrl + D`).**

-Builtins:

`echo` avec l'option `-n`
`cd` avec un chemin relatif ou absolu
`pwd` sans options
`export` sans options
`unset` sans options
`env` sans options ou arguments
`exit` sans options

SHELL

-D’abord commençons par savoir c’est quoi shell :
shell est une interface qui permet aux utilisateurs d’interagir avec le système d’exploitation à travers des lignes de commande.

Echo:

La commande echo de Linux répète simplement ce que vous lui demandez de répéter. Il s’agit d’une fonction extrêmement simple, mais absolument nécessaire pour ce genre d’actions. Sans *echo*, vous ne pourriez obtenir aucune sortie visible de vos scripts shell
Syntaxe :
echo [option] [string]

```
$ echo un exemple

un exemple

$ echo $PWD

/Users/Name/Desktop

$ echo "Vous êtes dans le répertoire : $PWD"

Vous êtes dans le répertoire : /Users/Name/Desktop.
```

Cd :

La commande cd est particulièrement importante, car elle vous permet de basculer très rapidement d’un répertoire à un autre. Cette commande Linux propose plusieurs fonctions pratiques.

Chdir : est une fonction standard en c utilisée pour changer le répertoire de travail courant d’un programme.

Retourne **0** si le changement de répertoire est réussi.

Retourne **-1** en cas d'erreur, et dans ce cas, la variable globale `errno` est définie pour indiquer la nature de l'erreur .

Pwd :

print working directory, afficher le répertoire de travaille actuelle .

getcwd :

(Get current working directory), une fonction standard en c qui renvoie le chemin absolu du répertoire de travail courant sous forme de chaîne de caractères,

ft_putendl_fd

est une fonction personnalisée qui affiche la chaîne cwd(chemin de répertoire actuelle) suivie d'un \n, dans le fd1(qui correspond à la sortie standard c'est à dire la console o.

Exit :

permet de quitter le shell avec un code de sortie si aucun argument ou un argument non numérique est passé, elle sort avec un code par défaut de o.

- **`exit(exit_status)`** terminé le programme en retournant le code de sortie spécifié par `exit_status`. Le shell ou le système d'exploitation interprétera ce code pour savoir comment le programme s'est terminé (succès ou échec).

Unset :

Utilisez la commande unset pour supprimer les variables pendant l'exécution du programme. Elle peut supprimer à la fois les fonctions et les variables shell. À l'aide de la commande unset, vous pouvez annuler les valeurs et les attributs des variables et des fonctions shell.

The syntax is:

```
unset var_name_here
unset [options] var
```

Env:

env est une commande shell pour Unix et les systèmes d'exploitation de type Unix. Elle permet d'imprimer une liste de variables d'environnement ou d'exécuter un autre utilitaire dans un environnement modifié sans avoir à modifier l'environnement existant.