OS et programmation système

Projet - Mini Shell

Magniadas Guillaume

December 18, 2018

1 Le projet

Il nous était demandé de créer un Mini Shell à partir d'une base fournie. Cette base était déjà fonctionnelle mais notre objectif était de l'améliorer et de rajouter des fonctions basiques d'un interprète Shell classique (Exemple : cd, &&, &)
Pour se faire je suis donc partie de la base fournie et j'ai décidé de créer une fonction qui s'occupera de d'abord traiter les caractères prioritaires.
La suite de ce rapport sera rédigé dans l'ordre d'incrémentation des fonctions dans le projet.

2 Lancerfonc

Lancerfonc est donc la première fonction que j'ai ajouté au programme, elle fait la même chose plusieurs fois mais effectue des tests différents, elle parcourt le tableau de string et vérifie si une chaine de caractères spécifiques s'y trouve et effectue une tâche spécifique si c'est le cas. Elle était assez vide au début, mais s'est rapidement étoffée à chaque ajout de fonctionnalité (Elle a commencé avec &, &&, cd, exit). Il est très simple de rajouter une fonctionnalité, il suffit de la rajouter au bon endroit, car oui l'ordre est important et rien n'est laissé au hasard dans ce code.
(Note : J'avoue avoir utilisé une facilité en utilisant un jump pour le ; mais mon autre solution était de faire une boucle juste après le découpage de la commande mais cela ne correspondait pas du tout avec le code que j'avais déjà écrit, j'ai donc décidé d'utiliser un jump, après beaucoup de tests il ne cause aucun bug.)

3 lancerfoncappart

Lanceroncappart est la fonction qui, suite au découpage, va gérer le tableau de strings, il va d'abord appeler lancerfonc, puis va s'occuper de créer un enfant si il faut et de lancer la commande. Au début du programme, lancerfonc était appelé directement après le découpage et tout ce que fait lancerfoncappart était géré dans le main mais j'ai pris la décision de gérer cela à part pour rendre le code plus modulable et surtout pour le rendre réutilisable ailleurs que dans le main, rendant ainsi très simple l'utilisation des programmes créer pendant les TP tels que && et pipes, en ayant juste à remplacer les exec par lancerfoncappart.
La fonction prend en paramètre child pour savoir si il s'agit d'un enfant qui a lancé la fonction (Pour éviter de créer des zombies et de bloquer le programme).

4 Metacar

Metacar est une fonction pour traiter les métacaractères juste après le découpage du strings. Elle ne contient en réalité que * mais est modulable et pourrait en accepter d'autres et même des macro, par exemple si je veux transformer toute les strings "salut" en "bonjour", un simple test après la première boucle suffirait. Pour * elle va d'abord chercher la présence d'une étoile, créer un tableau de strings des éléments du répertoire courant, puis regarder si il y a des caractères à gauche puis à droite et va chercher selon si il y a des caractères à droite, à gauche ou des deux coté, ou si il n'y en a aucun, les strings correspondants dans le tableau de strings créé et va remplacer le strings de l'étoiles par ceux-ci (Si l'étoile est seule, elle va simplement remplacer l'étoile par tous les strings du tableau créé).

5 Trace

Normalement, le code est assez commenté pour comprendre où se trouve le traitement de chaque caractère, et est assez simple pour comprendre son fonctionement.
Voici une trace d'utilisation inspiré de l'exemple sur la page web du projet :

```
[magnat@Pc-Magnat Projet]$ ./shell
/home/magnat/Documents/Universite/OS/Projet ? cd ..
/home/magnat/Documents/Universite/OS ? echo coucou
coucou
/home/magnat/Documents/Universite/OS ? sleep 3 & ; echo coucou
coucou
/home/magnat/Documents/Universite/OS ? ls bar && echo bar existe
ls: impossible d'accéder à 'bar': Aucun fichier ou dossier de ce type
/home/magnat/Documents/Universite/OS ? mkdir bar
/home/magnat/Documents/Universite/OS ? ls bar && echo bar existe
bar existe
/home/magnat/Documents/Universite/OS ? echo coucou && echo toi
coucou
toi
/home/magnat/Documents/Universite/OS ? sleep 2 && echo fou &
/home/magnat/Documents/Universite/OS ? fou

/home/magnat/Documents/Universite/OS ? echo coucou > fichier
/home/magnat/Documents/Universite/OS ? touch bonjour bar/bonjour bar/ca bar/va
/home/magnat/Documents/Universite/OS ? ls bar foo 2> /dev/null
bar:
bonjour  ca  va
/home/magnat/Documents/Universite/OS ? ls bar foo > /dev/null
ls: impossible d'accéder à 'foo': Aucun fichier ou dossier de ce type
/home/magnat/Documents/Universite/OS ? ls bar foo &> fichier
/home/magnat/Documents/Universite/OS ? cat fichier
ls: impossible d'accéder à 'foo': Aucun fichier ou dossier de ce type
bar:
bonjour
ca
va
/home/magnat/Documents/Universite/OS ? echo coucou >> fichier
```

```
/home/magnat/Documents/Universite/OS ? ps -o 2>> fichier
/home/magnat/Documents/Universite/OS ? cat fichier
ls: impossible d'accéder à 'foo': Aucun fichier ou dossier de ce type
bar:
bonjour
ca
va
coucou
erreur: le spécificateur de format doit suivre -o

Usage:
 ps [options]

 Essayez 'ps --aide <simple|liste|sortie|threads|divers|tous>'
  ou 'ps --aide <s|l|o|h|d|t>'
 pour plus d'aide.

Pour plus de détails, consultez ps(1).
/home/magnat/Documents/Universite/OS ? echo -e couleur\nperuque | cut -c 1-3 | sor
cou
per
/home/magnat/Documents/Universite/OS ? ls
bar  Cours   Projet  TD3  TD5  TD7  test
bonjour  fichier  TD10   TD4  TD6  TD8  YO
/home/magnat/Documents/Universite/OS ? echo *
bar bonjour Cours fichier Projet TD10 TD3 TD4 TD5 TD6 TD7 TD8 test YO
/home/magnat/Documents/Universite/OS ? echo TD*
TD10 TD3 TD4 TD5 TD6 TD7 TD8
/home/magnat/Documents/Universite/OS ? echo *jet
Projet
/home/magnat/Documents/Universite/OS ? touch cajet saljet testeurjet
/home/magnat/Documents/Universite/OS ? echo *jet
cajet Projet saljet testeurjet
/home/magnat/Documents/Universite/OS ? touch Poulpejet
/home/magnat/Documents/Universite/OS ? echo P*jet
Poulpejet Projet
/home/magnat/Documents/Universite/OS ? exit
Bye
[magnat@Pc-Magnat Projet]$
```