Projet SHELL

Description du projet

L'objectif de ce projet est de réaliser un SHELL, c'est-à-dire un logiciel permettant, en ligne de commande, de contrôler le système d'exploitation. Un tel programme attend de l'utilisateur des commandes sous forme de chaines de caractères qui doivent être interprétées par le SHELL. Le projet ne cherche pas à mettre en place un langage de programmation tel qu'il a été expliqué en cours, mais uniquement :

- de gérer des variables ;
- de pouvoir lire des commandes et les exécuter.

Une commande peut être une commande interne au SHELL, ou une commande externe. Les commandes externes sont disponibles dans l'arborescence de fichiers du système.

Dans notre projet, les commandes internes se limiteront à :

> echo : affiche une chaine de caractères à l'écran
> pwd : affiche le répertoire courant
> addpath path : ajoute au chemin de recherche d'un exécutable le chemin path
> delpath : remise à zéro du chemin de recherche des exécutables
> showpath : affiche tous les chemins dans lesquels s'effectue la recherche des exécutables

L'exécution des commandes externes devra déterminer le chemin de la commande dans le système de fichiers, soit en entrant au clavier le chemin absolu de la commande, soit en utilisant une variable interne contenant l'ensemble des chemins dans lesquels on peut chercher une commande.

Une fois l'exécution des commandes supportées, on gèrera la redirection des sorties, soit avec > ou >>, et on essaiera de supporter le pipe (|) qui permet de rediriger la sortie d'une commande en entrée d'une deuxième.

Dans la ligne de commande, on considère que l'espace est un délimiteur. Si une chaine de caractères doit contenir un espace, elle sera incluse entre des guillemets.

Langage de programmation et recommandations

Le langage de programmation utilisé sera le langage C sous Linux. On pourra bien sûr s'appuyer sur les fonctions systèmes vues en cours de programmation système. On veillera à organiser au mieux le programme et à commenter le code pour plus de lisibilité.

Etapes du projet

Ces étapes sont indicatives, allant des fonctionnalités les plus faciles à implémenter vers les plus difficiles. Dans toute la suite on supposera que le caractère espace est un délimiteur (c'est-à-dire qu'il sépare systématiquement deux mots), sauf lorsqu'il est inclus dans une chaine de caractères.

Gestion des variables et commandes internes

Les variables sont définies lorsqu'on leur affecte une variable qui peut être uniquement une chaine de caractères. Lorsque la chaine de caractère ne contient pas d'espace, la variable sera définie par la commande « var=chaine » (sans espaces avant et après le signe d'affectation '='), ou si elle contient des espaces « var="ceci est une chaine" ». Une variable pourra être accédée à n'importe quel moment en utilisant la valeur « $var », qui sera alors remplacée par la valeur de la variable.

Le traitement des commandes internes doit être réalisé par le SHELL, lorsqu'il les reconnaît.

Par exemple :

```
> var=IMT
> echo $var
IMT
> var=IMT Atlantique
Atlantique : command not found
> var="IMT Atlantique"
> echo $var
> IMT Atlantique
```

Gestion de l'exécution des commandes

Les fonctionnalités attendues en terme d'éxecution des commandes sont les suivantes (du plus facile à implémenter au plus difficile) :

1. Lire une commande et « parser » (reconnaître) la ligne de commande sans argument et l'exécuter. Si une commande est interne au SHELL, l'exécuter. Si elle est externe elle va entrainer l'exécution d'un exécutable présent dans le système de fichier. Par exemple, être capable de lire au clavier la commande « /bin/ls », l'exécuter et afficher son résultat. Si la commande n'existe pas, retourner un message d'erreur.
2. Lire une commande et ses arguments, être capable de l'exécuter et afficher son résultat. Par exemple, être capable de lire au clavier la commande « /bin/ls -al », l'exécuter et afficher son résultat. Si un argument est une variable, cet argument sera remplacé par la valeur de la variable. Si la commande n'existe pas, retourner un message d'erreur.
3. Gérer le chemin dans lequel le shell cherche les commandes et être capable d'exécuter « ls -al ». La gestion des chemins dans l'arborescence de fichiers se fera au travers des commandes internes « addpath » et « delpath ».
4. Gérer la redirection d'une sortie à l'écran en écrivant le résultat dans un fichier. Par exemple être capable d'exécuter « ls -al > fichier.txt » et afficher le résultat avec « cat fichier.txt ».
5. Gérer la concaténation d'une sortie à l'écran à la fin d'un fichier existant. Par exemple en étant capable d'exécuter « ls -al >> fichier.txt ».
6. Gérer l'exécution d'une commande en background. Par exemple, « ls & ».
7. Gérer la redirection d'une sortie d'une commande en entrée d'une autre via une pipe (un tube en français) et afficher le résultat de deux commandes en séquences. Par exemple en étant capable d'exécuter « ls -al | grep e ».

Exemple de commandes

Votre shell doit être capable de gérer les commandes suivantes :

```
> delpath
> addpath /bin
> addpath /usr/local/bin
> showpath
/bin:/usr/local/bin
> fichier="mon fichier.txt"
> touch $fichier fichier2.txt
> ls -al $fichier
> ls -al > resultat.txt
> ls -al | grep fichier
```