TP de Shell numéro 1 : Bash introduction

ISIMA ZZ 1

Un programme Shell sous UNIX est contenu dans un fichier texte. Après une brève entête, il liste ligne par ligne les commandes à exécuter dans une syntaxe totalement comparable à ce qu'il faut taper dans une fenêtre interactive (un terminal). Il est possible de séparer les commandes avec un ;

Il existe différents Shell UNIX, nous limiterons l'étude présente à l'utilisation de la syntaxe du Bourne Shell qui est présent sur l'ensemble des systèmes UNIX.

Commençons par un exemple simple, ouvrez un fichier texte (`toto`) et tapez le code suivant :

```
#!/bin/bash
echo "Kenavo les ZZ !"
```

La première ligne de ce code indique que vous voulez que ce programme soit interprété par le programme /bin/sh ; la deuxième qu'il faut écrire sur l'écran "Kenavo les ZZ". Pour que le programme fonctionne, il vous faudra taper de plus :

```
chmod a+rx toto
./toto
```

La première ligne met les droits nécessaires pour l'exécution du programme, la ligne suivante lance le programme. Taper echo $PATH vous devez voir un . dans ce qui s'affiche. Cette variable liste les chemins dans lesquels les programmes sont recherchés et le . signifie qu'il inclut dans cette recherche le répertoire courant.

Nous allons commencer l'apprentissage des diverses techniques du Shell en construisant un programme qui produit un petit annuaire.

1. Faire un fichier texte qui contient ceci :

```
Christophe Gouinaud;50 41;gouinaud@isima.fr
Leon Bobo;30 24;bobo@isima.fr
Loic Yon;22 22;yon@isima.fr
Christophe Lefort;50 41;lefort@isima.fr
```

   Utilisez pour cela un éditeur de texte et appelez le `tel.dat`

2. Affichez le fichier (utilisation d'une commande) (faire une sauvegarde de `tel.dat`)
   Faire à l'aide de la commande `cut` un programme qui affiche tous les noms. `cut` permet de couper une colonne en lui donnant un séparateur. Par exemple `cut -d";" -f 1,2` vous renvoie les noms et les numéros de tel.
   Pour cela, on commencera par cutter une première fois le fichier pour avoir les noms et prénoms, puis on « cuttera » une deuxième fois pour ne garder que les noms. Une commande sera chaînée à une autre à l'aide d'un pipe.
   Ajouter des espaces (en nombre variable) entre les prénoms et les noms. La commande précédente fonctionne-t-elle toujours ? Pour résoudre ce problème, vous pouvez utiliser `tr`, afin de réduire toute suite d'espace par un seul espace.

3. Rangez des valeurs dans une liste
   Les variables en Shell, ne se déclarent pas, sont de type chaîne de caractères. On leur affecte une valeur avec = et on récupère leur valeur en les préfixant par un `$`. Une liste peut être mise dans une variable pour une exploitation ultérieure, par exemple :

```
 #!/bin/sh
LISTE="un deux trois quatre"
for i in $LISTE
```

```
do
  echo $i
done
```

Les résultats d'une commande peuvent être affectés à une variable en utilisant des accents graves :

```
#!/bin/sh
echo "calcul du nombre de fichiers .dat"
NB=`ls *.dat | wc -l`
echo "$NB fichiers ont pour extension .dat"
```

En utilisant les deux notions décrites ci-dessus, écrire un programme qui affiche pour chaque fichier de votre répertoire principal uniquement les droits et la taille. Pour cela, il vous faudra affecter les résultats d'un premier `ls` à une liste puis, en bouclant sur cette liste, affichez les droits des fichiers à l'aide de `ls -l`.

4. Cherchons dans l'annuaire

   La commande `read mavar` permet de lire au clavier et d'affecter la variable `mavar`.
   La commande `grep toto file.dat` permet de ne sortir que les lignes qui contiennent le mot `toto` dans le fichier `file.dat`.
   La commande `sort` permet de trier et la commande `uniq` de supprimer les doublons.
   Faire un programme qui recherche une personne dans l'annuaire en triant les résultats et en virant les éventuels doublons.

5. Cherchons plus vite

   On va utiliser les arguments du Shell, c'est-à-dire ce que l'on écrit derrière la commande. On rappelle que :

$*	liste des arguments
$n	nième argument ($1, $2, …)

   On peut tester de la façon suivante :

```
#!/bin/sh
echo $*
for i in $*
do
   echo $i
done
```

   Faire un programme (`prog`) qui recherche `toto` dans l'annuaire, quand on tape `prog toto`.

6. Rajoutons un truc dans l'annuaire

   Faire un programme qui ajoute ou modifie une ligne du fichier annuaire. Pour cela, utiliser un fichier temporaire dans `/tmp`. A ce propos, un fichier temporaire se manipule de la façon suivante :

```
#!/bin/sh
TMP="/tmp/toto.$$"
echo "Blabla" > $TMP
echo "Blibli" >> $TMP
rm $TMP
```

   Le signe `$$` est une variable particulière dont la valeur est le numéro du processus courant.

7. Question subsidiaire

   Faire un programme qui tue tous vos processus `firefox` à l'aide de la commande `ps -aux` et `kill`.

8. Question encore plus subsidiaire

   Faire `man bash`, cela pourra être utile pour la prochaine fois et particulièrement ce qui concerne les structures de contrôle et les expressions régulières.