Introduction aux systèmes d'exploitation (IS1)

TP 10 – Structures `if` et boucles `for`

Comme vous l'avez probablement déjà remarqué, le `shell` est un véritable langage de programmation. Dans ce dernier TP, nous allons introduire deux composantes de la programmation en `shell` : les structures `if` et les boucles `for`.

1 Structures `if`

Nous avons déjà vu que la commande `test` combinée avec les connecteurs `&&`, `||` et `;` nous permettait d'exécuter des commandes en fonction du résultat de `test`. À la place des connecteurs `&&`, `||` et `;` on peut aussi utiliser un structure conditionnelle dont la syntaxe est la suivante :

```
if commande1 ; then commandes2 ; else commandes3 ; fi
```

Cela peut aussi s'écrire ainsi :

```
if commande1
then commandes2
else commandes3
fi
```

Cela correspond au comportement suivant :
- exécution de la commande *commande1*
- exécution de la suite de commandes *commandes2* si *commande1* retourne `0` ;
- exécution de la suite de commandes *commandes3* dans les autres cas.

Dans la plupart des cas, on utilise la commande `test` (que vous avez déjà rencontrée) pour tester une condition. Deux exemples typiques sont `test $x -eq $y` pour tester si les variables `x` et `y` ont le même contenu, ou bien `test -e truc` pour savoir si `truc` désigne un fichier ou un répertoire existant. D'autres options utiles sont entre autre `-f`, `-d`, `-r`, `-w`, `-x`, `-ne`, `-ge`, `-le`, `-gt`, `lt`,... Consultez le manuel de `test` !

Exercice 1 – *Structure `if`*

1. À l'aide d'une conditionnelle, écrivez une commande qui affiche le contenu de votre fichier `.bashrc` s'il existe, et la phrase *« Vous n'avez pas de fichier* `.bashrc`*.»* sinon.
2. À l'aide d'une conditionnelle, écrivez un script qui compare deux fichiers et affiche un message indiquant le résultat de la comparaison (identiques ou différents). (Utilisez la commande `diff`.)
3. Modifiez le script pour que le résultat de la comparaison soit redirigé vers le fichier `fichier` tout en ayant au préalable vérifié que vous possédez bien les droits pour le faire.
4. Écrivez une commande qui salue l'utilisateur en fonction de l'heure qu'il est : *«Bonjour»* avant 18 heures et *«Bonsoir»* après 18 heures et avant minuit (Utilisez la commande `date +%H`).

Correction.

1.

```
if test -e ~/.bashrc
then cat ~/.bashrc
else echo pas de fichier .bashrc
fi
```

2.

```
if diff file1 file2
then echo "meme fichier"
else echo "fichiers differents"
fi
```

3.

```
if test -w fichier
then if diff file1 file2
     then echo "meme fichier"
     else echo "fichiers differents"
     fi >> fichier
else echo "pas le droit d'ecrire dans fichier"
fi
```

4.

```
if test $(date +%H) -lt 18
then echo "Bonjour"; else echo "Bonsoir"
fi
```

2 Boucles `for`

La boucle `for` permet d'exécuter successivement une suite de commandes, pour chaque valeur que l'on donne à un paramètre. (Comme en `java` !) La syntaxe de la boucle `for` en `shell` est

`for` *`var`* `in` *`liste`* `; do` *`commandes`* `; done`

1. Renommez tous les éléments du répertoire courant en ajoutant le préfixe `BLA` à leur nom.
2. Écrivez une commande qui crée cinquante fichiers `fic1`, ..., `fic50`. (Indice : la commande `seq 1 50` retourne la liste des entiers de 1 à 50.)
3. Numérotez chaque élément du répertoire courant en ajoutant un numéro (croissant) au début du nom de chaque élément du répertoire. Indice : `i=$(($i+1))`

Correction.

1.
```
for x in *
do mv $x BLA$x
done
```
2. `for i in $(seq 1 50) ; do touch fic$i ; done`
3.
```
for x in *
do mv $x BLA$x
done
```
4.
```
i=1
for x in *
do
mv $x $i$x
i=$(($i+1))
done
```

3 Structures `case, while, until, select, ...`

Les plus curieux d'entre vous peuvent se documenter d'avantage sur les possibilités offertes par le `shell`. Une source conseillée est le site `http ://www.tuteurs.ens.fr/unix/shell/boucle.html`