R1.04 – TP 5

Table des matières

Nous finissons la découverte de bash avec les scripts. Jusqu'ici, nous avons utilisé des commandes de manières interactive. Les scripts vont nous permettre de regrouper plusieurs commandes, utiliser des choix, des boucles et des fonctions dans un fichier qui devient une nouvelle commande.

1 Éléments de base d'un script

Un script est un fichier texte que l'on rend exécutable. Il n'y a pas d'extension obligatoire, mais il est habituel d'utiliser `.sh`).

```
#!/bin/bash

echo Bonjour
echo le
echo monde
```

La première ligne du fichier indique le *path* de bash. Le caractère `#` est le caractère pour débuter un commentaire.

Nous pouvons utiliser des variables comme expliqué au Tp n°2 avec : affectation par `=` (sans espace) et consultation par `$variable` ou `$variable`.

Il est possible de forcer l'évaluation arithmétique d'une expression par `((expr))` et d'utiliser la valeur ainsi calculée avec `$`. Exemple :

```
$ ((c=3))
$ echo $c
3
$ d=$((1+2))
$ echo $d
```

```
3
$ b=$((a=2))
$ echo $a
2
$ echo $b
2
```

Il est également possible de lancer l'évaluation d'une commande : `()` et d'utiliser le résultat avec `$`. Ceci est appelé **substitution de commande**

```
# obtenir les permissions du répertoire courant avec son path absolu
$ ls -ald $(pwd)$
```

Comme pour une commande ordinaire, il est possible de passer des arguments à notre script. Ces arguments sont accessibles par des **variables positionnelles** : `$0` (la commande elle-même), `$1`, `$2`, ... ou `$@` pour les avoir tous. Si un argument n'a pas été fourni, la variable associée vaut la chaine vide, d'où l'habitude d'encadrer ces variables de guillemets doubles. Il est également possible de donner une valeur par défaut à une valeur de variable : `$var:-valDefaut`. Et, le nombre d'arguments passés est `$#`.

```
#!/bin/bash

echo $1
echo ${2 :-0}
```

Question.

- Étudiez **attentivement** les exemples précédents
- Consultez le man de bash et recherchez la section `Paramètres positionnels` ou `Positional Parameters` si votre man est en anglais.
- Écrivez un script qui affiche le nombre d'arguments du script ainsi que 2 fois ce nombre
- Écrivez un script qui affiche son propre nombre de lignes.
- en utilisant la variable d'environnement `RANDOM` (consultez le man de bash), produisez un script affichant une valeur aléatoire comprise entre 0 et 100.

2 Expression conditionnelle et choix

Une expression conditionnelle est mise entre `[ ]`. Il existe trois types de test :

1. test sur un fichier : `-r` accessible en lecture, `-w` accessible en écriture, `-x` accessible en exécution, `-d` est un répertoire, `-f` est un fichier, ...
2. test numérique : `-eq` égalité, `-ne` différence, `-lt` plus petit, `-le` plus petit ou égal, `-gt` plus grand, `-ge` plus grand ou égal
3. test sur une chaine de caractères : `==` ou `=` égalité, `!=` différence, `-z` taille nulle, ...

La valeur d'une expression conditionnelle est consultable avec `$?`. Attention, `0` correspond à `vrai` et `1` à `faux`. Ces expressions conditionnelle sont combinables avec `-a` et, `-o` ou et `!` non. Elles peuvent également être utilisées dans des choix ou des enchainements de commandes :

Choix :

```
if [ ... ]
then
  cmds1
elif [ ... ]
  cmds2
else
  cmds3
fi
```

qu'on peut compacter sur une seule ligne en remplaçant les retours à la ligne par des `;`

Enchainement de commandes :

- `cmd1 ; cmd2` : `cmd2` est toujours exécutée (après `cmd1`)
- `cmd1 && cmd2` : `cmd2` est exécutée si `cmd1` renvoie un code de retour égal à 0
- `cmd1 || cmd2` : `cmd2` est exécutée si `cmd1` renvoie un code de retour égal à 1

Exemple :

```
# touch fichier.txt  pour être sûr d'avoir un fichier
$ if [ -r fichier.txt -a 1 -lt 2 ] ; then echo "ok" ; fi
ok
$  [ -r fichier.txt ] && echo "ok"
ok
```

Question.

- Étudiez **attentivement** les exemples précédents
- Consultez le man de bash et recherchez la section `CONDITIONS` ou `CONDITIONAL EXPRESSIONS` si votre man est en anglais.
- Écrivez un script qui reçoive, comme seul paramètre, un nom de répertoire et qui affiche, selon le cas, l'un des trois messages suivants : `ce répertoire n'existe pas`, ou `ce répertoire existe et il est vide`, ou `ce répertoire existe et n'est pas vide`.
- Écrivez un script qui reçoive trois nombres en arguments et qui teste s'ils sont donnés en ordre croissant. Le script renvoie (par `exit`) un code de retour 0 si la commande est conforme (3 arguments donnés en ordre croissant) et 1 sinon (pas trois arguments ou pas le bon ordre).

3 Boucles

Bash permet de faire des boucles. En particulier (il en existe d'autres) :

- boucle `while` : `while [ ... ]; do cmds ; done`
- boucle `for` : `for var in list ; do cmds ; done`

Exemple :

```
#!/bin/bash

i=1
while [ $i -lt 10 ]
do
  echo -n "$i "
  ((i++))
done
echo
```

Question.

- Étudiez **attentivement** les exemples précédents
- Consultez le man de bash et recherchez la section `Commandes composées` ou `Compound Commands` si votre man est en anglais.
- Écrire un script qui affiche tous ses paramètres en utilisant une boucle `while` et `shift`
- en utilisant `for` et `seq` faire afficher les entier de 1 à 10