Scripts Shell

👁 `sed`, `exit`, `head`, `test`... (cf. document commandes)

✎ Exercice 1

En supposant que le répertoire courant est la racine `/`, que produisent les séquences de commandes suivantes :

1.	2.	3.	4.
`a=1`	`a=1`	`a=1`	`a=1`
`b=$a`pwd``	`b=$a `pwd``	`b="$a `pwd`"`	`b='$a `pwd`'`
`echo $b`	`echo $b`	`echo $b`	`echo $b`

1 - 1/repertoire courant
2 - Erreur car on ne peut pas avoir d'espace dans une ligne où il y a une affectation
3 - 1 /repertoire courant car on a mis les ""
4 - $a `pwd` car aucun metacaractere n'est interprété entre les '

✎ Exercice 2

On suppose que l'on dispose d'un fichier `calepin.txt`, contenant des noms et des numéros de téléphone rangés selon le modèle suivant :

```
DUPONT␣Jean␣05.61.75.18.47
DURAND␣Martin␣09.23.45.32.56
MARTIN␣Yvonne␣02.23.34.45.56
DUPONTEL␣Albert␣05.32.57.39.66
```

1 - sed -e 's/0/+33 /'

écrire la ligne de commande permettant d'afficher le calepin :

1. ajouter l'indicatif +33 à la place du 0 au début des numéros de téléphone,
2. la même chose sauf pour les numéros en 09,
3. inverser nom et prénom.

✎ Exercice 3

Écrire le script `chercher.sh` qui recherche dans `calepin.txt` les personnes ayant un nom qui commence par le premier paramètre passé à ce script et dont le numéro de téléphone se termine par les deux chiffres passés en deuxième paramètre, et qui affiche les informations relatives à ces personnes

```
#! bin/sh
grep "^$1 " calepin.txt | grep "$2$"
```

✎ Exercice 4

La commande `file` affiche une ligne contenant la chaîne « executable » lorsqu'on lui passe en paramètre le nom d'un fichier « binaire exécutable ».

En utilisant cette commande, écrire un script qui affiche « OK » si le fichier dont le nom lui est donné en paramètre correspond à un « binaire exécutable ». Ce script renverra le code de retour 0 en cas de succès, le code 1 sinon.
Par exemple :

```
$ file programme1
programme1: ELF 64-bit LSB executable, x86-64, version 1 (SYSV)...
$ ./verif_exec.sh programme1
OK
$ echo $?
0
$ file texte.txt
texte.txt: Unicode text, UTF-8 text
$ ./verif_exec.sh texte.txt
$ echo $?
1
```

✎ Exercice 5

A l'aide de la commande `test`, écrire un fragment de script qui vérifie que le script a été appelé avec 1 paramètre. Dans le cas contraire, un message d'erreur doit être affiché sur `stderr` puis le script doit s'arrêter avec le code de retour `1`.

1 Éléments pour l'écriture des scripts

1.1 Grandes étapes d'un script

Un script est généralement constitué des parties suivantes :

- vérification du nombre de paramètres ;
- vérification de la validité des paramètres ;
- traitement.

1.2 Vérification de la validité des paramètres

1. Pour un nom de fichier
   (a) fichier à lire :
   - existence (`test -f`)
   - droit de lecture (`test -r`)

   (b) fichier à modifier :
   - existence (`test -f`)
   - droit de lecture (`test -r`)
   - droit d'écriture (`test -w`)

2. Pour un nom de répertoire
   (a) répertoire à consulter :
   - existence (`test -d`)
   - droit de lecture (`test -r`)
   - droit d'exécution (`test -x`)

   (b) répertoire à modifier :
   - existence (`test -d`)
   - droit de lecture (`test -r`)
   - droit d'exécution (`test -x`)
   - droit d'écriture (`test -w`)

✎ Exercice 6

Écrire le script `info.sh` dont la syntaxe d'appel est : `info.sh fichier` qui affiche sur stdout les informations suivantes :

1. Nom du fichier
2. Fichier (ou Répertoire)
3. Droit de lecture (ou Pas le droit de lecture)
4. Droit d'écriture (ou Pas le droit d'écriture)
5. Droit d'exécution (ou Pas le droit d'exécution)

✎ Exercice 7 (test de la validité d'un entier)

Écrire le script `compare2.sh` dont la syntaxe d'appel est :

`compare2.sh` $\underline{\texttt{entier}_1}$ $\underline{\texttt{entier}_2}$

qui compare les deux entiers strictement positifs $\underline{\texttt{entier}_1}$ et $\underline{\texttt{entier}_2}$. Si $\underline{\texttt{entier}_1}$>$\underline{\texttt{entier}_2}$ alors il affiche `OK` et retourne 0 sinon il renvoie 1.

 Attention : Vous devez vérifier que $\underline{\texttt{entier}_1}$ et $\underline{\texttt{entier}_2}$ sont bien des entiers strictement positifs. Dans le cas contraire un message d'erreur doit être affiché sur `stderr`.

✎ Exercice 8 (Traitement des paramètres passés au script)

Écrire le script `parametres.sh` dont la syntaxe d'appel est :

`parametres.sh` [$\underline{\texttt{param}}$...]

qui affiche tous les paramètres du script selon le format suivant :

`paramètrei=valeur du` i^{me} `paramètre`

Par exemple :

```
$ parametres.sh a b "e f"
paramètre1=a
paramètre2=b
paramètre3=e f
```

Indication :

```
for param in "$@"
do
  # Traitement de $param
done
```