

IF 110 : Système d'exploitation - TP4

Claire Capdevielle - Rémi Giraud
ccapdevi@labri.fr, remi.giraud@labri.fr

2015-2016

1 Exécution d'une commande

1.1 Valeur de retour d'une commande

Dans les TP précédents, les variables `PATH`, `DISPLAY` tout comme `FOO` ont en fait été définies dans votre environnement, et vous pouvez en modifier leur valeur. Mais il existe quelques variables particulières affectées par le shell lui même. L'une d'entre elles se nomme `?`.

Question 1 Exécutez la commande `ls -l` et affichez le contenu de la variable `?`.

Question 2 Exécutez la commande `ls -z` et affichez le contenu de la variable `?`.

La variable `$` retourne le PID du shell courant. Afficher donc son contenu, et vérifiez en sa valeur à l'aide de la commande `ps -l`.

Question 3 Testez les commandes suivantes :

```
$ [ 2 = 2 ] ; echo $?  
$ [ 2 = 3 ] ; echo $?  
$ test 2 = 2 ; echo $?  
$ test 2 = 3 ; echo $?\
```

1.2 Action conditionnelle

Question 4 Dans votre fenêtre terminal, exécutez la commande `ls -l && echo OK || echo KO`. Puis exécutez la commande `ls -z && echo OK || echo KO`. Quel est le rôle des opérateurs `&&` et `||` ? Examinez dans chacun des deux cas précédents, le contenu de la variable `?`. Expliquez son contenu. Modifiez la commande `ls -z && echo OK || echo KO` de telle sorte que le message d'erreur généré soit redirigé vers `/dev/null`.

Question 5 Réécrire les commandes de l'exercice précédent en utilisant la structure de contrôle `If` en une ligne de commande et dans un script que l'on appellera `script_if` (faites passer la commande `ls -l` en paramètre).

2 Les script shells et mécanismes de substitution

Un fichier script contient une suite de commandes shell. Pour exécuter le script, deux méthodes sont possibles. Soit vous tapez le nom du shell suivi du nom du script à interpréter :

```
$ bash mon-script
```

soit vous tapez directement le nom du script :

```
$ ./mon-script
```

Dans ce dernier cas, le fichier script doit posséder la permission en exécution et pour garantir son exécution par un shell de type **sh**, la première ligne de ce fichier doit indiquer le chemin absolu du shell qui sera utilisé. La syntaxe de cette ligne est la suivante :

```
#!/bin/sh
```

Si cette ligne est omise, c'est le shell de type **bash** qui est utilisé par défaut. Dans tous les cas, prenez la bonne habitude d'insérer systématiquement cette ligne particulière dans vos scripts shells.

Question 6 Créez un fichier exécutable **cmd.sh** contenant :

```
#!/bin/sh
```

```
ps -f
```

Dans votre fenêtre **terminal**, lancez la commande **cmd.sh**. Si tout va bien, vous devriez voir s'afficher la liste de vos processus et en particulier un processus (à peu de chose près) de la forme **/bin/sh cmd.sh**. Modifiez la ligne **#!/bin/sh** afin d'utiliser un autre shell que **sh** et réexaminez le résultat produit par l'exécution de la commande **cmd.sh**.

Question 7 Ecrire un petit script qui affiche "Bonjour *login*, il est 16 heures" (où *login* est votre nom de login, et 16 est l'heure actuelle. Bien évidemment, vous utiliserez la notion de variable).

2.1 Les variables prédéfinies

Le shell offre des variables prédéfinies facilitant la programmation sous le shell. Vous avez déjà pu observer le rôle des variables **?** ou **\$**. Cherchez dans le manuel de **bash** le rôle des variables **#**, *****, **@**, **0**, **1**, **2**, **3**, etc.

Question 8 Insérez dans un script shell, une commande permettant d'afficher le nom absolu du script lancé (chemin depuis la racine + nom de la commande). Pour en vérifier le bon fonctionnement, renommez votre fichier en **autre_nom.sh** et exécutez le.

Question 9 Insérez une commande affichant le nombre de paramètres du fichier de commandes de la manière suivante :

$$nomvar = 'nb_param'$$

où *nomvar* est le nom de la variable contenant cette information et *nb_param* son contenu. Vérifiez que cela fonctionne en lançant le fichier de commandes avec plusieurs paramètres.

Question 10 De la même façon, insérez une commande affichant la liste des paramètres du fichier de commandes. Trouvez une autre façon d'écrire la commande précédente pour obtenir le même résultat.

Question 11 Lisez le manuel de **shift**. Ecrivez un script qui affiche la liste de ses paramètres commençant par le troisième paramètres.

2.2 Itérations

Question 12 Tester la boucle **for** au moyen des programmes suivants :

<pre>for N in un deux trois quatre do echo "***\$N**" done</pre>	<pre>for N in \$(ls) do echo "Fichier -> \$N" done</pre>
--	---

Question 13 Ecrire un script listant chacun de ses paramètres en utilisant une boucle `while` avec la commande `shift` ou `for`. On appellera ce script `liste.sh`. Tester la avec la commande suivante `./liste.sh "a b" 'c' d e`

2.3 Substitution de commandes et de variables

Question 14 Examinez les scripts qui suivent,

`petit-script-4` :

```
#!/bin/sh
date
echo Il est $4\\
```

`petit-script-5` :

```
#!/bin/sh
set 'date'
echo Il est $4\\
```

et exécutez les commandes :

```
$ petit-script-4 donald riri fifi loulou picsou
```

```
$ petit-script-5 donald riri fifi loulou picsou
```

afin d'interpréter le rôle de la commande : `set` $\prec$ *commande* $\succ$.

Question 15 Écrire un script qui affiche l'heure sous la forme : `Il est 17:45:26`.

Consultez le manuel de `bash` afin d'examiner le rôle de la variable `IFS`.

Question 16 En modifiant la variable `IFS`, écrire un script qui affiche : `Il est 17h 45mn`.

Question 17 Écrire un script `prepa` prenant en paramètre un nom de fichier : $\prec nom \succ$ qui crée dans le répertoire courant un fichier exécutable $\prec nom \succ$ dont la première ligne est `#!/bin/bash` suivi d'une ligne vide.

Question 18 Écrire, sans utiliser de structure de contrôle, les commandes suivantes :

1. `nf` : affiche le nombre de fichiers ne commençant pas par un point, du répertoire courant ¹ ;
2. `ra` : affiche oui si le répertoire courant est le répertoire d'accueil et non sinon ² ;
3. `prc` : affiche la profondeur du répertoire courant ³ ;

Question 19 Écrire le script `ifdef` tel que l'exécution de `ifdef` $\prec nom \succ .h$ place en tête de $\prec nom \succ .h$ les lignes

-
1. Vous utiliserez les commandes `echo`, `wc` et `pwd`.
 2. Vous utiliserez les commandes `test`, `pwd`, `echo` et les opérateurs `&&` et `||`.
 3. Vous utiliserez les commandes `set`, `pwd`, `shift`, `echo` ainsi que la variable `IFS`.

```
\#ifndef \meta{NOM}\_H\  
\#define \meta{NOM}\_H\  

```

et la ligne

```
\#endif /* \meta{NOM}\_H */
```

à la fin. On pourra utiliser la commande `tr(1)` pour les conversions de minuscules en majuscules.

3 Enchaînement des substitutions

Question 20 Expliquez le comportement de chacune des lignes de commandes suivantes :

```
A='echo *'  
$A  
echo $A  
echo '$A'  
echo "$A"
```

De même avec cette fois ci le **remplacement de commandes** : `A='echo *'`

Question 21 En observant le comportement des lignes de commandes suivantes, expliquez ce que fait `eval`.

```
CMD=echo  
ARG='$$'  
LIG='$CMD [$ARG]'  
$LIG  
eval $LIG  
eval eval $LIG  
eval eval eval $LIG
```

4 Environnement

Question 22 Observez et expliquez la suite de commandes :

```
echo $A  
A=aaaaa  
echo $A  
bash  
echo $A  
exit  
export A  
bash  
echo $A  
A=bbbbb  
echo $A  
exit  
echo $A
```

Question 23 Que fait le script suivant :

```
#!/bin/sh

echo " --> $$"
```

Testez ce script et comparez avec le résultat affiché en entrant directement la commande **echo \$\$** dans votre bash. Expliquez ce qui se produit à l'exécution. Essayez en préfixant la commande par **.** ou par **source**. Que peut-on en conclure ?

Question 24 Que fait le script suivant :

```
#!/bin/sh

if [ $# = 0 ]
then
    cd ..
else
    cd $1
fi
echo " --> `pwd`"
```

Testez cette commande et expliquez ce qui se produit à l'exécution. Essayez en préfixant la commande par **.** ou par **source**. Que peut-on en conclure ?

Question 25 Faites un script qui range les fichiers dans une arborescence du type année/mois/-jour à partir de la date de modification.

5 Mini Projet noté - Exercice Quizz Capitales

D'après <http://www.lix.polytechnique.fr/~cousineau/Partage/IS1> modifié par Aurélien Esnard.

Le but de cet exercice est de réaliser un script "quizz.sh" pour tester ses connaissances sur les capitales du monde. Récupérez le fichier capitales.txt dont chaque ligne contient un pays et sa capitale séparés par une virgule. Les noms de ville et de capitale sont sans accents.

Le fonctionnement minimal souhaité du script est le suivant :

- Si l'utilisateur entre la commande "quizz.sh -v ville", le script demande le nom du pays dont la capitale est ville et vérifie la réponse .

- Si l'utilisateur entre la commande "quizz.sh -p pays", le script demande le nom de la ville qui est capitale de pays, puis vérifie la réponse.

Question 26 Écrivez le script quizz.sh remplissant les fonctionnalités minimales. Vous pourrez utiliser la commande "grep" pour récupérer la ligne du fichier capitales correspondant à un pays ou une ville donnée et la commande "cut" pour extraire d'une ligne la ville ou le pays.

Question 27 En cas d'erreur de l'utilisateur dans le format des paramètres, faites afficher au script un message d'aide rappelant la syntaxe.

Question 28 Votre script utilise le fichier capitales.txt pour fonctionner. Décidez d'un endroit où stocker ce fichier, et faites afficher un message d'erreur par le script si le fichier est absent.

Question 29 (optionnel) Vous pouvez rajouter des fonctionnalités plus avancées à votre script. Voici quelques suggestions bonus :

- ignorer les accents et majuscules dans la vérification de la réponse (utilisez la commande *sed*);
- poser plusieurs questions à la suite (nombre fixé à l'avance ou choix du type "continuer (o/n)?");
- afficher un score, l'enregistrer dans un fichier ;
- si le deuxième argument est omis, tirer une ligne au hasard ;
- poser des questions pour toute la liste dans le désordre.