Chapitre 4
Manipulation des variables

Durée : 1 heure 55

Mots-clés

variables utilisateur, script shell, chmod, variables réservées, paramètres de position, shift, exit, read, expr, opérateurs logiques et arithmétiques

Objectif

Ce chapitre vous permettra de :

- Manipuler des variables utilisateur : chaînes, nombres, environnement et paramètres.
- Rédiger des scripts shell.

Prérequis

Pour valider les prérequis nécessaires, avant d'aborder le TP, répondez aux questions ci-après :

1. Pour définir une variable utilisateur, quelle commande doit-on exécuter ?

a. `set var1 valeur1`

b. `var1=valeur1`

c. `var1 = valeur1`

2. Laquelle des syntaxes suivantes est correcte ?

a. `Message= "Bonjour tout le monde"`

b. `Message=Bonjour tout le monde`

c. `Message = "Bonjour tout le monde"`

3. Pour supprimer une variable, on utilise la commande **`unset`** :

a. Vrai.

b. Faux.

4. Un script shell est :

a. un fichier qui contient des commandes shell.

b. un fichier binaire.

c. un fichier qui dispose de la permission 'x'.

d. un fichier compilé.

e. un fichier interprété.

5. Un script shell peut-il être lancé dans le shell courant ?

a. Vrai.

b. Faux.

6. Un script shell peut être lancé par un processus shell fils :

a. Vrai.

b. Faux.

7. Un script shell peut contenir :

a. des définitions de variables d'environnement.

b. des structures conditionnelles.

c. des boucles.

d. des appels à d'autres scripts shell.

8. Parmi la liste suivante, quelles sont les variables réservées du shell ?

a. `$!`

b. `$&`

c. `$*`

d. `${11}`

9. La variable `$0` représente le nom du script shell en cours d'exécution ?

a. Vrai.

b. Faux.

10. Quelle commande permet de donner le droit d'exécution au propriétaire du script nommé *essai1* ?

a. `chmod a+x essai1`

b. `chmod u+e essai1`

c. `chmod u+x essai1`

11. Pour visualiser le contenu de la variable **$#**, on lance la commande suivante :

a. `echo #`

b. `echo $#`

c. `echo ${#}`

12. Ces affirmations sont-elles exactes ?

a. **`$@`** et **`$*`** contiennent la même information.

b. **`"$@"`** et **`"$*"`** contiennent la même information.

c. **`$*`** utilise les valeurs de la variable **`IFS`** comme séparateur de champs.

d. **`$@`** utilise les valeurs de la variable **`IFS`** comme séparateur de champs.

Corrigé p. 189

Énoncé 4.1 Les variables utilisateur

Durée estimative : 10 minutes

1. Définissez une variable nommée **`Nom`** dont le contenu est **ENI**.

2. Définissez une variable nommée **`Collection`** avec le contenu : **Ressources Informatiques**.

3. Affichez la chaîne de caractères suivante : **Les Editions ENI. Collection Ressources Informatiques**. Utilisez vos deux variables **`Nom`** et **`Collection`** définies précédemment.

4. Définissez une variable **`Var1`** avec le contenu : **bon**

5. Nous voulons afficher le mot : bonjour en se servant du contenu de la variable **`Var1`**. Laquelle des commandes suivantes devons-nous exécuter ?

```
$ echo $Var1jour
$ echo $(Var1)jour
$ echo ${Var1}jour
```

À quoi servent les accolades ? À quoi servent les parenthèses ?

6. Exécutez les commandes suivantes :

```
$ echo ${var2 :?"var2 n'est pas encore définie"}
$ var2=Bonjour
$ echo ${var2 :?"var2 n'est pas encore définie"}
$ echo ${lejour:=`date +%d`}
$ lejour=24
$ echo ${lejour:=`date +%d`}
$ echo "Jour : ${lejour:+01}"
```

Quel est l'intérêt de la première commande ?
Quel est l'intérêt de la quatrième commande ?
Quelle est la différence entre la quatrième et la dernière commande ?

Indices pour l'énoncé 4.1

6. *Consultez le manuel d'aide de la commande* **`date`** *pour plus d'informations sur les options utilisées.*

Corrigé p. 191

Énoncé 4.2 Les variables globales et les variables locales

Durée estimative : 15 minutes

1. Créez un alias **`cx`** dont le contenu permet de rendre un fichier exécutable. Comment pouvez-vous rendre cet alias permanent pour toutes les sessions ?

2. Créez un fichier nommé *`Affiche`* renfermant les commandes suivantes :

```
#!/bin/sh
# La première ligne indique au système que /bin/sh est le programme
# shell à utiliser pour exécuter les commandes du script.

# Affiche deux chaînes de caractères

echo "Mon premier script shell"
echo "Chaque ligne du fichier est une commande"

# Le script retourne 0, valeur synonyme de succès.
exit 0
```

3. Après avoir saisi le script et l'avoir enregistré, n'oubliez pas de le rendre exécutable à l'aide de l'alias créé précédemment.
`$ cx Affiche`

4. Pour exécuter le script, tapez la commande suivante :
`$ ./Affiche`

5. Créez un script nommé *`AfficheVarEnv`* qui affiche le contenu des variables d'environnement suivantes : **`HOME`**, **`PATH`**, **`USER`** et **`PWD`**. Chaque contenu devra être précédé d'un commentaire correspondant à l'intitulé de la variable. Exécutez ce script.

6. Créez un script nommé *VarLocGlob* renfermant les commandes suivantes :

```
#!/bin/sh

# Manipulation de variables locales et globales

echo "Mon répertoire de connexion est : $HOME"
echo "Mon nom de connexion est : $(logname) "
echo "Nous sommes le : `date +%D`"

Nom=Torvalds
Prenom=Linus
echo "Linux a été créé par $Nom $Prenom à l'université d'Helsinki"

exit 0
```

Exécutez ce script.

7. Créez un script nommé *VarLocGlob1* qui effectue les opérations suivantes :
- Afficher le contenu du répertoire courant.
- Afficher le shell de l'utilisateur.
- Afficher le nom de la machine.
- Définir une variable appelée **os** avec comme contenu le nom du système d'exploitation.
- Définir une variable appelée **ver** contenant le numéro de version du système d'exploitation.
- Afficher le contenu des deux variables **os** et **ver**.

Indices pour l'énoncé 4.2

1. *Consultez le chapitre Exécution et environnement shell ou la page du manuel de la commande* ***alias****.*

3. *Utilisez la commande* ***chmod*** *avec l'option adéquate.*

4. *Si vous n'avez pas le répertoire courant (.) dans votre variable* ***PATH****, le* ***./*** *devant le* et **6.** *nom du script est indispensable, sinon il suffit de taper directement le nom du script.*

7. *Utilisez la commande* ***uname*** *avec les options adéquates.*

Corrigé p. 192

Énoncé 4.3 Les variables de paramètres : $@, $*, $#...

Durée estimative : 40 minutes

1. Écrivez le script suivant nommé *Parametres* :

```
#!/bin/sh

# Parametres : Variables de paramètre ou de position

echo "Nom du script en cours d'exécution : $0"
echo "Nombre de paramètres transmis : $#"
echo "Numéro du processus du script en cours d'exécution $0 : $$"

echo "Paramètre 1 du script $0 : $1"
echo "Paramètre 2 du script $0 : $2"
echo "Paramètre 3 du script $0 : $3"
echo "Liste de tous les paramètres du script $0 : $*"

exit 0
```

   Une fois le script rendu exécutable, lancez-le de différentes manières :
   - Sans paramètre.
   - Avec un seul paramètre.
   - Avec deux paramètres séparés par un espace.
   - Avec trois paramètres séparés par des espaces.
   - Avec quatre paramètres séparés par des espaces.

2. Créez un script nommé *AfficheRep* qui affiche les caractéristiques et le contenu d'un répertoire dont le nom est donné en paramètre du script.

3. Créez un script nommé *ChercheUser* qui cherche un utilisateur dont le nom de connexion est donné en paramètre au script.

4. Créez un script *vi2* qui prend en argument un nom de fichier. Le script réalise une sauvegarde du fichier dans le répertoire */tmp* avant de lancer l'éditeur de textes vi pour afficher et/ou modifier son contenu. Le fichier de sauvegarde aura comme nom *nom_parametre_1.numéro_processus_script_en_cours*.