Administration système

Chapitre 6 - Les scripts shell bash

Gorgoumack SAMBE

Université Assane Seck de Ziguinchor

Version 1.0 [1]

1. Novembre 2022

Objectifs

être capable de :

1. implémenter un script shell et le lancer ;
2. utiliser des expressions et des variables sur bash ;
3. utiliser des structures conditionnelles sur bash ;
4. utiliser des structures itératives sur bash ;
5. utiliser des fonctions sur bash.

Plan

Plan

Les bases

- Simple **fichier texte**
- Permet d'**automatiser** une série d'opérations.
- Constitué de :
  - une ou plusieurs commandes
  - variables et expressions
  - structures de contrôles
  - structures itératives
  - fonctions
  - ...

Mon premier script shell

Helloworld

1. **Édition**
   1. saisie du code dans un éditeur :

```
#!/bin/bash
# fichier : hello.sh
# Affiche un salut a l'utilisateur
echo "Bonjour"
```

   2. enregistrement : hello.sh

2. **Exécution**

   $ bash hello.sh

Plan

Shebang, commentaires et variables

- **shebang** : indique au système l'interpréteur à utiliser
  #!/bin/bash
  #!/usr/bin/python
- **Commentaires** : commence par #
- **Variables**
  - déclaration : mavar=valeur
  - accès : $mavar
- **Tableaux**
  - déclaration
    - tab=('val0' 'val1' 'val2')
    - tab=(['un']="v0" ['deux']="v1")
    - declare -a tableau
    - read -a tableau
  - Accès : ${tab[1]}
  - Opérations
    - indices
      ${nom-tableau[*]}
    - nombre d'éléments
      ${#nomtableau[*]}
    - éléments
      ${nom-tableau[@]}
    - ajout
      tableau[${#tableau[*]}]=element

Entrées/sorties et Expressions

- **Entrées/sorties**
  1. **Sorties** : echo
     echo donner une valeur
     echo bonjour $nom
  2. **Entrées** : read
     read -p "Votre Nom" nom
     echo Hello $nom
- **Expressions arithmétiques**
  1. **syntaxe à parenthèses** : $((expression))
     $ echo $((3*5))
  2. **syntaxe à crochet** :$[expression]
     $ echo $[x*y]
  3. **commande let** : let expression
     $ let n=3*5

Structures conditionnelles

- **La structure if**

```
if condition ; then
    Instructions
elif condition ; then
    Instructions
elif condition; then
    ...
else
    Alternative
fi
```

- **La structure case**

```
case EXPRESSION in
    CASE1) Commandes;;
    CASE2) Commandes;;
    ...
    CASEn) Commandes;;
esac
```

- **Exemple**

```
#!/bin/bash
read -p "saisir le mot l3in ou m1in : " var
if [ $var = "l3in" ] ; then
    echo "Licence 3 info"
elif [ $var = "m1in" ];then
    echo "Master 1 info"
else
    echo "formation inconnue"
fi
```

- **Exemple**

```
echo -n "Votre réponse :"
read REPONSE
case $REPONSE in
    O* | o*) REPONSE="OUI" ;;
    N* | n*) REPONSE="NON" ;;
    *) REPONSE="PEUT-ETRE !";;
esac
```

Test conditionnel

- Condition
  1. Commande **test** ou sa version étendue **[[condition]]**
  2. syntaxe entre crochet : **[condition]**
- Opérateurs

Numérique		Chaînes		Fichiers			
op.	sémantique	op.	sémantique	op.	sémantique	op.	sémantique
-eq	égalité	-z	chaîne vide	-e	existe	-f	fichier simple
-ne	inégalité	-n	chaîne non vide	-d	répertoire	-L	lien symbolique
-lt	inférieur	==	chaînes ident.	-s	fichier non vide	-r	lisible
-le	inf. ou égal	!=	chaînes diff.	-w	modifiable	-x	exécutable
-gt	sup. strict			-nt	plus récent que	-ot	plus vieux que
-ge	sup. ou égal			-O	on est owner du fichier	-G	on a le même groupe que le fichier

- Exemple :

```
if [[ x -gt 10 \&\& x -lt 20 ]] ; then
    echo "Moyenne"
fi
```

Structures itératives

- **La structure while**

```
while CONDITION;do
    COMMANDES
done
```

- **La structure until**

```
until CONDITION;do
    COMMANDES
done
```

- **La structure for**

```
for ((init.;cond.;action))
do
    instructions
done
```

```
for Variable in LIST;do
    COMMANDES
done
```

- **break** : quitter une boucle

- **Exemple**

```
while [ $reponse != 'oui' ];do
    read -p 'Dites oui : ' reponse
done
```

- **Exemple**

```
until [ $reponse == 'oui' ]; do
    read -p 'Dites oui : ' reponse
done
```

- **Exemples**

```
for((i=0;i<100;i=i+1));do
    echo $i
done
```

```
for variable in 'v1' 'v2' 'v3';do
    echo "La variable vaut $variable"
done
```

- **continue** : quitter une itération

Fonctions

- **Déclaration**
  1. ```
     function nomFonction {
         instructions
         ...
     }
     ```
  2. ```
     nomFonction {
         instructions
         ...
     }
     ```
- **Appel de fonction**

  nomFonction
- **Exemple :**

```
#!/bin/bash
#helloworld2.sh
#définition de la fonction
function hello
{
    echo "Bonjour tout le monde"
}
#appel de la fonction
hello
```

- **Paramètres de fonction/script**
  1. **$0** : nom de script/fonction ;
  2. **$1**, . . . , **$n** : arguments ;
  3. **$#** : nombre d'arguments ;
  4. **$*** : tous les arguments comme un mot ;
  5. **$@** : tous les arguments séparés.