

# Scripting Shell

## Module 06 – Exécution conditionnelle (test, if, case)



1

Scripting Shell



### Objectifs

- Comprendre les structures conditionnelles
- Écrire des tests
- Comparer des nombres, des chaînes



2

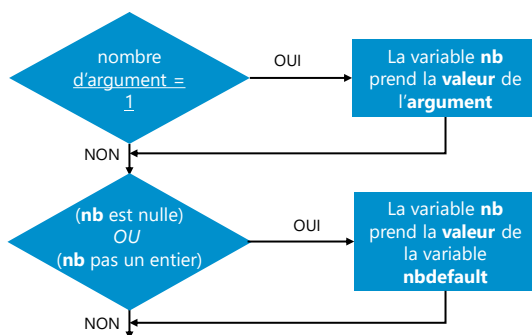
# Contextes outils et structures de conditionnement



3

## Contextes outils et structures de conditionnement

Dans de nombreux cas pratiques, la réalisation d'actions est conditionnée.



Dans cet exemple, la valeur de la variable **nb** est conditionnée

- Au nombre d'arguments passé au script
- Puis à la présence ou non de contenu dans cette variable.



4

- En Scripting Shell, pour traduire cet algorithme, nous utiliserons :
  - Une structure de contrôle comme la structure **if**
  - Une expression de test (qui sera intégrée) à la structure de contrôle.
- Ainsi, la première condition serait traduite ainsi :

Structure de contrôle if	Test de la condition	Structure complète
<pre>if condition ;then     nb=\$1 fi</pre>	<pre>[[ \$nbarg -eq 1 ]]</pre>	<pre>if [[ \$nbarg -eq 1 ]] ; then     nb=\$1 fi</pre>

- Dans ce module seront successivement abordées la structure **if** puis les expressions de tests.



# L'instruction if



L'instruction **if** est utilisée pour structurer la réalisation conditionnée de commandes.

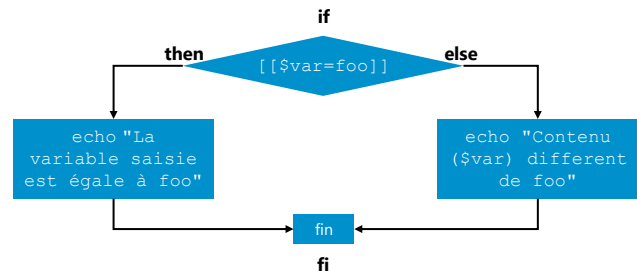
Si

*la condition est remplie*

**alors** je fais telle action

**sinon** je fais telle action

Fin du **si**



- Cette instruction requiert au minimum :
  - Un début : **if**
    - L'exécution d'une commande (un test de condition par exemple)
    - L'exécution d'au moins une action si la condition est vérifiée : **then**
  - Une fin : **fi**
- Cette instruction peut être complétée par :
  - L'exécution d'autres actions si la condition est vérifiée
  - L'exécution d'actions à réaliser si la condition n'est pas vérifiée : **else**



Convention de structure de la boucle **if**

- Il existe 2 syntaxes pour l'écriture de la structure **if** :

## Moderne

```
if condition ; then
    Action1
else
    Action2
fi
```

## Classique

```
if condition
then
    action1
else
    action2
fi
```

- Dans ce support, la forme moderne de la structure sera prise pour les exemples.
- Mais vous pouvez choisir la méthode de votre choix.

Utilisation minimale de **if**

- Dans l'exemple ci-dessous, la condition de réalisation de l'action d'affichage est conditionnée par le résultat du test.



```
if [[ "$LOGNAME" = root ]] ; then
    echo "Ne pas se connecter en root"
fi
```

- Autre exemple d'utilisation conditionné par le code retour d'une commande (**ls**) :




```
if ls /tmp ; then
    echo "/tmp existe"
fi
```



Utilisation de **if** avec **else**


**if** s'utilise souvent avec **else** pour choisir une action ou une autre



```
if [[ "$LOGNAME" = root ]] ; then
    echo "Ne pas se connecter en root"
else
    echo "Bienvenue sur la machine $HOSTNAME"
fi
```

Utilisation de **if** avec **else**

Il est possible de stipuler plusieurs actions dans une structure **if** :



```
if [[ "$LOGNAME" = root ]] ; then
    echo "Ne pas se connecter en root"
    echo "Le script va quitter"
    exit 1
else
    echo "Bienvenue sur la machine $HOSTNAME"
    echo "Le script continue"
fi
```



## Imbrication de structures **if**

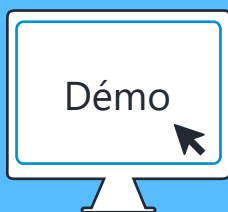
- Il est possible d'imbriquer des structures **if** entre elles. Dans l'exemple ci-dessous, on fait d'autres tests si le premier n'est pas rempli.
- Dans le cas d'imbrication d'un **if** dans un **else**, on peut utiliser la structure imbriquée **elif**.

### Imbrication de structures **if**

```
if <condition> ; then
    <action>
else
    if <condition> ; then
        <action>
    else
        <action>
    fi      # fin du second if
fi        # fin du premier if
```

### Utilisation du **elif**

```
if <condition> ; then
    <action>
elif <condition> ; then
    <action>
else
    <action>
fi      # fin commune des 2 if
```



L'instruction **if**



# Évaluation des conditions



- Le conditionnement de commandes s'appuie sur l'évaluation de conditions.
- Afin d'évaluer une ou plusieurs conditions, il est possible d'utiliser les commandes suivantes :

<code>test condition</code>	Commande externe de test utilisable quel que soit le Shell
<code>[ condition ]</code>	Identique à la commande test
<code>[[ condition ]]</code>	Commande interne à certains Shells (dont le Bash) qui prend en charge des fonctionnalités supplémentaires
<code>(( condition ))</code>	Commande réservée à l'utilisation de tests arithmétiques





- Suite à l'exécution de ces commandes, leur code retour peut avoir pour valeur :
  - 0 : l'expression est vraie
  - 1 : l'expression est fausse
- Dans ce cours, les tests sont réalisés au moyen de `[ [ ] ]`.
- Certains éléments abordés ne sont pas pris en charge par les autres commandes de tests. `[ ]` et `test`



- Lors de l'évaluation d'une condition, les critères d'évaluation sont liés au type d'élément à évaluer, celui-ci pouvant être :
  - Un entier (chiffre ou nombre)
  - Une chaîne (mot ou phrase)
  - Une composante du système de fichier (fichier, répertoire, etc.)
- Les principaux opérateurs de tests sont présentés dans ce cours.



Opérateurs de comparaison sur les **entiers**

- Prendre garde à ce que la variable testée soit déclarée en type entier (**typeset -i variable**).

<b>-eq</b>	égalité
<b>-ne</b>	différence
<b>-gt</b>	supérieur
<b>-ge</b>	supérieur ou égal
<b>-lt</b>	inférieur
<b>-le</b>	inférieur ou égal



```
if [[ $nbre -eq 2 ]] ; then
    echo 'nombre choisi : 2'
fi
```

Opérateurs de comparaison sur les **chaînes**

<b>=</b>	égalité
<b>!=</b>	différence
<b>-n \$nomvar</b>	variable non nulle
<b>-z \$nomvar</b>	variable nulle



```
if [[ $prenom = sofia ]] ; then
    echo 'Bienvenue Sofia'
fi
```



```
if [[ -z $prenom ]] ; then
    echo 'Merci de saisir
votre prénom'
fi
```



Opérateurs de comparaison sur les **fichiers**

<b>-d nomrep</b>	répertoire
<b>-f nomfic</b>	fichier ordinaire
<b>-w nomfic</b>	droit d'écriture
<b>-r nomfic</b>	droit de lecture
<b>-x nomfic</b>	droit d'exécution
<b>-s nomfic</b>	taille de nomfic > 0



```
if [[ -d $rep ]] ; then
    ls -l $rep
fi
```



```
if [[ -f $fic ]] ; then
    cat $fic
else
    echo "$fic absent"
fi
```



## Combinaison de critères

- Il est possible de cumuler plusieurs critères au sein d'un même test.
- On utilise alors les options suivantes :

<b>&amp;&amp;</b>	et logique
<b>  </b>	ou logique
<b>!</b>	négation
<b>( )</b>	regroupement



```
if [[ -f fic1 && ( -d rep1 || -d rep2 ) ]] ; then
    echo "fic1 est présent, ainsi que rep1 ou rep2"
fi
```



## Les caractères spéciaux dans les chaînes



23

## Les caractères spéciaux dans les chaînes

### Métacaractères du Shell

- Les expressions de test sont interprétées par le Shell.
- Il est donc possible d'utiliser des caractères spéciaux dans celles-ci.
- Rappel :

<b>*</b>	0 à n caractères
<b>?</b>	1 caractère quelconque
<b>[...]</b>	1 caractère parmi ceux entre crochets
<b>[^...]</b>	1 caractère autre que ceux indiqués entre crochets (ou [!...])



24

## Facteurs d'occurrence

? (...)	0 à 1 fois l'expression
* (...)	0 à n fois l'expression
+ (...)	1 à n fois l'expression
@ (...)	1 fois l'expression
! (...)	0 fois l'expression
* (...   ... )	De 0 à n fois l'expression 1 ou l'expression 2. Valable également pour les caractères ? * + @ !



```
chaine=+123
if [[ "$chaine" = ?{[+-]}+([0-9]) ]] ; then
    echo "$chaine est un nombre valide"
fi
```

Pour la prise en compte de facteurs d'occurrences, l'option **extglob** doit être activée dans le Shell d'interprétation du script.



```
$ shopt
$ shopt -s extglob
```



## Illustration d'un cas pratique



- Illustration de test valide ou non une expression :



```
[[ -d /data ]] && {  
    mkdir /data/perso  
    cp -a /home/penthiuim/documents/* /data/perso  
}
```

- Dans cet exemple, le test `[[ -d /data ]]` vérifie l'existence ou non du répertoire `/data`
- Si le répertoire existe, le code retour de test est égal à 0. Le regroupement de commandes qui suit est alors effectué.
- S'il n'existe pas, le code de sortie test diffère de 0. Le regroupement de commandes est ignoré.

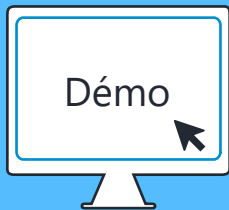


- Cet exemple pourrait être intégré dans une instruction `if` :



```
if [[ -d /data ]] ; then  
    mkdir /data/perso  
    cp -a /home/penthiuim/documents/* /data/perso  
fi
```





## Les opérateurs de tests



## La structure case



La structure de contrôle **case** est adaptée au test d'un contenu (généralement stocké dans une variable) pouvant contenir un ensemble de valeurs et des actions liées à ces valeurs.

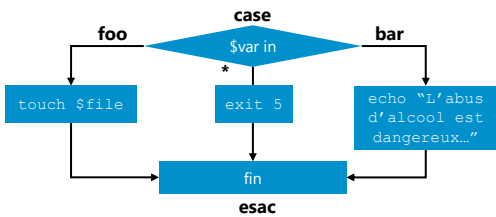


Algorithme d'illustration

La variable **var (\$var)** peut contenir les valeurs : « **foo** », « **bar** » ou toute autre chaîne.

<b>foo</b>	Une action de création de fichiers est réalisée
<b>bar</b>	Une action d'affichage est réalisée
<b>Toute autre valeur (*)</b>	Sortie du script avec un code d'erreur

Début du **case**  
On compare la valeur de **\$var** à "**foo**" ou à "**bar**" ou à "**\***"  
Si la valeur est "**foo**" je fais une action  
Si la valeur est "**bar**" je fais une autre action  
Si la valeur est "**\***" je fais une troisième action  
Fin du **case**





## Syntaxe d'utilisation



```
case $nomvar in
    expr)
        action1
        action2
        ;;
    expr2)
        action
        ;;
esac
```



33

## Exemple d'utilisation



```
case $saisie in
    p)
        echo "la lettre est p"
        ;;
    [0-9])
        echo "un chiffre"
        ;;
    a|b)
        echo "a ou b"
        echo "\"?"
        ;;
    *)
        echo "n'importe quel autre caractère"
        ;;
esac
```

La liste de valeurs possibles est interprétée par le Shell, ce qui autorise l'utilisation de métacaractères ou d'expressions complexes.



Pour le premier des cas vérifiés, les actions qui lui sont associées sont effectuées suivi de la sortie immédiate de la structure. Aucun autre cas ne sera appliqué.



34



- 
- La structure case
-