Scripting Shell

Module 04 – Les variables dans les scripts

Objectifs

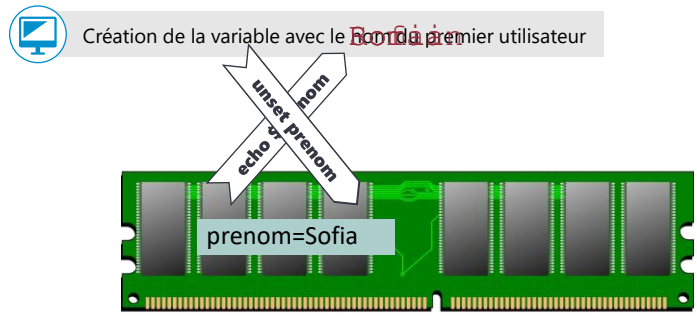
- Comprendre l'intérêt des variables
- Utiliser des variables dans ses scripts
- Manipuler les variables
- Récupérer une saisie utilisateur dans une variable

- Les variables sont des éléments essentiels à la réalisation de scripts.
- Elles permettent de stocker temporairement des informations au cours du déroulement du script.
- Ce contenu peut être affiché ou réutilisé par d'autres commandes.

Il est important de bien appréhender les mécanismes régissant les variables.

rappel | Fonctionnement d'une variable

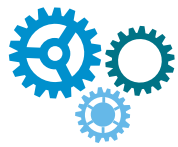
Etape 1



Les variables peuvent être classées en 4 catégories.

Les variables locales	Les variables d'environnement	Les variables réservées	Les constantes (variables en lecture seule)
Les noms doivent être en minuscule (utilisation de doubles quotes " pour séparer les mots)	Variable système (comme HOME) ou variable exportée	Il est impossible d'utiliser leurs noms pour des variables locales ou d'environnement	Variable en lecture seule via `declare -r <var>`. Leurs noms doivent être en majuscule (utilisation de "_" pour séparer les mots

Les principales actions réalisées sur les variables



- Affectation d'un contenu
- Utilisation de son contenu
- Exportation de la variable
- Réaffectation de son contenu

Utilisation de variables locales

Une variable locale est utilisable dans l'environnement courant.

- Pour affecter un contenu à une variable : **`nom=valeur`**

```
errormsg="une erreur est survenue, contactez votre administrateur système"
logdir="/var/log/"
```

- Pour utiliser le contenu d'une variable, on préfixe son nom par le caractère dollar `$` , généralement entouré de doubles quotes.

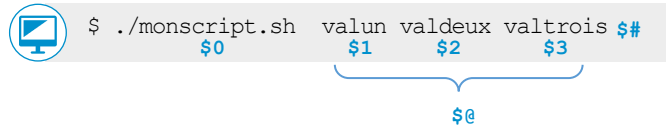
```
[...]
echo "$errormsg"
mkdir -p "$logdir" 2>/dev/null
```

Les variables réservées

- Ce sont des variables dont le nom est réservé par le Shell qui en gère le contenu.
- Elles sont très utilisées dans les scripts et les enchaînements de commandes.

$$	Numéro du processus en cours
$?	Code retour (0 si vrai, different de 0 si faux) `exit 2` -> provoque la sortie avec le code retour "2"
$!	Numéro du dernier processus lancé en arrière-plan
$#	Nombre de paramètres reçus par le script
$1 à $9	Paramètres de la commande (1 pour le premier, 2 pour le deuxième, etc.)
$0	Nom de la commande ou du script
$@ ou $*	Valeurs des paramètres reçus encadrés par " ". Certaines versions Unix concatènent les valeurs avec *.

Exemple



Variables affectées

$#	3 *Trois paramètres reçus*
$0	./monscript.sh
$1	valun
$2	valdeux
$3	valtrois
$@	valun valdeux valtrois

Modification du contenu d'une variable d'environnement

Modification du contenu d'une variable d'environnement

Une variable d'environnement est une variable qui est héritée dans tous les sous-Shell.

- Pour créer une variable d'environnement, on peut :
  - Soit exporter une variable locale

```
export NOMVAR
```

  - Soit créer (ou modifier) et exporter directement une nouvelle variable

```
export NOMVAR=valeur
```

Commande `set`

- La commande **set** permet notamment d'affecter des valeurs aux variables réservées 1, 2 ...

```
$ set "$LOGNAME" $(uname -n)
$ echo "$1 $2"
user20 serveur102
```

- Si aucun argument n'est donné, toutes les variables du Shell sont affichées.

Commande read

- La commande **`read`** est utilisée pour affecter un contenu saisi par l'utilisateur à une variable.

Syntaxe `Read [options] <variable(s)>`

- Cette commande génère l'attente de saisie clavier validée (par Entrée) et procède au stockage de cette saisie dans la (ou les) variable(s) spécifiée(s).

```
echo "Veuillez entrer votre nom : "
read nom
echo "Bonjour $nom"
```

- Il est possible de substituer la commande **`echo`** par l'option **`-p`** de **`read`**.

```
read -p "Veuillez entrer votre nom : " nom
```

- Si plusieurs paramètres séparés par des espaces sont fournis à **read**, alors chaque mot saisi est affecté à la variable correspondante. Les saisies en surnombre sont affectées au dernier nom de variable indiqué.

```
read -p " Saisissez votre nom et prénom : " nom prenom
       Thouin Frédéric dit penthium
echo "Nom saisi : $nom "
Nom saisi : Thouin
echo "Prénom saisi : $prenom "
Prénom saisi : Frédéric dit penthium
```

- Pour éviter ce type de désagrément, il est possible de déclarer une variable *tampon* non utilisée.

```
read -p " Saisissez votre nom et prénom : " nom prenom tampon
       Thouin frédéric dit penthium
[…]
echo "Prénom saisi : $prenom "
Prénom saisi : frédéric
```

- Il est éventuellement possible ultérieurement de détruire cette variable tampon

```
unset tampon
```

- Les variables