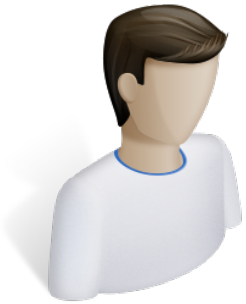
CS230 - Programmation shell sous Unix

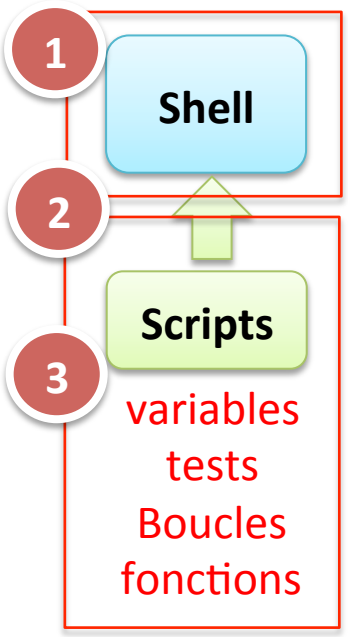
Cours 3 – Les scripts shell

Dr. Bassem DEBBABI

Objectif du cours 3



Utilisateur



Outils

Binaires exécutables

Appels système

Système de fichiers

Système d'exploitation UNIX

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Le Shell

- Un shell intègre :
  - Les notions de variable, d'opérateur arithmétique, de structure de contrôle, de fonction, présentes dans tout langage de programmation classique, mais aussi
  - Des opérateurs spécifiques (ex : |, ;)
- Mais attention :
  - On ne peut pas manipuler des structures de données complexes comme par exemple l'utilisation de pointeurs,
  - L'oubli ou l'ajout d'un caractère espace provoque facilement une erreur de syntaxe

Les scripts shell

- Écriture d'un script

```
#!/bin/bash

# mon premier script

echo –n "La date du jour est :"
date
```

- La notation #! En premier ligne d'un script *bash* précise au shell courant quel interpréteur doit être utilisé.
- La notation # indique un commentaire.
- On peu exécuter une commande directement par écrire son nom (ou son chemin absolu)

- Exécution d'un script
  - sh nomDuScript # ou
  - chmod u+x nomDuScript (attribution des droits d'exécution)
  - ./nomDuScript

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
    - Lecture du contenu
    - Affectation de valeurs
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Manipulation des variables

- Lecture
  - Pour rappeler le contenu d'une variable (locale ou d'environnement),
    - il suffit de faire précéder son nom par le caractère $
    - ou en utilisant ${var} si la variable est utilisée dans un texte.

```
$ echo $HOME
/home/debbabi/
$ a=hell
$ echo $a
Hell
$ b=world
$ echo ${a}o $b
hello world
```

Manipulation des variables

- Affectation de valeur
  - Affectation directe (=)
    - var1=12
    - var1="hello"
    - declare –r var # constante
    - var="abc"
  - La commande *read*
  - Substitution de commandes

Manipulation des variables

La commande « read »

Syntaxe $	`read variable [variable...]`	Lire les informations sur l'entéee standard et les affectés aux variables définies.

- l'espace est un séparateur au niveau de la ligne de commandes.
- L'option **–p** de **read** affiche une chaîne d'appel avant d'effectuer la lecture ;
  - read -p "Entrez une valeur:" v
- Lorsque la commande interne **read** est utilisée sans argument, la ligne lue est enregistrée dans la variable prédéfinie du shell **REPLY**.
- Exemples : on suppose qu'un script contient la commande suivante :
  - read var1 var2 var3
  - L'utilisateur saisit la chaîne suivante :

`chaine1 chaine2 chaine3`

Variable	Valeur
var1	chaine1
var2	chaine2
var3	chaine3

`chaine1 chaine2`

Variable	Valeur
var1	chaine1
var2	chaine2
var3	NULL

`chaine1 chaine2 chaine3 chaine4 chaine5`

Variable	Valeur
var1	chaine1
var2	chaine2
var3	chaine3 chaine4 chaine5

Manipulation des variables

- Substitution de commandes
  - Réassigne la sortie d'une commande ou même de multiples commandes à une variable
  - La forme classique de la substitution de commande utilise l'apostrophe inverse `...`

```
a_supprimer=`cat nomfichiers`
# nomfichiers est une fichier qui contient
# une liste de fichiers à effacer
```

  - La forme alternative d'une substitution de commande utilise la syntaxe $(...)

```
a_supprimer2=$(ls *.txt)
```

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Les paramètres

- Les paramètres de position
  - référencé par un ou plusieurs chiffres : 8 , 0 , 23
  - L'assignation de valeurs à des paramètres de position s'effectue :
    - soit à l'aide de la commande interne set
    - soit lors de l'appel d'un fichier shell ou d'une fonction shell.
  - ATTENTION :
    - on ne peut utiliser ni le symbole **=**, ni la commande interne **read** pour affecter directement une valeur à un paramètre de position.

Les paramètres

- La commande « set »

Syntaxe $	`set arg1...`	affecte une valeur à un ou plusieurs paramètres de position.

  - En numérotant ses arguments suivant leur position. La numérotation commence à **1**.

```
$ set ab cd ef gh
$ echo $1
ab
$ echo $4
gh
```

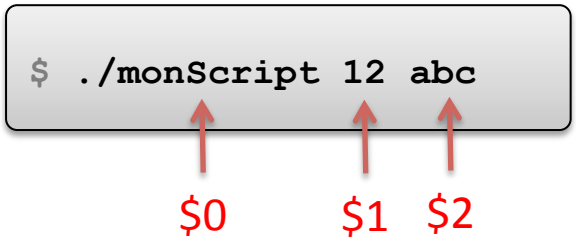
- La commande **set --** rend indéfinie la valeur des paramètres de position préalablement initialisés.
- La commande shift [n] décale la numérotation des paramètres de position ayant une valeur.

$**set a b c d e**
=> *1 2 3 4 5*

$**shift 2**
=> a b c d e
=> *1 2 3*

Les paramètres

- Dans un fichier shell, les paramètres de position sont utilisés pour accéder aux valeurs des arguments qui ont été passés lors de son appel :
  - cela signifie qu'au sein du fichier shell, les occurrences de $1 sont remplacées par la valeur du premier argument,
  - celles de $2 par la valeur du deuxième argument, etc.
  - Le paramètre spécial $# contient le nombre d'arguments passés lors de l'appel.
  - Le paramètre de position 0 contient le nom complet du programme shell qui s'exécute.



$# → 2

$* → "12 abc"

$@ → "12" "abc"

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Déclaration des tableaux de données

- Oui on peut manipuler les tableaux de données !

- Principe

  - MAYA=(un deux trois quatre) # déclaration du tableau
  - echo ${MAYA[2]} # affiche trois
  - echo $MAYA # affiche uniquement le premier élément
  - echo ${MAYA[*]} # affiche tous les éléments
  - Echo ${#MAYA[*]} # affiche le nombre d'éléments initialisés

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Les commandes de gestion des variables du shell

- set
  - La commande "set" sans arguments affiche la liste des variables locales au shell.
- variable=valeur
  - Définition d'une variable
- unset variable
  - La commande "unset" suivi d'un nom de variable permet d'effacer celle-ci de la zone des variables locales du shell.
- export variable
  - La commande "export" suivie du nom d'une variable, permet de placer une variable définie de la zone locale au shell vers la zone d'environnement.
- env et printenv
  - Les commandes "env" et "printenv" listent les variables de la zone d'environnement et les valeurs qui leur sont affectées.

Les variables les plus usuelles du shell Unix

Variable	Signification
PATH	Référence les chemins d'accès scrutés lors de l'exécution d'une commande
HOME	Référence le répertoire de login. C'est le répertoire par défaut de la commande "cd"
PWD	Le chemin du dossier courant
SHELL	Le nom du shell par défaut
USER	Le nom de l'utilisateur en cours
IFS	Séparateur interne du champ de saisie

- En général, les noms des variables utilisateur sont en lettres minuscules tandis que les noms des variables prédéfinies (du shell ou de commandes unix) sont en majuscules.

Plan du cours 3

- Écriture et exécution d'un script
- Les variables
  - Manipulation des variables
  - Les paramètres
  - Tableaux
  - Variables système
  - Variables spéciales

Variables spéciales

Variable	Valeur retournée
#	Nombre d'arguments
*	List des paramètres de positions sous forme d'une unique chaîne de caractères
@	produit autant de chaînes que de paramètres de positions initialisés
$	Numéro du process shell dans lequel s'exécute le script
!	Numéro du process de la dernière commande lancée en arrière plan (grâce au caractère &)
-	Liste des options du shell positionnées avec la commande set
?	Code de retour de la dernière commande exécutée
REPLY	Contient la valeur lu par la commande *read* si aucune variable n'est spécifiée.