

Systeme d'exploitation

Scripting Bash

(7)

Appel de scripts
Chaînes de caractères

Script appelant un script

- Les scripts peuvent s'appeler entre eux à la manière de fonctions :
 - le résultat peut être :
 - une modification de variables globales (*à l'interpréteur*)
 - un affichage de messages (*capturés par la suite*)
 - une valeur retournée
 - **return** : n'est utilisable que si le script est appelé par **source**
 - **exit** : attention si le script est appelé par **source**
(arrêt de l'interpréteur courant)

Retour de valeur

- Exécution par un **shell fils** et capture du retour
 - shell fils => espaces indépendants
 - pas de visibilité ascendante : appelant ne voit pas les variables de appele
 - pas de visibilité descendante : appele ne voit pas les variables de appelant

appelant.sh

```
a=deux
b=trois
c=$1
d=quatre
./appele.sh un
e=$?
echo "a=$a b=$b c=$c d=$d e=$e"
```

shell fils

appele.sh

```
a=2
b=$0
c=$1
echo "a=$a b=$b c=$c d=$d"
exit 4
```

\$d inconnu

```
./appelant.sh 1 :
a=2 b=./appele.sh c=un d=
a=deux b=trois c=1 d=quatre e=4
```

Capture de l'affichage

- Exécution par un **shell fils** et capture de l'affichage

appelant.sh

```
a=deux  
b=trois  
c=$1  
d=quatre  
e=$(./appele.sh un)  
echo "a=$a b=$b c=$c d=$d e=$e"
```

shell fils

appele.sh

```
a=2  
b=$0  
c=$1  
echo "a=$a b=$b c=$c d=$d"  
exit 4
```

\$d inconnu



./appelant.sh 1 :

a=2 b=./appele.sh c=un d=

a=deux b=trois c=1 d=quatre e=a=2 b=./appele.sh c=un d=

Exécution par source

- Exécution par le même shell : lancement par source
 - variables globales au shell => partagées

appelant.sh

```
a=deux
b=trois
c=$1
d=quatre
source ./appele.sh un
e=?
echo "a=$a b=$b c=$c d=$d e=$e"
```

un seul shell



appele.sh

```
a=2
b=$0
c=$1
echo "a=$a b=$b c=$c d=$d"
return 4
```

./appelant.sh 1

a=2 b=./appele.sh c=un d=quatre

a=2 b=./appele.sh c=un d=quatre e=4

Appel récursif

- Un script peut s'appeler lui-même
 - Exemple : Script récursif réduit à une fonction
 - `./liste.sh *` : *affiche le contenu du répertoire courant*

liste.sh

```

if ! [ -z $1 ]
then
  echo $1
  shift
  $0 $*
fi
exit
  
```

appel du script par le script

Initialisation des variables : **`${...}`**

`${var}` : pour éviter une ambiguïté

`a=A ; echo ${a}b` \Rightarrow `Ab` (*alors que* `echo $ab` \Rightarrow rien)

`${var?}` : message d'erreur si non définie + **`exit 1`**

`${a?}` \Rightarrow ...: *a: paramètre vide ou non défini*

`${var?message}` : idem à ci-dessus en affichant un message spécifique

`${a?non affecté}` \Rightarrow ...: *a: non affecté*

`${var-val}` : valeur de *var* si déjà définie, *val* dans le cas contraire

`a=A ; c=${a-X} ; d=${b-X}`

`echo "a=$a b=$b c=$c d=$d"` \Rightarrow `a=A b= c=A d=X`

`${var=val}` : *var* si déjà définie, *val* sinon **`et $var`** se voit affectée la valeur de *val*

`a=A ; c=${a=X} ; d=${b=X}`

`echo $a $b $c $d` \Rightarrow `a=A b=X c=A d=X`

Variable = commande

- **eval** : exécuter une commande dont le nom est dans une variable.

- Exemple :

```
instruction=date  
echo $instruction  
eval $instruction
```

```
# variable contenant la chaîne "date"  
=> affiche date  
=> affiche la date courante  
équivalent à : echo $($instruction)
```

Ex : sécurisation de commandes saisies en ligne (test préalable)

```
echo "Saisissez vos commandes (exit pour sortir)"  
while read -p "Entrez une commande > " commande  
do  
    eval $commande &>/dev/null || # test exécution correcte  
    { echo "Commande invalide"; continue; }  
    eval $commande # exécution de la commande  
done
```


Suppression dans des **variables** chaînes

- **Suppression** dans une **variable** chaîne des sous-chaînes fidèles au modèle
(métacaractères linux *pas expression régulière*)
 - `${chaîne#modele}` supprime la plus **courte** sous-chaîne **gauche**
 - `${chaîne##modele}` supprime la plus **longue** sous chaîne **gauche**
 - `${chaîne%modele}` supprime la plus **courte** sous-chaîne **droite**
 - `${chaîne%%modele}` supprime la plus **longue** sous-chaîne **droite**
- Exemples avec `var="123 abc 456 def"`
 - `${var#[0-9]}` : supprime jusqu'au 1^{er} chiffre : **23 abc 456 def**
 - `${var##*[0-9]}` : supprime jusqu'au dernier chiffre : **def**
 - `${var%[0-9]*}` : supprime jusqu'au 1^{er} chiffre (par la droite) **123 abc 45**
 - `${var%%*[0-9]*}` : supprime toute la chaîne

Traitement de **variables** chaînes (*bash*)

- **Extraction** d'une sous-chaîne à une position donnée
 - **`${chaîne:indice}`** : sous chaîne débutant à *l'indice* spécifié
 - **`${chaîne:indice:longueur}`** : idem en spécifiant la longueur

Ex : `texte="choucroute" ; echo ${texte:4} => croute`
- **Remplacement** d'occurrences d'une sous-chaîne
 - **`${chaîne/old/new}`** : remplace la 1^{ère} occurrence de *old* par *new*
 - **`${chaîne//old/new}`** : remplace toutes les occurrences de *old* par *new*
 - **`${#chaîne}`** : **Longueur** d'une chaîne contenue dans une variable

Ex :

 - `texte="rantanplan" ; echo ${texte/an/o} => rotanplan`
 - `texte="rantanplan" ; echo ${texte//an/o} => rotoplo`
 - `slt="Bonjour" ; echo ${#slt} # le $ n'est pas repris , affiche 7`
 - Rmq : `${#1}` longueur du 1^{er} paramètre (\$1)