

Programmation SHELL

Syntaxe Standard :

Commentaires

\$0 : Nom de l'exécutable

\$1 .. \$9 : Arguments passés lors de l'exécution

\$# : Nombre des arguments

\$* : Concaténation de tous les arguments en une seule chaîne

\$@ : Tous les arguments (mieux pour les parcours)

Var=val # Créer une variable / Lui affecter une valeur (pas d'espaces)

\$Var # Accéder à la valeur d'une variable

↳ echo \$Var # Afficher la valeur d'une variable

read Var # Affecter l'entrée de l'utilisateur à une variable

read Var1 Var2 ... # Lire un mot (de la ligne entrée) par variable

\$((expr_arith)) # Évaluer une expression arithmétique (sans espaces et sans \$)

`commande` # Évaluer le résultat d'une commande

↳ Var=`commande` # Affecter le résultat d'une commande à une variable

\$? : Déroulement de la dernière commande effectuée (0 : réussie | ≠0 : échouée)

exit n # Quitter le programme en indiquant son déroulement par n

ligne1 ; ligne2 # Séparer des lignes du script (; ≡ nouvelle ligne)

Tests sur les fichiers :

-e f : existe	-r f : lisible	-d f : dossier
-s f : non vide	-w f : modifiable	-f f : fichier
-z f : vide	-x f : exécutable	-L f : lien symbo
f1 -nt f2 : plus récent	f1 -ot f2 : plus ancien	f1 -ef f2 : même fichier
-O f : appartient à l'utilisateur	-G f : appartient au groupe	

Tests sur les entiers : e1 -option e2

-eq : = -ge : ≥ -gt : > -le : ≤ -lt : < -ne : ≠

Tests sur les chaînes :

-n "ch" : non vide	"ch1" = "ch2" : identiques
-z "ch" : vide	"ch1" != "ch2" : différent

Combinaisons de tests :

```
test condition 1 && test condition 2 ≡ [ condition 1 ] && [ condition 2 ]
```

```
test condition 1 || test condition 2 ≡ [ condition 1 ] || [ condition 2 ]
```

Structure IF :

```
if test condition 1 ≡ if [ condition 1 ] (espaces requises)
```

```
then
```

```
    traitement 1
```

```
elif condition 2
```

```
then
```

```
    traitement 2
```

```
else
```

```
    traitement par défaut
```

```
fi
```

Boucle WHILE :

```
while condition
```

```
do
```

```
    traitement
```

```
done
```

Boucle FOR :

```
for Var ≡ for Var in "$@" (parcourir les arguments)
```

```
do
```

```
    traitement
```

```
done
```

```
—
```

```
for Var in val1 val2 val3 ... (parcourir les valeurs val1, val2, val3, etc.)
```

```
—
```

```
for Var in {vali..valf} (parcourir les valeurs entre vali et valf)
```

```
—
```

```
for((Var = vali; Var <= Valf; Var++)) (syntaxe C)
```

Structure SWITCH :

```
case $Var in
  exprReg1)
    traitement 1
    ;;
  exprReg2)
    traitement 2
    ;;
  *)
    traitement par défaut
esac
```

Lecture d'un fichier ligne par ligne :

```
while read Var
do
    traitement
done < Fichier
```

Variables d'environnement :

\$USER : Nom de l'utilisateur courant
\$HOME : Chemin du répertoire personnel
\$PATH : Liste de chemins des répertoires contenant les exécutable accessibles au SHELL