Initiation à Unix

Cours 2 - Programmation Shell

Université Henri Poincaré

Master Génomique et Informatique

Automne 2008

Introduction

- L'interprète de commandes (shell) permet d'interagir avec le système
- Exécution de commandes (modification / consultation de l'état du système)
- Utilisation avancée : combinaison de commandes par exemple en redirigeant les entrées/sorties
- Programmation shell : combinaison de commandes au sein d'un *script* dans le but d'automatiser certains tâches
- Un Script shell correspond à un fichier exécutable d'extension .sh et débutant par : `#!/bin/sh`

Plan

Les variables

- Association entre un nom et un contenu (chaîne de caractères, nombre entier)
- Affectation d'une valeur à une variable au moyen du symbole =
  Exemple: `$ x="ceci est une variable"`
- Lecture de la valeur d'une variable au moyen du symbole $
  Exemples:
  `$ echo $x`
  `$ echo "oui, "$x`
- NB: les noms de variables peuvent être des entiers, dans ce cas, ils doivent être entourés de {} à partir de 10 :
  `$ echo ${10}`
- Attention : une variable non-définie contient la chaîne vide!

Les variables (suite)

- Découpage, dans le contenu d'une variable, de la plus courte chaîne de caractères satisfaisant un motif au moyen de l'opérateur #. Exemple :
  `$ x="ceci est une variable de cecette"`
  `$ echo ${x#*ce}` → `tte`

- Découpage de la plus longue chaîne au moyen de ##. Exemple :
  `$ x="ceci est une variable de cecette"`
  `$ echo ${x##*ce}` → `ci est une variable de cecette`

- Découpage de la fin de chaîne via % et %%. Exemple :
  `$ x="ceci est une variable de cecette"`
  `$ echo ${x%ce*}` → `ceci est une variable de ce`
  `$ echo ${x%%ce*}` →

Les variables (suite)

- Possibilité de passer des paramètres à un script shell au moyen d'**arguments**
- Les arguments sont définis dans la ligne de commande, à la suite du nom de l'exécutable, et séparés par des espaces
  Exemple : `$ print.sh fichier`
- Dans le script, le nom de l'exécutable est associé à la variable $0, les arguments aux variables $1, $2, ...
- La variable $# contient le nombre entier d'arguments du script
- La variable $* contient la concaténation de tous les arguments

Plan

Expressions arithmétiques

- Le shell peut évaluer des expressions arithmétiques délimitées par `$(( ))`
- Exemple:

```
$ n=1
$ echo $(( n + 1 ))
$ p = $(( n * 5 / 2 ))
$ echo $p
```

Plan

Evaluation de commandes

- Possibilité de stocker le résultat d'une commande dans une variable
- Utilisation de la *backquote* (‘)

  Exemple :

```
$ n=‘ls | wc -l‘
$ echo $n
```

  $\longrightarrow$ 50

Plan

Structures de contrôle

Instruction `if`

Syntaxe 1 :

```
if [ condition ]
then
    action1
fi
```

Syntaxe 2 :

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

Structures de contrôle (suite)

Exemples :

```
if [ $# = 0 ]
then
    echo "$0 : Aucun argument recu !"
fi

if cp "$1" "$1%"
then
    echo "sauvegarde de $1 reussie"
else
    echo "sauvegarde du fichier $1 impossible"
fi
```

Structures de contrôle (suite)

Instructions `if` imbriquées

```
if [ condition 1 ]
then
    action1
elif [ condition 2 ]
then
    action2
elif [ condition 3 ]
    action3
...
else
    actionN
fi
```

Structures de contrôle (suite)

Instruction `for`

```
for var in liste
do
    commandes
done
```

Exemple :

```
for file in *.sh
do
    cat $file
done
```

Structures de contrôle (suite)

Instruction `while`

```
while [ condition ]
do
    commandes
done
```

Exemple :

```
while [ "$var1" != "fin" ]
do
  echo "Variable d'entrée #1 (quitte avec fin) "
  read var1
  echo "variable #1 = $var1"
  echo
done
```

Structures de contrôle (suite)

Instruction case

```
case valeur_de_variable in
  val1)
    commandes
    ;;
  val2)
    commandes
    ;;
    ...
  *)
    commandes
esac
```

Structures de contrôle (suite)

Exemple #1 :

```
case $# in
  0) echo "aucun parametre"
     echo "Syntaxe :  $0 <nom d'utilisateur>";;
  1) echo "1 parametre passe au programme : $1";;
  2) echo "2 parametres passes au programme : $1 et $2";;
  *) echo "TROP DE PARAMETRES !"
esac
```

Structures de contrôle (suite)

Exemple #2 :

```
echo "Voulez vous continuer le programme ?"
read reponse
case $reponse in
  [yYoO]*) echo "Ok, on continue";;
  [nN]*) echo "$0 arrete"
         exit 0;;
  *) echo "ERREUR de saisie"
     exit 1;;
esac
```

Plan

Opérateurs de comparaison

Tests sur les fichiers (et sur les répertoires) :

- -e fichier Vrai si le fichier/répertoire existe
- -s fichier Vrai si le fichier à une taille supérieure à 0
- -z fichier Vrai si le fichier fait 0 octet (donc si il est vide)
- -r fichier Vrai si le fichier/répertoire est lisible
- -w fichier Vrai si le fichier/répertoire est modifiable
- -x fichier Vrai si le fichier est exécutable ou si le répertoire est accessible
- -O fichier Vrai si le fichier/répertoire appartient à l'utilisateur
- -G fichier Vrai si le fichier/répertoire appartient au groupe de l'utilisateur
- -b nom Vrai si nom représente un périphérique (pseudo-fichier) de type bloc (disques et partitions de disques généralement)

Opérateurs de comparaison (suite)

Tests sur les fichiers (et sur les répertoires, suite) :

- -c nom — Vrai si nom représente un périphérique (pseudo-fichier) de type caractère (terminaux, modems et port parallèles par exemple)
- -d nom — Vrai si nom représente un répertoire
- -f nom — Vrai si nom représente un fichier
- -L nom — Vrai si nom représente un lien symbolique
- -p nom — Vrai si nom représente un tube nommé
- f1 -nt f2 — Vrai si les deux fichiers existent et si f1 est plus récent que f2
- f1 -ot f2 — Vrai si les deux fichiers existent et si f1 est plus ancien que f2
- f1 -ef f2 — Vrai si les deux fichiers représentent un seul et même fichier

Opérateurs de comparaison (suite)

Tests sur les entiers :

- entier1 -eq entier2 Vrai si entier1 est égal à entier2
- entier1 -ge entier2 Vrai si entier1 est supérieur ou égal à entier2
- entier1 -gt entier2 Vrai si entier1 est strictement supérieur à entier2
- entier1 -le entier2 Vrai si entier1 est inférieur ou égal à entier2
- entier1 -lt entier2 Vrai si entier1 est strictement inférieur à entier2
- entier1 -ne entier2 Vrai si entier1 est différent de entier2

Opérateurs de comparaison (suite)

Tests sur les chaînes de caractères :

- -n "chaîne" Vrai si la chaîne n'est pas vide
- -z "chaîne" Vrai si la chaîne est vide
- "chaine1" = "chaine2" Vrai si les deux chaînes sont identiques
- "chaine1" != "chaine2" Vrai si les deux chaînes sont différentes

Ressources en ligne

- les pages man !
- `http://www.linux-france.org/article/memo/node80.html`
- `http://www-gtr.iutv.univ-paris13.fr/Cours/Mat/Systeme/TDTP2003/tp03.html`
- `http://pagesperso-orange.fr/gleu/absfr.tuxfamily.org/abs-2.3-fr/`
- Remerciements: ce cours a été réalisé à partir du support de Catherine Eng.