

# Unix: Scripts Shell

ASR1 Système

Département d'informatique  
IUT Bordeaux 1

Novembre 2010

# Fonction et Structures de contrôle

# Fonction et Structures de contrôle

## 1 Fonction

## 2 Décisions

- Code de retour (exit status)
- if-then-else-fi
- La commande `test`
- Enchaînement conditionnel
- if-then-elif-else, forme générale

## 3 Structure de contrôle : case

- Présentation
- Motifs d'un case

## 4 Boucle while

- Boucle while
- Syntaxe
- Exemple
- Boucle while-read
- break

## 5 Boucle for

# Fonctions

Un script peut comporter des **fonctions**, avec des **paramètres positionnels**

## Syntaxe

```
function nom-de-fonction
{
    commande
    commande
    ...
}
```

## Avantages :

- découpage logique,
- code plus facile à lire
- fonctions réutilisables

# Fonctions

## Exemple

```
1 #!/bin/bashrc
2
3 function archiver
4 {
5     tar -czf /var/svgd/$1.tgz $2
6 }
7
8 archiver photos /home/login/photos
9 archiver musique /home/login/musique
```

# Fonctions

Remarque : Par défaut, les variables sont **communes** (globales)

## Exemple 2

```
1  #!/bin/bashrc
2  destination=/var/svgd
3
4  function archiver
5  {
6      tar -czf $destination/$1.tgz $2
7  }
8
9  archiver photos /home/login/photos
10 archiver musique /home/login/musique
```

# Variables locales

On peut déclarer des **variables locales** dans une fonction

## Exemple

```
1 #!/bin/bashrc
2 destination=/var/svgd
3
4 function archiver
5 {
6     local nom=$(basename $1)
7     tar czf $destination/$nom.tgz
8 }
9
10 archiver /home/login/photos
11 archiver /home/login/musique
```

# Code de retour (exit status)

- Le **code de retour** d'un programme indique si il s'est bien terminé.
- Le code de retour de la dernière commande est dans la variable \$?
- Par convention : 0 = OK.

## Exemple

```
$ ls -ld /tmp
drwxrwxrwt 10 root root 12288 déc 22 17:32 /tmp
$ echo code de retour = $?
code de retour = 0
```

```
$ ls -l qdsdq
ls: ne peut accéder à qdsdq: Aucun fichier ou répertoire de ce type
$ echo code de retour = $?
code de retour = 2
```

# Codes de retour et documentation

Les codes de retour des commandes sont décrits dans le manuel

Exemple : `man ls`

...

```
Exit status is 0 if OK, 1 if minor problems,  
2 if serious trouble.
```

# exit

- Par défaut, un script retourne le code de sa dernière commande
- Il peut retourner un code spécifique par `exit [code]`

## Exemple

```
1 #!/bin/bash
2 echo "Something strange happened"
3 exit 42
```

# Structure de contrôle `if`

La structure de contrôle `if-then-else-fi` utilise le code de retour d'une commande

## Exemple

```
1 # usage:
2 #   compiler.sh prefixe
3
4 if g++ -o $1 $1.cc
5 then
6     echo " la compilation s'est bien passée"
7 else
8     echo " il y a eu un problème"
9 fi
```

# La commande test

Le code de retour de la commande test dépend d'une **condition**.

**Exemple** : `test -f nomFichier`  
indique si le fichier existe.

Application : compiler.sh

```
1  #!/bin/bash
2
3  if test -f $1.cc
4  then
5      g++ -Wall -o $1 $1.cc
6      echo Compilation terminée
7  else
8      echo "Erreur : _pas_de_fichier_$1.cc"
9      exit 1
10 fi
```

# La commande test

Autre notation : [ condition ]

## Différents tests

```
[ -d nom ]           # nom désigne un répertoire
[ chaine1 = chaine2 ] # comparaison de chaînes
[ chaine1 != chaine2 ]
[ chaine1 \
```

# Exercice

Écrire une commande qui affiche le maximum de deux paramètres.

## Exemple d'exécution

```
$ max.sh 37 421  
421
```

# Enchaînement conditionnel

## Syntaxe

```
COMMANDE1 && COMMANDE2
```

```
COMMANDE1 || COMMANDE2
```

Exécute la seconde commande seulement si la première a réussi (&&) ou échoué (||)

## Exemple

```
g++ prog.cc && echo OK
```

## Application : max.sh

```
1  #!/bin/bash
2
3  max=$1
4  [ $2 -ge $max ] && max=$2
5  echo $max
```

# if-then-elif-else, forme générale

## Syntaxe

```
if condition
then
    COMMANDES
elif condition
then
    COMMANDES
...
else
    COMMANDES
fi
```

# Structure de contrôle case

La structure de contrôle case choisit les commandes à exécuter en fonction d'un **sélecteur**

- semblable au `switch` de C++
- le sélecteur est une chaîne de caractères

## Exemple

```
1  #!/bin/bash
2  # Usage : archiver nom-de-repertoire
3
4  echo "Format _=_normal_gz_?"
5  read format
6  case "$format" in
7      gz)
8          option=z ;
9          suffixe=tgz ;
10         ;;
11     normal | "" )
12         option= ;
13         suffixe=tar ;
14         ;;
15     *)
16         echo "format '$format' non reconnu" >&2
17         exit 1
18         ;;
19  esac
20  prefixe=$(basename $1)
21  tar -c{option}f $prefixe.$suffixe $1
```

# Motifs d'un case

Plusieurs motifs pour un même cas

## Exemple

```
case $response in
oui | o )
    echo "d'accord"
    ;;
non | n )
    echo "tant pis"
    ;;
esac
```

# Motifs d'un case

Les motifs du case peuvent être des expressions régulières

## Exemple

```
case $response in
[o0][Uu][iI] | [o0] )
    echo "d'accord"
    ;;
[nN][o0] )
    echo "tant pis"
    ;;
*)
    echo "quoi ?"
    ;;
esac
```

# Boucle while

La boucle `while` (tant que) exécute les commandes de manière répétée tant que la première commande réussit

## Structure

```
while COMMANDE
do
  COMMANDES
done
```

# Boucle while

## Exemple

```
1  #!/bin/bash
2  # fac nombre
3  # calcule la factorielle d'un nombre
4  # exemple:   fac 5
5
6  i=1
7  f=1
8  while [ $i -le $1 ]
9  do
10     let f=f*i
11     let i++
12 done
13 echo $f
```

# Boucle while-read

Permet d'exploiter le contenu d'un fichier ligne par ligne.

## Structure

```
while read ligne
do
    commandes
done < fichier
```

## Exemple

```
while read numero nom
do
    printf "| %12s | %-30s |\n" $numero $nom
done < $nomFichier
```

# L'instruction break

L'instruction `break` permet de sortir d'une boucle

## Exemple

```
while true
do
    ....

    echo "voulez-vous arrêter ?"
    read reponse
    [ $reponse = oui ] && break
    ...
done
```

# Boucle for

La boucle for définit une variable prend les valeurs d'une liste de mots.

## Forme générale

```
for VAR in LISTE
do
    COMMANDE
    COMMANDE
    ....
done
```

## Exemple 1

```
for f in *.cc
do
    astyle --style=gnu $f
done
```

# Boucle for

## Exemple 2

```
for f in *.cc
do
    echo "le fichier $f contient $(wc -l $f) lignes"
done
```

**Note :** Une boucle for permet aussi de lire un fichier ligne par ligne ou mot par mot

## Lire un fichier ligne par ligne

```
old_IFS=$IFS # sauvegarde du séparateur de champ
IFS=$'\n'    # nouveau séparateur de champ (fin de ligne)
for ligne in $(cat fichier)
do
    commande
done
IFS=$old_IFS # rétablissement du séparateur de champ
```

# Boucle for : Exercices

Écrire une commande qui calcule la somme de ses paramètres (en nombre illimité)

## Exemple d'exécution

```
$ somme 100 3 20  
123
```

Analyser le script

## Script

```
r=1  
for i in $(seq 1 $1)  
do  
    let r*=i  
done
```

A quoi sert la commande seq ?

# Exercices

## Analyser le script

### Script

```
for f in $(find ~ -name '*.cc' -ctime -7)
do
  ls -l "$f"
done
```

- Quel est le rôle de `find ~`
- Quel est le rôle de `find ~ -name '*.cc'`
- Quel est le rôle de `find ~ -name '*.cc' -ctime -7r`
- Pourquoi met-on les guillemets dans `ls` ?