Fonctions

Introduction au Bash

Fonctions — Principes

- Une fonction est
  - Une encapsulation d'une série de lignes de code pour une exécution future
  - Appelée comme une simple commande
  - Exécutée séquentiellement, il n'y a pas de création de nouveau processus (contrairement au cas de l'appel d'une commande)
- Une fonction permet
  - Plus de légèreté dans le code, suppression des répétitions de blocs
  - La récurrence

Fonctions — Déclaration

```
# Déclaration de la fonction 'salut'
#
function salut() # nom
{                # Début
  # ...
  # Code
  # ...
}                # Fin
```

```
biotope% ./script.sh
Salut !
biotope%
```

```
#!/bin/bash

# Déclaration
#
function salut() {
  echo Salut \!
}

# Appel
#
salut
```


Fonctions — Variables

```
#!/bin/bash

var1=23

function ajoute_8_a_var1() {
  echo "fonction, var1 : $var1"
  var1=$(($var1 + 8))
  echo "fonction, var1 : $var1"
}

echo "var1 : $var1"
ajoute_8_a_var1
echo "var1 : $var1"
```

```
biotope% ./script.sh
var1 : 23
fonction, var1 : 23
fonction, var1 : 31
var1 : 31
biotope%
```

Fonctions Variables

```
#!/bin/bash

function affiche() {
  var2=27
  echo fonction, var2 : $var2
}

echo var2 : $var2
affiche
echo var2 : $var2
```

```
biotope% ./script.sh
var2 :
fonction, var2 : 27
var2 : 27
biotope%
```


Fonctions Variables

```
#!/bin/bash

function affiche() {
  var2=27
  echo fonction, var2 : $var2
}

affiche
echo var2 : $var2
var2=$(($var2 + 12))
echo var2 : $var2
affiche
echo var2 : $var2
```

```
biotope% ./script.sh
fonction, var2 : 27
var2 : 27
var2 : 39
fonction, var2 : 27
var2 : 27
biotope%
```


Fonctions Variables

```
#!/bin/bash

function affiche() {
  local var2=27
  echo fonction, var2 : $var2
}

affiche
echo var2 : $var2
var2=$(($var2 + 12))
echo var2 : $var2
affiche
echo var2 : $var2
```

```
biotope% ./script.sh
fonction, var2 : 27
var2 :
var2 : 12
fonction, var2 : 27
var2 : 12
biotope%
```


Fonctions Paramètres

- Les paramètres sont passés sur la ligne d'appel de la fonction
- Les paramètres sont accessibles sous forme de variables portant le numéro du paramètre

  Ex. $1, $2 et $3 dénotent respectivement les paramètres 1, 2 et 3 passés à la fonction
- La variable $# contient le nombre de paramètres passés à la fonction
- La variable $* contient l'ensemble des paramètres sous forme de liste

Fonctions Paramètres

```
#!/bin/bash

function affiche() {
  echo fonction, paramètre1 : $1
  echo fonction, paramètre2 : $2
}

affiche 12 17
echo \#
affiche 12
echo \#
affiche 12 17 24
```

```
biotope% ./script.sh
fonction, paramètre1 : 12
fonction, paramètre2 : 17
#
fonction, paramètre1 : 12
fonction, paramètre2 :
#
fonction, paramètre1 : 12
fonction, paramètre2 : 17
biotope%
```


Fonctions Paramètres

```
#!/bin/bash

function affiche() {
  echo nombre de paramètres : $#
  echo liste des paramètres : $*
}

affiche 12 17 16 24 a 11 "chaîne de caractères"
```

```
biotope% ./script.sh
nombre de paramètres : 7
liste des paramètres : 12 17 16 24 a 11 chaîne de caractères
biotope%
```


Fonctions Paramètres

```
#!/bin/bash

function affiche() {
  local compteur=1
  echo arguments : $#
  for i in $*;
  do
    echo "arg($compteur) : $i"
    compteur=$(($compteur + 1))
  done
}

affiche 12 17 24 32 a 3
```

```
biotope% ./script.sh
arguments : 6
arg(1) : 12
arg(2) : 17
arg(3) : 24
arg(4) : 32
arg(5) : a
arg(6) : 3
biotope%
```


Fonctions Paramètres

```
#!/bin/bash

function addition() {
  local total=0
  for i in $*;
  do
    total=$(($total + $i))
  done
  echo total : $total
}

addition 1 2 3 4 5 6
```

```
biotope% ./script.sh
total : 21
biotope%
```

```
0 + 1   = 1
1 + 2   = 3
3 + 3   = 6
6 + 4   = 10
10 + 5  = 15
15 + 6  = 21
21
```

Fonctions Récurrence

```
#!/bin/bash

function inc() {
  echo -n "$1 "
  inc $(($1 + 1))
}

inc 0
```

ATTENTION :
INTERDICTION D'ESSAYER
CE SCRIPT ☺

```
biotope% ./script.sh
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41^C
biotope%
```


Fonctions Récurrence

```
#!/bin/bash

function rebour() {
  if [ $1 -gt 0 ];
  then
    echo $1
    rebour $(($1 - 1))
  else
    echo Terminé
  fi
}

rebour 10
```

```
biotope% ./script.sh
10
9
8
7
6
5
4
3
2
1
Terminé
biotope%
```


Interception des Signaux

Introduction au Bash

Interception des Signaux Signaux

- SIGINT – (valeur : 2)
  - Interruption reçue du clavier
    - Crtl-C
    - kill –2
- SIGTERM – (valeur : 15)
  - Signal de terminaison
    - kill
- SIGKILL – (valeur : 9)
  - Signal de mort sans appel (insaisissable)
    - kill -9
    - kill -KILL

Interception des Signaux 'trap'

```
#!/bin/bash

# Interception du signal INT,
# envoyé par défaut par Crtl-C
# Ici, le signal est ignoré

trap '' INT

while [ 1 ];
do
  sleep 1
done
```

```
biotope% ./script.sh
^C
^C
^C
^Z

[1]+ Stopped     ./script.sh
biotope% kill %1

[1]+ Stopped     ./script.sh
biotope%
[1]+ Terminated  ./script.sh
biotope%
```


Interception des Signaux 'trap'

```
#!/bin/bash

trap niet TERM

function niet() {
  echo Non, je ne meurs pas
}

while [ 1 ]; do
  sleep 1
done
```

```
biotope%
biotope% ./script.sh &
[1] 27109
biotope% kill %1
biotope% Terminated
Non, je ne meurs pas

biotope% kill %1
biotope%
Non, je ne meurs pas

biotope%
```


Interception des Signaux 'trap'

```
#!/bin/bash

trap niet TERM

function niet() {
  echo Non, je ne meurs pas
}

while [ 1 ]; do
  sleep 1
done
```

```
biotope% kill -9 %1
biotope%
[1]+ Killed      ./script.sh
biotope%
```


Passage d'Arguments

Introduction au Bash

Passage d'Arguments Principe

- Même principe que pour les fonctions
  - Les paramètres sont passés sur la ligne d'appel du script, comme fait pour les commandes
  - Les paramètres sont accessibles sous forme de variables portant le numéro du paramètre, $1, $2 et $3 dénotent respectivement les paramètres 1, 2 et 3 passés au script
  - La variable $# contient le nombre de paramètres passés au script
  - La variable $* contient l'ensemble des paramètres sous forme de liste
  - La variable $0 contient le nom du script

Passage d'Arguments Principe

- Si une fonction est appelée
  - A l'intérieur de la fonction, la valeur des variables $# et $1 à $n (où n = nombre d'arguments), est remplacée par les valeurs relatives à l'appel de la fonction
  - La valeur de $0 ne change pas
  - Une fois la fonction terminée, les valeurs initiales (avant l'appel de la fonction) des variables sont restaurées
  - On ne peut donc pas récupérer les paramètres passés au shell à l'intérieur d'une fonction, à moins de les passer à la fonction avec $*

Passage d'Arguments Exemple

```
#!/bin/bash

echo nom du script : $0
echo nombre de paramètres : $#
echo liste des paramètres : $*
```

```
biotope% ./script.sh 12 17 16 24 x 11 "chaîne de caractères"
nom du script : ./script.sh
nombre de paramètres : 7
liste des paramètres : 12 17 16 24 x 11 chaîne de caractères
biotope%
```


Passage d'Arguments Exemple

```
#!/bin/bash

total=0
for i in $*;
do
  total=$(($total + $i))
done
echo total : $total
```

```
0 + 1   = 1
1 + 2   = 3
3 + 3   = 6
6 + 4   = 10
10 + 5  = 15
15 + 6  = 21
21
```

```
biotope% ./script.sh 1 2 3 4 5 6
total : 21
biotope%
```

Passage d'Arguments Récurrence

```
#!/bin/bash

if [ $1 -gt 0 ];
then
  echo $1
  $0 $(($1 - 1))
else
  echo Terminé
fi
```

ATTENTION :
UN NOUVEAU SHELL EST
APPELE A CHAQUE FOIS

```
biotope% ./script.sh 10
10
9
8
7
6
5
4
3
2
1
Terminé
biotope%
```


Paramétrage de l'Environnement

Introduction au Bash

Param. de l'Env. Choix du shell

- Un fichier à la racine de votre compte vous permet de spécifier le shell par défaut
  - **~/.pshrc** le **~** désigne le répertoire principal c'est à dire, **/u/votrelogin**
  - Shells possibles : bash, tcsh, csh, etc.
- Manipulation
  - Editer le fichier **~/.pshrc** et remplacer la ligne existante par **SHELL=bash**
  - Le fichier ne doit contenir qu'une seule ligne

Ex. /u/gorsen/.pshrc : `SHELL=bash`

Param. de l'Env. DIRO

- Pour pouvoir utiliser le compilateur Java dans le cadre du cours IFT1010, vous devez éditer le fichier **.bashrc** qui se trouve à la racine de votre compte et vous assurer qu'il contient les lignes suivantes :

/u/votrelogin/.bashrc :

```
. ~/.profile
source /u/casino/casino.bashrc
export CLASSPATH=.:/u/dift1010/RessourcesJava/
inclure jdk
```

Param. de l'Env. Fichiers

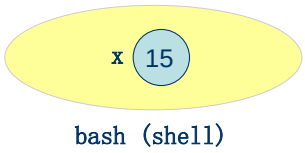
- Login
  - /etc/profile
  - ~/.bash_profile
  - ~/.bash_login
  - ~/.profile
- Ouverture de terminal
  - ~/.bashrc
- Logout
  - ~/.bash_logout

Param. de l'Env. Variables

- var=valeur
  - Même fonctionnement que dans un script, création de la variable avec assignation de la valeur donnée
  - Existe uniquement dans la session actuelle
- unset var
  - Effacement de la variable var
- export var=valeur
  - Création de la variable avec assignation de valeur
  - Existe dans la session actuelle et sera exportée dans l'environnement des commandes appelées

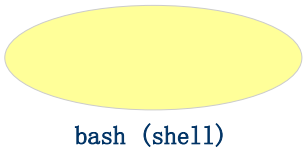
Param. de l'Env. Variables

- var=valeur




```
biotope% x=15
biotope% echo $x
15
biotope%
```

- unset var

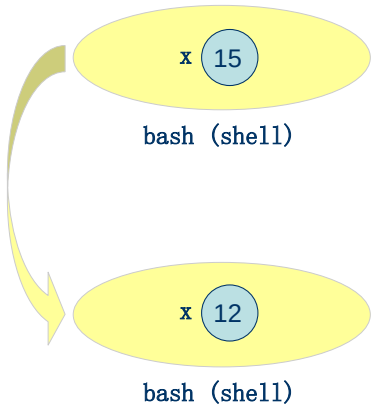



```
biotope% unset x
biotope% echo $x

biotope%
```


Param. de l'Env. Variables

- var=valeur

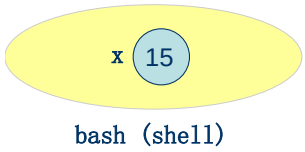



```
biotope% x=15
biotope% echo $x
15
biotope% bash
biotope% echo $x

biotope% x=12
biotope% echo $x
12
biotope%
```

Param. de l'Env. Variables

- export var=valeur




```
biotope% export x=15
biotope% echo $x
15
biotope% bash
biotope% echo $x
15
biotope% unset x
biotope% exit
biotope% echo $x
15
biotope%
```


Param. de l'Env. Variables

- Accès aux variables d'environnement à l'intérieur d'un script bash

```
#!/bin/bash

echo $x
```

```
biotope% x=15
biotope% ./script.sh

biotope% export x=15
biotope% ./script.sh
15
```

- En réalité, le lancement d'un script bash est comme le lancement d'un nouveau shell dans lequel on exécute le code qui se trouve à l'intérieur dudit fichier

Param. de l'Env. Variables

- Affichage : **export**
- Variables d'environnement les plus usuelles
  - HISTSIZE="1000"
  - HOSTNAME="hamurabi.iro.umontreal.ca"
  - HOME="/u/gorsen"
  - PATH=/usr/bin:/usr/local/bin:...
  - PS1="\\h:<\\W>% "
  - PWD="/u/gorsen/tmp"
  - USER="gorsen"
  - SHELL="/bin/bash"

Param. de l'Env. Invite

- PS1=... ou bien export PS1=... (propagation)
- Options courantes
  - \d : "Jour Mois Date"
  - \h : Nom d'hôte
  - \j : Nombre de taches
  - \t : Heure HH:MM:SS
  - \u : Nom d'usager
  - \W : Répertoire courant
  - \\ : Caractère \

```
biotope% pwd
/u/gorsen/tmp
biotope% PS1="\h:<\W>% "
biotope:<tmp>%
biotope:<tmp>% cd /u
biotope:</u>%
biotope:<tmp>% PS1="\d> "
Thu Nov 06>
Thu Nov 06> PS1="\u on \h> "
gorsen on biotope>
gorsen on biotope>
```

Param. de l'Env. Historique

- Commande : **history**
- Effacement de l'historique : **history –c**
- Limite de l'historique : variable **HISTSIZE**
- Rappel d'une commande :
  - **!<numéro de la commande>**
    Ex. **!21** rappelle la commande 21
  - **!<premières lettres de la commande>**
    Ex. **!rm** rappelle la dernière commande effectuée commençant par les lettres 'rm'
    Attention cela peut être très dangereux

Param. de l'Env. Historique

```
hamurabi:<gorsen>% history -c
hamurabi:<gorsen>% ls
HTML  PhD  archives  docs  local  progs  tmp
hamurabi:<gorsen>% pwd
/u/gorsen
hamurabi:<gorsen>% history
    1  ls
    2  pwd
    3  history
hamurabi:<gorsen>% !2
pwd
/u/gorsen
hamurabi:<gorsen>%
```


Param. de l'Env. Alias

- Commande : alias
  - alias <nom>=commande
    Ex. **alias del=rm**
  - alias <nom> "commande" si cette dernière comporte des paramètres
    Ex. **alias rm="rm –rf"**
    Attention, cela peut être très dangereux
- Il est aussi possible de
  - Supprimer un alias avec : **unalias <nom>**
  - Combiner des commandes avec '**;**' ou '**&**'
  - Faire des alias sur des alias (enchaînement)

Param. de l'Env. Alias

```
hamurabi:<gorsen>% alias h=history
hamurabi:<gorsen>% alias p="pwd ; ls"
hamurabi:<gorsen>% p
/u/gorsen
HTML  PhD  archives  docs  local  progs  tmp
hamurabi:<gorsen>% alias
alias p='pwd ; ls'
alias h='history'
hamurabi:<gorsen>% unalias h
hamurabi:<gorsen>% alias
alias p='pwd ; ls'
hamurabi:<gorsen>% h
-bash: h: command not found
```

Param. de l'Env. ~/.bashrc

- Exécuté au moment de l'ouverture de session
- Vous permet de conserver les paramètres de votre environnement
  - Editez votre fichier ~/.bashrc
  - Ajoutez-y ce que vous voulez garder
    - Alias
    - Déclarations de variables
    - Modifications de variables
    - Commandes devant être lancées lors du login ou de l'ouverture d'un terminal
    - Etc.

Param. de l'Env. ~/.bash_logout

- Exécuté au moment de la fermeture de session
- Vous permet de lancer des commandes qui vous semblent importantes
  - Nettoyage de certains répertoires
  - Changement des permissions pour vous assurer que tout est sécuritaire
    - chmod –R go=u-rwx ~/
    - chmod –R go=u-w ~/HTML
  - Effacement de l'historique
  - Etc.

Param. de l'Env. Conseils

- Faites attention avec la manipulation des alias
  - alias rm='rm –rf' peut être plutôt destructeur
  - Remplacez trop de commandes par des alias et vous serez perdus en arrivant sur un système au sein duquel vous n'avez pas cet environnement
- N'appelez jamais le bash dans le ~/.bashrc
  - Souvenez vous du principe de récurrence
  - Pensez aux administrateurs du réseau
- Faites attention en manipulant les permissions
- Expérimentez, amusez vous