



Compléments sur bash

CSC3102 – Introduction aux systèmes d'exploitation
Elisabeth Brunet & Gaël Thomas





Plan

- Variables notables
- Code de retour d'un processus
- Alias de commandes
- Fichier de configuration bash
- Filtrage de fichiers par motif

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - cd , cd ~ et cd \$HOME sont des commandes équivalentes
 - PS1 : prompt (défaut \$)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut >)

```
$ if  
>
```

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - cd , cd ~ et cd \$HOME sont des commandes équivalentes
 - PS1 : prompt (défaut \$)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut >)

```
$ if  
> [ 0 == 0 ]; then echo 'yes!'; fi  
yes!  
$
```

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - `cd`, `cd ~` et `cd $HOME` sont des commandes équivalentes
 - PS1 : prompt (défaut `$`)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut `>`)

```
$ if  
> [ 0 == 0 ]; then echo 'yes!'; fi  
yes!  
$ PS2="++++ "  
$
```

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - cd , cd ~ et cd \$HOME sont des commandes équivalentes
 - PS1 : prompt (défaut \$)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut >)

```
$ if
> [ 0 == 0 ]; then echo 'yes!'; fi
yes!
$ PS2="++++ "
$ if
++++
```

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - `cd`, `cd ~` et `cd $HOME` sont des commandes équivalentes
 - PS1 : prompt (défaut `$`)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut `>`)

```
$ if
> [ 0 == 0 ]; then echo 'yes!'; fi
yes!
$ PS2="++++ "
$ if
++++ [ 0 == 0 ]; then echo 'yes!'; fi
yes!
$
```

Variables notables

- Bash définit des variables d'environnement notables :
 - HOME : chemin absolu du répertoire de connexion
 - `cd`, `cd ~` et `cd $HOME` sont des commandes équivalentes
 - PS1 : prompt (défaut `$`)
 - PS2 : prompt en cas de commande sur plusieurs lignes (défaut `>`)

```
$ if
> [ 0 == 0 ]; then echo 'yes!'; fi
yes!
$ PS2="++++ "
$ if
++++ [ 0 == 0 ]; then echo 'yes!'; fi
yes!
$ PS1="ceci est un prompt: "
ceci est un prompt:
```


La variable d'environnement PATH

- `PATH` : ensemble de chemins séparés par des deux points (`:`)

Typiquement : `PATH=/bin:/usr/bin`

- Lorsque `bash` essaye d'exécuter `cmd`

- Si `cmd` contient un `/`, lance l'exécutable de chemin `cmd`

Exemple : `./truc.sh`, `/bin/truc.sh`

- Sinon

- Si `cmd` est une commande interne

(c.-à-d, directement exécutable par `bash`), exécute la commande

Exemple : en général, les commandes `read` ou `echo`

- Sinon, `bash` cherche `cmd` dans les répertoires du `PATH`

Exemple : `test.sh` \Rightarrow `/bin/test.sh` puis `/usr/bin/test.sh`

- Sinon, `bash` affiche `Command not found`

La variable d'environnement PATH

- La commande `which` indique où se trouve les commandes
`which cmd` : indique le chemin complet de `cmd` en utilisant `PATH`

La variable d'environnement PATH

Attention : il est fortement déconseillé de mettre `.` dans `PATH` (surtout si `.` est en tête du `PATH`)

- Avantage : mettre `.` dans `PATH` évite le `./` pour trouver les commandes du répertoire de travail
(`$ script.sh` au lieu de `$./script.sh`)
- Mais n'importe quel virus/malware peut alors créer un cheval de troie en :
 - Plaçant un script nommé `ls` dans le répertoire `/tmp`
 - Attendant tranquillement que l'administrateur entre dans `/tmp`
 - Attendant ensuite que l'administrateur lance `ls` dans `/tmp`,
=> lancement du `ls` du malware avec les droits administrateurs

La malware a pris le contrôle de la machine !

Code de retour d'un processus

- Un script peut renvoyer un code de retour avec `exit n`
 - Ce code de retour peut être utilisé dans les `if` et `while`
0 ⇒ vrai (ou ok), autre ⇒ faux (ou problème)
 - Sémantique du code de retour parfois cryptique ⇒ utiliser `man`
- Code de retour dernière commande stocké dans la variable `$?`

```
$
```

```
#!/bin/bash  
  
exit $1
```

replay.sh

Code de retour d'un processus

- Un script peut renvoyer un code de retour avec `exit n`
 - Ce code de retour peut être utilisé dans les `if` et `while`
0 ⇒ vrai (ou ok), autre ⇒ faux (ou problème)
 - Sémantique du code de retour parfois cryptique ⇒ utiliser `man`
- Code de retour dernière commande stocké dans la variable `$?`

```
$ ./replay.sh 42  
$
```

```
#!/bin/bash  
  
exit $1
```

replay.sh

Code de retour d'un processus

- Un script peut renvoyer un code de retour avec `exit n`
 - Ce code de retour peut être utilisé dans les `if` et `while`
0 ⇒ vrai (ou ok), autre ⇒ faux (ou problème)
 - Sémantique du code de retour parfois cryptique ⇒ utiliser `man`
- Code de retour dernière commande stocké dans la variable `$?`

```
$ ./replay.sh 42
$ echo $?
42
$
```

```
#!/bin/bash

exit $1
```

replay.sh

Code de retour d'un processus

- Un script peut renvoyer un code de retour avec `exit n`
 - Ce code de retour peut être utilisé dans les `if` et `while`
0 ⇒ vrai (ou ok), autre ⇒ faux (ou problème)
 - Sémantique du code de retour parfois cryptique ⇒ utiliser `man`
- Code de retour dernière commande stocké dans la variable `$?`

```
$ ./replay.sh 42
$ echo $?
42
$ if ./replay.sh 0; then echo coucou; fi
coucou
$
```

```
#!/bin/bash

exit $1
```

replay.sh

Code de retour d'un processus

- Un script peut renvoyer un code de retour avec `exit n`
 - Ce code de retour peut être utilisé dans les `if` et `while`
0 ⇒ vrai (ou ok), autre ⇒ faux (ou problème)
 - Sémantique du code de retour parfois cryptique ⇒ utiliser `man`
- Code de retour dernière commande stocké dans la variable `$?`

```
$ ./replay.sh 42
$ echo $?
42
$ if ./replay.sh 0; then echo coucou; fi
coucou
$ if ./replay.sh 1; then echo coucou; fi
$
```

```
#!/bin/bash

exit $1
```

replay.sh

Code de retour d'un processus

- Un script peut renvoyer un code de retour

Attention : contrairement à `bash`, dans quasiment tous les autres langages de programmation, la valeur faux vaut 0 et la valeur vrai vaut autre chose que 0

(car dans le cas 0 = faux/1 = autre, le `ou` logique est une simple addition dans \mathbb{Z} et le `et` logique est une simple multiplication dans \mathbb{Z})

```
$ .  
$ echo  
42  
$ if  
could  
$ if  
$
```

replay.sh

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

- Création :

```
alias cmd='...'
```

- Suppression :

```
unalias cmd
```

- Consultation :

```
alias
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

■ Création :
`alias cmd='...'`

■ Suppression :
`unalias cmd`

■ Consultation :
`alias`

```
$ ls  
d      f1      test.sh  
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

- Création :

```
alias cmd='...'
```

- Suppression :

```
unalias cmd
```

- Consultation :

```
alias
```

```
$ ls
d      f1      test.sh
$ alias ls='ls -a'
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

- Création :

```
alias cmd='...'
```

- Suppression :

```
unalias cmd
```

- Consultation :

```
alias
```

```
$ ls
d      f1      test.sh
$ alias ls='ls -a'
$ ls
.      ..      d      f1      test.sh
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

■ **Création :**
`alias cmd='...'`

■ **Suppression :**
`unalias cmd`

■ **Consultation :**
`alias`

```
$ ls
d      f1      test.sh
$ alias ls='ls -a'
$ ls
.      ..      d      f1      test.sh
$ alias
alias ls='ls -a'
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

■ **Création :**
`alias cmd='...'`

■ **Suppression :**
`unalias cmd`

■ **Consultation :**
`alias`

```
$ ls
d      f1      test.sh
$ alias ls='ls -a'
$ ls
.      ..      d      f1      test.sh
$ alias
alias ls='ls -a'
$ unalias ls
$
```

Alias de commande

- Sert à (re)définir le nom d'une commande
 - Pour créer des noms abrégés ou passer des options

■ **Création :**
`alias cmd='...'`

■ **Suppression :**
`unalias cmd`

■ **Consultation :**
`alias`

```
$ ls
d      f1      test.sh
$ alias ls='ls -a'
$ ls
.      ..      d      f1      test.sh
$ alias
alias ls='ls -a'
$ unalias ls
$ ls
d      f1      test.sh
$
```


Fichiers de configuration bash

- Exécutés automatiquement au démarrage de `bash`
 - La prise en compte d'une modification de configuration impose le redémarrage de `bash` (ou l'utilisation de la **source** `~/ .bashrc`)
- Configuration
 - Globale du système d'exploitation par l'administrateur
 - Fichier `/etc/profile`
 - Pour son compte par l'utilisateur
 - Fichier `~/ .bashrc` (+ d'autres fichiers non étudiés dans ce cours)
- Opérations typiquement réalisées :
 - Affectation de variables : `PATH`, `PS1`, etc.
 - Déclaration de variables liées à des logiciels installés en sus
 - Création d'alias
 - Positionnement du masque des droits d'accès
 - Etc.

Filtrage de fichiers par motif (1/3)

- Bash peut filtrer des noms de fichiers en suivant un motif
 - * ⇒ une chaîne de caractères quelconque (même vide)
 - ? ⇒ substitue **un** caractère quelconque

```
$ ls                # contenu du répertoire
CSC3101 CSC3102 CSC3601 CSC3602 NET3101 NET3102
$
```

Filtrage de fichiers par motif (1/3)

- Bash peut filtrer des noms de fichiers en suivant un motif
 - * ⇒ une chaîne de caractères quelconque (même vide)
 - ? ⇒ substitue **un** caractère quelconque

```
$ ls # contenu du répertoire
CSC3101 CSC3102 CSC3601 CSC3602 NET3101 NET3102
$ echo CSC*1 # les cours CSC se terminant par 1
CSC3101 CSC3601
$
```

Filtrage de fichiers par motif (1/3)

- Bash peut filtrer des noms de fichiers en suivant un motif
 - * ⇒ une chaîne de caractères quelconque (même vide)
 - ? ⇒ substitue **un** caractère quelconque

```
$ ls # contenu du répertoire
CSC3101 CSC3102 CSC3601 CSC3602 NET3101 NET3102
$ echo CSC*1 # les cours CSC se terminant par 1
CSC3101 CSC3601
$ echo CSC?1?? # les cours CSC de semestre 1
CSC3101 CSC3102
$
```

Filtrage de fichiers par motif (2/3)

■ Filtre suivant un ensemble de caractères

- [...] → un caractère dans l'ensemble donné
- [!...] → un caractère hors de l'ensemble donné

■ Ensemble

- Liste de caractères : [aeiouy] [!aeiouy]
- Un intervalle : [0-9] [a-zA-Z] [!A-F]
- Ensembles prédéfinis :
 - [[:alpha:]] : caractères alphabétiques
 - [[:lower:]] / [[:upper:]] : alphabet minuscule / majuscule
 - [[:digit:]] : chiffres décimaux [0-9]

Filtrage de fichiers par motif (2/3)

```
$ ls
CSC3102 CSC3501 CSC4501 CSC5001 NET3101 NET3102
$ echo CSC[45]*      # cours de 2A et 3A
CSC4502 CSC5001
```



En route pour le TP!