

Scripting Shell

Module 05 – Les caractères spéciaux du script



1

Scripting Shell



Objectifs

- Savoir enchaîner plusieurs commandes
- Conditionner une commande au résultat d'une autre
- Regrouper les commandes
- Colorier la console



2

Enchaînement de commandes



3

2 types d'enchaînement de commandes

Inconditionnel

Conditionnel



4

; | L'enchaînement sans condition

Pour enchaîner des commandes quel que soit le résultat de leur exécution, on utilise le « ; » entre celles-ci.



```
$ mkdir /data ; touch /data/file1 ; cat /home/bonjour.txt  
mkdir: cannot create directory './data': Permission denied  
touch: cannot touch './data/file1': No such file or directory  
Bonjour le monde !
```

- Dans cet exemple, l'utilisateur n'a pas le droit de créer un répertoire à la racine (mkdir en erreur).
- La commande de création d'un fichier dans le répertoire échoue aussi (touch en erreur).
- La dernière commande (sans lien avec les précédentes) aboutit.



; | L'enchaînement sans condition

Dans un script, le traitement des commandes est séquentiel.
Le « ; » est substitué par un retour à la ligne.



```
$ mkdir /data ; touch /data/file1 ; cat /home/bonjour.txt
```



```
mkdir /data  
touch /data/file1  
cat /home/bonjour.txt
```

Traitement
séquentiel

& &

Le "et" logique

L'opérateur « && » conditionne l'enchaînement de commandes au résultat d'exécution positif de la commande précédente.

- Cette structure peut se traduire par :

*Si la **commande aboutit alors** je fais la suivante.*



```
$ mkdir /data && touch /data/file1
mkdir: cannot create directory './data': Permission denied

$ mkdir ./data && touch ./data/file1
$ ls -l ./data/
-rw-r--r-- 1 Gilles Gilles 0 2021-12-26 02:12 file1
```



& &

Le "et" logique

Le système s'appuie sur le code retour de la commande précédente (variable réservée `$?`) :

- Si `$? = 0` (pas d'erreur) : alors la commande suivante est exécutée
- Si `$? ≠ 0` (erreur) : alors la commande suivante n'est pas exécutée



|| Le "ou" logique

- L'opérateur « || » conditionne l'enchaînement de commandes à l'échec d'exécution de la commande précédente.
- Cette structure peut se traduire par :

*Si la **commande n'a pas abouti alors** je fais la suivante.*



```
mkdir ./data 2>/dev/null || echo "Erreur de création du répertoire 'data'"  
Erreur de création du répertoire 'data'
```



&& Le "et" logique

Le système s'appuie sur le code retour de la commande précédente (variable réservée `$?`) :

- Si `$? = 0` (pas d'erreur) : alors la commande suivante n'est pas exécutée
- Si `$? ≠ 0` (erreur) : alors la commande suivante est exécutée





- Enchaînement conditionnel et inconditionnel de commandes
-



Regroupements de commandes




- Les regroupements de commandes peuvent être utilisés pour :
 - Exécuter plusieurs commandes dans un même environnement.
 - Rediriger le résultat de plusieurs commandes vers un fichier ou un tube.
- Deux syntaxes de regroupement de commandes sont utilisables : (« regroupement ») et {« regroupement »}.



Regroupement avec création de sous-shell

- Pour regrouper dans un environnement enfant l'exécution de plusieurs commandes, on utilise la syntaxe suivante :

 \$ (cmd1 ; cmd2)

- Dans un script, pour plus de lisibilité, le regroupement sera écrit ainsi :

 (
 cmd1
 cmd2
)

Exemple :
pour afficher le nombre total de fichiers
contenus dans /bin et /usr/bin

```
$ (ls /bin/ ; ls /usr/bin) | wc -l
```



Regroupement sans création de sous-shell

- Pour regrouper dans un environnement enfant l'exécution de plusieurs commandes, on utilise la syntaxe suivante :



```
$ { cmd1 ; cmd2 ; cmd3 ; }
```



À la différence de la syntaxe du `()`, un `" ; "` doit être mis juste avant l'accolade fermante. Ne pas oublier le point-virgule suivant la dernière commande et les espaces après et avant les accolades.

- Dans un script, pour plus de lisibilité, le regroupement sera écrit ainsi :



```
{  
    cmd1  
    cmd2  
}
```



Utiliser les couleurs dans le Shell



- Il est possible d'utiliser quelques couleurs dans le Shell ou dans les scripts.
- Elles peuvent être utiles pour mettre en valeur des informations importantes.

Couleur	Caractères	Fond	Attribut des caractères
Noir	30	40	0 : aucun
Rouge	31	41	1 : gras
Vert	32	42	4 : souligné
Jaune	33	43	7 : inversé
Bleu	34	44	8 : invisible
Magenta	35	45	9 : barré
Cyan	36	46	
Gris clair	37	47	



- Pour coloriser une information, les codes couleur seront à encadrer pour :
 - Annoncer la couleur : `\033[` ou `\e[`
 - Finir la définition de couleur : `m`
 - Remettre la valeur initiale : `\033[0m`

Exemple

```
$ echo -e "voici du \033[1;32mvert\033[0m"
voici du vert
```





-
- Les couleurs dans le shell
-