Chapitre 6

Système d'Exploitation UNIX

Mécanismes d'interprétation du Shell (Redirections & Tubes)

BOUKRI KHALIL

Types de Shell

Shell	Nom	Description
Bourne Shell	sh	Shell disponible sur toute plateforme UNIX
C shell	csh	Shell développé par BSD
Korn shell	ksh	Bourne Shell étendu par l'AT&T
Bourne Again Shell	bash	Version améliorée de sh et csh. Fourni le plus souvent avec Linux.
Zero Shell	zsh	shell avec beaucoup de fonctionnalités : typage, substitution et complétion très poussées
Tenex	tcsh	csh étendu
rc	rc	Implémentation pour UNIX du shell de Plan 9
es	es	Extension de rc

Caractères spéciaux de Shell

Caractères	Description
tabulation, espace	Délimiteur de mot
retour chariot	Fin de la commande à exécuter
&	Lance une commande en tâche de fond
; ; ;	Séparateur de commande
∗?[][^]	Substitution de noms de fichiers
&& ||!	Opérateurs booléens
' " \	Caractères de quotation
<><<>> ‘ |<><< & > & << − >|	Opérateurs de redirection d'entrées sorties
$	Valeur d'une variable
#	Début de commentaires
(){}	Groupement de commande

Métacaractères: Quotation

’ " ` \ changent la façon dont le Shell interprète les caractères spéciaux.

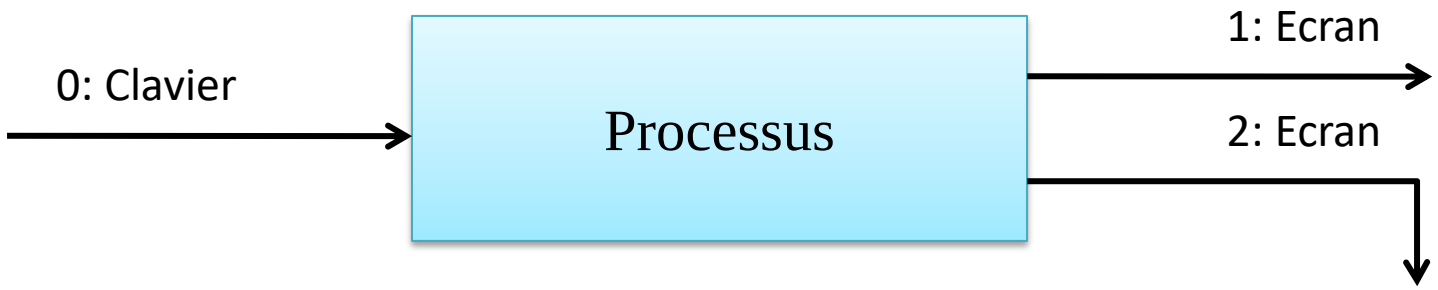
Symbole	Signification
’ (single-quote)	le shell ignore tout caractère spéciaux entre deux ’
" (double-quote)	le shell ignore tout caractère spéciaux entre deux ", à l’exception de $ et \ et ‘
\ (antislash ou backslash)	le shell ignore le caractère spécial suivant le \
‘ (backquote ou antiquote)	le shell exécute ce qu’il y a entre deux ‘

Métacaractères: Quotation

Exemples

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[emsig1@localhost ~]$ whoami
emsig1
[emsig1@localhost ~]$ echo `whoami` et le chemin: $HOME et c*
emsig1 et le chemin: /home/emsig1 et cours
[emsig1@localhost ~]$
[emsig1@localhost ~]$
[emsig1@localhost ~]$ echo "`whoami` et le chemin: $HOME et c*"
emsig1 et le chemin: /home/emsig1 et c*
[emsig1@localhost ~]$
[emsig1@localhost ~]$ echo '`whoami` et le chemin: $HOME et c*'
`whoami` et le chemin: $HOME et c*
[emsig1@localhost ~]$
```

Flux: Redirections



- L'entrée standard (stdin : 0) : le flux d'entrée du programme (par défaut, ce qui est tapé au clavier)
- La sortie standard (stdout : 1) : le flux de sortie du programme (par défaut, il sera affiché à l'écran)
- L'erreur standard (stderr : 2) : le flux d'erreur du programme (par défaut, il sera affiché à l'écran)

Métacaractères de redirection

Syntaxe	Commande
cmd < fic	l'entrée de la commande provient du fichier
cmd << etq	l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette
cmd > fic	la sortie de la commande est placé dans le fichier
cmd >> fic	la sortie de la commande est mise à la suite du fichier
cmd 2 > fic	redirige la sortie d'erreur de la commande dans le fichier
cmd 2 >> fic	redirige la sortie d'erreur de la commande à la suite du fichier
cmd_1 | cmd_2	passe la sortie de la commande 1 comme entrée de la commande 2

Redirection de la sortie standard

Syntaxe:

cmd > fichier

☛ redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.

cmd >> fichier

☛ redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

Exemple:

```
$ echo "Premier contenu ajouté" > file1
$ cat file1
Premier contenu ajouté

$ echo "Deuxième contenu ajouté " >> file1
$ cat file1
Premier contenu ajouté
Deuxième contenu ajouté

$ echo "Troisième contenu ajouté" > file1
$ cat file1
Troisième contenu ajouté
$
```

Redirection de la sortie d'erreur

Syntaxe:

cmd 2> fichier

☛ redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.

cmd 2>> fichier

☛ redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

Exemple:

```
$ ls abc file1 file2
ls : abc : No such file or directory
file1 file2
$ ls abc file1 file2 > sortiestd
ls : abc : No such file or directory
$ cat sortiestd
file1 file2
$ ls abc file1 file2 2> erreurstd
file1 file2
$ cat erreurstd
ls : abc : No such file or directory
$ ls abc file1 file2  2> erreurstd  >sortiestd
$ cat erreurstd
ls : abc : No such file or directory
$
```

Redirection de l'entrée standard

Syntaxe:

cmd < fichier

☛ l'entrée de la commande provient du fichier

cmd << ETQ

☛ l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

Exemple:

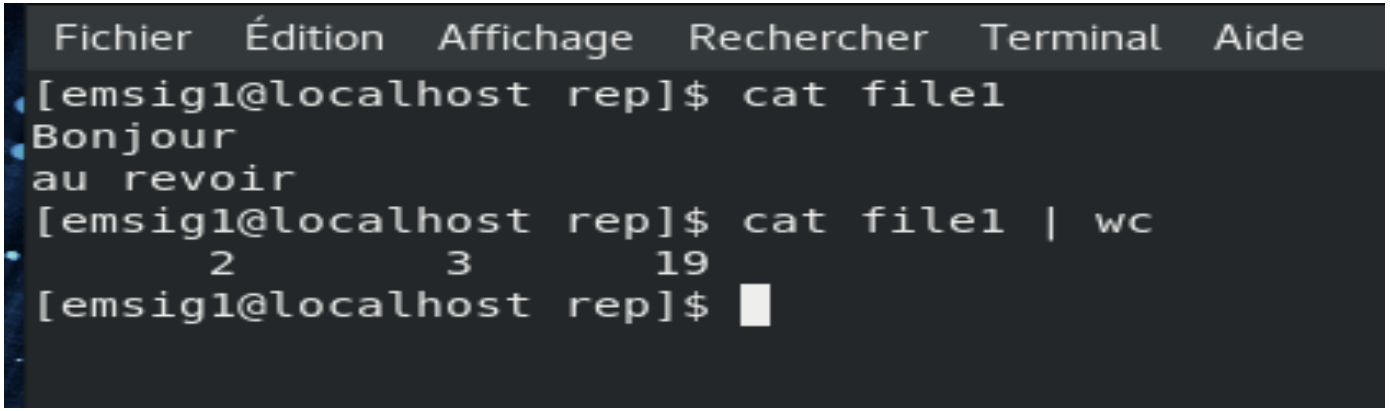
Tube (Pipe) entre flux d'entrée et flux de sortie

Syntaxe:

cmd 1 | cmd 2

☛ redirige la sortie de la commande1 vers l'entrée de la commande 2

cmd1 | cmd2 <==> **cmd1 > fictmp**
cmd2 < fictmp
rm fictmp



Tube (Pipe) entre flux d'entrée et flux de sortie

Syntaxe:

cmd 1 | cmd 2

☛ redirige la sortie de la commande1 vers l'entrée de la commande 2

La commande « xargs »

Certaines commandes ne savent pas lire leur entrée standard:

➔ ls , rm, cp, ln, mv

Exemple :

ls | rm aucun sens

➔ ls | xargs rm