Regroupement de commandes

Le regroupement de commandes peut être utilisé pour :

- rediriger la sortie écran de plusieurs commandes vers un même fichier ou vers un tube ;
- faire exécuter plusieurs commandes dans le même environnement.

Exemple

Seule la sortie standard de la deuxième commande est redirigée dans le fichier ***resultat****.*

```
$ date ; ls > resultat
mar jan 28 05:16:30 CET 2003
$ cat resultat
FIC
fichier
$
```

Les parenthèses **( )** et les accolades **{ }** permettent de regrouper les commandes. Dans le premier cas, les commandes sont exécutées à partir d'un shell enfant, dans le deuxième cas à partir du shell courant.

1. Les parenthèses

Syntaxe

```
(cmde1 ; cmde2 ; cmde3)
```

Avec les parenthèses, un shell enfant est systématiquement créé et c'est ce dernier qui traite la ligne de commande (avec duplications ultérieures si nécessaire).

Premier exemple

Ici, l'utilisateur se sert des parenthèses pour rediriger la sortie standard de deux commandes :

```
$  (date ; ls) > resultat
$ cat resultat
mar jan 28 05:21:36 CET 2003
FIC
fichier
$
```

Les figures 14, 15 et 16 représentent le mécanisme interne associé. Le shell courant (PID=201) se duplique **(1)**. Le shell enfant (PID=205) s'occupe d'abord de la redirection **(2)**, se duplique ensuite pour l'exécution de la commande externe **`date`** **(5)**. Lorsque cette dernière est terminée **(7)**, se duplique à nouveau pour exécuter **`ls`** **(8)**. Grâce au mécanisme d'héritage, les deux commandes utilisent le même offset **(3)**. Donc les écritures dans le fichier se succèdent.

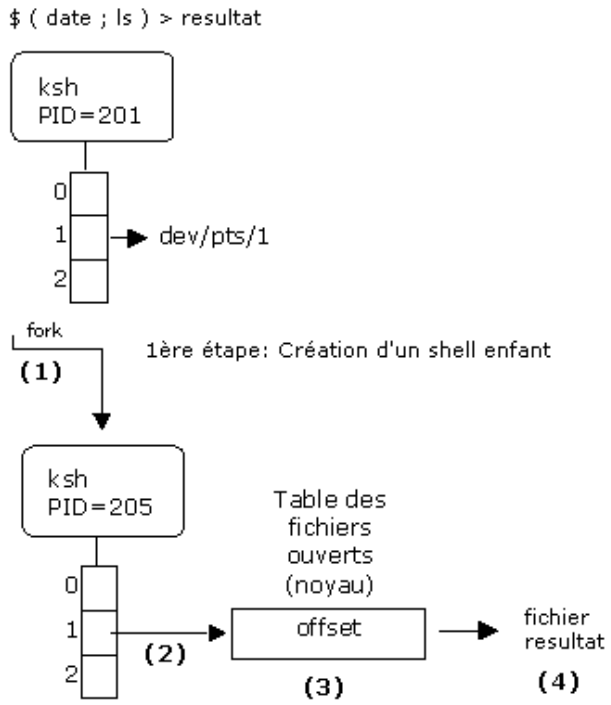

Figure 14 : Premier exemple de regroupement avec parenthèses - Première étape

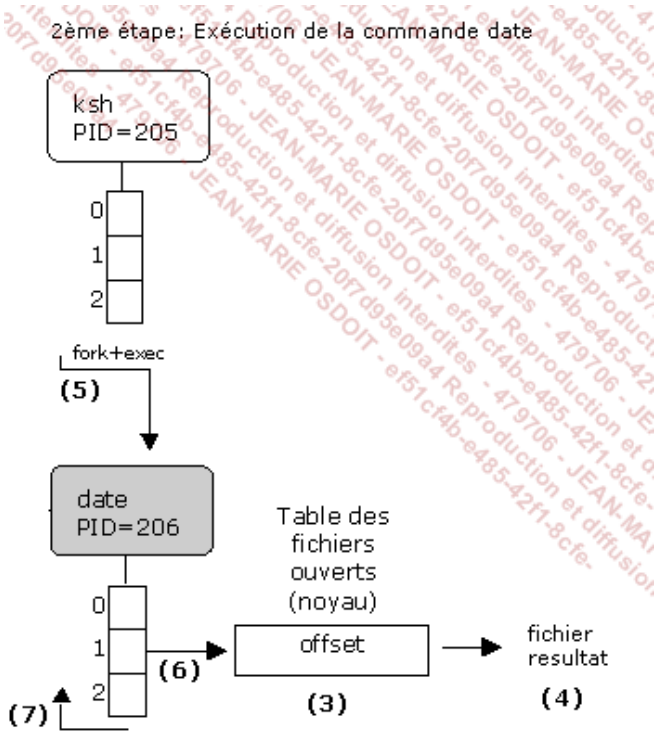

Figure 15 : Premier exemple de regroupement avec parenthèses - Deuxième étape

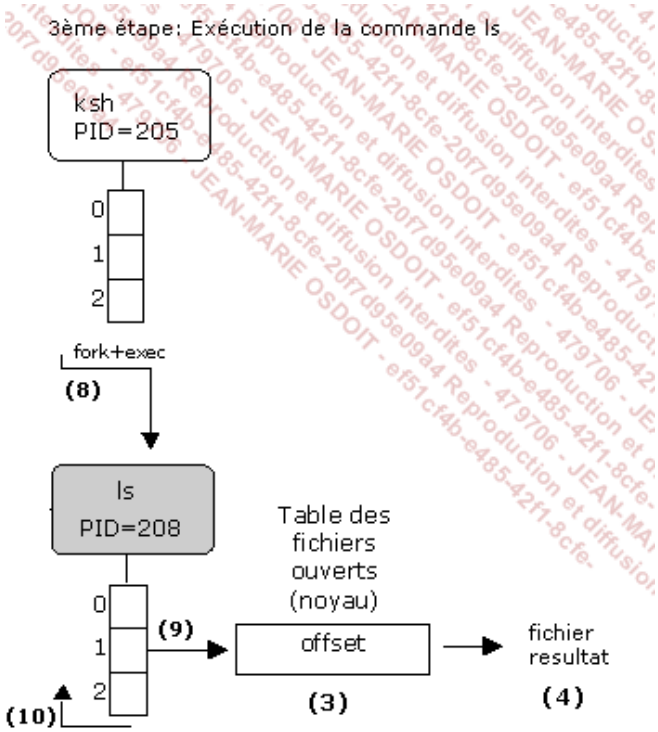

Figure 16 : Premier exemple de regroupement avec parenthèses - Troisième étape

Deuxième exemple

*Les commandes **pwd** et **ls** ont pour répertoire courant **/tmp** :*

```
$ pwd
/home/christie
$ (cd /tmp ; pwd ; ls)  > listefic
$ cat listefic
/tmp                          (résultat de pwd)
dcopNYSrKn                    (liste des fichiers de /tmp)
listetmp
...
```

*Lorsque l'exécution des trois commandes est terminée, le shell de premier niveau reprend la main. Son répertoire courant est toujours **/home/christie**.*

```
$ pwd
/home/christie
$
```

Les figures 17 et 18 représentent le mécanisme interne associé. Le shell de travail a pour répertoire courant **/home/christie (1)**. Comme dans l'exemple ci-dessus, il y a création d'un shell enfant **(2)**. Ce dernier va exécuter lui-même la commande interne **cd (3)** (le répertoire courant du shell enfant change **(4)**), puis la commande interne **pwd (5)** (qui affiche **/tmp**), puis se duplique **(6)** pour l'exécution de la commande externe **ls** qui hérite du répertoire **/tmp (7)**.

Lorsque toutes les commandes sont exécutées **(8)**, le shell de premier niveau reprend le contrôle. Son répertoire courant est toujours **/home/christie**.

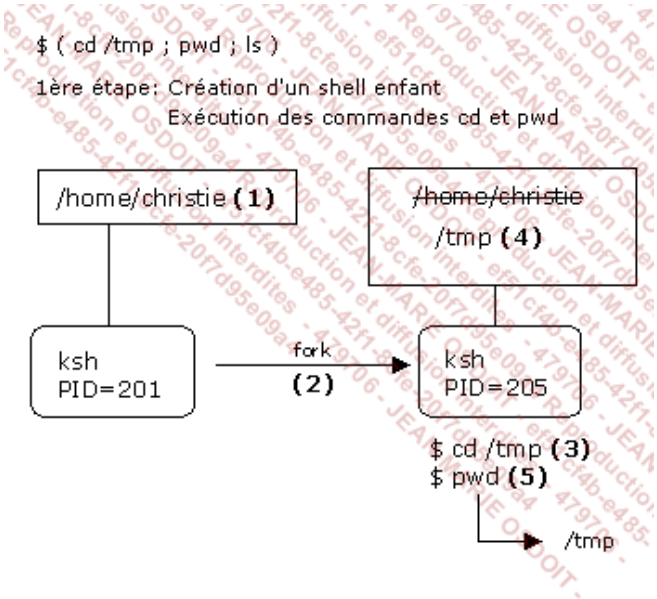

Figure 17 : Deuxième exemple de regroupement avec parenthèses - Première étape

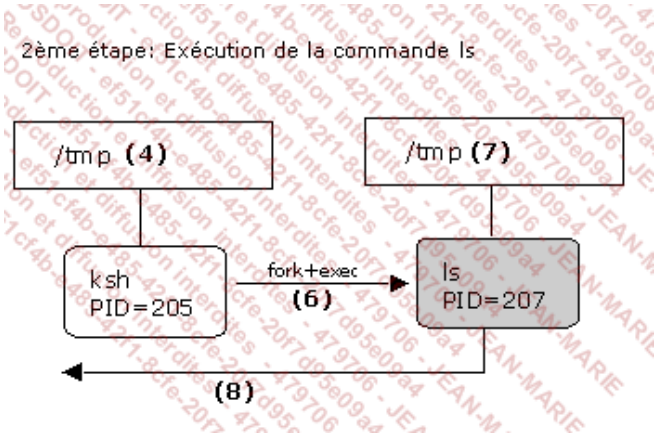

Figure 18 : Deuxième exemple de regroupement avec parenthèses - Deuxième étape

2. Les accolades

Syntaxe

```
{ cmde1 ; cmde2 ; cmde3 ; }
```

- Les accolades ouvrante et fermante doivent être respectivement suivies et précédées par un espace.
- La dernière commande doit être suivie d'un ;.

La ligne de commande est traitée par le shell courant (avec duplications ultérieures si nécessaire).

Premier exemple

Les deux commandes suivantes produisent le même résultat, mais la version avec accolades est plus rapide :

```
$ ( date ; ls) > resultat
$ { date ; ls ; } > resultat
```

Les figures 19, 20, 21 et 22 représentent le mécanisme interne associé aux accolades. Le shell de travail sauvegarde ses associations **descripteur-fichier** courantes **(1)**, traite lui-même la redirection demandée **(2)**, se duplique pour

l'exécution de la commande externe **`date` (5)**, puis lorsque cette dernière est terminée **(7)**, se duplique à nouveau pour exécuter **`ls` (8)**. Lorsque les commandes sont terminées, le shell de premier niveau reprend la main **(10)** et restaure son environnement **descripteur-fichier (11)**.

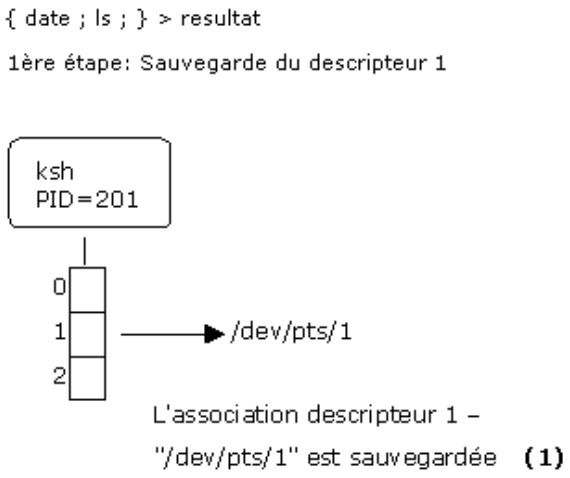

Figure 19 : Premier exemple de regroupement avec accolades - Première étape

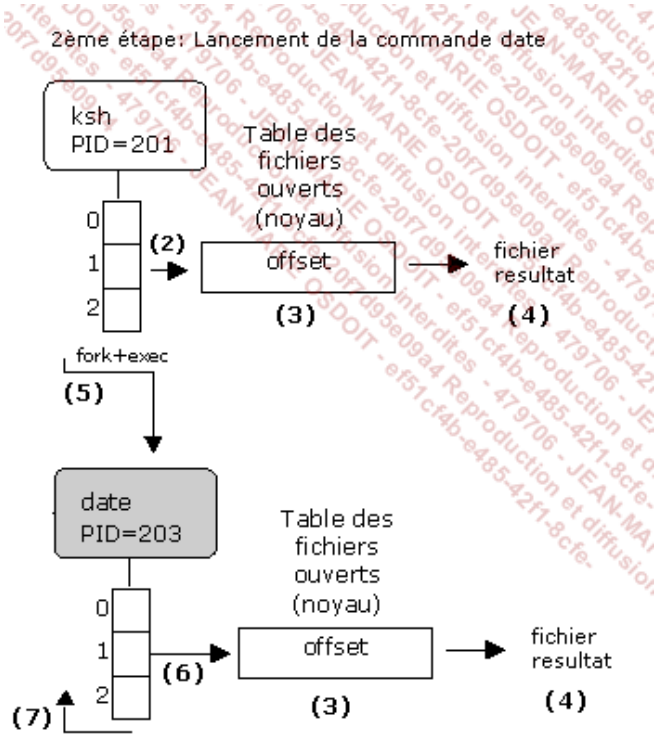

Figure 20 : Premier exemple de regroupement avec accolades - Deuxième étape

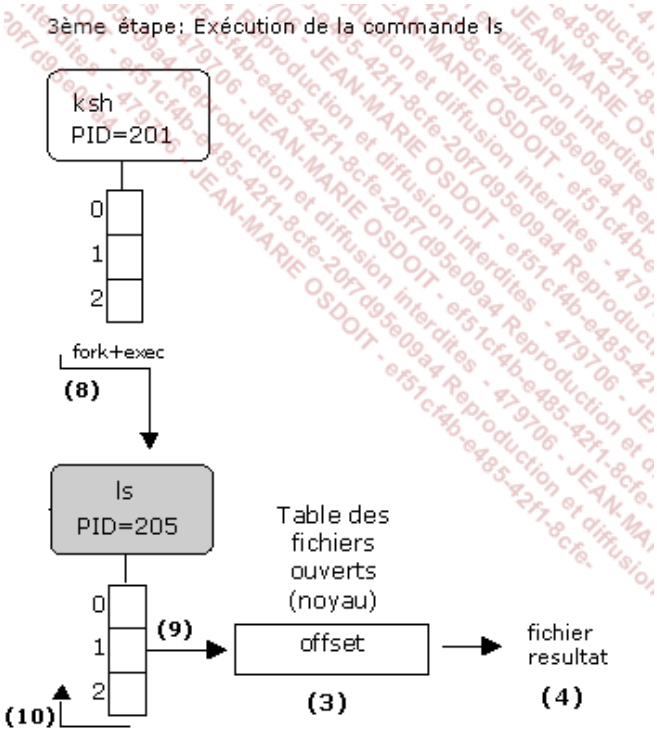

Figure 21 : Premier exemple de regroupement avec accolades - Troisième étape

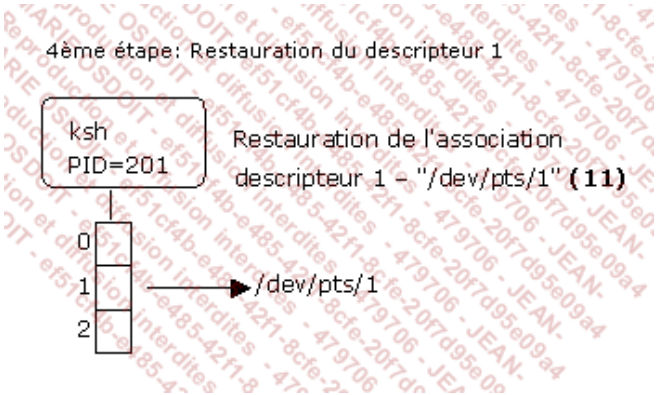

Figure 22 : Premier exemple de regroupement avec accolades - Quatrième étape

Deuxième exemple

Ici, l'environnement du shell de premier niveau va être modifié, ce qui n'est pas forcément très intéressant :

```
$ pwd
/home/christie
$
$ { cd /tmp ; pwd ; ls ; } > listefic
$
$ cat listefic
/tmp
dcopNYSrKn
listetmp
$ pwd
/tmp
$
```

Les figures 23 et 24 représentent le mécanisme interne associé. Le shell de travail sauvegarde ses associations **descripteur-fichier** courantes, traite lui-même la redirection demandée. Il exécute ensuite la commande interne **cd (1)** (son répertoire courant change **(2)**), puis la commande interne **pwd (3)** (qui affiche donc **/tmp**), puis se duplique

(4) pour l'exécution de la commande externe **ls** qui hérite du répertoire **/tmp (5)**. Lorsque **ls** est terminée, le shell de travail reprend la main et restaure son environnement **descripteur-fichier**. Par contre, son répertoire courant reste **/tmp**. La commande **pwd** le confirme **(6)**.

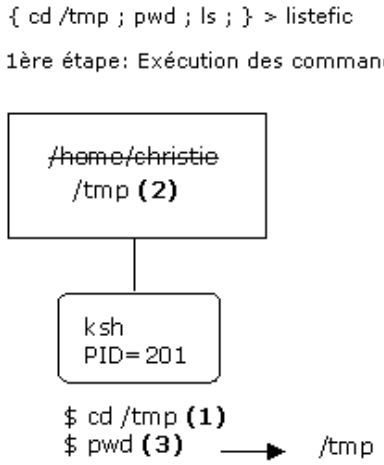

Figure 23 : Deuxième exemple de regroupement avec accolades - Première étape

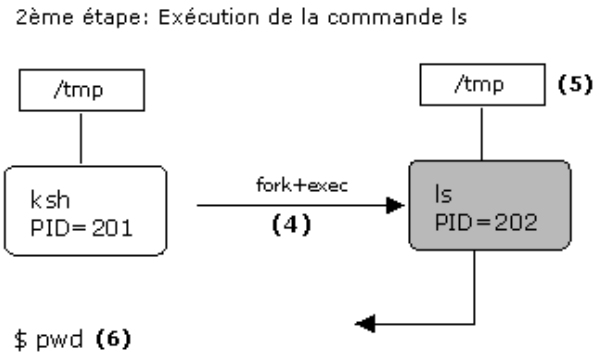

Figure 24 : Deuxième exemple de regroupement avec accolades - Deuxième étape

3. Conclusion

Les parenthèses sont plus utilisées que les accolades pour les deux raisons suivantes :

- leur syntaxe est plus simple à utiliser ;
- quel que soit le jeu de commandes, on est toujours sûr de retrouver l'environnement de travail initial.

L'utilisation des accolades se justifiera dans le cas d'une recherche de performance.