Variables et Pipe

D. Bogdaniuk – P. Portejoie

Antoine Gicquel	**04/11/2016**
1A2	

L'exécution d'une commande peut se faire à l'intérieur d'une autre ; par exemple pour récupérer le résultat de la première comme paramètre de la seconde. Le caractère anti-apostrophe ou backquote (`) encadrant une commande permet une telle exécution :

```
>set var=`ls`
>echo $var
admin/ ailleurs/ bin/ cours/ doc/ ftp/ mbox nsmail/
recherche/ tex_inputs@ tex_inputs.perso@ texput.log tmp/
```

A noter l'équivalence des 2 écritures suivantes :

- `@ x = ( $y + $z ) * 2 ; echo $x`
- `echo `expr \($y + $z \) * 2``
  qui s'écrit encore plus simplement `expr \( $y + $z \) \* 2` puisque `echo` est ici inutile, `expr` affichant son propre résultat.

D'autre part, le point-virgule (`;`) est la marque de séquentialité et permet de placer plusieurs commandes sur une même ligne. Elles sont exécutées les unes après les autres :

```
>who;ps
portejoi     ttyp1   Sep 28 09:29   (:0.0)
portejoi     ttyp3   Sep 28 10:53   (:0.0)
portejoi     ttyp5   Sep 28 17:27   (kb-b035-8.univ-ubs.fr)
  PID TT STAT  TIME COMMAND
  181 p1 IW    0:01 -csh (tcsh)
  188 p1 S     7:07 firefox
  227 p3 IW    0:00 -csh (tcsh)
  258 p4 S     0:00 -csh (tcsh)
  424 p4 R     0:00 ps
```

Enfin, la commande `ypcat` (cf man) permet d'afficher les clés d'une base de données NIS dont la carte (*map*) est passée en paramètre et donc de connaître les ressources du réseau. Ainsi `ypcat passwd` permet de connaître la liste des personnes ayant un compte sur le réseau de l'UBS (autres cartes : group, hosts, services, networks…) :

```
> ypcat passwd
…
portejoi:x:2211:200:Portejoie Philippe, philippe.portejoie@univ-ubs.fr:
/ubs/home/prof/portejoi:/bin/bash
…
```

Chaque entrée du fichier est considérée comme une ligne. Vous remarquerez en TP que le format des lignes n'est pas tout à fait le même selon que celle-ci décrit un étudiant ou un enseignant, mais vous y repérerez toutefois votre *euid* (user id, ici 2211) et votre *gid* (group id, ici 200).

TD – TP

Rappel : les exercices sont à faire en C-shell (csh). Le langage de commande initial à l'ouverture d'un terminal étant le bash vous devrez donc veiller à utiliser en tout premier lieu la commande `csh` (ou `/bin/csh`) afin d'y travailler en C-shell

Exercice 1

1/ Visualisez le contenu de la variable d'environnement `PATH`. Rappelez son rôle.

PATH est la liste des répertoires de recherche des commandes

```
[e1600718@ens-iutva-0391 ~]$ echo $PATH
/usr/lib64/ccache:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin
:/ubs/home/etud/2016/e1600718/.local/bin:/ubs/home/etud/2016/e1600718/bi
n
[e1600718@ens-iutva-0391 ~]$
```

2/ Sauvegardez le contenu de `PATH` dans `OLDPATH`

```
[e1600718@ens-iutva-0391 ~]$ setenv OLDPATH $PATH
[e1600718@ens-iutva-0391 ~]$ echo $OLDPATH
/usr/lib64/ccache:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin
:/ubs/home/etud/2016/e1600718/.local/bin:/ubs/home/etud/2016/e1600718/bi
n
[e1600718@ens-iutva-0391 ~]$
```

3/ `Créez un alias (cf man) de la commande ls que vous nommerez monls` (vous devrez utiliser le nommage absolu : `/bin/ls`)

```
[e1600718@ens-iutva-0391 ~]$ alias monls /bin/ls
[e1600718@ens-iutva-0391 ~]$ monls
Bureau                    'SPAD Mes Projets'
conf                      'SPAD Préférences'
datamodeler               Sqldeveloper
Documents                 Téléchargements
```

...

4/ Tapez la suite de commandes `setenv PATH .. ; echo $PATH ; clear`
Décrivez ce qu'il se passe. Expliquez

```
[e1600718@ens-iutva-0391 ~]$ setenv PATH .. ; echo $PATH ; clear
..
clear: Commande introuvable.
[e1600718@ens-iutva-0391 ~]$
```

Premièrement, nous fixons la variable d'environnement PATH à la valeur .. ; puis nous l'affichons ; et enfin nous exécutons la commande clear mais comme PATH renvoie sur .. ; la commande est donc introuvable.

5/ Lancez la commande `ls`. Lancez la commande `/bin/ls`. Lancez la commande `monls`. Que se passe-t-il à chaque fois ? Justifiez

La commande ls ne fonctionne pas car PATH vaut .. et ne vas donc pas chercher la commande dans /bin/, en revanche en exécutant /bin/ls fonctionne car on spécifie le chemin absolue. Le alias monls fonctionne de la même façon que /bin/ls.

6/ Redonnez à la variable `PATH` son ancienne valeur

```
[e1600718@ens-iutva-0391 ~]$ setenv PATH $OLDPATH
[e1600718@ens-iutva-0391 ~]$ echo $PATH
/usr/lib64/ccache:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin
:/ubs/home/etud/2016/e1600718/.local/bin:/ubs/home/etud/2016/e1600718/bi
n
[e1600718@ens-iutva-0391 ~]$
```

7/ Lancez la commande `ps`. Que se passe-t-il ? Justifiez

```
[e1600718@ens-iutva-0391 ~]$ ps
  PID TTY          TIME CMD
 1651 pts/0    00:00:00 bash
 2012 pts/0    00:00:00 csh
 2725 pts/0    00:00:00 ps
[e1600718@ens-iutva-0391 ~]$
```

La commande ps afficher l'état des processus en cours. On a donc notre session csh lancé depuis la session bash mais également la commande ps elle même

8/ Placez-vous dans le répertoire `SYS` créé lors d'un TP précédent (au besoin créez-le) et copiez-y le fichier `ps` du répertoire `/ubs/forum/prof/1tin01/ASR/M1101/tpsUNIX`

```
[e1600718@ens-iutva-0391 ~]$ cd ~/SYS/ && cp ~/tpsUNIX/ps ./
[e1600718@ens-iutva-0391 ~/SYS]$
```

9/ Visualisez le contenu du fichier copié afin d'en évaluer le rôle, puis lancez normalement la commande `ps` avec l'espoir de voir fonctionner ce script. Mais que se passe-t-il réellement ? Expliquez

```
[e1600718@ens-iutva-0391 ~/SYS]$ cat ./ps
#!/bin/tcsh
echo "La commande ps est ex@cut@e localement!"
[e1600718@ens-iutva-0391 ~/SYS]$
```

Ce script affiche simplement « La commande ps est exécutée localement! » cependant lorsqu'on lance ps ; cela exécute la commande ps et non pas le script ps car ~/SYS n'existe pas dans PATH.

10/ Lancez la commande `./ps`. Que se passe-t-il ? Justifiez

Pour forcer l'exécution du scipt on peut utiliser ./ps ainsi l'interpréteur de commande n'ira pas chercher la commande dans PATH mais dans le répertoire local.

```
[e1600718@ens-iutva-0391 ~/SYS]$ ./ps
La commande ps est exécutée localement!
[e1600718@ens-iutva-0391 ~/SYS]$
```

11/ Lancez la commande `demo`. Que se passe t-il ? Pourquoi ?

La commande est introuvable car inexistante dans les repertoires dans PATH.

```
[e1600718@ens-iutva-0391 ~/SYS]$ demo
demo: Commande introuvable.
[e1600718@ens-iutva-0391 ~/SYS]$
```

12/ Copiez dans SYS le fichier `demo` du répertoire `/ubs/forum/prof/1tin01/ASR/M1101/tpsUNIX` puis lancez à nouveau la commande `demo` (vous aurez éventuellement à redéfinir les droits). Que se passe-t-il ? Justifiez

```
[e1600718@ens-iutva-0391 ~/SYS]$ cp ~/tpsUNIX/demo ./
[e1600718@ens-iutva-0391 ~/SYS]$ ls
demo  ps
[e1600718@ens-iutva-0391 ~/SYS]$ demo
demo: Commande introuvable.
[e1600718@ens-iutva-0391 ~/SYS]$ ls -l
total 8
-rwxr--r--. 1 e1600718 etud 44  4 nov.  08:31 demo
-rwxr--r--. 1 e1600718 etud 59  4 nov.  08:21 ps
[e1600718@ens-iutva-0391 ~/SYS]$
```

C'est le même cas que tout à l'heure, l'interpréteur va chercher la commande demo dans les répertoires de PATH et non pas dans le répertoire courant alors que le fichier demo a les droit d'exécution.

13/ Faites-en sorte de réussir à lancer la commande `demo` depuis le répertoire courant.

```
[e1600718@ens-iutva-0391 ~/SYS]$ ./demo

Je suis une d@monstration.

[e1600718@ens-iutva-0391 ~/SYS]$
```

14/ Tapez la commande setenv PATH .:$PATH
Visualisez le contenu de la variable PATH. Commentez

```
[e1600718@ens-iutva-0391 ~/SYS]$ setenv  PATH  .:$PATH
[e1600718@ens-iutva-0391 ~/SYS]$ echo $PATH
.:/usr/lib64/ccache:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sb
in:/ubs/home/etud/2016/e1600718/.local/bin:/ubs/home/etud/2016/e1600718/
bin
[e1600718@ens-iutva-0391 ~/SYS]$
```

La commande a simplement ajouté .: au debut de PATH.

15/ Lancez à nouveau la commande demo
Expliquez la différence avec ce que vous aviez observé en 11/ 12/ et 13/. Justifiez

```
[e1600718@ens-iutva-0391 ~/SYS]$ demo

Je suis une d@monstration.

[e1600718@ens-iutva-0391 ~/SYS]$
```

À préset, la commande se lance car le répertoire courant representé par . est présent dans PATH.

16/ Lancez à nouveau la commande ps
Que constatez-vous ? Evaluez les risques de la manip effectuée en 14/

```
[e1600718@ens-iutva-0391 ~/SYS]$ ps
La commande ps est exécutée localement!
[e1600718@ens-iutva-0391 ~/SYS]$
```

Maintenant, lorsqu'on éxecute ps, ce n'est plus la commande UNIX mais le scipt car . est présent avant /bin/ dans le PATH. Le principale dangé de la commade effectuée en 14/ est de créer des ambiguités.

17/ Redonnez à la variable `PATH` son ancienne valeur (celle sauvegardée dans `OLDPATH`)

```
[e1600718@ens-iutva-0391 ~/SYS]$ setenv PATH $OLDPATH
[e1600718@ens-iutva-0391 ~/SYS]$
```

Exercice 2

1/ Affichez l'ensemble des variables d'environnement (donnez la commande après l'avoir testée, mais ne recopiez pas l'ensemble des résultats)

```
[e1600718@ens-iutva-0391 ~/SYS]$ printenv
XDG_VTNR=1
XDG_SESSION_ID=2
HOSTNAME=ens-iutva-0391.univ-ubs.fr
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/e1600718
TERM=xterm-256color
SHELL=/bin/bash
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4402
```

...

2/ Affichez l'ensemble des variables locales. Observez le résultat et commentez la première ligne affichée (celle commençant par)

On affiche les variables locales avec la commande set.

3/ Affichez le nombre de personnes ayant un compte sur le réseau de l'UBS

```
[e1600718@ens-iutva-0391 ~/SYS]$ ypcat passwd | wc --lines
16875
[e1600718@ens-iutva-0391 ~/SYS]$
```

Il y a au total 16875 comptes sur le réseau de l'UBS.

4/ Affichez le nombre de groupes d'utilisateurs du réseau de l'UBS

```
[e1600718@ens-iutva-0391 ~/SYS]$ ypcat group | wc --lines
2286
[e1600718@ens-iutva-0391 ~/SYS]$
```

Il y a au total 2286 groupes sur le réseau de l'UBS.

5/ Calculez le nombre moyen d'utilisateurs par groupe (plusieurs commandes sur une même ligne)

```
[e1600718@ens-iutva-0391 ~/SYS]$ set x1 = `ypcat passwd | wc --lines`
[e1600718@ens-iutva-0391 ~/SYS]$ set x2 = `ypcat group | wc --lines`
[e1600718@ens-iutva-0391 ~/SYS]$ @ x3 = $x1 / $x2
[e1600718@ens-iutva-0391 ~/SYS]$ echo Resultat : $x3
Resultat : 7
[e1600718@ens-iutva-0391 ~/SYS]$
```

6/ Calculez le nombre moyen d'utilisateurs par groupe en une seule commande (vous utiliserez la commande *expr*)

```
[e1600718@ens-iutva-0391 ~/SYS]$ echo Resultat : ; expr `ypcat passwd | wc --lines`
 / `ypcat group | wc --lines`
Resultat :
7
[e1600718@ens-iutva-0391 ~/SYS]$
```

7/ Idem 6/, mais en pourcentage d'utilisateurs par groupe (nbgroup * 100 / nbutil)

```
[e1600718@ens-iutva-0391 ~/SYS]$ echo Resultat : ; expr `ypcat group | wc
 --lines` \* 100 / `ypcat passwd | wc --lines`
Resultat :
13
[e1600718@ens-iutva-0391 ~/SYS]$
```

Exercice 3

```
total 1951
-rw-r--r--   1 portejoi     prof       315069 Dec 17  2010 ensI1.jpg
-rw-r-----   1 portejoi     prof       414900 Mar 10  2010 modele crtp.ps
-rw-r-----   1 portejoi     prof        12875 Mar 14  2010 netscape.ps
-rw-r-----   1 portejoi     prof        94275 Jan 27  2010 plan.txt
-rw-r-----   1 portejoi     prof       600897 Apr  1  2010 tp.txt
5 Fichiers dans /home/prof/portejoi/
```

a/ Ecrivez une séquence de commandes (possible en 2 commandes) qui force l'affichage du résultat sous la forme donnée ci-dessus. On y trouve tout d'abord, en nombre de blocs, la taille totale des fichiers du répertoire courant, sous-répertoires compris, puis les caractéristiques des différents fichiers (notez que ces 2 résultats sont le fruit d'une seule commande que vous connaissez bien), et enfin le nombre de fichiers contenus dans ce répertoire, par comptage du nombre de **mots** retournés par `ls`, suivi d'une chaîne de caractères et enfin de son nom.

Testez-la depuis votre répertoire d'accueil, mais comme dans un premier temps on ne tiendra pas compte de la présence du caractère espace dans certains noms de fichiers, vous constaterez que le nombre de ces derniers sera erroné. La solution précise sera établie en b/,

```
[e1600718@ens-iutva-0391 ~/SYS]$ ls -l ; echo `ls | wc -w` Fichers dans `pwd`
total 8
-rwxr--r--. 1 e1600718 etud 44  4 nov.  08:31 demo
-rw-r--r--. 1 e1600718 etud  0  4 nov.  09:19 'fichier avec des espaces'
-rwxr--r--. 1 e1600718 etud 59  4 nov.  08:21 ps
6 Fichers dans /ubs/fukuisaurus/home.1/8/e1600718/SYS
[e1600718@ens-iutva-0391 ~/SYS]$
```

b/ Sachant que la commande `sed "s/a/b/g" fichier` remplace toutes les occurrences (`g`) de `a` par `b` dans `fichier`, ré-écrivez la commande précédente de façon à pouvoir prendre en compte l'espace dans un nom de fichier (et donc l'ignorer). Testez-la depuis votre répertoire d'accueil.

```
[e1600718@ens-iutva-0391 ~/SYS]$ ls -l ; echo `ls | sed "s/\ //g" | wc -w` Fichers dans `pwd`
total 8
-rwxr--r--. 1 e1600718 etud 44  4 nov.  08:31 demo
-rw-r--r--. 1 e1600718 etud  0  4 nov.  09:19 'fichier avec des espaces'
-rwxr--r--. 1 e1600718 etud 59  4 nov.  08:21 ps
3 Fichers dans /ubs/fukuisaurus/home.1/8/e1600718/SYS
[e1600718@ens-iutva-0391 ~/SYS]$
```