Variables du C-shell (csh)

D. Bogdaniuk – P. Portejoie

Gicquel Antoine	**14/10/16**
1A2	

NB : on suppose pour tout le TD que l'interpréteur est en *C-shell* (*csh*), mais sous UNIX, il existe d'autres langages de commande. En TP vous devrez prendre quelques précautions pour qu'il en soit ainsi car le langage de commande initial à l'ouverture d'un terminal est le *bash*.

Il est possible dans un interpréteur de commandes (un shell) de définir des variables, c'est à dire, d'associer un nom et une valeur. Le shell manipule deux types de variables :

- les **variables locales** qui ne sont visibles que dans l'interpréteur de commandes où elles ont été définies
- les **variables globales** ou **variables d'environnement**, qui sont visibles dans l'interpréteur où elles ont été définies ainsi que dans les commandes, programmes ou shells, appelés à partir de celui-ci

Les variables locales

Les variables locales ne sont visibles que dans l'interpréteur shell qui les a définies. Par convention, leur nom est toujours en minuscule. A cet égard, UNIX faisant la distinction entre minuscules et majuscules, il ne faut pas confondre les deux variables chemin et CHEMIN. Le contenu d'une variable est accessible en préfixant son nom par $.

Plusieurs commandes permettent de manipuler les variables locales :

- **set**
  sans paramètre permet de lister les variables locales créées et de connaître leur contenu
- **set <*variable*>=<*valeur*>**
  permet d'affecter une valeur à une variable. Par exemple pour créer une variable compteur à 1, il suffit de donner la commande : `set compteur=1`
- **set <*variable*>**
  permet de créer une variable sans contenu. Elle est dite booléenne car c'est son existence ou son absence qui est testée
- **unset <*variable*>**
  permet de détruire une variable. Par exemple, pour détruire la variable compteur, il suffit de donner la commande : `unset compteur`
- **echo <$*variable*>**
  permet d'afficher le contenu d'une variable. Par exemple, pour afficher le contenu de la variable compteur, il suffit de lancer la commande : `echo $compteur`

Certaines variables ont un sens prédéfini :

- **noclobber** quand cette variable existe, il est impossible d'effectuer une redirection vers un fichier existant
- **home** indique le répertoire d'accueil à la connexion
- **prompt** définit l'invite de la ligne de commande
- **cwd** contient le nom du répertoire courant
- Enfin la variable **$<** permet de saisir une information sur l'entrée standard (le clavier).
  Par exemple : `set saisie=$<`

Les variables sont utilisables pour effectuer des calculs. Ces derniers sont réalisés par la commande : `@` ***<variable> = <expression>***

Par exemple, l'addition de deux variables `a` et `b` dans la variable `c` se fait par :
```
@ c = $a + $b
```

Remarque : **Notez les espaces de la formule précédente** ; ils sont indispensables.
Les opérateurs disponibles sont ceux de C ou de java.

Les variables d'environnement

Les variables d'environnement sont visibles dans les programmes appelés à partir du shell qui les a créées. Ces variables sont également transmises aux programmes appelés par celui-ci et ainsi de suite. Par convention, leur nom est toujours en majuscule.

Comme leur nom l'indique, les variables d'environnement permettent de transmettre des informations sur le système aux autres programmes. Quelques variables d'environnement doivent être connues :

- **HOME** contient le nom du répertoire de connexion
- **PATH** contient la liste des répertoires où l'interpréteur doit chercher les commandes à exécuter
- **SHELL** contient le nom de l'interpréteur de commandes à la connexion
- **LANG** contient le langage utilisé
- **PRINTER** contient le nom de l'imprimante par défaut
- **DISPLAY** contient le nom de la machine sur laquelle l'affichage s'effectue

Plusieurs commandes permettent de manipuler les variables d'environnement :

- **env**, **setenv** ou **printenv**
  permet de lister les variables globales créées et de connaître leur contenu
- **setenv <*VARIABLE*>** <***valeur***>
  permet d'affecter une valeur à une variable. Par exemple pour créer une variable `COMPTEUR` à `1`, il suffit de lancer la commande : `setenv COMPTEUR 1`
- **unsetenv <*VARIABLE*>**
  permet de détruire une variable. Par exemple, pour détruire la variable `COMPTEUR`, il suffit de lancer la commande : `unsetenv COMPTEUR`
- **echo <$*VARIABLE*>**
  permet d'afficher le contenu d'une variable. Par exemple, pour afficher le contenu de la variable `COMPTEUR`, il suffit de lancer la commande : `echo $COMPTEUR`

Les variables d'environnement sont en fait recopiées dans l'environnement des programmes appelés. Ces derniers peuvent modifier leur contenu, mais pour l'interpréteur appelant les valeurs restent inchangées. Cette notion est illustrée par la figure suivante :

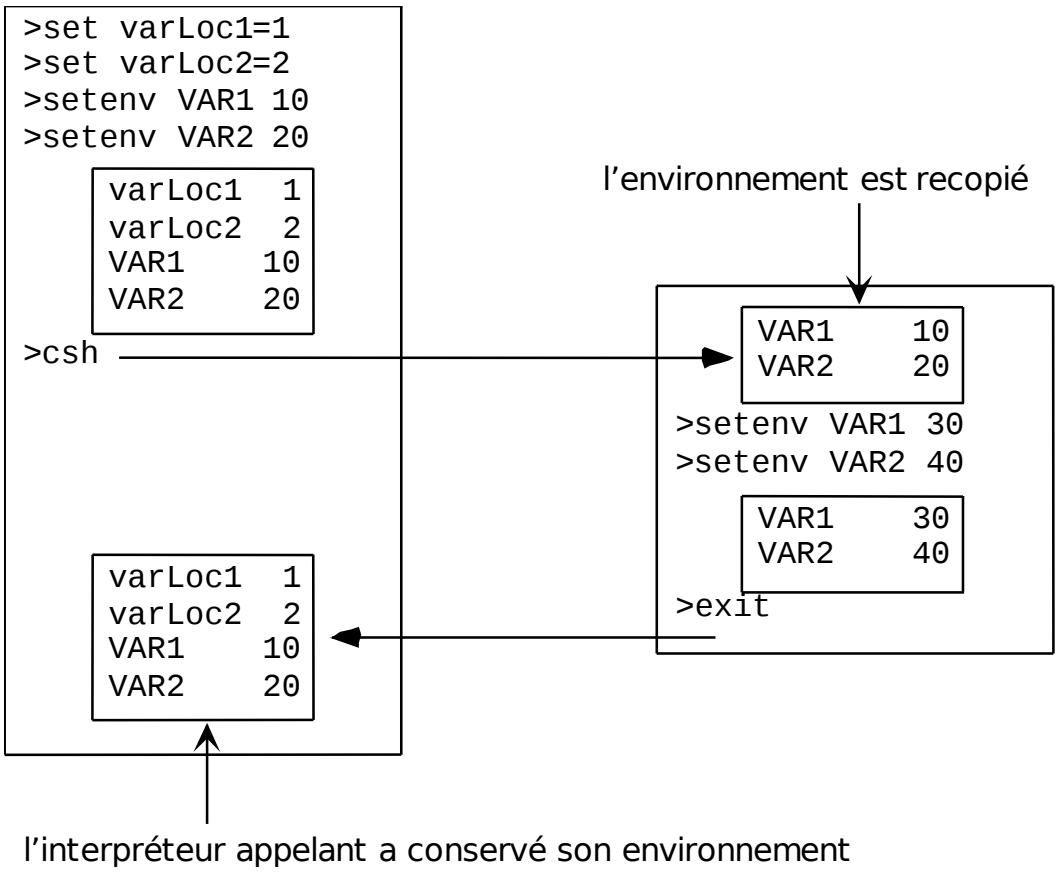

Fig 0 - Exemple de portée des variables

Question préliminaire

Suite à la lecture de la présentation, trouvez-y une variable booléenne prédéfinie du système. Rappelez-en le rôle.

Exercice 1

1 - Les instructions étant exécutées dans l'ordre, donnez pour chaque `echo` la valeur affichée :

Commande	Affichage

`set compteur=1`
`setenv X 10`
`echo $compteur`
`echo $X`
`set X=20`
	
	1
	10

`csh`
`echo $compteur`
`echo $X`
`set solde=2`
`setenv Y 20`
`@ total = $solde + $solde`
`echo $total`
	
	pas définie
	10
	
	4

`csh`
`echo $X`
`echo $Y`
`set solde=100`
`setenv Y 30`
`exit`
	
	10
	20

`echo $Y`
`echo $solde`
`exit`
	20
	2

`echo $Y`
`echo $X`
`unset X`
`echo $X`
`echo $compteur`
	pas définie
	20
	
	10
	1

2 - Que se passe-t-il si on remplace unset X par unsetenv X dans la dernière série de commandes ?

→ La variable X désigne à la fois une variable locale et une variable globale. Les variables locales sont privilégiés mais si on supprime la varibla globale par un unsetenv X, alors echo $X affichera la valeur de la variable locale c'est à dire 20.

Remarque : il est préférable (par convention) de nommer les variables locales en minuscules et globales en majuscules, mais pas indispensable (cf ci-dessus) avec les risques d'ambiguïté que cela comporte.

Exercice 2

Pour chaque commande `echo`, donnez le résultat affiché :

Commande	Affichage

`set a=10`
`set b=20`
`@ a = $b + 1`
`echo $a`
	21

`echo $a + $b`	21 + 20
`@ c = $a + $b` `echo $a + $b = $c`	21 + 20 = 41
`set c=44` `echo $a + $b = $c`	21 + 20 =44

`setenv X I`
`setenv Y U`
`setenv Z T`
`setenv NOM $X$Y$Z`
`echo $NOM`
	IUT

Exercice 3

Le fichier `conf` contient le script suivant (pour des raisons pratiques celui utilisé en TP sera légèrement différent) :

```
#!/bin/csh
1echo $X
setenv X 5
2echo $x
set x=3
if ( $# == 0 ) then                # $# contient le nombre d'arguments
               setenv X 10
               exit(0)
endif
setenv Y $1                        # $1 référence l'argument de rang 1
3echo $X
4echo $Y
```

NB : *source lance la commande dans le shell courant alors que ./ lance un nouveau shell ; exit est implicite en fin de script.*

Quel est l'affichage produit par chacune des commandes `echo` dans les 3 cas suivants (dans chacun des cas n'hésitez pas à représenter graphiquement les variables utilisées) :

1er cas	2ème cas	3ème cas
setenv X 8 **set x=4**	**setenv X 8** **set x=4**	**setenv X 8** **set x=4**
./conf	**./conf 2**	**source conf 2**
1echo $X	*1echo $X*	*1echo $X*
2echo $x	*2echo $x*	*2echo $x*
3echo $X	*3echo $X*	*3echo $X*
4echo $Y	*4echo $Y*	*4echo $Y*
echo $X	**echo $X**	**echo $X**
echo $Y	**echo $Y**	**echo $Y**
echo $x	**echo $x**	**echo $x**

Exercice 4

Ecrivez un script en csh qui demande par 2 fois à l'utilisateur d'entrer 1 entier au clavier (affichage d'un message puis saisie) et en affiche la somme et la moyenne sous la forme :

Entier1 : --
Entier2 : --
Somme = --
Moyenne = –

En TP :

NB : les exercices sont à faire en *C-shell* (*csh*), mais le langage de commande initial à la connexion est le *bash* et par conséquent toute ouverture de terminal lance un interpréteur en *bash*. Vous devrez donc veiller à y utiliser en tout premier lieu la commande `csh` (ou `/bin/csh` selon configuration) afin d'y travailler en *C-shell*.

Exercice préliminaire

Affichez la valeur des variables d'environnement suivantes : HOME, USER, DISPLAY, TERM, SHELL, PATH. Dans votre compte-rendu donnez-en une copie d'écran et rappelez pour chacune d'elles à quoi elles correspondent.

```
[e1600718@ens-iutva-0225 e1600718]$ echo $HOME
/ubs/home/etud/2016/e1600718
[e1600718@ens-iutva-0225 e1600718]$ echo $USER
e1600718
[e1600718@ens-iutva-0225 e1600718]$ echo $DISPLAY
:0
[e1600718@ens-iutva-0225 e1600718]$ echo $TERM
xterm-256color
[e1600718@ens-iutva-0225 e1600718]$ echo $SHELL
/bin/bash
[e1600718@ens-iutva-0225 e1600718]$ echo $PATH
/usr/lib64/ccache:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/ubs/ho
me/etud/2016/e1600718/.local/bin:/ubs/home/etud/2016/e1600718/bin
[e1600718@ens-iutva-0225 e1600718]$
```

HOME : le répertoire d'accueil par défaut
USER : le nom d'utilisateur par défaut
DISPLAY : le tty ou est affiché l'information par défaut
TERM : le profil du terminal (avec support couleur par exemple)
SHELL : le shell par défaut
PATH : endroit où sont situés les exécutables

Exercices 1 et 2

Reprenez les exercices 1 et 2 vus en TD dans un terminal, en testant et vérifiant les résultats (donnez-en une copie).

1)

```
[e1600718@ens-iutva-0027 e1600718]$ cd
[e1600718@ens-iutva-0027 ~]$ csh
[e1600718@ens-iutva-0027 ~]$ set compteur=1
[e1600718@ens-iutva-0027 ~]$ setenv X 10
[e1600718@ens-iutva-0027 ~]$ echo $compteur
1
[e1600718@ens-iutva-0027 ~]$ echo $X
10
[e1600718@ens-iutva-0027 ~]$ set X=20
[e1600718@ens-iutva-0027 ~]$ csh
[e1600718@ens-iutva-0027 ~]$ echo $compteur
compteur: Variable pas définie.
[e1600718@ens-iutva-0027 ~]$ echo $X
10
[e1600718@ens-iutva-0027 ~]$ set solde=2
[e1600718@ens-iutva-0027 ~]$ setenv Y 20
[e1600718@ens-iutva-0027 ~]$ @ total = $solde + $solde
[e1600718@ens-iutva-0027 ~]$ echo $total
4
[e1600718@ens-iutva-0027 ~]$ csh
[e1600718@ens-iutva-0027 ~]$ echo $X
10
[e1600718@ens-iutva-0027 ~]$ echo $Y
20
[e1600718@ens-iutva-0027 ~]$ set solde=100
[e1600718@ens-iutva-0027 ~]$ setenv Y 30
[e1600718@ens-iutva-0027 ~]$ exit
exit
[e1600718@ens-iutva-0027 ~]$ echo $Y
20
[e1600718@ens-iutva-0027 ~]$ echo $solde
2
[e1600718@ens-iutva-0027 ~]$ exit
exit
[e1600718@ens-iutva-0027 ~]$ echo $Y
Y: Variable pas définie.
[e1600718@ens-iutva-0027 ~]$ echo $X
20
[e1600718@ens-iutva-0027 ~]$ unset X
[e1600718@ens-iutva-0027 ~]$ echo $X
10
[e1600718@ens-iutva-0027 ~]$ echo $compteur
1
[e1600718@ens-iutva-0027 ~]$
```

2)

```
[e1600718@ens-iutva-0027 e1600718]$ csh
[e1600718@ens-iutva-0027 ~]$ set a=10
[e1600718@ens-iutva-0027 ~]$ set b=20
[e1600718@ens-iutva-0027 ~]$ @ a = $b + 1
[e1600718@ens-iutva-0027 ~]$ echo $a
21
[e1600718@ens-iutva-0027 ~]$ echo $a + $b
21 + 20
[e1600718@ens-iutva-0027 ~]$ @ c = $a + $b
[e1600718@ens-iutva-0027 ~]$ echo $a + $b = $c
21 + 20 = 41
[e1600718@ens-iutva-0027 ~]$ set c=44
[e1600718@ens-iutva-0027 ~]$ echo $a + $b = $c
21 + 20 = 44
[e1600718@ens-iutva-0027 ~]$ setenv X I
[e1600718@ens-iutva-0027 ~]$ setenv Y U
[e1600718@ens-iutva-0027 ~]$ setenv Z T
[e1600718@ens-iutva-0027 ~]$ setenv NOM $X$Y$Z
[e1600718@ens-iutva-0027 ~]$ echo $NOM
IUT
[e1600718@ens-iutva-0027 ~]$
```

Exercice 3

Afin de repartir sur de bonnes bases fermez le terminal, puis rouvrez-en un nouveau (n'oubliez pas d'y faire **csh**). Reprenez ensuite chacun des cas de l'exercice 3 vu en TD, en testant et vérifiant les résultats (donnez-en une copie).

Vous trouverez le script **conf** *dans le forum habituel. Pour des raisons purement pratiques, il diffère légèrement de celui proposé en TD, mais remplit les mêmes fonctions.*

1)

```
[e1600718@ens-iutva-0027 e1600718]$ csh
[e1600718@ens-iutva-0027 ~]$ setenv X 8
[e1600718@ens-iutva-0027 ~]$ set x=4
[e1600718@ens-iutva-0027 ~]$ ./conf
8
x: Variable pas définie.
[e1600718@ens-iutva-0027 ~]$ echo $X
8
[e1600718@ens-iutva-0027 ~]$ echo $Y
Y: Variable pas définie.
[e1600718@ens-iutva-0027 ~]$ echo $x
4
```

2)

```
[e1600718@ens-iutva-0027 e1600718]$ csh
[e1600718@ens-iutva-0027 ~]$ setenv X 8
[e1600718@ens-iutva-0027 ~]$ set x=4
[e1600718@ens-iutva-0027 ~]$ ./conf 2
8
x: Variable pas définie.
5
2
[e1600718@ens-iutva-0027 ~]$ echo $X
8
[e1600718@ens-iutva-0027 ~]$ echo $Y
Y: Variable pas définie.
[e1600718@ens-iutva-0027 ~]$ echo $x
4
[e1600718@ens-iutva-0027 ~]$
```

3)

```
[e1600718@ens-iutva-0027 e1600718]$ csh
[e1600718@ens-iutva-0027 ~]$ setenv X 8
[e1600718@ens-iutva-0027 ~]$ set x=4
[e1600718@ens-iutva-0027 ~]$ source conf 2
8
4
5
2
[e1600718@ens-iutva-0027 ~]$ echo $X
5
[e1600718@ens-iutva-0027 ~]$ echo $Y
2
[e1600718@ens-iutva-0027 ~]$ echo $x
3
[e1600718@ens-iutva-0027 ~]$
```

Exercice 4

Écrivez le script demandé si ce n'est déjà fait, testez-le et vérifiez les résultats.

tp5_asr

```
1  #!/bin/csh
2
3  echo "Entrez la première valeur"
4  @ x1=$<
5  echo "Entrez la deuxème valeur"
6  @ x2=$<
7
8  @ somme = $x1 + $x2
9  @ moyenne = $somme / 2
10
11 echo "Somme : " + $somme
12 echo "Moyenne : " + $moyenne
13
14 exit(0)
15
```

Test et vérification :

```
[e1600718@ens-iutva-0027 ~]$ csh
[e1600718@ens-iutva-0027 ~]$ ./tp5_asr
Entrez la première valeur
15
Entrez la deuxème valeur
10
Somme :  + 25
Moyenne :  + 12
[e1600718@ens-iutva-0027 ~]$
```

→ le csh ne supporte pas les nombres flottants donc la moyenne est 12 au lieu de 12.5.

Question subsidiaire (fonction du temps et de vos capacités) :

Réécrivez le script précédent afin que les 2 entiers soient passés en arguments

tp5_asr_bonus

```
1   #!/bin/csh
2
3   if ( $# != 2 )  then
4           echo "2 arguments attendus"
5           exit(0)
6   endif
7
8   @ somme = $1 + $2
9   @ moyenne = $somme / 2
10
11  echo "Somme : " + $somme
12  echo "Moyenne : " + $moyenne
13
14  exit(0)
15
```

```
[e1600718@ens-iutva-0390 ~]$ ./tp5_asr_bonus
2 arguments attendus
[e1600718@ens-iutva-0390 ~]$ ./tp5_asr_bonus 5
2 arguments attendus
[e1600718@ens-iutva-0390 ~]$ ./tp5_asr_bonus 5 20
Somme :  + 25
Moyenne :  + 12
[e1600718@ens-iutva-0390 ~]$
```