Ressourcesinformatiques

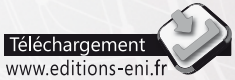
Programmation shell sous Unix/Linux

sh, ksh, bash

(avec exercices corrigés)

4ième édition

Christine DEFFAIX RÉMY



Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **RI4PRO** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 3
Paramétrage de l'environnement de travail

Chapitre 4
Les bases de la programmation shell

Chapitre 5

Aspects avancés de la programmation shell

Chapitre 6
Les expressions régulières

Chapitre 7
La commande sed

Chapitre 8
Le langage de programmation awk

Chapitre 9
Les commandes filtres

Chapitre 10
Solutions des exercices

Annexes

Chapitre 3
Paramétrage de l'environnement de travail

1. Variables d'environnement

Les thèmes abordés dans ce chapitre permettront à l'utilisateur de paramétrer son environnement de travail en tenant compte du shell utilisé.

Un certain nombre de variables sont définies dans l'environnement du shell. Elles contiennent des informations nécessaires au fonctionnement de l'interpréteur et/ou des commandes lancées à partir de celui-ci.

1.1 Liste des variables

La commande **set** donne la liste des variables définies dans le shell courant.

Exemple

```
$ set
HOME=/home/christie
LOGNAME=christie
PATH=/usr/bin:/bin
PS1='$ '
PS2='> '
TERM=vt100
...
```

1.2 Affichage de la valeur d'une variable

Le caractère spécial **$** du shell permet d'accéder au contenu d'une variable.

Exemple

```
$ echo $HOME
/home/christie
$
```

1.3 Modification de la valeur d'une variable

Le shell permet d'initialiser ou de modifier des variables.

Exemple

```
$ variable=valeur
$ echo $variable
valeur
$$
```

Si la valeur contient des caractères spéciaux du shell ($, >, espace...), il faut empêcher le shell d'interpréter ceux-ci en entourant la valeur avec des simples quotes.

Remarque

Utiliser des simples quotes est l'une des trois manières de masquer des caractères en shell. Ce point sera détaillé ultérieurement.

Exemple

Le symbole ">" (redirection) doit être masqué, l'espace (séparateur de mots sur la ligne de commande) également :

```
$ variable='mot1 mot2 =>'
$ echo $variable
mot1 mot2 =>
$
```

Remarque

Il ne faut pas mettre d'espace autour du signe **=**. *Le shell ne comprendrait pas qu'il s'agit d'une affectation.*

1.4 Principales variables

Les variables présentées ci-dessous possèdent une valeur au niveau du shell de connexion. D'autres variables peuvent être définies ultérieurement.

1.4.1 HOME

Cette variable contient la valeur du répertoire d'accueil de l'utilisateur. Elle ne doit pas être modifiée.

1.4.2 PATH

La variable PATH contient une liste de répertoires qui sont explorés par le shell lorsqu'il doit lancer une commande externe.

Remarque

En aucun cas, une commande n'est recherchée dans le répertoire courant si celui-ci ne figure pas dans la variable PATH.

Exemples

```
$ echo $PATH
/usr/bin:/bin
$
```

La commande **date** est trouvée :

```
$ date
Tue Jan 28 17:51:23 MET 2014
$
```

En effet, elle se situe dans le répertoire **/usr/bin** :

```
$ find / -name date 2> /dev/null
/usr/bin/date
$
```

La commande **ping** n'est pas trouvée :

```
$ ping localhost
ksh: ping:  not found
$
```

La commande est située dans le répertoire **/usr/sbin** qui n'est pas cité dans la variable PATH :

```
$ find / -name ping 2> /dev/null
/usr/sbin/ping
$
```

Le répertoire courant n'est pas exploré s'il n'est pas cité dans PATH :

```
$ cd /usr/sbin
$ ping localhost
ksh: ping:  not found
$
```

Modifier le contenu de la variable PATH :

```
$ PATH=$PATH:/usr/sbin
$ echo $PATH
/usr/bin:/bin:/usr/sbin
$
```

La commande **ping** est trouvée :

```
$ ping localhost
localhost is alive
$
```

Rechercher une commande dans le répertoire courant

Pour qu'une commande soit recherchée dans le répertoire courant, il faut ajouter en fin de variable PATH la chaîne "**:.**" ou simplement le caractère "**:**".

Exemple

```
PATH=/usr/bin:/usr/local/bin:/home/christie/bin:.
```

Équivalent à :

```
PATH=/usr/bin:/usr/local/bin:/home/christie/bin:
```

1.4.3 PWD

ksh	bash

Cette variable contient la valeur du répertoire courant. Elle est mise à jour par le shell dès que l'utilisateur change de répertoire. Cette variable peut être utilisée en ksh pour faire apparaître la valeur du répertoire courant dans le prompt.

1.4.4 PS1

Cette variable contient la chaîne de caractères représentant le prompt principal.

Exemple

```
$ echo $PS1
$
$ PS1='Entrez une commande => '
Entrez une commande => date
Thu Jan 30 17:27:51 MET 2014
Entrez une commande =>
```

Avec le ksh et le bash, il est possible de paramétrer son prompt de telle façon qu'il contienne en permanence la valeur du répertoire courant.

Faire apparaître la valeur du répertoire courant dans le prompt en ksh

Il faut se servir de la variable PWD.

Exemple

Ici, le prompt est composé de deux caractères : le symbole "$" suivi d'un espace (cf. Figure 1) :

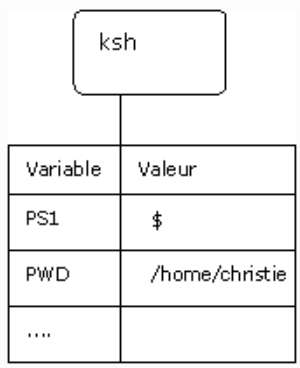


Figure 1 : Initialisation de PS1 avec le répertoire courant (1)

```
$
$ echo -$PS1-
-$ -
$
```

Le répertoire courant est **/home/christie** :

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

Initialisation de PS1 avec la chaîne de caractères 'PWD' ; il faut empêcher le shell de substituer $PWD par sa valeur au moment de l'affectation, donc il faut protéger l'expression avec des quotes (cf. Figure 2) :

```
$ PS1='$PWD$␣'
```