Capítulo 3
Configuración del entorno de trabajo

1. Presentación

Los temas abordados en este capítulo permitirán al usuario configurar su entorno de trabajo teniendo en cuenta el shell utilizado.

Se definen una serie de variables en el entorno del shell. Estas contienen la información necesaria para el funcionamiento del intérprete o de los comandos ejecutados por este.

2. Variables de entorno

2.1 Listado de variables

El comando **set** devuelve la lista de las variables definidas en el shell actual.

Ejemplo

```
$ set
HOME=/home/cristina
LOGNAME=cristina
PATH=/usr/bin:/bin
PS1='$ '
PS2='> '
```

```
TERM=vt100
...
```

2.2 Mostrar el valor de una variable

El carácter especial **$** del shell permite acceder al contenido de una variable.

Ejemplo

```
$ echo $HOME
/home/cristina
$
```

2.3 Modificación del valor de una variable

El shell permite inicializar y modificar variables.

Ejemplo

```
$ variable=valor
$ echo $variable
valor
$
```

Si el valor contiene caracteres especiales del shell (**$**, >, espacio...), hay que impedir que el shell los interprete poniendo el valor entre apóstrofes.

Observación

Utilizar apóstrophes es una de las tres maneras posibles de enmascarar caracteres en shell. Ver el capítulo Las bases de la programación shell - Caracteres de protección.

Ejemplo

El símbolo > (redirección) tiene que enmascararse, el espacio (separador de palabras en la línea de comandos) también:

```
$ variable='palabra1 palabra2 =>'
$ echo $variable
palabra1 palabra2 =>
$
```

Observación

No hay que poner espacios alrededor del símbolo **=**. *El shell no comprendería que se trata de una asignación.*

2.4 Variables principales

Las variables presentadas a continuación tienen un valor definido en el momento de conexión. Otras variables pueden ser definidas posteriormente.

La modificación de una variable de entorno en línea de comandos es válida solo en el shell actual. Para que las modificaciones sean tomadas en cuenta en todos los shells, debemos utilizar los archivos de configuración (ver Los archivos de entorno en ese capítulo).

2.4.1 HOME

Esta variable contiene el valor del directorio de inicio del usuario. No debe ser modificada.

2.4.2 PATH

La variable PATH contiene una lista de directorios que el shell explora cuando este debe invocar un comando externo.

Observación

En ningún caso, un comando se buscará en el directorio actual si este no figura en la variable PATH.

Ejemplos

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

El comando **date** se conoce:

```
$ date
Fri Jan 28 17:51:23 MET 2023
$
```

En efecto, se encuentra en el directorio **/usr/bin**:

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

El comando **ping** no se conoce:

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

El comando se encuentra dentro del directorio **/usr/sbin**, que no está incluido en la variable PATH:

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

El directorio actual no se explora si no se cita en PATH:

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

Modificar el contenido de la variable PATH:

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

El comando **ping** ahora se conoce:

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

Buscar un comando en el directorio actual

Para que un comando se busque en el directorio actual, hay que añadir al final de la variable PATH la cadena "**:.**" o simplemente el carácter "**:**".

Ejemplo

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

Es equivalente a:

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

2.4.3 PWD

Esta variable solo está disponible en bash y ksh y contiene el valor del directorio actual. Cada vez que el usuario cambia de directorio, el shell se encarga de actualizarla. Esta variable puede utilizarse en ksh para mostrar el valor del directorio actual en el prompt.

2.4.4 PS1

Esta variable solo está disponible en bash y ksh y contiene la cadena de caracteres que representan el prompt principal.

Ejemplo

```
$ echo $PS1
$
$ PS1='Entre un comando => '
Entre un comando => date
Mon Jan 30 17:27:51 MET 2023
Entre un comando =>
```

En ksh y en bash, es posible configurar el prompt de tal forma que contenga permanentemente el valor del directorio actual.

Mostrar el directorio actual en el prompt en ksh

Se usa la variable PWD.

Ejemplo

A continuación, el prompt se compone de dos caracteres: el símbolo "$" seguido de un espacio (ver Figura 1):

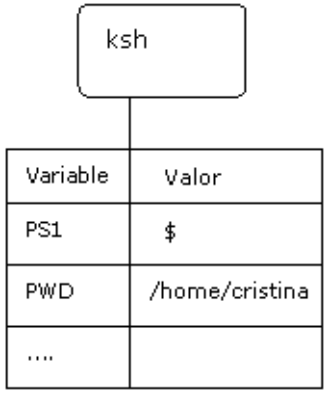

Figura 1: Inicialización de PS1 con el directorio actual (1)

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

El directorio actual es **/home/cristina**:

```
$
$ pwd
/home/cristina
$
```

Inicialización de PS1 con la cadena de caracteres 'PWD' ; es necesario impedir que el shell substituya $PWD por su valor en el momento de la asignación; por lo tanto, hay que proteger la expresión con apóstrofes (ver Figura 2):

```
$  PS1='$PWD$␣'
```

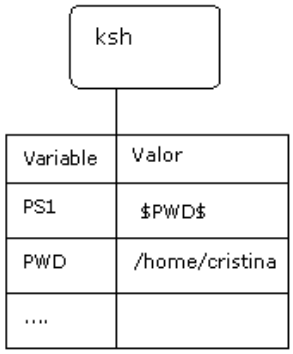

Figura 2: Inicialización de PS1 con el directorio actual (2)

El shell debe mostrar ahora su prompt. Va a buscar el valor de PS1 (PWD). La variable PWD se evalúa y se remplaza por su valor (actualmente `/home/cristina`):

```
/home/cristina$
```

Cambio de directorio:

```
/home/cristina$ cd /tmp
```

El shell actualiza inmediatamente la variable PWD, que ahora vale "/tmp". Después tiene que mostrar su prompt. Vuelve a buscar el valor de PS1 (PWD) y lo evalúa. Por lo tanto, PS1 tiene por valor "/tmp" (ver Figura 3):

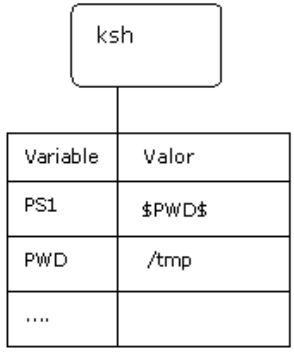

Figura 3: Inicialización de PS1 con el directorio actual (3)