Las variables

Se distinguen tres tipos: usuario, sistema y especiales. El principio consiste en poder asignar un contenido a un nombre de variable, en general una cadena de caracteres o valores numéricos.

1. Nomenclatura

Un nombre de variables obedece a ciertas reglas:

- Se puede componer de letras minúsculas, mayúsculas, cifras, caracteres de subrayado.
- El primer carácter no puede ser una cifra.
- El tamaño de un nombre suele ser ilimitado (pero no hay que pasarse tampoco).
- Las convenciones quieren que las variables de usuario estén en minúsculas para diferenciarlas de las variables de sistema. A elección del usuario.

2. Declaración y asignación

Se declara una variable en cuanto se le asigna un valor. Se efectúa la asignación con el signo **=**, sin espacio antes ni después del signo.

```
var=Hola
```

3. Acceso y visualización

Puede acceder al contenido de una variable colocando el signo **$** delante del nombre de la variable. Cuando el shell encuentra el **$**, intenta interpretar la palabra siguiente como si fuera una variable. Si existe, entonces se sustituye el $nombre_variable por su contenido, o

por un texto vacío en el caso contrario. Se habla también de referenciado de variable.

```
$ ruta=/tmp/seb
$ ls $ruta
...
$ cd $ruta
$ pwd
/tmp/seb
$ cd $ruta/dir1
$ pwd
/tmp/seb/dir1
```

Para visualizar la lista de las variables, se utiliza el comando **env**. Muestra las variables de usuario y de sistema, nombre y contenido.

```
$ env

HOSTTYPE=x86_64
SSH_CONNECTION=192.168.25.1 54564 192.168.25.26 22
LESSCLOSE=lessclose.sh %s %s
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
LANG=es_ES.UTF-8
WINDOWMANAGER=/usr/bin/gnome
LESS=-M -I -R
JAVA_ROOT=/usr/lib64/jvm/jre-11-openjdk
HOSTNAME=localhost.localdomain
CONFIG_SITE=/usr/share/site/x86_64-unknown-linux-gnu
CSHEDIT=emacs
GPG_TTY=/dev/pts/2
AUDIODRIVER=pulseaudio
LESS_ADVANCED_PREPROCESSOR=no
COLORTERM=1
JAVA_HOME=/usr/lib64/jvm/jre-11-openjdk
ALSA_CONFIG_PATH=/etc/alsa-pulse.conf
MACHTYPE=x86_64-suse-linux
QEMU_AUDIO_DRV=pa
MINICOM=-c on
QT_SYSTEM_DIR=/usr/share/desktop-data
OSTYPE=linux
```

```
XDG_SESSION_ID=2
USER=alejandro
PAGER=less
MORE=-sl
PWD=/home/alejandro
...
_=/usr/bin/env...
```

Una variable puede contener caracteres especiales, particularmente el espacio. El ejemplo siguiente no funciona:

```
$ c=Hola amigos
amigos: not found
$ echo $c
```

Para hacer que funcione, hay que cerrar los caracteres especiales uno por uno, o colocarlos entre comillas o apóstrofes.

```
c=Hola\ Amigos # Solución pesada
c="Hola amigos" # Solución correcta
c='Hola amigos' # Solución correcta
```

La principal diferencia entre las comillas y los apóstrofes es la interpretación de las variables y de las sustituciones. " y ' se cierran mutuamente.

```
$ a=Julio
$ b=César
$ c="$a $b conquistó la Galia"
$ d='$a $b conquistó la Galia'
$ echo $c
Julio César conquistó la Galia
$ echo $d
$a $b conquistó la Galia
$ echo "Linux es genial"
Linux es genial
```

```
$ echo 'Linux "demasiado bien"'
Linux "demasiado bien"
```

4. Supresión y protección

Se suprime una variable con el comando **unset**. Puede proteger una variable en modo escritura y contra su supresión con el comando **readonly**. Una variable en modo de sólo lectura, incluso vacía, es exclusiva. No existe ningún medio de sustituirla en escritura y de suprimirla, salvo saliendo del shell.

```
$ a=Julio
$ b=César
$ echo $a $b
Julio César
$ unset b
$ echo $a $b
Julio
$ readonly a
$ a=Nerón
a: is read only
$ unset a
a: is read only
```

5. Export

Por defecto una variable es accesible únicamente desde el shell donde ha sido definida. El siguiente ejemplo declara la variable **a** en el entorno del shell actual y luego trata de visualizar su valor a través de un script lanzado desde este mismo shell. El script no reconoce la variable **a**: no se visualiza nada.

```
$ a=Julio
```

```
$ echo 'echo "a=$a"' > ver_a.sh
$ chmod u+x ver_a.sh
$ ./ver_a.sh
a=
```

El comando **export** permite exportar una variable de manera que su contenido sea visible por los scripts y otros subshells. Se pueden modificar las variables exportadas en el script, pero estas modificaciones sólo se aplican al script o al subshell. Esta vez, el primer script puede acceder a la variable **a** exportada. Pero las modificaciones siguen siendo locales en el script. Una vez éste ha terminado, la modificación desaparece.

```
$ export a
$ ./ver_a.sh
a=Julio
$ echo 'a=Nerón ; echo "a=$a"' >> ver_a.sh
$ ./ver_a.sh
a=Julio
a=Nerón
$ echo $a
Julio
```

6. Llaves

Las llaves básicas {} permiten identificar el nombre de una variable. Imaginemos la variable archivo que contiene el nombre de archivo «lista». Quiere copiar lista1 en lista2.

```
$ archivo=lista
$ cp $archivo1 $archivo2
cp: operando archivo que falta
Para saber más, haga: «cp --help».
```

No funciona, ya que no se interpreta $archivo sino $archivo1 y $archivo2, que no existen.

```
$ cp ${archivo}2 ${archivo}1
```

En este caso, esta línea equivale a:

```
$ cp lista2 lista1
```

7. Llaves y sustitución condicional

Las llaves disponen de una sintaxis particular.

```
{variable:Sustitución}
```

Según el valor o la presencia de la variable, es posible sustituir su valor por otro.

Sustitución	Significado
`{x:-texto}`	Si la variable x está vacía o es inexistente, el texto cogerá su sitio. En caso contrario, es el contenido de la variable el que prevaldrá.
`{x:=texto}`	Si la variable x está vacía o es inexistente, el texto cogerá su sitio y se convertirá en el valor de la variable.
`{x:+texto}`	Si la variable x está definida y no vacía, el texto cogerá su sitio. En el caso contrario, una cadena vacía coge su lugar.
`{x:?texto}`	Si la variable x está vacía o es inexistente, se interrumpe el script y se visualiza el mensaje de texto. Si el texto está ausente, se visualiza un mensaje de error estándar.

```
$ echo $nombre

$ echo ${nombre:-Juan}
Juan
$ echo $nombre

$ echo ${nombre:=Juan}
Juan
$ echo $nombre
Juan
$ echo ${nombre:+"Valor definido"}
Valor definido
$ unset nombre
$ echo ${nombre:?Variable ausente o no definida}
nombre: Variable ausente o no definida
$ nombre=Juan
$ echo ${nombre:?Variable ausente o no definida}
Juan
```

8. Variables de sistema

Además de las variables que el usuario puede definir por sí mismo, se inicia el shell con un cierto número de variables predefinidas útiles para ciertos comandos y accesibles por el usuario. Se puede modificar el contenido de estas variables de sistema, pero hay que ser cuidadoso, ya que algunas tienen una incidencia directa sobre el comportamiento del sistema.

Variable	Contenido
HOME	Ruta de acceso del directorio de usuario. Directorio por defecto en caso de uso de CD.

PATH	Lista de directorios, separados por:, donde el shell va a buscar los comandos externos y otros scripts y binarios. La búsqueda se hace en el orden de los directorios insertados.
PS1	Prompt String 1, cadena que representa el prompt estándar visualizado en la pantalla por el shell a la espera de inserción de comando.
PS2	Prompt String 2, cadena que representa un prompt secundario en caso de que se deba completar la inserción.
IFS	Internal Field Separator, lista de los caracteres que separan las palabras en una línea de comandos. Por defecto se trata del espacio, de la tabulación y del salto de línea.
MAIL	Ruta y archivo que contiene los mensajes del usuario.
SHELL	Ruta completa del shell en curso de ejecución.
LANG	Definición del idioma a utilizar, así como del juego de caracteres.
USER	Nombre del usuario en curso.
LOGNAME	Nombre del login utilizado durante el inicio de sesión.
HISTFILE	Nombre del archivo del historial, en general $HOME/.sh_history.
HISTSIZE	Tamaño en número de líneas del historial.
OLDPWD	Ruta de acceso del directorio al que se ha accedido anteriormente.
PS3	Define la línea de comandos para escribir un select.

PWD	Ruta actual.
RANDOM	Genera y contiene un número aleatorio entre 0 y 32767.

9. Variables especiales

Se trata de variables accesibles únicamente para lectura y cuyo contenido suele ser contextual.

Variable	Contenido
$?	Código de retorno del último comando ejecutado.
$$	PID del shell activo.
$!	PID del último proceso iniciado en segundo plano.
$-	Las opciones del shell.

```
$ echo $$
23496
$ grep memoria lista
$ echo $?
1
$ grep ratón lista
ratón  óptico 30     15
$ echo $?
0
```

```
$ ls -lR >pepito.txt 2>&1 &
26675
$ echo $!
26675
```

10. Longitud de una cadena

Es posible obtener la longitud de una cadena con el carácter #.

```
$ a=Julio
$ echo "Longitud de $a: ${#a}"
Longitud de Julio: 5
```

11. Tablas y campos

Linux cuenta con dos medios para declarar una tabla: la utilización de los corchetes **[ ]** y la creación global. Bash solamente soporta las tablas de una sola dimensión. El primer elemento es 0 y no hay una talla máxima, depende de la memoria. Para acceder al contenido de la tabla, hay que poner la variable Y el elemento entre llaves **{}**.

```
$ Nombre[0]="Julio"
$ Nombre[1]="Román"
$ Nombre[2]="Francisco"
$ echo ${Nombre[1]}
Román
```

o:

```
$ Nombre=(Julio Román Francisco)
$ echo ${nombre[2]}
```

```
Francisco
```

Para listar todos los elementos:

```
$ echo ${Nombre[*]}
Julio Román Francisco
```

Para conocer el número de elementos:

```
$ echo ${#Nombre[*]}
3
```

Si el índice es una variable, no se pone $ delante:

```
$ idx=0
$ echo ${Nombre[idx]}
Julio
```

12. Variables tipadas

Las variables pueden ser de tipo numérico entero (integer) con el comando **typeset -i**. La ventaja es que permite efectuar cálculos y comparaciones sin pasar por expr. El comando **let** o (**(...)**) permite cálculos sobre variables.

Operador	Función
`+ - * /`	Operaciones sencillas
`%`	Módulo
`< > <= >=`	Comparaciones, 1 si verdadero, 0 si falso
`== !=`	Igual o diferente
`&& ||`	Comparaciones relacionadas con un operador lógico
`& | ^`	Lógico binario AND OR XOR

```
$ typeset -i resultado
$ resultado=6*7
$ echo $resultado
42
$ resultado=Error
ksh: Error: bad number
$ resultado=resultado*3
126
$ typeset -i add
$ add=5
$ let resultado=add+5 resultado=resultado*add
$ echo $resultado
50
```