Shell Scripts en Linux

Variables

Para asignar el valor a una variable simplemente debemos usar el signo =

```
nombre_variable=valor_variable
```

Es importante no dejar espacios ni antes ni después del =

Para recuperar el valor de dicha variable sólo hay que anteponer el símbolo de dólar $ antes del nombre de la variable:

```
$nombre_variable
```

Las variables pueden tomar prácticamente cualquier nombre, sin embargo, existen algunas restricciones:

- Sólo puede contener caracteres alfanuméricos y guiones bajos
- El primer carácter debe ser una letra del alfabeto o "_" (este último caso se suele reservar para casos especiales).
- No pueden contener espacios.
- Las mayúsculas y las minúsculas importan, "a" es distinto de "A".
- Algunos nombres som usado como variables de entorno y no los debemos utilizar para evitar sobrescribirlas (HOME, PATH…)

Bucles (for)

La sintaxis general de los bucles es la siguiente:

```
for VARIABLE in LISTA_VALORES;
do
      COMANDO 1
      COMANDO 2
      ...
      COMANDO N
done
```

Donde la lista de valores puede ser un rango numérico:

```
      for VARIABLE in 1 2 3 4 5 6 7 8 9 10;
      for VARIABLE in {1..10};
```

una serie de valores:

```
      for VARIABLE in file1 file2 file3;
```

o el resultado de la ejecución de un comando:

```
      for VARIABLE in $(ls /bin | grep s.[aeiou]);
```

Ejemplo :

```
#!/bin/bash
for numero in {1..10..2};
do
  echo Estos números van de dos en dos: $numero
done
```

Condicionales (if)

La sintaxis básica de un condicional es la siguiente

```
if [ condición ]
then
     # instrucciones a ejecutar si la condición es cierta
fi
```

O si queremos que en caso de que la condición sea falsa ejecute otras instrucciones:

```
if [ condición ]
then
     # instrucciones a ejecutar si la condición es cierta
else
     # instrucciones a ejecutar en caso de que la condición sea
falsa
fi
```

Incluso se pueden añadir más condiciones concatenando más if:

```
if [ CONDICIÓN 1 ]
then
  COMANDO 1 si se cumple la condición 1
elif [ CONDICIÓN 2 ]
then
  COMANDO 2 si se cumple la condición 2
else
  COMANDO 3 si no se cumple la condición 2
Fi
```

Operadores condicionales entre números:

operador	significado
-lt	menor que (<)
-gt	mayor que (>)
-le	menor o igual que (<=)
-ge	mayor o igual que (>=)
-eq	igual (==)
-ne	no igual (!=)

Operadores condicionales entre cadenas de texto:

operador	significado
=	igual, las dos cadenas de texto son exactamente idénticas
!=	no igual, las cadenas de texto no son exactamente idénticas
<	es menor que (en orden alfabético ASCII)
>	es mayor que (en orden alfabético ASCII)
-n	la cadena no está vacía
-z	la cadena está vacía

Operadores condicionales con archivos

operador	Devuelve true si
-e fichero	fichero existe
-f fichero	fichero es un archivo normal (no es un directorio)
-s fichero	fichero NO tiene tamaño cero
-d fichero	fichero es un directorio
-r fichero	fichero tiene permiso de lectura para el user que corre el script
-w fichero	fichero tiene permiso de escritura para el user que corre el script
-x fichero	fichero tiene permiso de ejecución para el user que corre el script
Fichero1 -nt fichero2	Comprueba si fichero es más nuevo que fichero2.
fichero -ot fichero2	Comprueba si fichero es más viejo que fichero2.
-O fichero	Comprueba si el propietario del archivo y el user coinciden.
-G fichero	Comprueba si fichero y el user tienen el grupo idéntico.

```
#!/bin/bash
v1=zamora
v2=Zamora
if [ $v1 \> $v2 ]; then
      echo "$v1 es mayor que $v2"
else
      echo "$v1 es menor que $v2"
fi
```

Operadores AND, OR

Se pueden combinar múltiples pruebas utilizando los operadores AND (&&) u OR (||)

```
#!/bin/bash
dir=/home/pelaez
fichero=alpes.doc
if [ -d $dir ] && [ ! -s $fichero ]; then
      echo "El directorio existe y el fichero está vacío"
else
      echo " otra cosa"
fi
```