U4 Administración de Sistemas Operativos Linux

Parte III. Scripts

Implantación de Sistemas Operativos

Estructuras de repetición/bucles: for in

for variable **in** lista

do

instrucciones

done

- La variable toma en cada iteración uno de los valores que hay en la lista, de forma ordenada.
- Se puede emplear como lista la salida de una orden
- **¡OJO! La variable en el for va sin el $ porque sobre escribimos el valor en cada iteración, no queremos mostrarlo.**
- **La variable lista si que va con $ porque queremos leer el valor en cada iteración.**

- Ejemplo 1:

```
items="1 2 3 4 5"
for items in $items
do
    echo  $item
done
```

- Ejemplo 2a:

```
for file in `ls`
do
    echo $file
done
```

Estructuras de repetición/bucles: for in

Ejemplo 3:

```
for variable in " 1 2 3 4 Hola Adiós"
do
    echo valor: $variable
done
```

- Dada una variable recorre una lista.
- El separador por defecto será el espacio en blanco.
- Si ponemos la lista entre "" lo tomará como un único valor. No pondremos las comillas si queremos que tome cada variable por separado.
- Ejecutad el ejemplo quitando la comillas.

Estructuras de repetición/bucles: for in

- ¿Qué pasará con el ejemplo 2 si hay algún archivo con espacios en el nombre?

```
-rw-rw-r-- 1 marina marina    0 feb 28 19:03 'aaa bbb ccc'
-rwxrwxrwx 1 marina marina  361 feb 12 12:36  ejemplo_10.sh
-rwxrwxrwx 1 marina marina  569 feb 14 18:04  ejemplo_11.sh
```

- Ejemplo 2:

```
for file in `ls`
do
        echo $file
done
```

```
marina@marina-VirtualBox:~/scripts$ ./ej_FOR_IN_2.sh
aaa
bbb
ccc
ejemplo_10.sh
ejemplo_11.sh
```

- Cread archivo aaa bbb ccc y ejecutad script.
- ¿Cómo se podría evitar?

Estructuras de repetición/bucles: for in

- Podemos cambiar el separador por defecto (espacio) → **IFS="separador"**

  IFS es una variable de entorno.

  Salto de línea **IFS=$'\n'**

  Tabulador **IFS=$'\t'**

- Modificamos script para que el separador sea el salto de línea.
  - Ejemplo 2b

```
#!/bin/bash
IFS=$'\n'
for file in `ls`
do
        echo $file
done
```

Estructuras de repetición/bucles: for in

- ¿que pasa si lo hacemos con el comando ls –l y sin el salto de línea? (ej. 2c) ¿y si ponemos el salto de línea? (ej. 2d)
- Además de cambiar el separador por defecto, podemos eliminar los espacios de los nombres de los ficheros (ej. 4):

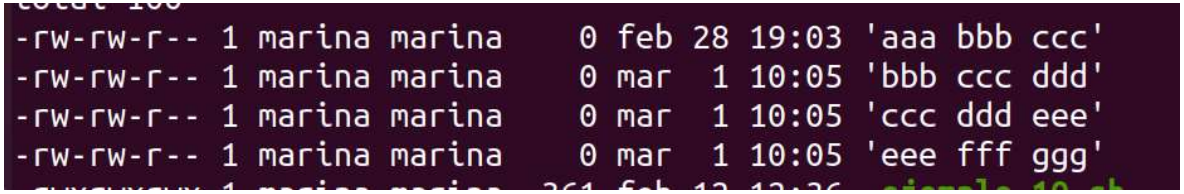
```
#!/bin/bash
IFS=$'\n'
for variable in `ls -l | tr -s ' ' | head -n 2`
do
        FicheroOriginal=`echo $variable | cut -d ' ' -f 9-`
        echo $FicheroOriginal
        NuevoFichero=`echo $FicheroOriginal | tr ' ' '_'`
        echo $NuevoFichero
done
```

En este caso lo hacemos solo para la línea 2, si queremos que recorra todo el ls –l, quitamos la parte de head –n 2.

Estructuras de repetición/bucles: for in

- Vemos que el comando se puede convertir en la lista que recorre la variable.

- ***-f 9-*** esta opción del cut significa que te quedes desde el campo 9 del fichero hasta el final.



-f 1- te quedas desde el campo 1 hasta el final.
-f -4 te quedas desde el principio hasta el campo 4.
-f 1-5 te quedas con los campos del 1 al 5.
-f 1,4,8 te quedas con los campos 1, 4 y 8.

Estructuras de repetición/bucles: for in

- En este caso hemos quitado los espacios del nombre, por lo que solo tengo el nombre del fichero, ahora tengo que traspasar el contenido:

```
#!/bin/bash
IFS=$'\n’
for variable in `ls -l | tr -s ' ' | head -n 2`
do
        FicheroOriginal=`echo $variable | cut -d ' ' -f 9-`
        echo $FicheroOriginal
        NuevoFichero=`echo $FicheroOriginal | tr ' ' '_'`
        echo $NuevoFichero
        mv $FicheroOriginal $Nuevo Fichero 2> /dev/null
done
```

Estructuras de repetición/bucles: for in

- También podemos pasarle argumentos y que éstos sean la lista del for.
- Si retomamos el ejemplo 1:

```
items="1 2 3 4 5"
for items in $items
do
    echo $item
done
```

- Y lo hacemos con argumentos (mas habitual) (ej. 5) pasamos argumentos 1 2 3 4 5

```
#!/bin/bash
for items in $*
do
        echo "ARGUMENTO: $items"
done
```

Estructuras de repetición/bucles: for in

- Creamos un fichero con nombre y apellidos y vamos a crear los usuarios para esos nombres (ej. 6)
- Cosas a tener en cuenta en función del formato del fichero:
  - Si tengo espacios, cambiar el separador por defecto al salto de línea, para quedarme con cada línea del fichero como variable.
  - Ver que campo me interesa del fichero y ver como lo tengo que tratar para poder extraer esa información.

```
#!/bin/bash
IFS=$'\n'
for items in `cat nombres`
do
        nombre=`echo $items | tr -s ' ' | cut -d ' ' -f 1`
#       useradd $nombre
        echo "$nombre añadido como usuario"
done
```

Estructuras de repetición/bucles: for in

- Otro forma de recorrer un fichero con **seq** (ej. 7)
  - **Seq 1 10** → Secuencia del 1 al 10
  - edad=20, **seq 1 $edad** → Secuencia del 1 al 20

```
#!/bin/bash
IFS=$'\n'
#calculo nº de líneas
lineas=`wc -l nombres | cut -d ' ' -f 1`
for variable in `seq 1 $líneas`
do
        linea=`head -n $variable  nombres | tail -n 1`
        nombre=`echo $linea | cut -d ' ' -f 1`
#       adduser $nombre
        echo "$nombre añadido como usuario"
done
```

Estructuras de repetición/bucles: for

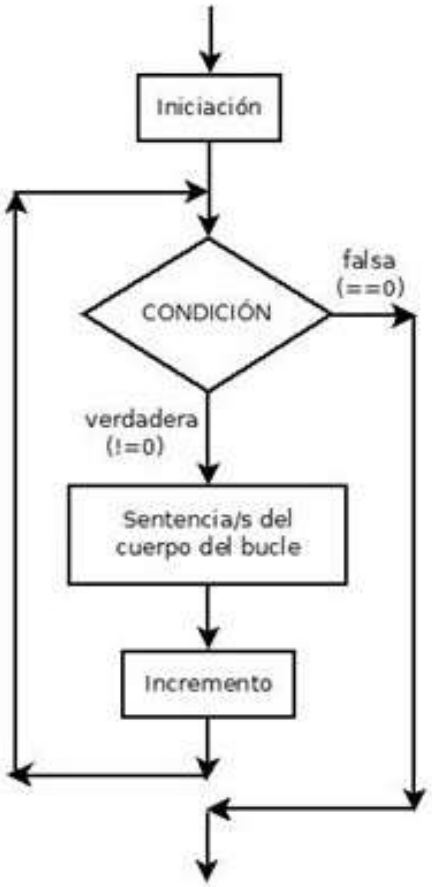
for ((inicialización; condición; incremento))

do

    instrucciones

done

- Las estructuras for son adecuadas siempre que sepamos el número de repeticiones necesarias en el bucle.
- Esta estructura da error con shell



Estructuras de repetición/bucles: for

- Ejemplo para recorrer un fichero (ej. 8)

```
#!/bin/bash
IFS=$'\n'
#calculo nº de lineas
lineas=`wc -l nombres | cut -d ' ' -f 1`
for ((x=1;x<=$lineas;x++))
do
        echo $x
        linea=`head -n$x nombres | tail -n 1`
        nombre=`echo $linea | cut -d ' ' -f 1`
#       adduser $nombre
        echo "$nombre añadido como usuario"
done
```

Estructuras de repetición/bucles: While

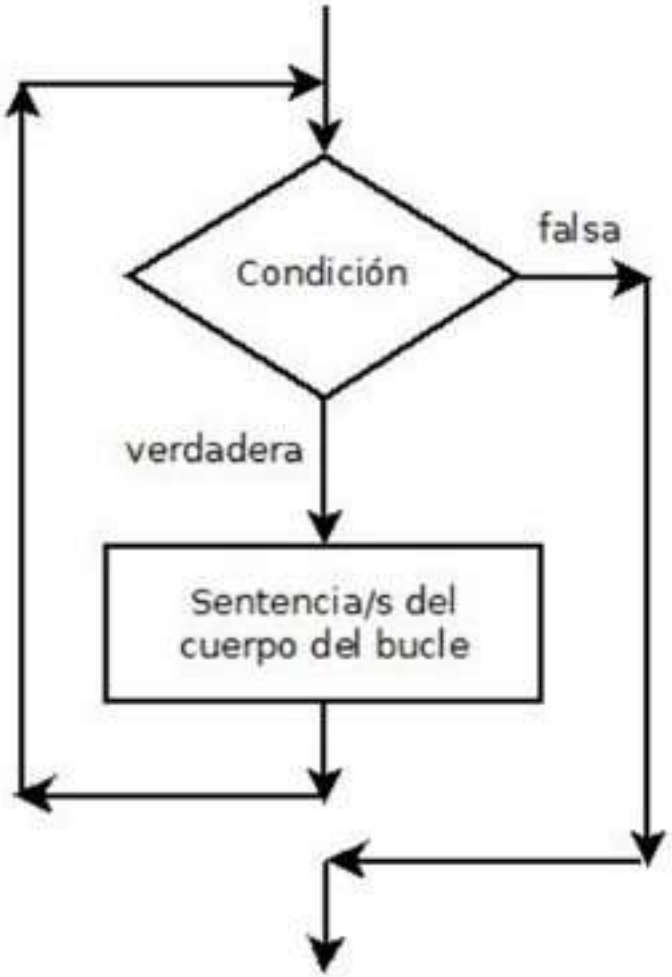
while [expresion]

do

instrucciones

done

- Las instrucciones se ejecutarán mientras se cumpla la expresión.
- Es necesario que la expresión se modifique dentro del bucle o crearemos un bucle infinito, si la condición se cumple siempre.



Estructuras de repetición/bucles: While

Ejemplo de bucle infinito:

```
#!/bin/bash
x=1
while [ $x -lt 10 ]
do
        echo "comienza la iteración"
        if [ $x -eq 4 ]
        then
                #let x=$x+1
                 echo “ahora x=4”
                continue
        fi
        echo "voy a incrementar x"
        let x=$x+1
done
```

¿Que tengo que hacer para que deje de ser un bucle infinito? (ej. 9)

Estructuras de repetición/bucles: While

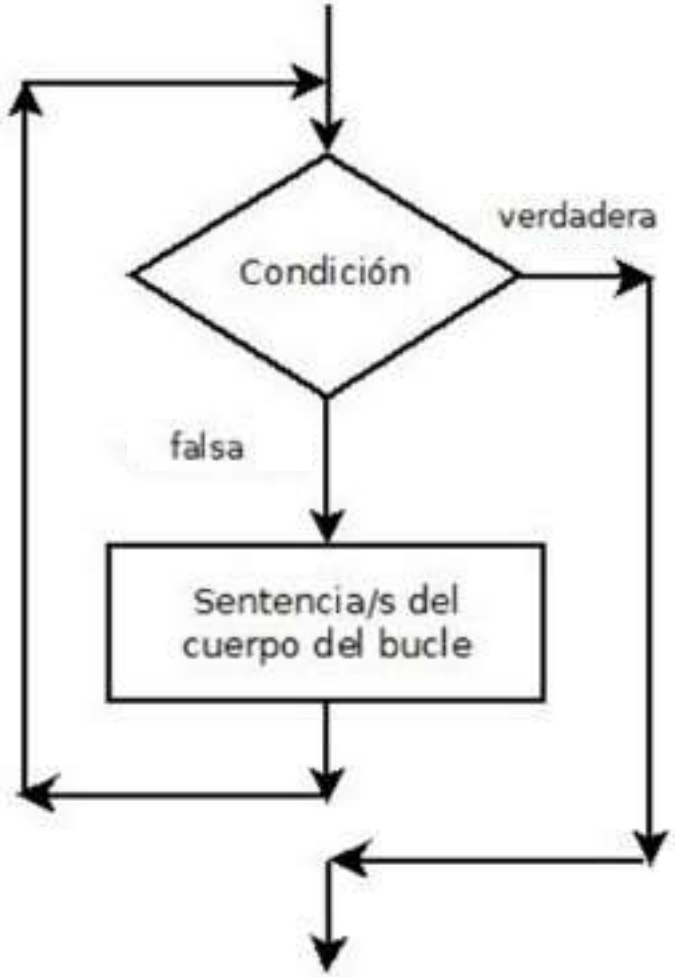
Mismo ejemplo con FOR (ej. 10):

```
#!/bin/bash
for ((x=1;x<10;x++))
do
        echo "comienza la iteracion"
        if [ $x -eq 4 ]
        then
                echo "ahora x=4"
                continue
        fi
        echo "voy a incrementar x"
done
```

Mismo ejemplo con FOR (ej. 11):

```
#!/bin/bash
for ((x=1;x<10;x++))
do
        echo "comienza la iteracion"
        if [ $x -eq 4 ]
        then
                echo "ahora x=4"
                break
        fi
        echo "voy a incrementar x"
done
```

Until



until [expresion]

do

instrucciones

done

- El funcionamiento de until es el opuesto al de while.
- Until ejecuta las instrucciones mientras la expresión es falsa y hasta que la expresión sea verdadera. Cuando sea verdadera parará.

Break y continue

- Podemos emplear break y continue para parar los bucles si es necesario:
  - **break** fuerza la salida del bucle (con break n indicamos de cuántos bucles queremos salir)
  - **continue** permite saltar las instrucciones siguientes a este comando hasta la siguiente iteración
- El uso de break y continue no se considera una buena práctica de programación.
- Estas instrucciones sólo se deben emplear de manera excepcional porque dificultan el seguimiento del flujo del programa y en la mayor parte de casos, se pueden suplir con un diseño adecuado del algoritmo solución.
- **exit** sales del programa, estés donde estés.
- **sleep** detiene el programa durante x segundos.