U4 Administración de Sistemas Operativos Linux

Parte II. Scripts

Implantación de Sistemas Operativos

Paso de argumentos a un script

- Un argumento es un parámetros que se le pasa a una función o a un programa. Es decir, son variables de entrada. Puede haber más de uno o ninguno.

- Para pasar argumentos a un script indicamos los valores, a continuación del script, de la siguiente manera:

./miscript.sh arg1 arg2 arg3 ... argn

- Se puede acceder al contenido de cada uno de los argumentos usando $X donde “X” es el número del argumento:

Paso de argumentos a un script

Ejemplo:

./soyUnScript.sh 1 hola 2 adios esto es un argumento
$1: Argumento nº1 → 1
$2: Argumento nº2 → hola
$3: Argumento nº3 → 2
$4: Argumento nº4 → adios
…

$0: Es el nombre del script → soyUnScript.sh
$*: Una lista con todos los argumentos → 1 hola 2 adios esto es un argumento
$#: Número de argumentos → 8

- Los argumentos van separados por espacios y en caso de insertar un metacaracter, como *, es necesario escaparlo → ./soyUnScript.sh *

Paso de argumentos a un script

- Si el argumento va entre comillas “” lo mete todo en una variable, si no cada cadena de caracteres separados por un espacio, irá a una variable distinta.
  - Ejemplo: “hola que tal” → 1 variable
    hola que tal → 3 variables
    1 2 3 “hola que tal” bien gracias → 6 variables

- Para hacer referencia a los argumentos dentro del script empleamos:
  - **$0**: nombre del script
  - **$1 a $9**: argumentos del 1 al 9
  - **${n}**: argumentos a partir del 10. Ej: ${10}
  - **$#: número de argumentos**
  - **$* o $@ todos los argumentos**
  - **$$** es el número de proceso actual (PID)
  - **$?** devuelve el código de salida del último comando ejecutado. Ejemplo:

    echo $((3/0)); $?

Paso de argumentos a un script

Ejemplos:

Tu script necesariamente necesita un argumento para ejecutarse

```
if [ $# -eq 1 ]
then
        echo “hago mi script”
        echo “ejecutando el script”
else
        echo “error, debes proporcionar un argumento”
fi
```

```
if [ $# -ne 1 ]
then
        echo “error, debes proporcionar un argumento”
else
        echo “ejecuto mi script”
fi
```

Paso de argumentos a un script

Ejemplo if_3.sh

```
#!/bin/bash

if [[ "$1" = "-h" || "$1" = "--help" ]]
then
        echo "debes pasar un argumento que sea un directorio"
elif [ $# -eq 1 ]
then
        echo "correcto"
else
        echo "error, es necesario un argumento"
fi
```

Paso de argumentos a un script

Ejemplo if_2.sh

```
#!/bin/bash

if [[ "$1" = "-h" || "$1" = "--help" ]]
then
        echo "debes pasar un argumento que sea un fichero"
elif [[ $# -eq 1 && -f $1 ]]
then
        cat $1
else
        echo "error, es necesario un argumento fichero"
fi
```

Paso de parámetros a un script

Valores por defecto

La expresión ${variable:-valor_por_defecto} permite establecer valores por defecto en caso de que no se produzca entrada. Si no me pasan un argumento, entonces pondré yo el valor que quiera a $1.

Ejemplo: valor.sh

```
#!/bin/bash
num=$((${1:-100}))
echo $num
```

Ejemplo: script suma.sh

```
#! /bin/bash
SUMA=$((${1:-0}+${2:-0}+${3:-0}))
echo $SUMA
```

Ejecución:

```
bash ./suma.sh 1 2 3
bash ./suma 1 2
```

La shell

Ejecución condicional de órdenes

- Podemos hacer uso de los operadores AND (&&) y OR (||) para enlazar la ejecución de 2 o más órdenes de la siguiente manera:

- Orden1 && Orden2: sólo ejecutará la orden 2 si la primera tiene éxito (es decir, la salida es 0). (si una orden falla ya no miramos el resto)

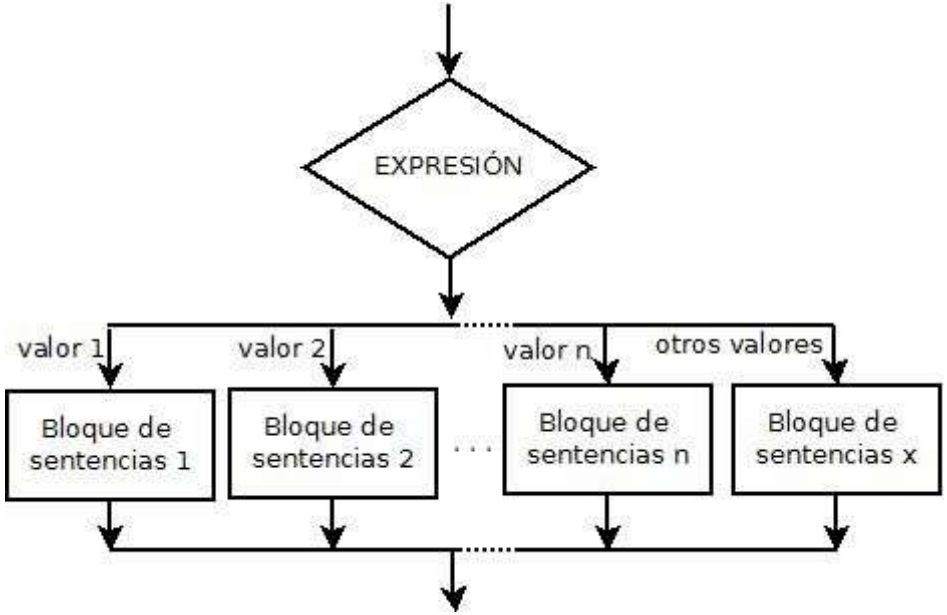
ls fichero && rm fichero

- Orden1 || Orden2: ejecutará la orden 2 si la 1 falla. (en cuanto una de las ordenes se ejecute con éxito, ya no se miran las demás)

ls fichero || touch fichero

Estructuras de control condicional: Case

- La instrucción **case** equivale a una estructura de if/else anidados. Comprobamos el valor de la condición y en función de ese valor hacemos una cosa u otra.
- Se utiliza sobre todo para hacer menús.



Estructuras de control condicional: Case

```
case $var in
    val1)
        instrucciones;;
    val2)
        instrucciones;;
    ...
    valn)
        instrucciones;;
    *)
        instrucciones;;
esac
```

- La última instrucción de cada opción se termina con ;; para indicar que ha terminado.
- Si no se pulsa ninguna opción correcta, entonces se ejecuta el *.
- El * equivale al “else” del “if”.
- Si las opciones que ponemos son letras, las tenemos que poner entre comillas “A)”

Estructuras de control condicional: Case

Ejemplo: case_1.sh

```
#!/bin/bash
echo “1. Eliminar fichero”
echo “2. Copiar fichero”
echo “3. Exit”
read –p “selecciona una opción” opc

case $opc in
        1)
                echo “fichero eliminado”;;
        2)
                echo “fichero copiado”;;
        3)
                echo “adiós”;;
        *)
                echo “esa opción no es correcta, adiós”;;
esac
```

Estructuras de control condicional: Case

- Podemos ejecutar varias órdenes en cada opción. Los ;; solo se pondrán al final de la última instrucción de esa opción.
- También podemos hacer uso de | para integrar dos opciones posibles, tanto si marcas una como otra se ejecutarán las mismas instrucciones

Ejemplo: case_2.sh

```
#!/bin/bash
read –p “escribe un nombre” nombre

case $nombre in
        alvaro | pepe)
                echo “tu nombre es $nombre ”
                echo “Bienvenido”;;
        *)
                echo “no te llamas ni alvaro ni pepe”
                echo “adios”;;
esac
```