

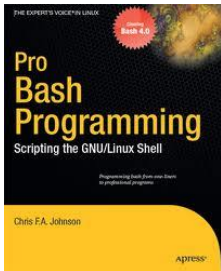
# Sistemas Operativos

sesión 13: *shell-scripting*

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

# Bibliografía



- Bash Programming:  
<http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>



- Advanced Bash-Scripting Guide:  
<http://www.tldp.org/LDP/abs/html/>

# Contenidos



- Ciclo de trabajo con *scripts*
- Shell-scripts como lenguaje de programación:
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - Expresiones
    - Redirecciones y tuberías
  - Sentencias de control
    - Alternativas e iterativas
  - Funciones
    - Uso de parámetros
- Ejemplos de *shell-scripts*

# Contenidos



- **Ciclo de trabajo con *scripts***
- Shell-scripts como lenguaje de programación:
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - Expresiones
    - Redirecciones y tuberías
  - Sentencias de control
    - Alternativas e iterativas
  - Funciones
    - Uso de parámetros
- Ejemplos de *shell-scripts*

# Ciclo de trabajo típico

1. Edición del *shell-script*

– `vi script.sh`

2. Cambio a ejecutable (si se necesita)

– `chmod a+x script.sh`

3. Ejecución del *shell-script*

– `./script.sh`



# Ciclo de trabajo típico

## 1. Edición del *shell-script*

– `vi hola.sh`

```
#!/bin/bash
```

```
echo "Hola mundo..."
```

## 2. Cambio a ejecutable (si se necesita)

– `chmod a+x hola.sh`

## 3. Ejecución del *shell-script*

– `./hola.sh`

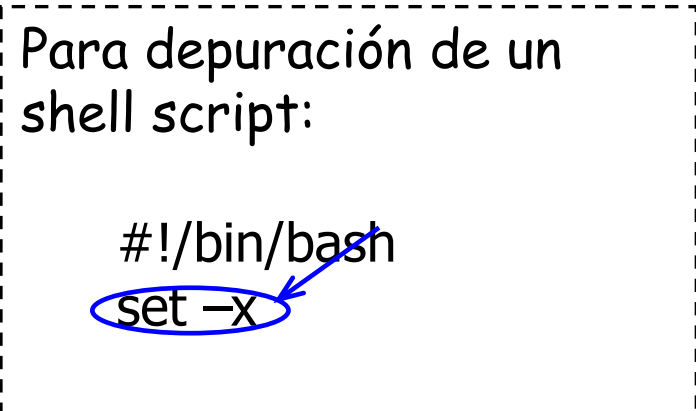
# Ciclo de trabajo típico

```
#!/bin/bash
set -x

# impresión de mensaje
echo "Hola mundo..."
```

Para depuración de un shell script:

```
#!/bin/bash
set -x
```



Comentarios:

```
# esto es un comentario
```



# Contenidos



- Ciclo de trabajo con *scripts*
- **Shell-scripts como lenguaje de programación:**
  - **Variable**
    - **Lectura y escritura de valores**
    - **Uso de comillas**
  - Expresiones
    - Redirecciones y tuberías
  - Sentencias de control
    - Alternativas e iterativas
  - Funciones
    - Uso de parámetros
- Ejemplos de *shell-scripts*



# Uso de variables

var.sh

```
#!/bin/bash
set -x

CHAR=a
STR="Hola mundo"
NUM=123

echo $CHAR
echo $STR
echo $NUM
```

- Es posible el uso de variables, como en otros lenguajes de programación
- Una variable en bash puede contener:
  - Número
  - Carácter
  - Tira de caracteres (string)
- No es necesario declarar una variable, solo asignar un valor a su referencia la crea.
- Al usarla es necesario anteponer el símbolo \$

# Lectura de valores por la entrada estándar

- Lectura de un valor:

```
#!/bin/bash  
echo "Por favor, indique su nombre"  
read NOMBRE  
echo "¡Hola $NOMBRE!"
```

- Lectura de varios valores a la vez:

```
#!/bin/bash  
echo "Por favor, indique su nombre"  
read NOMBRE APELLIDOS  
echo "¡Hola! $APELLIDOS, $NOMBRE !"
```

# Uso de variables

var3.sh

```
#!/bin/bash
```

```
set -x
```

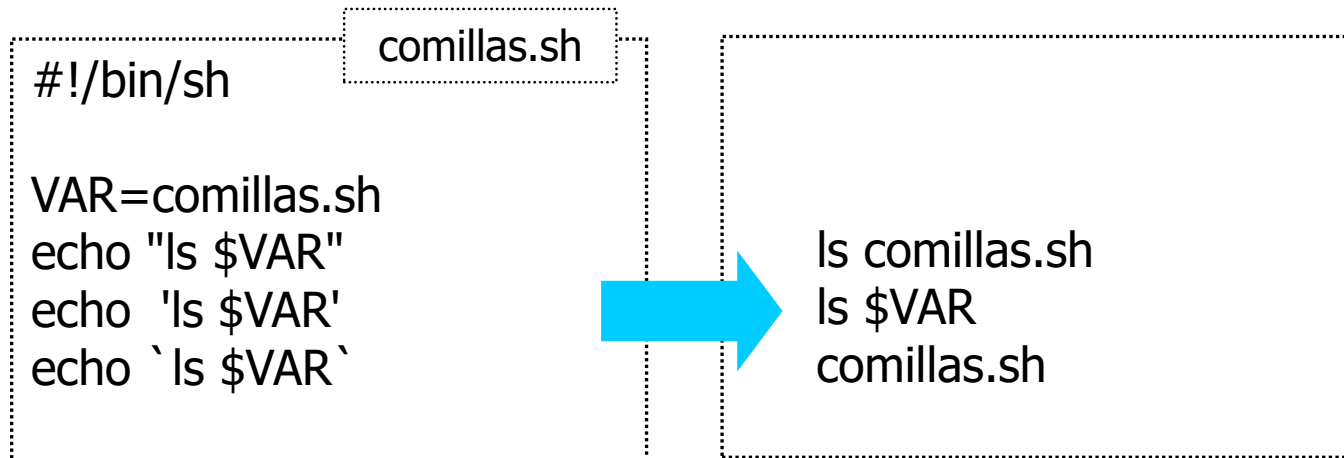
```
LISTADO=$(ls -l)
```

```
echo $LISTADO
```

- Es posible asignar la salida de la ejecución de un mandato.
  - Para ello se precisa la notación \$(mandato)
- A la hora de visualización los retornos de carro desaparecen.

# Uso de las comillas

- Tipo de comillas: “, ‘, `
- Diferentes usos de las comillas:



# Uso de las comillas

- **Ejercicio:**

- ¿Cuál es la salida del siguiente script?

```
comillas2.sh
#!/bin/sh

VAR="comillas.sh    comillas2.sh"
echo  ls $VAR
echo  "ls $VAR"
echo  'ls $VAR'
echo  `ls $VAR`
```



# Contenidos



- Ciclo de trabajo con *scripts*
- **Shell-scripts como lenguaje de programación:**
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - **Expresiones**
    - **Redirecciones y tuberías**
  - Sentencias de control
    - Alternativas e iterativas
  - Funciones
    - Uso de parámetros
- Ejemplos de *shell-scripts*

# Redirecciones

- Redirecciones de salida:
  - Salida estándar  
`ls -l > ls-l.txt`
  - Salida de errores  
`grep noesta * 2> grep-errores.txt`
  - Salida estándar y error juntas  
`grep noesta * 2>& grep-errores.txt`

# Tuberías (pipes)

- Tuberías:
  - Las tuberías permiten que la salida de un programa sea la entrada a otro (definido de forma simple).

– Ej)

```
ls -l | sed -e "s/[aeiou]/./g"
```

Permite cambiar todas las vocales por punto.

El carácter '.' es un metacaracter -> equivale a un carácter  
El carácter '\*' equivale a cero o más caracteres.



# Expresiones

expr.sh

```
#!/bin/bash
#set -x

wc -l /etc/fstab > /tmp/f1
L1=$(cut -f1 -d' ' /tmp/f1)
rm -fr /tmp/f1

wc -l /etc/fstab > /tmp/f2
L2=$(cut -f1 -d' ' /tmp/f2)
rm -fr /tmp/f2

SUMA=$(expr $L1 + $L2)

echo "el total líneas de"
echo -n "los ficheros es:"
echo " "$SUMA
```

- Como otros lenguajes es posible realizar operaciones aritméticas.
  - Ha de usarse expr indicando los operandos y el operador, separados todos por espacios en blanco.
  - El mandato expr (man expr) dispone de diferentes operadores: +, -, \\*, /, %, etc.

# Contenidos



- Ciclo de trabajo con *scripts*
- **Shell-scripts como lenguaje de programación:**
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - Expresiones
    - Redirecciones y tuberías
  - **Sentencias de control**
    - **Alternativas e iterativas**
  - Funciones
    - Uso de parámetros
- Ejemplos de *shell-scripts*

# Alternativas

- If...then...else:
  - El bloque del *else* es opcional.

```
#!/bin/bash

T1="igual"
T2="identico"

if [ "$T1" = "$T2" ]; then
    echo La expresión es evaluada verdadera
else
    echo La expresión es evaluada falsa
fi
```

# Alternativas

- If...then...else: ([man bash, expresiones condicionales](#))
  - Para string: =, !=
  - Para números: -eq, -ne, -lt, -le, -gt, -ge, etc.
  - Para ficheros: -f, -d, -s, -r, -w, -e, etc.

```
#!/bin/bash

T1="/tmp"
T2="/etc/fstab"

if [ -d $T1 ]; then
    echo "Es un directorio"
fi

if [ -a $T2 ]; then
    echo "Es un fichero y existe"
fi
```

# Uso de parámetros

- \$1, \$2, ... representan el primer, segundo, etc. parámetro
- \$# representa el número de parámetros
- Con el mandato `shift` se desplazan una posición
  - el \$2 pasa a ser el \$1, el \$3 pasa a \$2, y así sucesivamente

```
#!/bin/bash

if [ -z "$1" ]; then
    echo "uso: $0 <directorio>"
    exit
fi

if [ $# -ge 2 ]; then
    echo "demasiados parámetros."
    exit
fi
```

# Alternativas

- Case...in..esac:
  - Múltiples comprobaciones con case:

```
#!/bin/bash

variable=abc
case "$variable" in
    abc) echo "\$variable = abc" ;;
    xyz) echo "\$variable = xyz" ;;
esac
```

# Iteraciones

- for...do...done:
  - Bucles de iteración sobre un conjunto de palabras:

```
#!/bin/bash

LISTA=$(ls)

for I in $LISTA; do
    echo item: $I
done
```

# Iteraciones

- while...do...done:
  - Bucles de iteración sobre una condición tipo [0-n]:

```
#!/bin/bash

COUNTER=0
while [ $COUNTER -lt 10 ]; do
    echo El contador vale $COUNTER
    let COUNTER=COUNTER+1
done
```



# Iteraciones

- until...do...done:
  - Bucles de iteración sobre una condición tipo [1-n]:

```
#!/bin/bash

COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo El contador vale $COUNTER
    let COUNTER-=1
done
```

# Contenidos



- Ciclo de trabajo con *scripts*
- **Shell-scripts como lenguaje de programación:**
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - Expresiones
    - Redirecciones y tuberías
  - Sentencias de control
    - Alternativas e iterativas
  - **Funciones**
    - **Uso de parámetros**
- Ejemplos de *shell-scripts*


# Uso de funciones

- Declaración estilo *miFunción { miCódigo }*
- Uso con *miFunción*

```
#!/bin/bash

function quit
{
    exit
}

function hello
{
    echo "¡Hola!"
}

hello
quit
echo "¿llegará aquí?" 
```

# Parámetros a funciones

- Parámetros posicionales idénticos a los del script:

```
#!/bin/bash

function e
{
    echo $1 $2
}

e Hola Mundo
e "Hola mundo" "..."
```

# Variables locales

- Es posible el uso de variables locales, para ello se utiliza la palabra clave *local*:

```
#!/bin/bash
SALUDO=Hola
function hello
{
    local SALUDO=Mundo
    echo $SALUDO
}

echo $SALUDO
hello
echo $SALUDO
```

# Contenidos



- Ciclo de trabajo con *scripts*
- Shell-scripts como lenguaje de programación:
  - Variable
    - Lectura y escritura de valores
    - Uso de comillas
  - Expresiones
    - Redirecciones y tuberías
  - Sentencias de control
    - Alternativas e iterativas
  - Funciones
    - Uso de parámetros
- **Ejemplos de *shell-scripts***



# Ejemplo 1

- Escribir en un editor de texto el siguiente programa, y ejecutarlo en una shell:

```
./ejemplo1.sh
```

- El programa escribe los números del 3 al 1 y espera tantos segundos en escribir el siguiente como valor tenga el número escrito.



# Ejemplo 1

```
#!/bin/bash
#Esta es la primera línea del script

VALOR=3
while [ 0 -lt $VALOR ];
do
    echo "$VALOR"
    sleep $VALOR
    VALOR=`expr $VALOR - 1`
done
```





## Ejemplo 2

- Escribir en un editor de texto el siguiente programa, y ejecutarlo en una shell con:

```
./ejemplo2.sh 1 2 3 4 5
```

- y con:

```
./ejemplo2.sh 1 2 3 4 5
```



## Ejemplo 2

```
#!/bin/bash

if [ $# -ne 5 ];
then
    echo "Error, número de parámetros introducidos incorrecto"
    exit
fi

echo "Se ha introducido $* "
LISTADO="$*"

for I in $LISTADO;
do
    echo "parámetro - $I"
done
```



## Ejemplo 3

- Escribir en un editor de texto el siguiente programa, y ejecutarlo en una shell:

```
./ejemplo3.sh
```



# Ejemplo 3

```
#!/bin/bash
```

```
LS=`ls `
```

```
I=4
```

```
echo Lista de ficheros a procesar
```

```
echo "$LS"
```

```
echo
```

```
for J in $LS;
```

```
do
```

```
    echo $J
```

```
    head -n $I $J| tail -n 1
```

```
done
```

# Sistemas Operativos

sesión 13: *shell-scripting*

Grado en Ingeniería Informática

Universidad Carlos III de Madrid