



1º DAM/DAW Sistemas Informáticos

U6 - Shell Scripts

1 - Shell Scripts



¿Qué es un Script?

- En informática, un **script** es un término un tanto informal que se usa para designar a un **programa relativamente corto y/o sencillo**.
- Su traducción literal del inglés es guion.

Shell Scripting



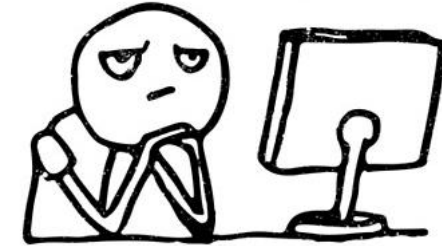
¿Qué es un Script?

- Los sistemas operativos son capaces de ejecutar directamente ciertos tipos de ficheros, que contienen secuencias de instrucciones o comandos, en modo texto, llamados scripts.
- Por ejemplo:
 - Windows: puede ejecutar ficheros con extensión “.bat”
 - **Linux: puede ejecutar ficheros con extensión “.sh”**
- Este tipo de fichero con scripts, también se conoce como **secuencia de comandos, fichero por lotes, shell script, ...**

¿Qué es un Script?

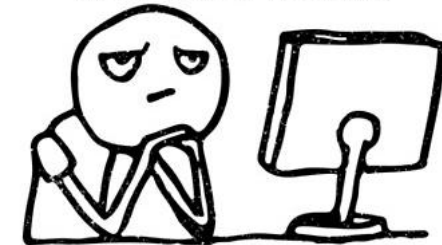
- Los scripts no son código compilado mediante una traducción completa y anticipada a código máquina, sino que **son ejecutados instrucción a instrucción por un intérprete que lee el archivo de código fuente en el momento.**
- Comúnmente, se denomina **Scripting** a la **acción de desarrollar scripts.**

THE CODE DOESN'T WORK



WHY?

THE CODE WORKS



WHY?

¿Qué es un Script?

- Con estos scripts, ejecutados a nivel de sistema operativo podemos realizar multitud de operaciones. Puesto que contamos, por ejemplo, con las siguientes **herramientas y capacidades**:
 - Facilita la **entrada de parámetros** y la **salida de resultados**.
 - Dispone de **variables, estructuras de control y todo tipo de operaciones lógicas y aritméticas**, para poder procesar información.

¿Qué es un Script?

- Permite **ejecutar de comandos propios de la línea de comandos** de forma integrada.
- Permite la **manipulación de archivos**, tanto para la lectura de información, como para la creación y actualización de archivos y estructuras de datos.
- Posibilita la **ejecución de otros scripts** mediante llamadas.
- ...

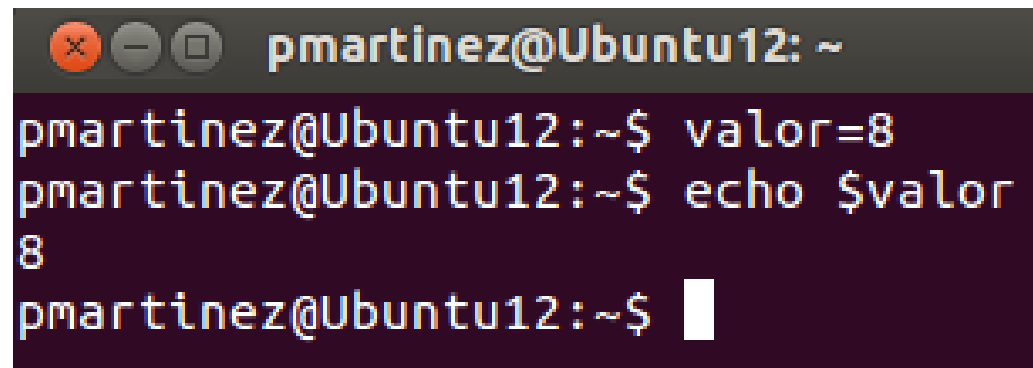
Shell Scripts

- Denominaremos **Shell Scripts** a los **scripts** que utilizamos a través de la línea de **comandos de Linux**.
- En este contexto, se puede entender **Shell Scripting** como un lenguaje de programación, puesto que va a disponer de conceptos y utilidades similares a las que disponemos con otros tipos de lenguajes de programación.
- **Utilizaremos ficheros con extensión “*.sh”**



Shell Scripts - Variables

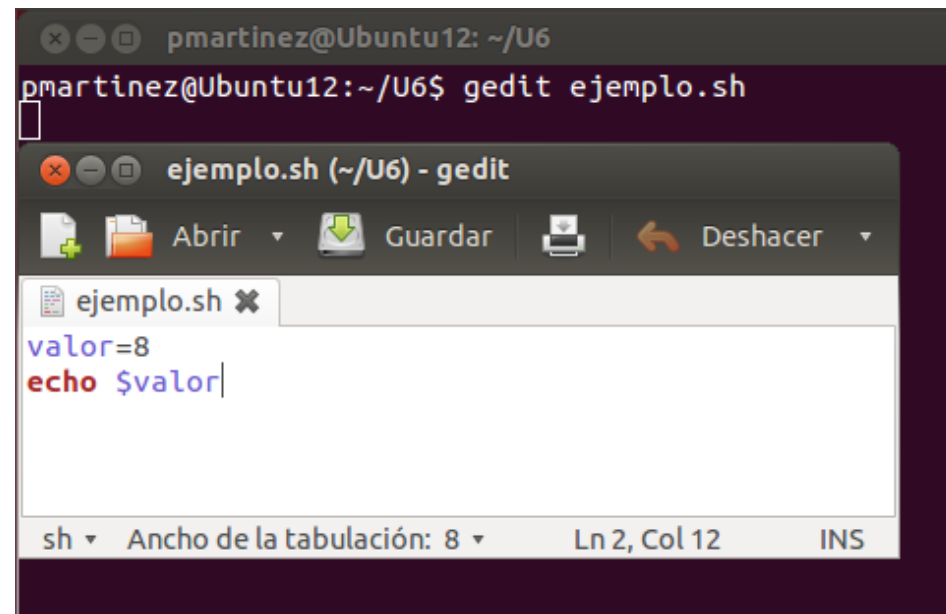
- Una **variable**, igual que en cualquier otro lenguaje de programación, hace referencia a un **nombre asociado a un contenido en memoria**.
- A diferencia de otros lenguajes, cuando trabajamos con shell scripts **no necesitamos asociar un tipo a las variables**.
- Podemos usar variables directamente a nivel de línea de comandos:



```
pmartinez@Ubuntu12: ~  
pmartinez@Ubuntu12:~$ valor=8  
pmartinez@Ubuntu12:~$ echo $valor  
8  
pmartinez@Ubuntu12:~$
```


Shell Scripts - Variables

- Si creamos un fichero, “ejemplo.sh” e indicamos como instrucciones los mismos comandos ejecutados desde el ejemplo anterior:



The image shows a terminal window and a gedit editor window. The terminal window has a title bar "pmartinez@Ubuntu12: ~/U6" and shows the command "pmartinez@Ubuntu12:~/U6\$ gedit ejemplo.sh" being executed. The gedit editor window has a title bar "ejemplo.sh (~/U6) - gedit" and a menu bar with "Abrir", "Guardar", and "Deshacer". The editor shows a file named "ejemplo.sh" with the following content:

```
valor=8  
echo $valor
```

The status bar at the bottom of the gedit window shows "sh", "Ancho de la tabulación: 8", "Ln 2, Col 12", and "INS".

Shell Scripts - Variables

- Podemos ejecutar el Shell script directamente desde la línea de comandos:

```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
bash: ./ejemplo.sh: Permiso denegado
pmartinez@Ubuntu12:~/U6$ ls -l
total 4
-rw-rw-r-- 1 pmartinez pmartinez 20 may  6 12:13 ejemplo.sh
```

****Cuidado**, necesitamos **permiso de ejecución** sobre el fichero:

```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ chmod u+x ejemplo.sh
pmartinez@Ubuntu12:~/U6$ ls -l
total 4
-rwxrw-r-- 1 pmartinez pmartinez 20 may  6 12:13 ejemplo.sh
```

Shell Scripts - Variables

- Una vez disponemos de permiso de ejecución, es suficiente con **escribir la ruta al fichero para ejecutarlo**:

```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ejemplo.sh
ejemplo.sh: no se encontró la orden
pmartinez@Ubuntu12:~/U6$
```

****Cuidado**, al indicar la ruta relativa, si el fichero está en la misma carpeta donde estamos ubicados, no hay que olvidar indicar “./ejemplo.sh”...

```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
8
pmartinez@Ubuntu12:~/U6$
```

Shell scripts - Variables

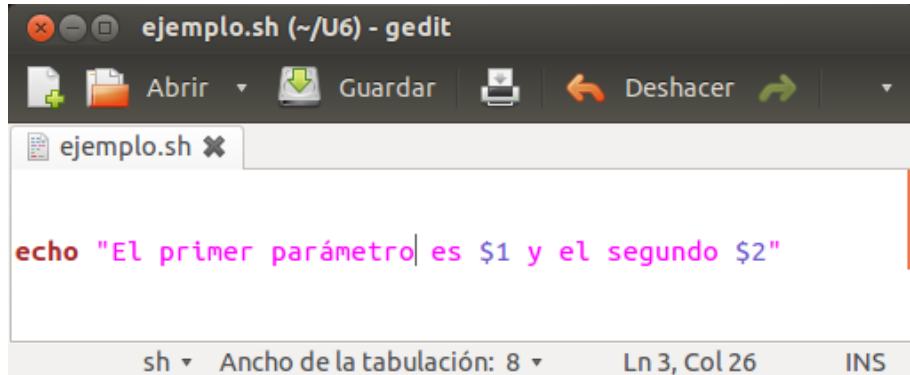
- Ahora sí, podemos confirmar que la ejecución del script se comporta igual que la secuencia de comandos introducida directamente mediante la línea de comandos:

```
pmartinez@Ubuntu12: ~  
pmartinez@Ubuntu12:~$ valor=8  
pmartinez@Ubuntu12:~$ echo $valor  
8  
pmartinez@Ubuntu12:~$
```

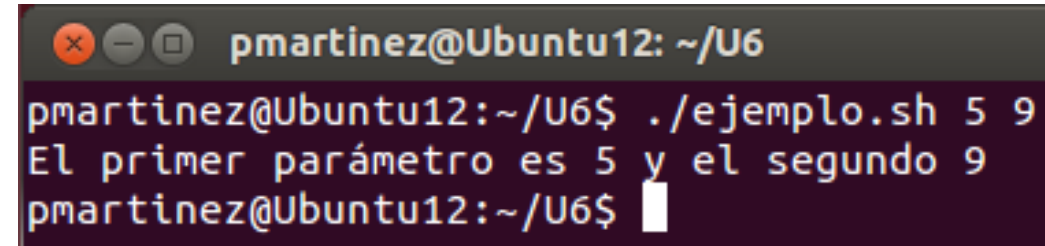
```
pmartinez@Ubuntu12: ~/U6  
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh  
8  
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Entrada/Salida

- Podemos **pasarle valores como parámetros a un Shell Script mediante teclado** en el momento de realizar la **llamada desde la línea de comandos**:
- En la imagen podemos ver como modificando el script de ejemplo, accedemos a los parámetros de la llamada (**\$1, \$2, ..., \$n**).



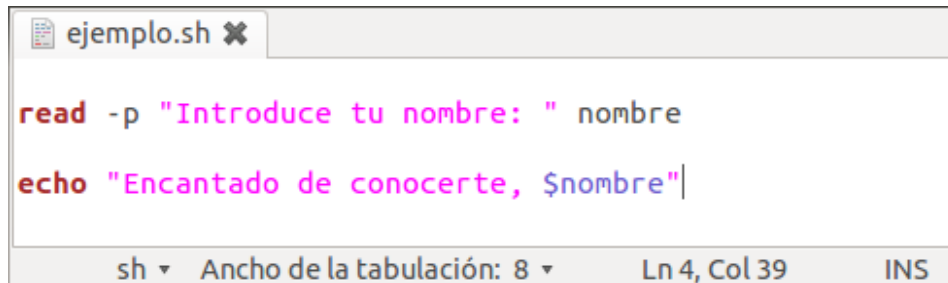
```
ejemplo.sh (~/.U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh ✕
echo "El primer parámetro es $1 y el segundo $2"
sh Ancho de la tabulación: 8 Ln 3, Col 26 INS
```



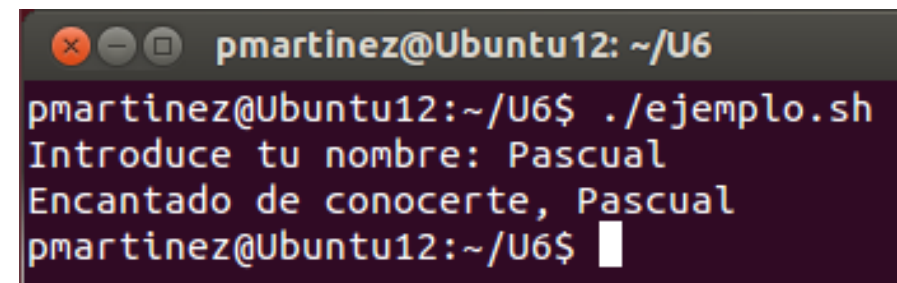
```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh 5 9
El primer parámetro es 5 y el segundo 9
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Entrada/Salida

- También podemos **pasarle valores como parámetros a un Shell Script mediante teclado durante la ejecución del script.**
- Desde el script, podremos recuperar estos valores mediante el **comando read** y el **parámetro -p.**



```
ejemplo.sh ✕  
  
read -p "Introduce tu nombre: " nombre  
echo "Encantado de conocerte, $nombre"  
  
sh ▾ Ancho de la tabulación: 8 ▾ Ln 4, Col 39 INS
```



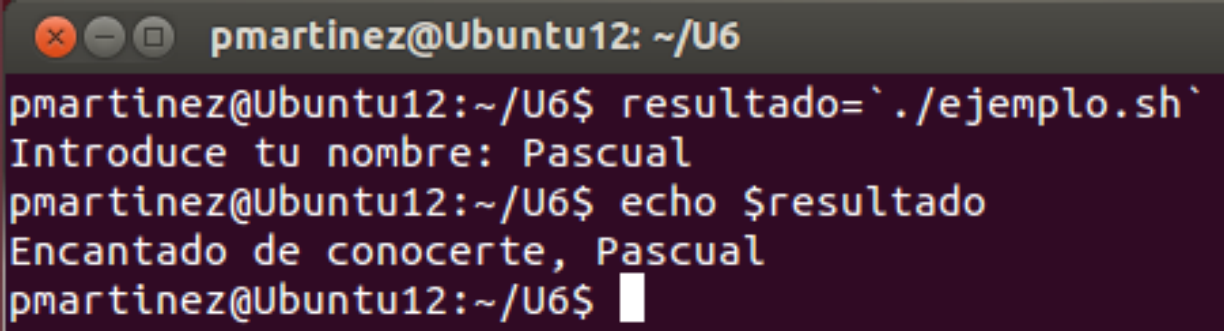
```
pmartinez@Ubuntu12: ~/U6  
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh  
Introduce tu nombre: Pascual  
Encantado de conocerte, Pascual  
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Entrada/Salida

- En los ejemplos que hemos visto, los scripts han expresado su salida mediante el comando `echo`, implicando en este caso que se muestre el resultado por pantalla, como salida estándar.
- También **es posible hacer que el resultado de un script, o de un comando, se guarde en una variable**, mediante una de estas sintaxis:

`variable=`./script.sh``

`variable=$(./script.sh)`



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ resultado=`./ejemplo.sh`
Introduce tu nombre: Pascual
pmartinez@Ubuntu12:~/U6$ echo $resultado
Encantado de conocerte, Pascual
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Operaciones aritméticas

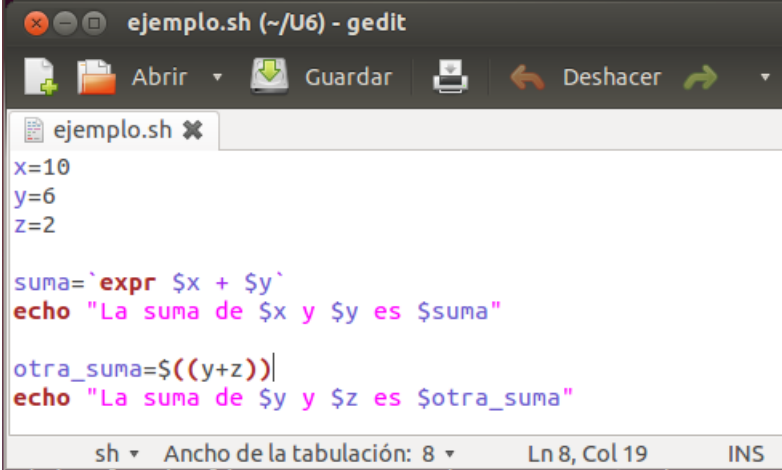
- Para realizar operaciones aritméticas podemos utilizar una de las siguientes dos sintaxis:

- expr** argumento1 **operador** argumento2

→ Nota: el operador de la multiplicación es *

variable=`expr arg1 **operador** arg2`

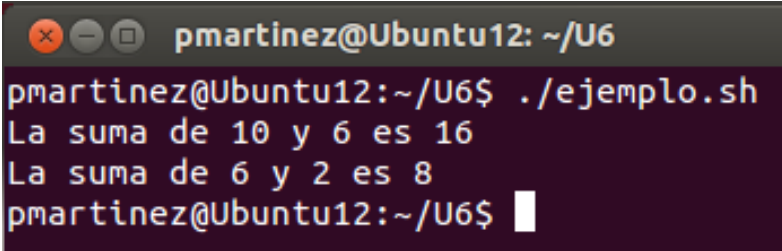
- \$((argumento1 operador argumento2))**



```
ejemplo.sh (~/.U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh
x=10
y=6
z=2

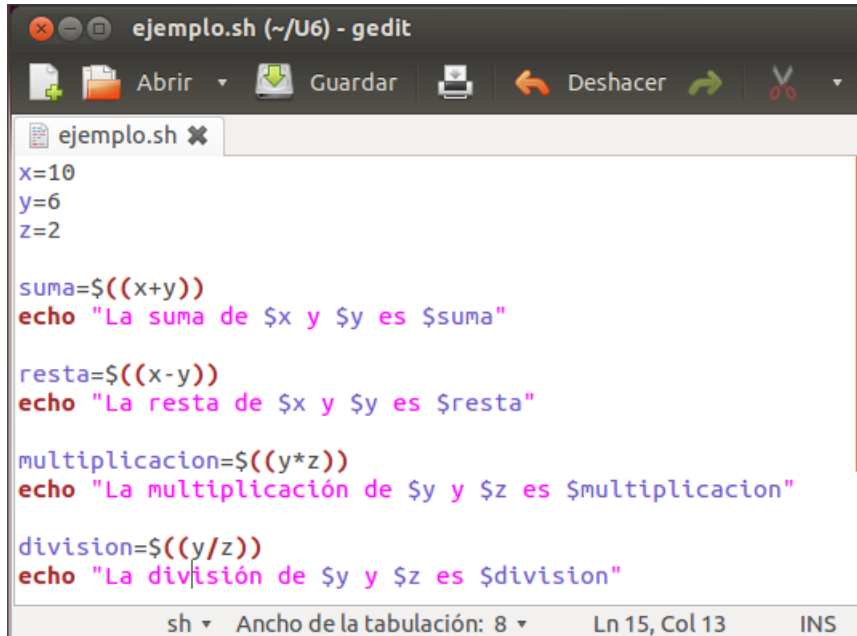
suma=`expr $x + $y`
echo "La suma de $x y $y es $suma"

otra_suma=$((y+z))
echo "La suma de $y y $z es $otra_suma"
sh Ancho de la tabulación: 8 Ln 8, Col 19 INS
```



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
La suma de 10 y 6 es 16
La suma de 6 y 2 es 8
pmartinez@Ubuntu12:~/U6$
```


Shell Scripts - Operaciones aritméticas - Ejemplo



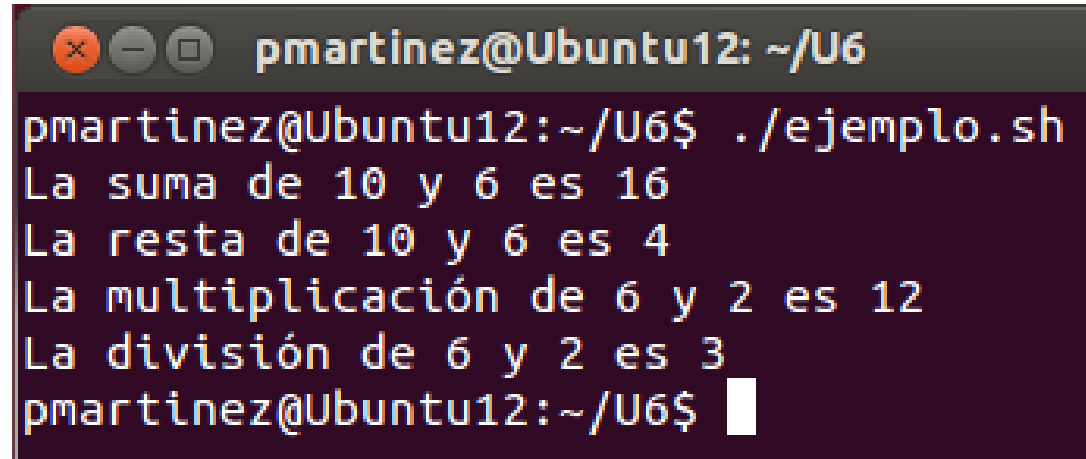
```
ejemplo.sh (~/U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh x
x=10
y=6
z=2

suma=$((x+y))
echo "La suma de $x y $y es $suma"

resta=$((x-y))
echo "La resta de $x y $y es $resta"

multiplicacion=$((y*z))
echo "La multiplicación de $y y $z es $multiplicacion"

division=$((y/z))
echo "La división de $y y $z es $division"
sh Ancho de la tabulación: 8 Ln 15, Col 13 INS
```



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
La suma de 10 y 6 es 16
La resta de 10 y 6 es 4
La multiplicación de 6 y 2 es 12
La división de 6 y 2 es 3
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Estructuras de control

- Podemos usar también estructuras de control en nuestros scripts:
 - **Estructura de control condicional: if**

if [condicion]; then

BLOQUE DE COMANDOS

else

BLOQUE DE COMANDOS

fi

Shell Scripts - Estructuras de control

- La condición de un **if** puede ser una de las indicadas en la siguiente tabla:

<code>cadena1 = cadena2</code>	Cierto si las cadenas son iguales.
<code>cadena1 != cadena2</code>	Cierto si las cadenas no son iguales.
<code>arg1 -eq arg2</code>	Cierto si los valores son iguales.
<code>arg1 -ne arg2</code>	Cierto si los valores son distintos.
<code>arg1 -lt arg2</code>	Cierto si el primer valor es menor que el segundo.
<code>arg1 -le arg2</code>	Cierto si el primer valor es menor o igual que el segundo.
<code>arg1 -gt arg2</code>	Cierto si el primer valor es mayor que el segundo.
<code>arg1 -ge arg2</code>	Cierto si el primer valor es mayor o igual que el segundo.

Shell Scripts - Estructuras de control

- Estructura de control condicional if - Varias condiciones anidadas

if [condicion1]; then

BLOQUE DE COMANDOS

else if [condicion2]; then

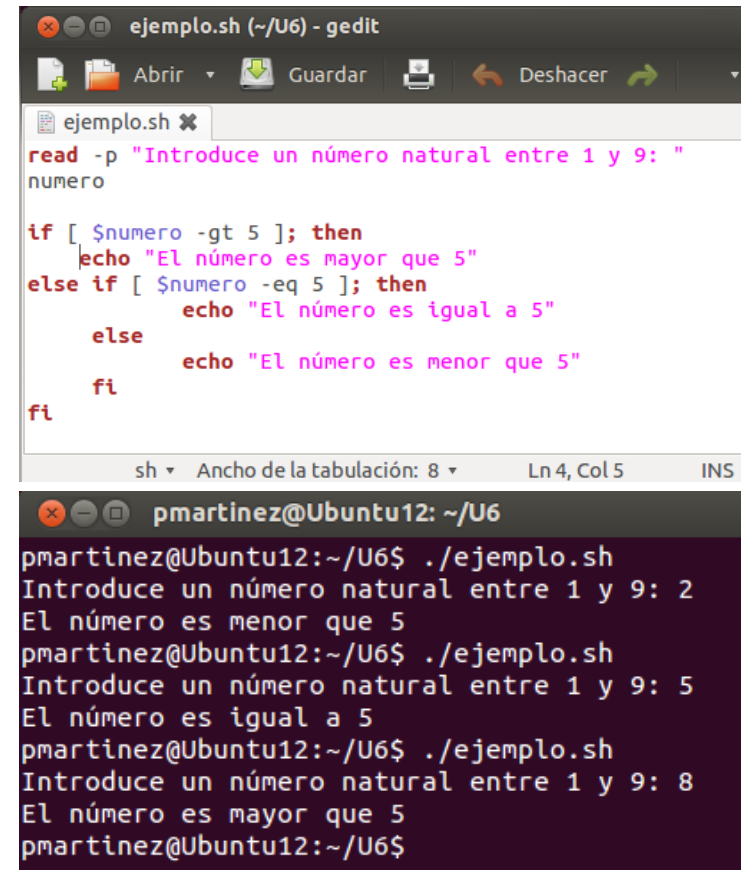
BLOQUE DE COMANDOS

else

BLOQUE DE COMANDOS

fi

fi



The image shows a shell script editor window titled 'ejemplo.sh (~ /U6) - gedit' and a terminal window below it. The script in the editor is as follows:

```
ejemplo.sh
read -p "Introduce un número natural entre 1 y 9: "
numero

if [ $numero -gt 5 ]; then
    echo "El número es mayor que 5"
else if [ $numero -eq 5 ]; then
    echo "El número es igual a 5"
else
    echo "El número es menor que 5"
fi
fi
```

The terminal window shows the execution of the script with three different inputs:

```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
Introduce un número natural entre 1 y 9: 2
El número es menor que 5
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
Introduce un número natural entre 1 y 9: 5
El número es igual a 5
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
Introduce un número natural entre 1 y 9: 8
El número es mayor que 5
pmartinez@Ubuntu12:~/U6$
```

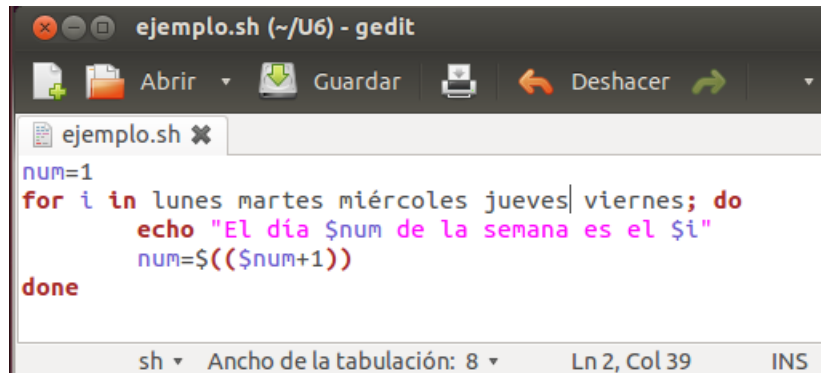
Shell Scripts - Estructuras de control

- Estructura de control iterativa: for

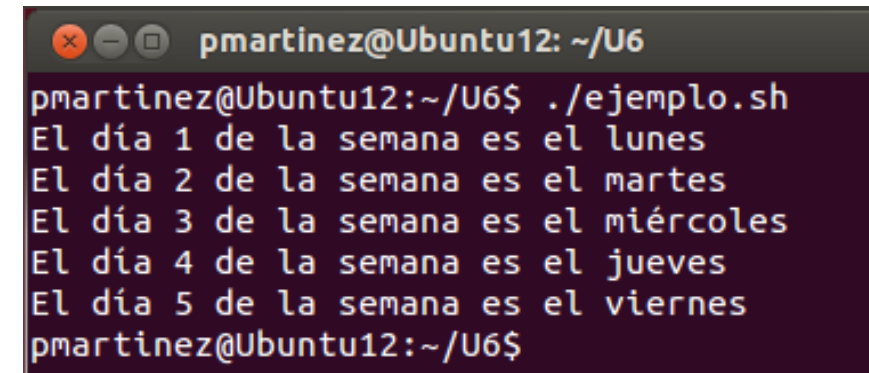
for variable in [lista o secuencia]; do

BLOQUE DE COMANDOS

done



```
ejemplo.sh (~/.U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh x
num=1
for i in lunes martes miércoles jueves viernes; do
    echo "El día $num de la semana es el $i"
    num=$((num+1))
done
sh Ancho de la tabulación: 8 Ln 2, Col 39 INS
```



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
El día 1 de la semana es el lunes
El día 2 de la semana es el martes
El día 3 de la semana es el miércoles
El día 4 de la semana es el jueves
El día 5 de la semana es el viernes
pmartinez@Ubuntu12:~/U6$
```

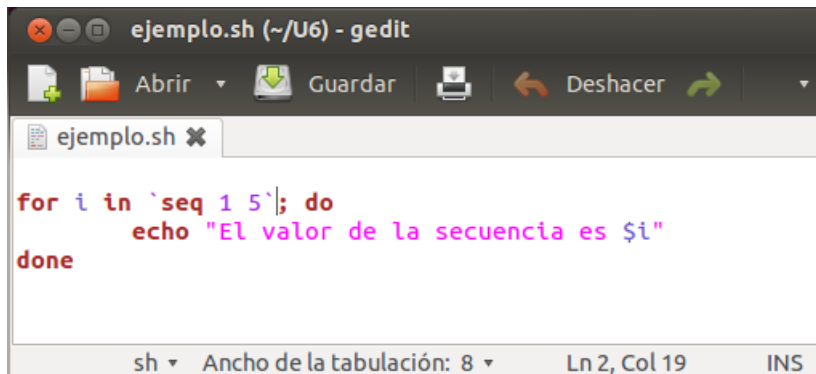
Shell Scripts - Estructuras de control

- Estructura de control iterativa: for

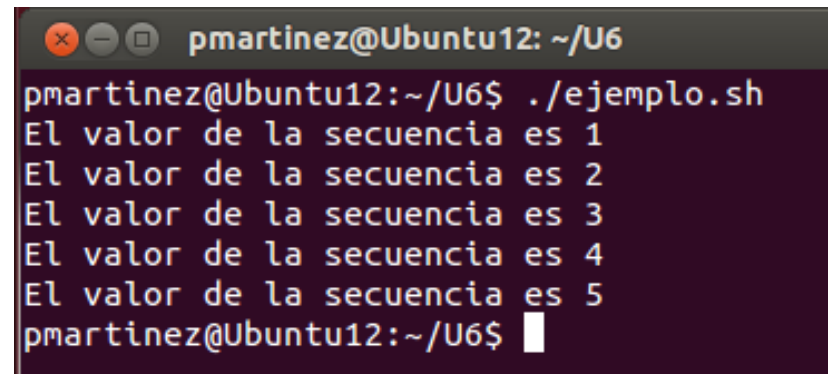
for variable in [lista o secuencia]; do

BLOQUE DE COMANDOS

done



```
ejemplo.sh (~/U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh x
for i in `seq 1 5`; do
    echo "El valor de la secuencia es $i"
done
sh Ancho de la tabulación: 8 Ln 2, Col 19 INS
```



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
El valor de la secuencia es 1
El valor de la secuencia es 2
El valor de la secuencia es 3
El valor de la secuencia es 4
El valor de la secuencia es 5
pmartinez@Ubuntu12:~/U6$
```

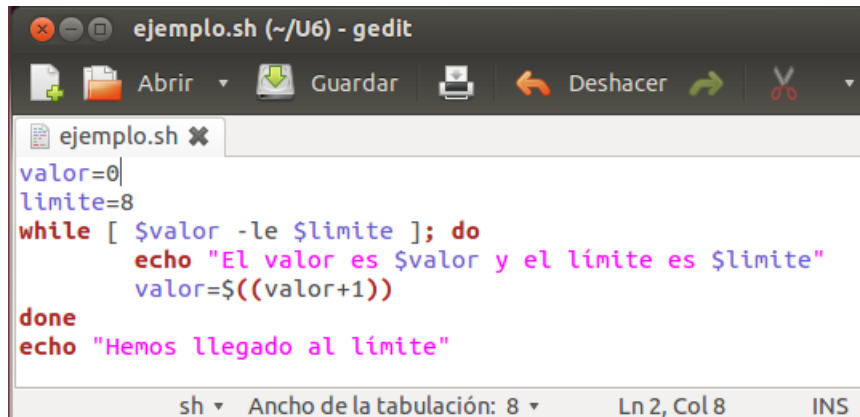
Shell Scripts - Estructuras de control

- Estructura de control iterativa: while

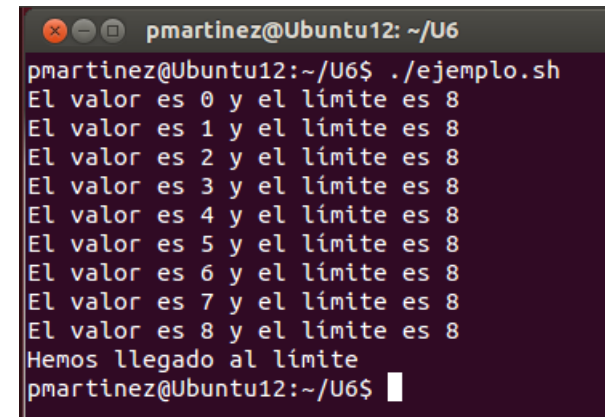
while [condicion]; do

BLOQUE DE COMANDOS

done



```
ejemplo.sh (~/.U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh x
valor=0
limite=8
while [ $valor -le $limite ]; do
    echo "El valor es $valor y el límite es $limite"
    valor=$((valor+1))
done
echo "Hemos llegado al límite"
sh Ancho de la tabulación: 8 Ln 2, Col 8 INS
```



```
pmartinez@Ubuntu12: ~/U6
pmartinez@Ubuntu12:~/U6$ ./ejemplo.sh
El valor es 0 y el límite es 8
El valor es 1 y el límite es 8
El valor es 2 y el límite es 8
El valor es 3 y el límite es 8
El valor es 4 y el límite es 8
El valor es 5 y el límite es 8
El valor es 6 y el límite es 8
El valor es 7 y el límite es 8
El valor es 8 y el límite es 8
Hemos llegado al límite
pmartinez@Ubuntu12:~/U6$
```

Shell Scripts - Ejemplo

- Mosaico...

```
ejemplo.sh (~/.U6) - gedit
Abrir Guardar Deshacer
ejemplo.sh
for i in `seq 1 10`; do
    contador=$i
    while [ $contador -le 10 ]; do
        contador=$((contador+1))
        if [ $contador -gt 5 ]; then
            echo -n " 0 "
        else
            echo -n " 1 |"
        fi
    done
    echo ""
done
```

```
pmartinez@Ubuntu12: ~/.U6
pmartinez@Ubuntu12:~/.U6$ ./ejemplo.sh
1 1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
1 1 0 0 0 0 0 0
1 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0
0 0 0 0
0 0 0
0 0
0
pmartinez@Ubuntu12:~/.U6$
```


Shell Scripts - Ejemplo

- Un shell script, generalmente, comienza con una línea especial llamada **shebang**.
- También conocida como hashbang o directiva de intérprete
- Se trata de una línea especial que indica al sistema qué intérprete de comandos debe utilizarse para ejecutar el script: **#!/bin/bash**



The screenshot shows a text editor window titled 'ejemplo.sh' with a path '~ / U6'. The window contains a shell script with the following lines:

```
1 #!/bin/bash
2
3 #...
4
5 #Cuerpo del script
6
7 #...
8
```