

# Administración de Sistemas Operativos



## U12. Programación shell

[www.adminso.es](http://www.adminso.es)



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

# COMANDOS BÁSICOS



## U12. Programación Shell (Linux)

### Contenido

#### Conceptos básicos

E/S de datos

Operaciones

Estructuras de control

Funciones

### Características

- El shell del sistema permite realizar programas que permiten automatizar y programar tareas de administración.
- Es un lenguaje de programación.
- Los programas son interpretados por el sistema.



## U12. Programación Shell (Linux)

### Contenido

#### Conceptos básicos

E/S de datos

Operaciones

Estructuras de control

Funciones

### Variables

Las variables permiten guardar información y a partir de ella poder realizar operaciones y tomar decisiones.

Ejemplo:

```
#!/bin/bash  
numero=5  
echo "el valor de la variable es "$numero
```

Cuando se accede al valor se pone el símbolo \$



## U12. Programación Shell (Linux)

### Contenido

#### Conceptos básicos

E/S de datos

Operaciones

Estructuras de control

Funciones

### Paso por parámetros

El paso por parámetros permite que los scripts reciban parámetros desde la línea de comandos.

Ejemplo:

```
#!/bin/bash
echo "El nombre del programa es "$0
echo "El primer parámetro recibido es "$1
echo "El segundo parámetro recibido es "$2
echo "..."
echo "En total se han recibido "$# parámetros"
```

**\$0 = 1º parámetro**  
**\$1 = 2ª parámetro**  
**..**

**Número total de parámetros recibidos**



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

# ENTRADA Y SALIDA DE DATOS



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### E/S por consola

Para leer un valor se utiliza *read*.

Ejemplo:

```
#!/bin/bash
echo -n "Introduce el valor de la variable: "
#el parámetro -n se utiliza para evitar el salto de línea
read numero
echo "El valor introducido es: "$numero
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Redirección de la E/S

La redirección de la E/S permite que el resultado de un comando sean los valores de entrada de otro comando.

Para realizar la redirección se utiliza los símbolos **>** y **<**

Ejemplo:

```
$ ls /etc/*.conf > ListaArchivos.txt
```

La salida del comando ls se guarda en el fichero

```
$ grep host <ListaArchivos.txt
```

Los datos del fichero se utilizan en el comando grep





## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Filtrado de textos

Los comandos más importantes para filtrar datos son:

- **grep.** Muestran las líneas que cumplen una determinada condición.
- **head.** Muestra las n primeras líneas de un archivo.
- **tail.** Muestra las n últimas líneas de un archivo.
- **cut.** Muestra una determinada columna.
- **sort.** Permite ordenar datos.



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### grep – Filtra la salida

```
$ less /etc/passwd | grep root
```

**Muestra las líneas del fichero /etc/passwd  
que contienen la palabra root**

### head y tail -- Muestra la n primeras o últimas líneas de un archivo

```
$ less /etc/passwd | head -n 5
```

```
$ less /etc/passwd | tail -n 5
```

**Muestra las n primeras (head) o últimas (tail) líneas del fichero**



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### cut – Muestra una determinada columna

```
$ less /etc/passwd | cut -d ":" -f1
```

**-d ":"** delimita las columnas por el símbolo :  
**-f1**. muestra la primera columna

### sort -- Ordena la salida

```
$ less /etc/passwd | cut -d ":" -f1 | sort  
$ less /etc/passwd | cut -d ":" -f3 | sort -n
```

**La opción -n se utiliza cuando se ordenan datos numéricos**



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

# OPERACIONES ARITMÉTICO O LÓGICAS



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

Como cualquier lenguaje de programación se pueden realizar operaciones aritmético lógicas sobre las variables.

Para realizar operaciones se utiliza el comando *expr* y para realizar comparaciones se utiliza el comando *test*.



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

**expr → Realiza operaciones**

Sintaxis:

```
$ expr arg1 op arg2 [op arg3...]
```

Ejemplo:

```
#!/bin/bash
echo -n "Introduce un valor: "
read var1
echo -n "Introduce un valor: "
read var2
resultado=$(expr $var1 \* $var2)
echo "El resultado de la multiplicación es "$resultado
```

**Multiplica 2 valores introducidos por teclado**



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

**expr → Realiza operaciones**

Tabla 12-1. Operadores de *expr*

Operador	Comentario
<b>Operadores aritméticos</b>	
+	Suma.
-	Resta.
\*	Multiplicación. El operador * va precedido de \ porque * ya tiene un significado en GNU/Linux.
/	División.
%	Resto de la división.
<b>Operadores relacionales</b>	
=	Igualdad.
!=	Diferentes.
>	Mayor.
>=	Mayor o igual.
<	Menor.
<=	Menor o igual.
<b>Operadores lógicos</b>	
	Or lógico.
&	And lógico.



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

**test → Realiza comparaciones**

Sintaxis:

```
$test - opcion archivo
```

Ejemplo:

```
test -f /etc/passwd
```

Mira si es un fichero

```
test 20 -lt 40
```

Indica si el 2º valor es menor que el 1º





## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

**test → Realiza comparaciones**

Tabla 12-2. Opciones del comando *test*

Opción	Descripción
<b>Archivos o directorios</b>	
-f	Devuelve verdadero (0) si el archivo existe y es un archivo regular (no es un directorio ni un archivo de dispositivo).
-s	Devuelve verdadero (0) si el archivo existe y si su tamaño es mayor que 0.
-r	Devuelve verdadero (0) si el archivo existe y tiene permisos de lectura.
-w	Devuelve verdadero (0) si el archivo existe y tiene permisos de escritura.
-x	Devuelve verdadero (0) si el archivo existe y tiene permisos de ejecución.
-d	Devuelve verdadero (0) si existe y es un directorio.
<b>Valores numéricos</b>	
-lt	Menor que
-le	Menor o igual que
-gt	Mayor que
-ge	Menor o igual que
-eq	Igual a
-ne	No igual a
<b>Conectores</b>	
-o	OR
-a	AND
!	NOT



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

# ESTRUCTURAS DE CONTROL



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

Las estructuras de control permiten cambiar el flujo de programa, en función del estado de las variables.

Las opciones que tenemos son:

- Toma de decisiones
  - If
  - Case
- Bucles
  - for
  - while



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Condición simple (if)

Las condiciones simples (if) permiten que en caso de cumplirse una determinada condición se ejecute un determinado código.

Ejemplo:

```
#!/bin/bash
echo -n "Introduce un valor: "
read var
if (( var < 10 ))
then
    echo "El valor es menor que 10"
else
    echo "El valor es mayor que 10"
fi
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Condiciones múltiples (case)

Cuando se realizan muchas condiciones sobre un mismo valor (p.e.: en un menú) la mejor opción es utilizar case.

Ejemplo:

```
#!/bin/bash
echo -n "Introduce un valor: "
read var1
case $var1 in
  1) echo " uno ";;
  2) echo " dos ";;
  3) echo " tres ";;
  4) echo " cuatro ";;
  *) echo "opcion incorrecta ";;
esac
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Bucle for

El bucle for se utiliza para ejecutar un código un determinado número de veces.

Ejemplo:

```
#!/bin/bash
for (( i = 0 ; i <= 5; i++ ))
do
    echo " $i "
done
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Bucle while

El bucle *while* permite ejecutar un código hasta que no se cumpla una determinada condición de salida.

Ejemplo:

```
#!/bin/bash
limite=5
i=0;
while (test $limite -gt $i)
do
    echo "Valor $i"
    let i=$i+1
done
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

# FUNCIONES





## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

Una función es un bloque de código que permite su reutilización de una forma fácil y sencilla.

### Ejemplo

```
#!/bin/bash

function mostrar_mensaje() {
    echo "Hola mundo!"
}

mostrar_mensaje;
```



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Ejemplos

```
#!/bin/bash
function suma (){
    resultado=$(expr $1 + $2)
    echo "a + b =" $resultado
}
a=5
b=10
suma $a $b
```

Recibo las variables como \$1, \$2..



## U12. Programación Shell (Linux)

### Contenido

Conceptos  
básicos

E/S de datos

Operaciones

Estructuras de  
control

Funciones

### Ejemplos

```
#!/bin/bash
function suma (){
    c=$(expr $a + $b)
    return $c
}
a=5
b=10
suma $a $b
resultado=$?
echo $resultado
```

Devuelvo el resultado con return

Con \$? Obtengo el resultado de la función

