

There are no translations available.

En esta segunda entrega veremos las estructuras de control y un par ejemplos muy interesantes...

Continuemos con este tutorial, ahora veremos los siguientes puntos:

- SCRIPTS EN LINUX: Estructuras de control
- SCRIPTS EN LINUX: Ejemplos de Scripts I.

4. Estructuras de control

Cómo en cualquier lenguaje de programación, necesitaremos una serie de estructuras de control que permitan modificar el flujo de ejecución de las instrucciones de nuestro script.

4.1 If

La sintaxis mas sencilla será:

if *condicion*

then **[bloque de comandos]**

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
[           else           ]
```

```
##### [bloque de comandos]
```

```
fi
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Si queremos añadir mas condiciones:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
if          condicion1
then        [bloque de comandos si la condición1 se cumple]
elif       condicion2
then
[bloque de comandos si la condición2 se cumple]
else
[bloque de comandos si las condiciones no se cumplen ni la
condición1 ni la condición2]
fi
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Normalmente la condición será el resultado de evaluar una primitiva o de ejecutar un comando.

Veamos un ejemplo:

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

```
pin="1234"
```

```
echo "Introduzca su pin"
```

```
read -s clave
```

```
if test "$clave" = "$pin"
```

```
then
```

```
    echo "Pin correcto"
```

```
    echo "Acceso permitido"
```

```
else
```

```
##### echo "Pin incorrecto"  
  
fi
```

En este sencillo ejemplo se nos pide que introduzcamos el pin (en nuestro caso lo hemos fijado a 1234), y dependiendo si lo introducimos bien o no, nos muestra una mensaje u otro. Más adelante veremos con mas profundidad las posibilidades del comando test.

Evaluando expresiones según código de error: && y ||

Existe otra manera de evaluar condiciones mediante la utilización de los caracteres especiales && y ||, que usan los códigos de error de un comando.

Es decir, cada comando ejecutado devuelve un valor de salida y como vimos, sacando el valor de '\$?', podíamos saber la salida del último proceso ejecutado.

El **operador '&&'** ejecuta un comando, y si es correcto ejecuta el siguiente, veamos un ejemplo:

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

```
if ls /var && ls /etc
```

```
then
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
##### echo "ok"
```

```
else
```

```
##### echo "error"
```

```
fi
```


Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

En este caso para que salga de mensaje ok, se tendrá que haber podido listar el directorio **/var**

y el

/etc

también, y sino dará error (esto puede ser usado cuando queremos comprobar la existencia de

los directorios para nuestros script):

El **operador** `'||'` ejecuta un comando y si no es correcto ejecuta el siguiente:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

`#!/bin/bash`

`if ls /var || ls /etc`

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

then

```
##### echo "ok"
```

else

```
##### echo "error"
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

fi

En este caso saldrá el mensaje "ok" si se ha podido listar correctamente alguno de los

directorios (**/var** o **/etc**), o los dos.

Comparación de ficheros y variables con test.

Para evaluar expresiones condicionales dentro de los 'if' se usa el comando 'test'. Veamos un

resumen en la siguiente tabla:

EXPRESIONES	DESCRIPCION

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-b fichero

Cierto si fichero existe y es un dispositivo de bloques.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-c fichero

Cierto si fichero existe y es un dispositivo de caracteres.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-d fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Cierto si fichero existe y es un directorio.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-e fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Cierto si fichero existe.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-f fichero

Cierto si fichero existe y es un fichero normal.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-g fichero

Cierto si fichero existe y tiene el bit de grupo activado.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-k fichero

Cierto si fichero tiene el bit de sticky activado.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-L fichero

Cierto si fichero existe y es un enlace simbólico.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-p fichero

Cierto si fichero existe y es una tubería nombrada.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-r fichero

Cierto si fichero existe y es legible.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-s fichero

Cierto si fichero existe y su tamaño es mayor que cero.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-S fichero

Cierto si fichero existe y es un socket.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-t [df]

Cierto si df está abierto en un terminal. Si df es omitido, se toma 1 (salida estándar) por defecto.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-u fichero

Cierto si fichero existe y tiene el bit de usuario activo.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-w fichero

Cierto si fichero existe y es escribible.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-x fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Cierto si fichero existe y es ejecutable.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-O fichero

Cierto si fichero existe y es propiedad del identificador efectivo del usuario.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-G fichero

Cierto si fichero existe y es propiedad del identificador efectivo del grupo.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20



fichero1 -nt fichero2

Cierto si fichero1 es más reciente (en base a la fecha de modificación) que fichero2.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20



fichero1 -ot fichero2

Cierto si fichero1 es más antiguo que fichero2.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20



fichero1 -ef fichero2

Cierto si fichero1 y fichero2 tienen el mismo número de dispositivo y de nodo-i.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-z cadena

Cierto si la longitud de cadena es cero.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

-n cadena

Cierto si la longitud de cadena no es cero.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

cadena

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Cierto si la longitud de cadena no es cero.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
cadena1 = cadena2
```

Cierto si las cadenas son iguales.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
cadena1 != cadena2
```

Cierto si las cadenas no son iguales.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

! expr

Cierto si expr es falsa.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

expr1 -a expr2

Cierto si expr1 y expr2 son ciertas.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

expr1 -o expr2

Cierto si expr1 o expr2 son ciertas.

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

arg1 OP arg2

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

OP es uno de los siguientes valores:

-eq, -ne, -lt, -le, -gt, o

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Veamos un ejemplo:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

#!/bin/bash

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

echo "dame primer número"

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

read primero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

echo "dame segundo número"

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

read segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

if test \$primero -lt \$segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

then

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

echo \$primero es menor que \$segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

else

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

if test \$primero -gt \$segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

then

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

echo \$primero es mayor que \$segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

else

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

echo \$primero es igual que \$segundo

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

fi

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

fi

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

4.2 For

El bucle 'for' es una estructura de control iterativa que permite repetir una sección del programa, un número fijo de veces. Su estructura es así:

```
for      variable      in [lista de palabras]
```

```
do
```

```
comandos
```

```
done
```

Otra manera de escribir nuestro bucle 'for' podría ser así:

```
for      variable      in [lista de palabras];      comandos; done
```

Veamos un ejemplo, en el que nos ayudamos de la ejecución del comando `seq` para cargar la variable:

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

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

for variable in `seq 1 100`

do

echo \$variable

done

Conseguiremos como salida un listado de los 100 primeros números.

4.3 While

Sirve para repetir un bucle mientras se cumpla la condición:

While *condición*

do

comandos

done

Veamos en siguiente ejemplo donde creamos una sucesión de números desde un origen, en

este caso el 33, hasta el final que marca la condición (el 100):

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

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

```
num=33
```

```
while [ $num -le 100 ]
```

```
do
```

```
echo "numero: $num"
```

```
num=`expr $num + 1`
```

```
done
```

4.4 Until

Igual que While pero el bucle se ejecuta mientras la condición sea falsa:

```
Until      condición
```


do

comandos

done

A continuación, el mismo ejemplo que el While, pero esta vez utilizando la estructura Until:

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

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
num=33
```

```
until [ $num -eq 100 ]
```

```
do
```

```
echo "numero: $num"
```

```
num=`expr $num + 1`
```

```
done
```

4.5 Case

Veamos la estructura:

case variable in

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

patron1) comandos condicion1;;

patron2) comandos condicion2;;

...

patron n) comandos condicion n;;

***) comandos si no se cumplen ninguna;;**

esac

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

En esta estructura cuando se encuentre un patron que coincida, se ejecuta la lista de

comandos hasta los ';;' y se termina la ejecución de 'case'. Veamos un ejemplo:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

`#!/bin/bash`

case \$1 in

start)

echo "Arranco el servidor"

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

;;

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

stop)

echo "Paro el servidor"

;;

restart)

\$0 stop

\$0 start

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

;;

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

*)

echo "Este script se arranca tecleando \$0 start o \$0 stop"

;;

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

esac

4.6 Select

A continuación se muestra la estructura:

```
select nombre_de_variable [in lista]
```

```
do
```

```
comandos
```

```
done
```

El siguiente ejemplo guarda en el fichero "ejemploselect" el resultado del comando ls, que lista

los archivos del directorio actual, y permite editarlo y salir:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

`#!/bin/bash`

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
echo "Elige lo que quieres hacer "
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

fichero=/home/ejemploselect

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

ls > \$fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

select opciones in listar editar salir

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

do

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

case \$opciones in

listar)

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

cat \$fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

;;

editar)

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

vi \$fichero

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

;;

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

salir)

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

break

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

;;

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

esac

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

done

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

5. SCRIPTS EN LINUX: Ejemplos de Scripts I.

Ya esta bien de teoría, vamos a ver un par de scripts del mundo real. Uno de ellos nos servira para comprobar que nuestra red local de "miles" de servidores está funcionando. El segundo script sirve para automatizar la creación de usuarios y servirá de punto de partida para otro algo más complejo que veremos despues.

5.1 Script par realizar un "ping" a todas las máquinas de nuestro sistema.

Supongamos que somos administradores o responsables de una red con cierto número de servidores y queremos realizar periodicamente la tarea de saber si están "levantados". Para ello escribiremos un script que lanzará un ping a todos nuestro servidores y nos avisará sonoramente cuando uno de ellos no responda.

Lanzamos el "vi" y creamos el siguiente script que llamaremos ping.sh

```
#!/bin/sh
```

```
#### Defino una función con todas las operaciones para poder llamarla recursivamente. Las fun  
#### se les llama o invoca
```

```
ejecucion()
```

```
{
```

```
clear
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

#####limpia la pantalla

```
for server in `cat mis_servers.lst`
```

```
#### Realiza un cat del fichero mis_servers.lst y almacena ciclicamente cada linea en la variable  
#### tantas veces como lineas tenga mis_servers.lst y almacena el contenido de la linea en la variable  
#### del for realiza lo que está entre el do y el done.  
do
```

```
echo  
echo Realizo un ping a la maquina $server  
echo  
ping -c 2 -A $server
```

```
### Manda dos paquetes de trafico icmp al destino almacenado en server. Emite un pitido si no responde  
done
```

```
###Ahora dentro de la funcion voy a llamar a la propia funcion para convertir el proceso en un bucle  
### espere 2 minutos antes de volver a ejecutarse
```

```
sleep 120  
ejecucion  
}  
ejecucion
```

```
#### Esta llamada externa es la que se va a ejecutar la primera vez.
```

Necesitamos crear un fichero "mis_servers.lst" con las ip de nuestros servidores:

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
root@ubuntu:~# cat mis_servers.lst
192.168.1.37
192.168.1.1
192.168.1.89
```

Entonces al ejecutar nuestro script obtenemos:

Realizo un ping a la maquina 192.168.1.37

```
PING 192.168.1.37 (192.168.1.37): 56 data bytes
64 bytes from 192.168.1.37: icmp_seq=0 ttl=128 time=0.244 ms
64 bytes from 192.168.1.37: icmp_seq=1 ttl=128 time=0.269 ms
```

```
--- 192.168.1.37 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.244/0.257/0.269/0.013 ms
```

Realizo un ping a la maquina 192.168.1.1

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=255 time=0.637 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=255 time=0.629 ms
```

```
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.629/0.633/0.637/0.004 ms
```

Realizo un ping a la maquina 192.168.1.89

PING 192.168.1.89 (192.168.1.89): 56 data bytes

--- 192.168.1.89 ping statistics ---

2 packets transmitted, 0 packets received, 100% packet loss

5.2 Script para crear usuarios en local

Vamos a realizar un script que nos facilite la creación de usuarios en una máquina LINUX. Este script es muy bueno porque aunque ejecutado localmente tampoco nos ahorra demasiado tiempo, puede usarse de base para un script que cree usuarios en máquinas remotas.

####Creamos una función que recoja los datos necesarios para la creacion de usuarios

```
#####  
###FUNCION QUE RECOGE LOS DATOS#####  
#####
```

```
peticion_datos()  
{
```

```
#### Borramos el contenido de las variables que vamos a usar. Por precaucion.
```

```
unset $user  
unset $grupo1  
unset $grupo2  
unset $shell  
unset $password  
unset $casa
```

Vamos pidiendo los datos que necesitamos

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
echo Nombre de usuario:  
read user
```

```
### Le proponemos al usuario el grupo "users", si no introduce ningun grupo, se lo asignamos
```

```
echo Grupo principal:[users]  
read grupo1  
if [ "$grupo1" = "" ]  
then  
grupo1=users  
fi
```

```
#### Puede no especificarse un grupo secundario  
echo Grupo Secundario:  
read grupo2  
if [ "$grupo2" = "" ]  
then  
grupo2=""  
else  
grupo2="-g '$grupo2'  
fi
```

```
####Si no especifica una shell se le asigna /bin/bash
```

```
echo Shell:[/bin/bash]  
read shell  
if [ "$shell" = "" ]  
then  
shell=/bin/bash  
fi
```

```
####Home del usuario, por defecto home/nombre_usuario
```

```
echo Home del usuario: [/home/$user]
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
read casa
if [ "$casa" = "" ]
then
casa=/home/$user
fi
```

```
####Existe el home?? Hay que crearlo???
echo Existe el home del usuario: [n]
read exist
if [ "$exist" = "" ] || [ "$exist" = "n" ]
then
param=-m
fi
```

#####Esta parte es la mas interesante. El comando useradd de linux permite especificar la password y se la encripta nosotros. Le pedimos la password al usuario del script y se la pasamos a un script. En dicho script la almacenamos en una variable.

```
echo Password:
read password
contrasena=`perl /root/scripts/crypt.pl $password`
```

El siguiente codigo es solo de control, para que muestre por pantalla el resultado de la encriptación.

```
echo _____
echo ----- DEBUG-----
echo _____
```

```
echo $contrasena
```

```
}
```

```
#####  
### FIN DE LA FUNCION #####  
#####
```

Como ya hemos especificado antes, las funciones no se ejecutan hasta que no se las llame

peticion_datos

####Ejecutamos el comando de creacion de usuario

/usr/sbin/useradd -G \$grupo1 \$grupo22 -d \$casa \$param -p \$contrasena \$user

El script de perl es muy sencillo. Debemos crear el fichero crypt.pl en la misma ubicación del script de creación de usuarios. El contenido del fichero debe ser:

```
#!/usr/local/bin/perl  
#  
# call it like so: perl crypt.pl password
```

```
srand (time());  
my $randletter = "(int (rand (26)) + (int (rand (1) + .5) % 2 ? 65 : 97))";  
my $salt = sprintf ("%c%c", eval $randletter, eval $randletter);  
my $plaintext = shift;  
my $crypttext = crypt ($plaintext, $salt);
```

```
print "${crypttext}";
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

Podemos comprobar que funciona haciendo:

```
root@ubuntu:~# perl crypt.pl funciona???
```

fFvYQ9cG.e.as

Lo que nos devuelve es la cadena encriptada equivalente a "funciona???".

Una vez que ya tenemos esto podemos probar la ejecución del script. Lo lanzamos y tenemos lo siguiente:

```
Nombre de usuario:  
javi  
Grupo principal:[users]
```

```
Grupo Secundario:
```

```
Shell:[/bin/bash]
```

```
Home del usuario: [/home/javi]
```

```
Existe el home del usuario: [n]
```

```
Password:  
javierete
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

----- DEBUG-----

ou47ezbisMOic

Es muy comodo porque hemos incluido opciones por defecto lo que nos permite responder a casi todas las preguntas pulsando ENTER. Ahora vemos lo que ha originado el script.:

FICHERO **/etc/passwd**:

.....

.....

gdm:x:108:118:Gnome Display Manager:/var/lib/gdm:/bin/false

ubuntu:x:999:999:Live session user,,,:/home/ubuntu:/bin/bash

sshd:x:109:65534::/var/run/sshd:/usr/sbin/nologin

javi:x:1000:1001::/home/javi:/bin/sh

FICHERO **/etc/shadow**:

ubuntu:U6aMy0wojraho:13869:0:99999:7:::

sshd!:13869:0:99999:7:::

javi:ou47ezbisMOic:13869:0:99999:7:::

Observamos que la ejecución ha generado entradas en dos ficheros. En el fichero **/etc/passwd** se almacenan los datos del usuario y en el **/etc/shadow** se almacena la password. También se comprueba que se ha creado el directorio "home" del usuario:

```
root@ubuntu:~# cd /home#
```

```
root@ubuntu:~# ls -lta#
```

Tutorial Shell Scripts II

Écrit par Javier Martínez Avedillo
Jeudi, 03 Avril 2008 21:20

```
total 0
drwxrwxrwt 30 root root 240 2007-12-22 11:22 ..
drwxr-xr-x 21 ubuntu ubuntu 740 2007-12-22 11:32 ubuntu
drwxr-xr-x 2 javi javi 140 2007-12-22 12:17 javi
drwxr-xr-x 4 root root 100 2007-12-22 12:17 .
```

¶ Para continuar viendo este tutorial, vaya al siguiente enlace:¶ [Tutorial Shell Scripts](#)
[III](#) .