

Buscar ...

**Menu**

f

...UX

t

in

p

# Script de Bash paso a paso tutorial (con ejemplos)

— Mokhtar Ebrahim Publicada: febrero 7, 2017 Última actualización: enero 10, 2019

En este tutorial, vamos a hablar sobre scripts de Bash o scripts de shell y cómo crear tu primer script de Bash. En realidad, se les llama scripts de shell en general, pero vamos a llamarlos scripts de Bash porque vamos a utilizar Bash entre los otros shells de Linux.

Existen los shell zsh, tcsh, ksh y otros shell.

En las publicaciones anteriores, vimos como utilizar el shell de bash y cómo utilizar **Comandos de Linux**.

El concepto de un script de Bash es ejecutar una serie de comandos para hacer su trabajo.

Para ejecutar múltiples comandos en un solo paso desde el shell, puede escribirlos en una línea y separarlos con punto y coma.

```
pwd ; whoami
```

f

🐦

n realidad, esto es un script de Bash!!

in

comando pwd se ejecuta primero, mostrando el directorio de trabajo

📌

tual, luego se ejecuta el comando whoami para mostrar a los usuarios que actualmente están conectados.

Puedes ejecutar múltiples comandos tanto como desee, pero con un límite. Puedes determinar tus max args usando este comando.

```
getconf ARG_MAX
```

Bueno, ¿y qué sucede si pones los comandos en un archivo, y cuando necesitamos ejecutar estos comandos, solo ejecutamos ese archivo? Esto es conocido como un script de Bash.

Primero, crea un nuevo archivo usando el **comando touch**. Al comienzo de cualquier script de Bash, debemos definir qué shell usaremos porque hay muchos shells en Linux, Bash shell es uno de ellos.

## Tabla de contenidos



### 1. Script de Bash Shebang

## 2. Establecer los permisos de scripts

### 3. Crea tu primer script Bash

### 4. Usando variables

#### 4.1. Variables de entorno

#### 4.2. Variables de usuario

### 5. Sustitución de comando

### 6. Cálculo matemático

### 7. Sentencia if-then

#### 7.1. Verificar si un usuario existe

### 8. Sentencia if-then-else

### 9. Combinar pruebas

#### 0. Comparaciones Numericas

#### 1. Comparaciones de cadenas

#### 2. Comparaciones de archivos



## Script de Bash Shebang

La primera línea que escribes al escribir un script bash es el (#!) Seguido del shell que utilizaras.

**#!** <=== este signo se llama shebang.

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

Si utilizas el signo de numeral (#) delante de cualquier línea en su script de Bash, esta línea se comentará, lo que significa que no se procesará, pero la línea anterior es un caso especial. Esta línea define qué shell utilizaremos, que es el Bash de shell en nuestro caso.

Los comandos de shell se ingresan uno por línea como este:

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

```
# This is a comment
```

```
# THIS IS A COMMENT
```

```
pwd
```

```
whoami
```

Puedes escribir múltiples comandos en la misma línea, pero debes separarlos con punto y coma, aunque es preferible escribir comandos en líneas separadas, esto hará que sea más fácil de leer luego.

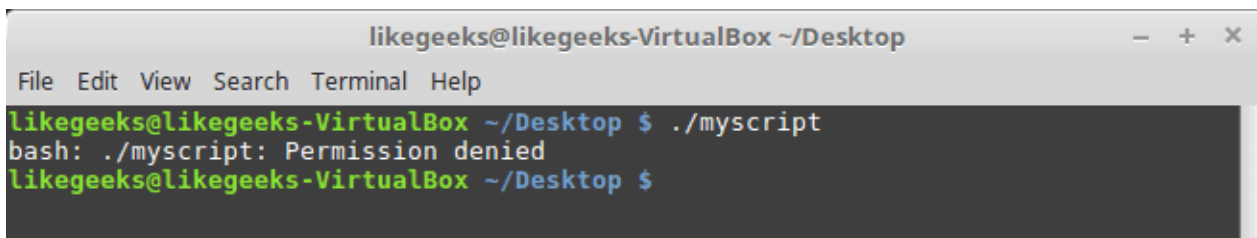
## f \_\_stablecer los permisos de scripts



Después de escribir tu script de Bash, guarda el archivo.



ora, configura ese archivo para que sea ejecutable, de lo contrario, mostrará permisos denegados. Puedes revisar cómo configurar permisos utilizando el **comando chmod**.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which results in the error message 'bash: ./myscript: Permission denied'. The prompt then returns to the user.

```
chmod +x ./myscript
```

Luego intenta ejecutarlo simplemente escribiéndolo en el shell:

```
./myscript
```

Y sí, se ejecuta.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ chmod +x ./myscript
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
/home/likegeeks/Desktop
likegeeks
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## Crea tu primer script Bash

Como sabemos por otras publicaciones, la impresión de texto se realiza mediante el comando echo.

Abre tu archivo y escribe esto:

```
#!/bin/bash

# our comment is here

echo "The current directory is:"

pwd

echo "The user logged in is:"

whoami
```

Mira el resultado:

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
The current directory is:
/home/likegeeks/Desktop
The user logged in is:
likegeeks
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

¡Perfecto! Ahora podemos ejecutar comandos y mostrar texto utilizando el comando echo.

Si no conoces el comando echo o cómo editar un archivo, te recomiendo que consultes los artículos acerca de los [comandos basicos de Linux](#)

## Usando variables

Las variables le permiten almacenar información para usarlas dentro del script.

f

t

uedes definir 2 tipos de variables en tu script de Bash:

in

Variables de entorno

p

Variables de usuario

## Variables de entorno

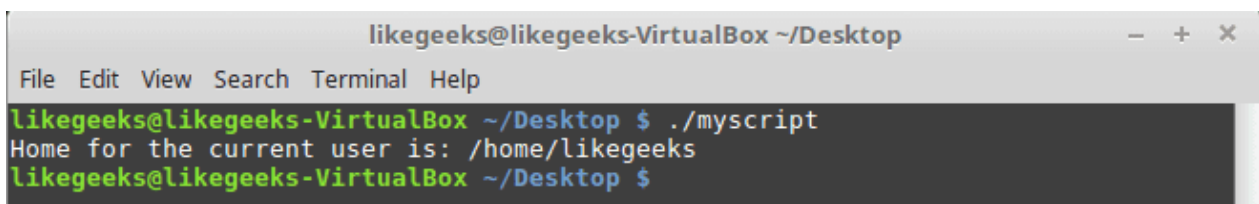
Algunas veces necesitas interactuar con las variables del sistema, puede hacerlo utilizando [variables de entorno](#).

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

```
# display user home
```

```
echo "Home for the current user is: $HOME"
```

Observa que ponemos la variable de sistema \$ HOME entre comillas dobles, e imprime la variable home correctamente.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows a prompt 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' followed by the command './myscript'. The output of the script is 'Home for the current user is: /home/likegeeks'. The prompt then changes to 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'.

## ¿Que pasa si queremos imprimir el signo de dolar?

```
echo "I have $1 in my pocket"
```

Debido a que la variable \$ 1 no existe, no funcionará. Entonces, ¿cómo superar eso?

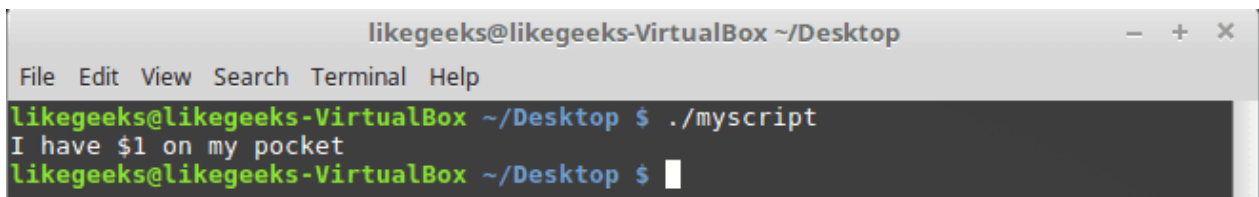
Puede usar el carácter de escape que es la barra invertida \ antes del signo de dólar como se puede observar:

f

🐦 `echo "I have \$1 in my pocket"`

in

📌 hora funciona!!

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which outputs 'I have \$1 on my pocket'. The prompt then returns to the shell.

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
I have $1 on my pocket
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## Variables de usuario

Además, puedes establecer y utilizar variables personalizadas en tu script.

Puedes llamar a las variables de usuario de la misma manera como se muestra a continuación:

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

```
# User variables
```

```
grade=5
```

```
person="Adam"
```

```
PERSON= Adam
```

```
echo "$person is a good boy, he is in grade $grade"
```

```
chmod +x myscript
```

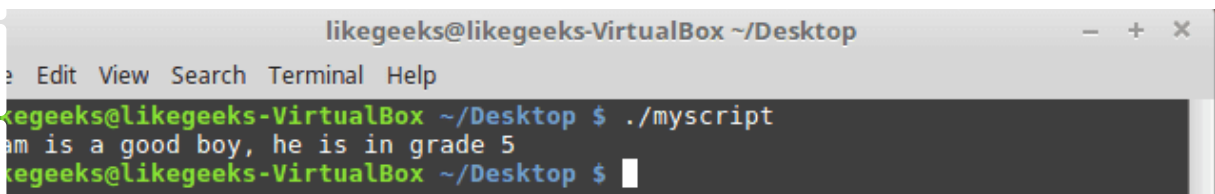
```
./myscript
```

f

t

in

p

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The terminal shows the command './myscript' being executed, which outputs 'Adam is a good boy, he is in grade 5'. The prompt returns to the shell.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Adam is a good boy, he is in grade 5
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## Sustitución de comando

Puedes extraer información del resultado de un comando utilizando la sustitución de comandos.

Puedes realizar la sustitución de comandos con uno de los siguientes métodos:

- El carácter comilla simple invertida (`).
- El formato \$ ().

Asegúrate de que cuando escribes el carácter de comilla invertida, no es la comilla simple.

Debes encerrar el comando con comillas invertidas de la siguiente forma:



```
mydir=`pwd`
```

O a la inversa:

```
mydir=$(pwd)
```

Entonces el script podría ser el siguiente:

f

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

🐦

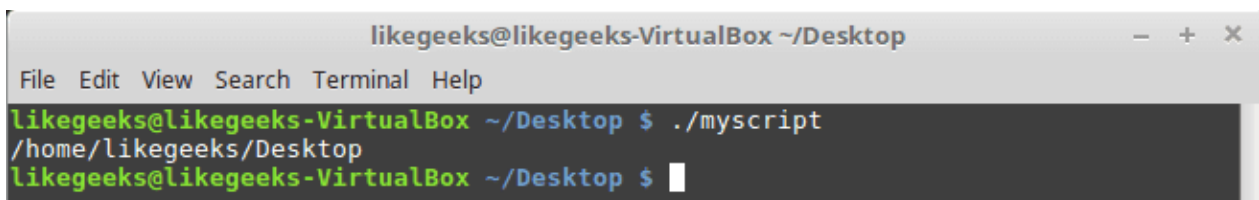
in

```
mydir=$(pwd)
```

p

```
echo $mydir
```

La salida del comando se almacenará en la variable mydir.

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which outputs '/home/likegeeks/Desktop'. The prompt then returns to the shell.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
/home/likegeeks/Desktop
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

## Cálculo matemático

Puedes realizar cálculos matemáticos básicos utilizando la sintaxis `$ ((2 + 2))`:

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

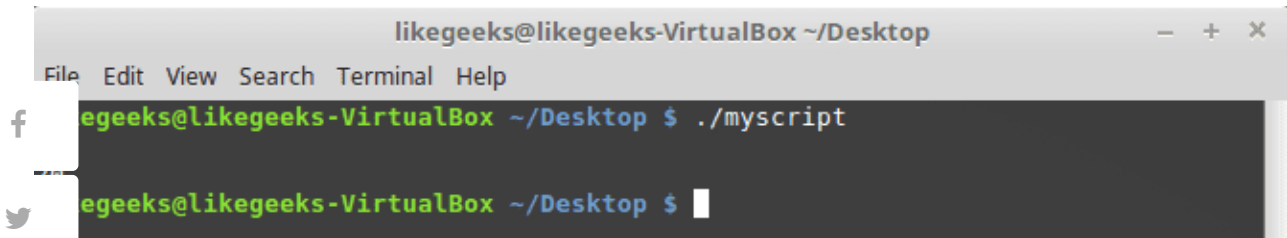
```
var1=$(( 5 + 5 ))
```

```
echo $var1

var2=$(( $var1 * 2 ))

echo $var2
```

Así de fácil.



## Sentencia if-then

Tus scripts de Bash necesitarán condicionales. Si el valor es menor que 10, haz esto, sino haz aquello. Puedes imaginar cualquier lógica que quieras.

La estructura más básica de la sentencia if-then es así:

```
if comand; then
```

```
hacer algo
```

```
fi
```

Aquí hay un ejemplo:

```
#!/bin/bash

if whoami; then
```

```
    echo "It works"

fi
```

Dado que el comando `whoami` devolverá mi usuario, la condición volverá a ser verdadera e imprimirá el mensaje.

Profundicemos y usemos otros comandos que conocemos.

## f Verificar si un usuario existe

Al vez estés buscando un usuario específico en el archivo de usuario `/etc/passwd` y si existe un registro, escríbelo en un mensaje.

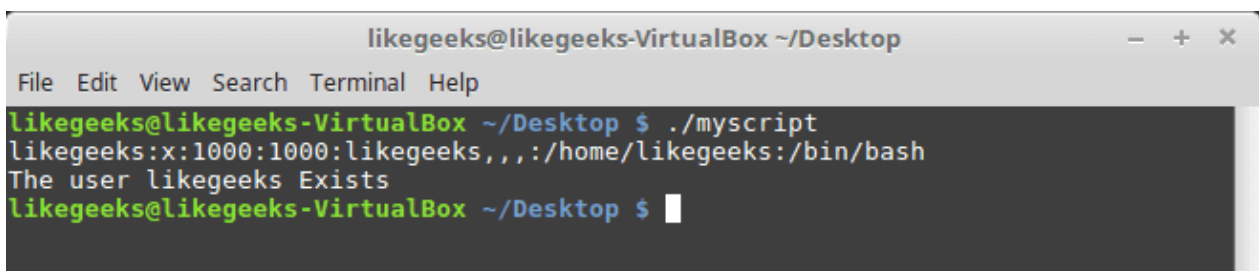
```
#!/bin/bash

user=likegeeks

if grep $user /etc/passwd; then

    echo "No such a user $user"

fi
```



```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
likegeeks:x:1000:1000:likegeeks,,,:/home/likegeeks:/bin/bash
The user likegeeks Exists
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Utilizamos el comando `grep` para buscar un usuario en el archivo `/etc/passwd` file. Puedes revisar nuestro tutorial acerca del [comando grep](#) en este enlace.

Su el usuario existe, el script de bash imprimira este mensaje.

¿Que sucede si un usuario no existe? El script terminara su ejecución sin decirnos que el usuario no existe. Ok, vamos a mejorar esto.

## Sentencia if-then-else

La sentencia if-then-else statement toma la siguiente estructura:

```
if   
do command; then   
do something   
else   
do something else   
fi
```

Si el comando se ejecuta y retorna cero; lo cual significa exito, no ejecutara los comandos despues de la sentencia else, por otro lado, si la sentencia if retorna un numero distinto de zero; lo cual significa que la condición no se cumple, en este caso, el shell ejecutara los comandos despues de la sentencia else.

```
#!/bin/bash

user=anotherUser

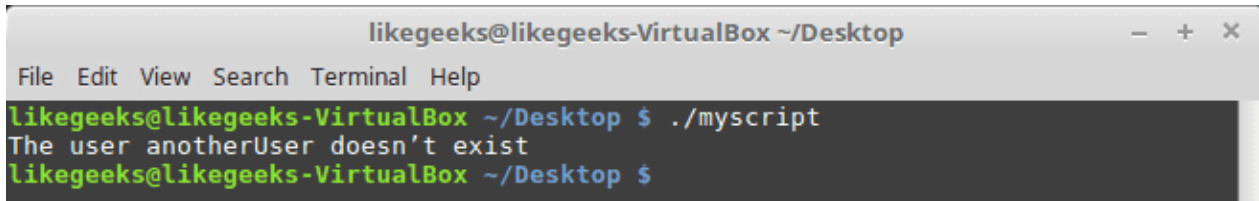
if grep $user /etc/passwd; then

    echo "The user $user Exists"

else
```

```
echo "The user $user doesn't exist"
```

```
fi
```

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows a prompt 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' followed by the command './myscript'. The output of the script is 'The user anotherUser doesn't exist'. The prompt then changes to 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'.

Vamos bien por ahora, continuemos.

f

ahora, que sucede si necesitamos mas sentencias else.

tw

eno, fácil, podemos lograr eso anidando sentencias de la siguiente

in

forma:

p

```
if condition; then
```

```
commands
```

```
elif condition2
```

```
then
```

```
commands
```

```
fi
```

Si el primer comando retorna zero; lo cual significa que se cumple la primera condición, ejecutara estos comandos, sino, si se cumple la segunda condición se ejecutara el bloque elif, por último si ninguno de estos se cumple ejecutara solo los comandos ultimos comandos.

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

```
user=anotherUser
```

```
user=anotheruser
```

```
if grep $user /etc/passwd; then
```

```
    echo "The user $user Exists"
```

```
elif ls /home; then
```

```
    echo "The user doesn't exist"
```

```
fi
```

f

Twitter

in

p

udes imaginar cualquier escenario aqui, tal vez si el usuario no existe, udes crearlo utilizando el comando useradd o hacer cualquier otra sa.

## Combianr pruebas

Puedes combinar multiples pruebas utilizando los comandos AND (&&) u OR (||).

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

```
dir=/home/likegeeks
```

```
name="likegeeks"
```

```
if [ -d $dir ] && [ -n $name ]; then
```

```
    echo "The name exists and the folder $dir exists."
```

```
else
```

```
    echo "One test failed"
```

```
fi
```

Este ejemplo retornara verdadero solo si ambas pruebas se cumplen, sino, fallara.

Ademas, puedes utilizar OR (||) de la misma manera:

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

```
f
```

```
dir=/home/likegeeks
```

```
Ⓣ
```

```
name="likegeeks"
```

```
in
```

```
if [ -d $dir ] || [ -n $name ]; then
```

```
Ⓟ
```

```
    echo "Success!"
```

```
else
```

```
    echo "Both tests failed"
```

```
fi
```

Este ejemplo retornara exito si una o ambas de las pruebas cumplen.

Fallara solo si ambas fallan.

## Comparaciones Numericas

Puedes realizar una comparación numérica entre dos valores numéricos utilizando comprobaciones numéricas de comparación como esta:

number1 -eq number2 Comprueba si number1 es igual a number2.

number1 -ge number2 Comprueba si number1 es más grande o igual number2.

number1 -gt number2 Comprueba si number1 es más grande que number2.

number1 -le number2 Comprueba si number1 es más pequeño o igual number2.

f mber1 -lt number2 Comprueba si number1 es más pequeño que mber2.

in mber1 -ne number2 Comprueba si number1 no es igual a number2.

p mo ejemplo, probaremos uno y el resto será el mismo.

Ten en cuenta que la sentencia de comparación se encuentra entre corchetes, como se muestra.

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

```
num=11
```

```
if [ $num -gt 10]; then
```

```
    echo "$num is bigger than 10"
```

```
else
```

```
    echo "$num is less than 10"
```

```
fi
```



```
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
11 is bigger than 10
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

El número es mayor que 10, por lo que ejecutará la primera instrucción e imprimirá el primer echo.

## Comparaciones de cadenas

f

t

in

p

edes comparar cadenas con una de las siguientes maneras:

ing1 = string2 Comprueba si string1 es idéntico a string2.

ing1 != string2 Comprueba si string1 no es idéntico a string2.

string1 < string2 Comprueba si string1 es menor que string2.

string1 > string2 Comprueba si string1 es mayor que string2.

-n string1 Comprueba si string1 es más mayor que cero.

-z string1 Comprueba si string1 tiene una longitud de cero.

Podemos aplicar la comparación de cadenas en nuestro ejemplo:

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

```
user="likegeeks"
```

```
if [ $user = $USER ]; then
```

```
    echo "The user $user is the current logged in use
```

```
fi
```

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
The user likegeeks is the current logged in user
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Una nota complicada sobre las comparaciones **mayor o menor de cadenas, DEBEN escaparse con la barra inclinada invertida** porque si usas solo el símbolo de mayor que, muestra resultados incorrectos.

f tonces deberías hacerlo así:



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

```
v1=text
```

```
v2="another text"
```

```
if [$v1 \> "$v2" ]; then
```

```
    echo "$v1 is greater than $v2"
```

```
else
```

```
    echo "$v1 is less than $v2"
```

```
fi
```

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
./myscript: line 5: [: too many arguments
text is less than another text
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Se ejecuta pero se muestra esta advertencia:

```
./myscript: line 5: [: too many arguments
```

Para solucionarlo, encierra \$ vals con comillas dobles, forzándolo a permanecer como una cadena:

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

```
v1=text
```

```
f v2="another text"
```

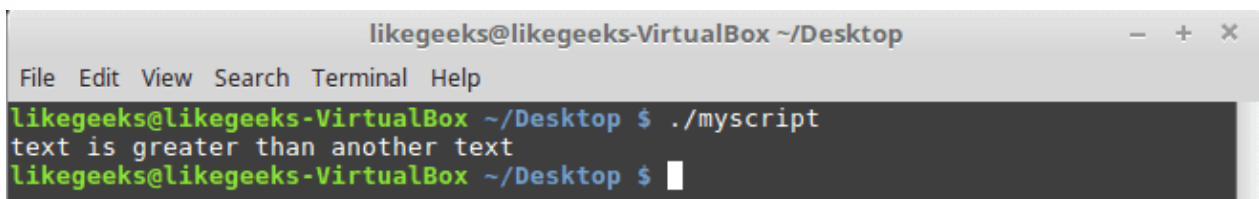
```
🐦 if [$v1 \> "$v2" ]; then
```

```
in     echo "$v1 is greater than $v2"
```

```
📌 else
```

```
     echo "$v1 is less than $v2"
```

```
fi
```

A screenshot of a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command './myscript' being executed, which outputs 'text is greater than another text'. The prompt then returns to the shell.

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File Edit View Search Terminal Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
text is greater than another text
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Una nota importante acerca del **mayor que y menor que** para las comparaciones de cadenas. Verifica el siguiente ejemplo para comprender la diferencia:

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

```
v1=Likegeeks
```

```
v2=likegeeks
```

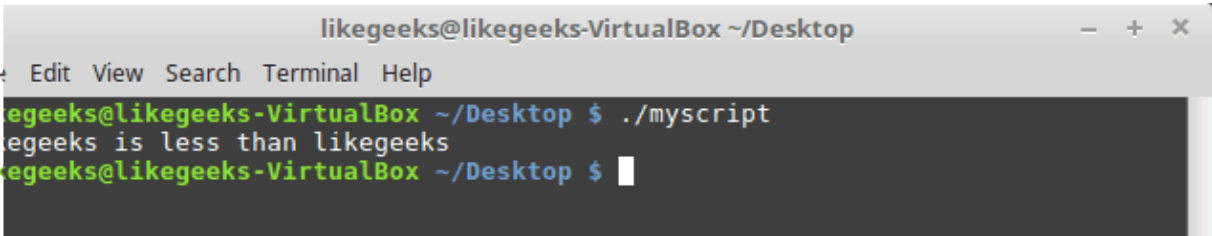
```
if [ $v1 \> $v2 ]; then

    echo "$v1 is greater than $v2"

else

    echo "$v1 is less than $v2"

fi
```

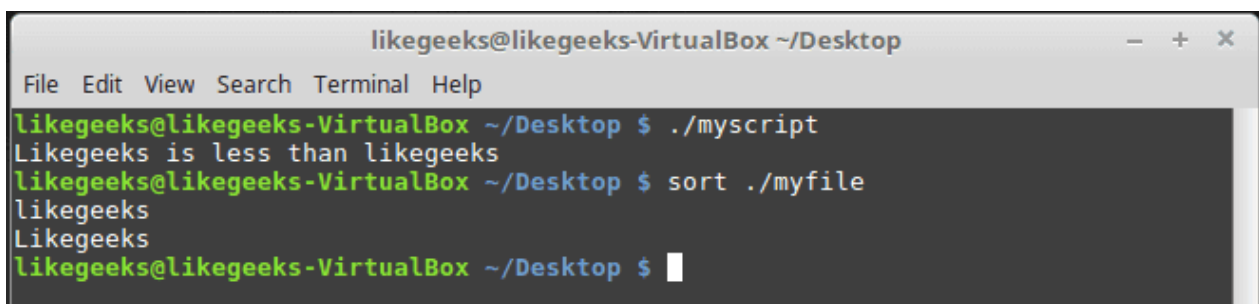
A terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop' with a menu bar (Edit, View, Search, Terminal, Help). The prompt is 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'. The user enters './myscript', and the output is 'likegeeks is less than likegeeks'. The prompt returns to 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' with a cursor.

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
likegeeks is less than likegeeks
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

```
sort myfile
```

likegeeks

Likegeeks

A terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'. The user enters './myscript', and the output is 'Likegeeks is less than likegeeks'. The user then enters 'sort ./myfile', and the output is 'likegeeks' followed by 'Likegeeks'. The prompt returns to 'likegeeks@likegeeks-VirtualBox ~/Desktop \$' with a cursor.

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ ./myscript
Likegeeks is less than likegeeks
likegeeks@likegeeks-VirtualBox ~/Desktop $ sort ./myfile
likegeeks
Likegeeks
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

La condición de prueba considera que las letras minúsculas son más grandes que las letras mayúsculas. A diferencia del comando ordenar que hace lo contrario.

La condición de prueba se basa en el orden ASCII, mientras que el comando de ordenamiento se basa en las órdenes de numeración de la configuración del sistema.

## Comparaciones de archivos

Puede comparar y verificar archivos utilizando los siguientes operadores:

`-d my_file` Comprueba si es una carpeta.

`-e my_file` Comprueba si el archivo está disponible.

`-f my_file` Comprueba si es un archivo.

`-r my_file` Comprueba si es legible.

`-n my_file -nt my_file2` Comprueba si `my_file` es más **nuevo** que `my_file2`.

`-o my_file -ot my_file2` Comprueba si `my_file` es más **viejo** que `my_file2`.

`-O my_file` Comprueba si el propietario del archivo y el usuario registrado coinciden.

`-G mi_archivo` Comprueba si el archivo y el usuario registrado tienen el grupo idéntico.

Como implican, nunca los olvidarás.

Vamos tomar uno de ellos y utilizarlo como ejemplo:

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

```
mydir=/home/likegeeks
```

```
if [ -d $mydir ]; then
```

```
    echo "Directory $mydir exists"
```

```
fi
```

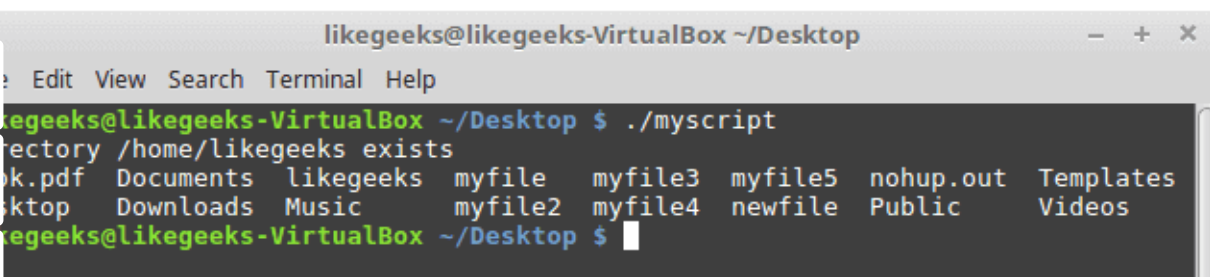
```
cu phnyu11
```

```
ls
```

```
else
```

```
echo "NO such file or directory $mydir"
```

```
fi
```



The screenshot shows a terminal window titled 'likegeeks@likegeeks-VirtualBox ~/Desktop'. The prompt is 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'. The user has entered './myscript'. The output of the script is displayed in two lines: 'Directory /home/likegeeks exists' followed by a directory listing: 'book.pdf Documents likegeeks myfile myfile3 myfile5 nohup.out Templates', 'Desktop Downloads Music myfile2 myfile4 newfile Public Videos'. The prompt returns to 'likegeeks@likegeeks-VirtualBox ~/Desktop \$'.

pero los voyas a escribir todos como un ejemplo. Simplemente escribe la comparación entre los corchetes tal como está y completa tu script normalmente.

Hay algunas otras características if-then avanzadas, pero las estudiaremos en otra publicación.

Eso es todo por ahora. Espero que lo disfrutes y sigas practicando más y más.

Gracias.

 Share on Facebook

 Tweet on Twitter











**Mokhtar Ebrahim**

Fundadora de LikeGeeks. Estoy trabajando como administrador de sistemas Linux desde 2010. Soy responsable de mantener, proteger y solucionar problemas de servidores Linux para múltiples clientes de todo el mundo. Me encanta escribir guiones de shell y Python para automatizar mi trabajo.



## 4 THOUGHTS ON “SCRIPT DE BASH PASO A PASO TUTORIAL (CON EJEMPLOS)”



**Tonydice:**

2019-06-29 a las 5:49 am

Hola.

Tengo una duda ¿Qué hace “-d” y “-n”?

*Responder*



**Mokhtar Ebrahimdice:**

2019-07-01 a las 1:33 pm

-d se utiliza para verificar la existencia del directorio.

-n para comprobar nombres

*Responder*



**Camilodice:**

2020-11-10 a las 1:42 pm

Hola muy bueno el tutorial. tengo una duda y es que estoy migrando un prpgrama shell de unix sco 5.06 de hace como 20 años a la version del Linux Centos 7.5. quiesiera saber si para el caso de ejemplo que te copio es solamente cambiar los comandos o debo programar de ceros el shell.? si es solo cambiar cuales serian los comando en esta version de linux?. muchas gracias por tu ayuda.

echo off

clear

#####

#####

## Programa : in103.sh

## Titulo : Listado de Actividades

## Enlaces : in103.r

## Fecha : 2 de Agosto de 1995

## Autor: Carlos Herrera.

##

```

###
## Parametros :
## $1 Path(path)
## $2 Numero de Copias (s_copias)
## $3 Salida (s_salida)
## $4 Archivo Destino (s_destino)
##
#####
#####
echo
echo
echo " P R O C E S A N D O .... "
echo "
lines 0;
select ac_codigo, ac_descripcion,
xci_descripcion
from t_activ, x_cia
where xci_key = 1;" > $UNIFYTMP/in103.s$$
SQL $UNIFYTMP/in103.s$$ > $UNIFYTMP/in103.dat$$
if test -s $UNIFYTMP/in103.dat$$
then
RPT $1in103.r $UNIFYTMP/in103.dat$$ > $UNIFYTMP/in103.lis$$
case $3 in
P) clear
more $UNIFYTMP/in103.lis$$ ;;
A) mv $UNIFYTMP/in103.lis$$ $4 ;;
I) lpr -#$2 $UNIFYTMP/in103.lis$$ ;;
esac
else
echo "No Existen Registros Seleccionados ....."
fi
sleep 3
rm -f $UNIFYTMP/in103.s$$ $UNIFYTMP/in103.dat$$
$UNIFYTMP/in103.lis$$

```

*Responder*



**Mokhtar Ebrahim**dice:

2020-11-13 a las 7:28 am

Hola,

Lo siento, no conozco los comandos equivalentes para Unix sco.  
Espero que pueda encontrar ayuda lo antes posible.

Saludos cordiales y lo siento de nuevo!



[Responder](#)

## DEJA UNA RESPUESTA

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con \*

### Comentario

f

t

in

p

Nombre \*

Correo electrónico \*

☐ Respuestas a mis comentarios ▼ Notify me of followup comments via e-mail. You can also [subscribe](#) without commenting.

**Publicar el comentario**

Email

**Suscribir**

## ESTE ES MI LIBRO PUBLICADO



## ANUNCIOS

## ANUNCIOS

## ÚLTIMAS PUBLICACIONES

Algoritmo Depth First Search en Python (múltiples ejemplos)

Tutorial de la matrix de correlacion de Python

Tutorial para NumPy where (Con ejemplos)

Salir/Terminar scripts en Python (Ejemplos simples)

20+ Ejemplos de multiplicación de matrices en NumPy

Cinco Cosas Que Debes Considerar Antes de "Desarrollar una APP"

Cifrado Cesar en Python (Tutorial de Cifrado de Texto)

Tutorial de loadtxt de NumPy( cargar datos de los archivos)

## **ANUNCIOS**

## **ANUNCIOS**

## **ARTÍCULOS RELACIONADOS**

15+ ejemplos para el comando yum update

Tutorial del comando find en Linux (con ejemplos)

15+ ejemplos para el comando cURL en Linux

Comando Grep en Linux (con ejemplos)

Entender los niveles de ejecución de Linux de la manera correcta

10+ ejemplos para cerrar un proceso en Linux

15+ Ejemplos para Listar Usuarios en Linux

Recuperar archivos eliminados en Linux (Tutorial para Principiantes)

## **ANUNCIOS**

## **ANUNCIOS**

## **ÚLTIMOS COMENTARIOS**

Alexis en Algoritmo Depth First Search en Python (múltiples ejemplos)

Mokhtar Ebrahim en 31+ ejemplos para el comando sed de Linux en la

manipulación de texto

Zayda en 31+ ejemplos para el comando sed de Linux en la manipulación de texto

Mokhtar Ebrahim en Tutorial Matplotlib (Graficar Gráficos Utilizando pyplot)

Arturo Salinas Calderón en Tutorial Matplotlib (Graficar Gráficos Utilizando pyplot)

**ANUNCIOS**

**ANUNCIOS**

in

**ELEGIDO PARA TI**

31+ ejemplos para el comando sed de Linux en la manipulación de texto

Ejemplos de la GUI de Python (Tutorial de Tkinter)

30 ejemplos para el comando Awk en el procesamiento de texto

Tutorial de Kivy – Construye aplicaciones con interfaces gráficas usando Python

Tutorial de Python SQLite3 (Programación de bases de datos)

Crea tu primer rastreador web con python usando scrapy

Tutorial de PyQt5- Ejemplos de programación con GUI de Python

Scripting de Bash en Linux Parte 3 – Parámetros y opciones

**ANUNCIOS**

**ANUNCIOS**

⬅ Acciones de Portapapeles: Copiar y Pegar en Windows

Scripting de bash Parte2 – ciclos For y While con ejemplos ➡

[Aviso Legal](#) [Política de privacidad](#) [Acerca de](#) [Contacto](#)



f



in

