


[LINUX](#)
[MAC OS](#)
[BSD](#)
[UNIX](#)
[SHELL SCRIPTING](#)
[SERVIDORES WEB](#)
[WINDOWS](#)
[VIRTUALIZACIÓN](#)
[BASES DE DATOS](#)
[SUPERVISIÓN](#)
[META SCG](#)
[Inicio](#) » [Confección de secuencias de comandos](#) » [BASH](#) » Ejemplo de Bash If-Then-Else

SOBRE ANDREAS POMAROLLI



Andreas se graduó de Informática y Bioinformática en la Universidad de Linz. Durante sus estudios, ha estado involucrado en una gran cantidad de proyectos de investigación que abarcan desde la ingeniería de software hasta la ingeniería de datos y al menos la ingeniería web. Su enfoque científico incluye las áreas de ingeniería de software, ingeniería de datos, ingeniería web y gestión de proyectos. Actualmente trabaja como ingeniero de software en el sector de TI, donde se dedica principalmente a proyectos basados en Java, bases de datos y tecnologías web.



Ejemplo de Bash If-Then-Else

Publicado por: [Andreas Pomarolli](#) en [BASH](#) 31 de enero de 2017

Este es un ejemplo de Bash If-Then-Else. A veces necesita especificar diferentes cursos de acción para tomar en un script de shell, dependiendo del éxito o fracaso de un comando. La construcción if le permite especificar tales condiciones.

¿Quieres dominar el scripting de BASH?

Suscríbete a nuestro boletín de noticias y descarga el libro de cocina de programación de **BASH** ahora mismo.

Para ayudarlo a comenzar a usar los scripts de BASH, hemos compilado una guía kick-ass con todos los principales comandos y casos de uso. ¡Además de estudiarlos en línea, puedes descargar el eBook en formato PDF!

Dirección de correo electrónico:

[Regístrate](#)

La siguiente tabla muestra una descripción general de todo el artículo:

Tabla de contenido

1. Introducción
2. Expresiones usadas con if-then-else
 - 1.1 Comprobación de archivos
 - 1.2 Comprobación de argumentos de línea de comando
 - 1.3 Probando la cantidad de argumentos
 - 1.4 Prueba de que existe un archivo



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

[Descárgalo gratis](#)

HOJA INFORMATIVA

113,332 expertos ya están actualizando semanales y gratuitos!

Únase a ellos ahora acceso exclusivo a las en el mundo de administrador sistema operativo, así como la programación integrada, sistemas de lenguaje de scripting y otras relacionadas.

Dirección de correo electrónico:

[Regístrate](#)

ÚNETE A NOSOTROS



Con **1.500** únicos miembros

programa de socios de **SCG** . IT

```
1 if TEST-COMMANDS; then CONSEQUENT-COMMANDS; else ALTERNATE-CONSEQUENT-COMMANDS; fi
```

La

TEST-COMMAND

lista se ejecuta, y si su estado de retorno es cero, la

CONSEQUENT-COMMANDS

lista se ejecuta. El estado de retorno es el estado de salida del último comando ejecutado, o cero si ninguna condición es verdadera.

A

TEST-COMMAND

menudo implica pruebas numéricas o de comparación de cadenas, pero también puede ser cualquier comando que devuelva un estado de cero cuando tenga éxito y algún otro estado cuando falle. Las expresiones unarias a menudo se utilizan para examinar el estado de un archivo. Si el argumento FILE para uno de los primarios tiene el formato

/dev/fd/N

, entonces se comprueba el descriptor de archivo "N". stdin, stdout y stderr y sus respectivas descripciones de archivos también se pueden usar para las pruebas.

La siguiente tabla contiene una descripción general de los llamados "primarios" que componen el

TEST-COMMAND

comando o la lista de comandos. Estas primarias se ponen entre corchetes para indicar la prueba de una expresión condicional.

Primario	Sentido
[-un archivo]	Verdadero si el ARCHIVO existe.
[-b ARCHIVO]	Es verdadero si FILE existe y es un archivo especial de bloque.
[-c ARCHIVO]	Es verdadero si FILE existe y es un archivo especial de caracteres.
[-d ARCHIVO]	Verdadero si ARCHIVO existe y es un directorio.
[-e ARCHIVO]	Verdadero si el ARCHIVO existe.
[-f ARCHIVO]	Verdadero si ARCHIVO existe y es un archivo regular.
[-g ARCHIVO]	Es verdadero si FILE existe y su bit SGID está configurado.
[-h ARCHIVO]	Verdadero si ARCHIVO existe y es un enlace simbólico.
[-k ARCHIVO]	Verdadero si ARCHIVO existe y su bit adhesivo está configurado.
[-p ARCHIVO]	Verdadero si FILE existe y es una tubería con nombre (FIFO).
[-r ARCHIVO]	Verdadero si ARCHIVO existe y es legible.
[-s FILE]	Verdadero si ARCHIVO existe y tiene un tamaño mayor que cero.
[-t FD]	Verdadero si el descriptor de archivo FD está abierto y se refiere a un terminal.
[-u ARCHIVO]	Es verdadero si FILE existe y su bit SUID (set user ID) está configurado.
[-w ARCHIVO]	Verdadero si ARCHIVO existe y puede escribirse.
[-x ARCHIVO]	Verdadero si FILE existe y es ejecutable.
[-O ARCHIVO]	Verdadero si el ARCHIVO existe y es propiedad de la identificación de usuario efectiva.
[-G ARCHIVO]	Verdadero si el ARCHIVO existe y es propiedad de la identificación de grupo efectiva.
[-L ARCHIVO]	Verdadero si ARCHIVO existe y es un enlace simbólico.
[-N ARCHIVO]	Verdadero si el ARCHIVO existe y ha sido modificado desde la última vez que se leyó.
[-S FILE]	Verdadero si FILE existe y es un socket.
[FILE1 -nt FILE2]	Verdadero si FILE1 se ha cambiado más recientemente que FILE2, o si FILE1 existe y FILE2 no.
[FILE1 -ot FILE2]	Verdadero si FILE1 es anterior a FILE2 o si FILE2 existe y FILE1 no.
[ARCHIVO1 -ef ARCHIVO2]	Verdadero si FILE1 y FILE2 se refieren al mismo dispositivo y los mismos números de inodo.
[-o OPTIONNAME]	Verdadero si la opción de shell "OPTIONNAME" está habilitada.
[-z STRING]	Es cierto de la longitud si "STRING" es cero.



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

[STRING1> STRING2]	Es verdadero si "STRING1" ordena después de "STRING2" lexicográficamente en la configuración regional actual.
[ARG1 OP ARG2]	"OP" es uno de los -eq, -ne, -lt, -le, -gt o -ge. Estos operadores binarios aritméticos devuelven verdadero si "ARG1" es igual a, no igual a, menor que, menor que o igual a, mayor que, o mayor que o igual a "ARG2", respectivamente. "ARG1" y "ARG2" son enteros.

Las expresiones se pueden combinar utilizando los siguientes operadores, enumerados en orden decreciente de precedencia:

Operación	Efecto
[! EXPR]	Verdadero si EXPR es falso.
[(EXPR)]	Devuelve el valor de EXPR. Esto puede usarse para anular la precedencia normal de los operadores.
[EXPR1 -a EXPR2]	Verdadero si tanto EXPR1 como EXPR2 son verdaderos.
[EXPR1 -o EXPR2]	Verdadero si EXPR1 o EXPR2 son verdaderos.

La `[[` (o prueba) incorporada evalúa expresiones condicionales usando un conjunto de reglas basadas en el número de argumentos. Se puede encontrar más información sobre este tema en la documentación de Bash.

Al igual que si se cierra con `fi`, el corchete de apertura debe cerrarse después de que se hayan enumerado las condiciones.

La

```
CONSEQUENT-COMMANDS
```

lista que sigue a la declaración de `then` puede ser cualquier comando de UNIX válido, cualquier programa ejecutable, cualquier script de shell ejecutable o cualquier declaración de shell, con la excepción de la `fi` de cierre.

La

```
ALTERNATE-CONSEQUENT-COMMANDS
```

lista que sigue a la declaración `else` puede contener cualquier comando de estilo UNIX que devuelva un estado de salida.

Es importante recordar que las sentencias `then` y `fi` se consideran separadas en el shell.

Por lo tanto, cuando se emiten en la línea de comando, están separados por un punto y coma. En una secuencia de comandos, las diferentes partes de la instrucción `if` suelen estar bien separadas.

2. Expresiones usadas con if-then-else

1.1 Comprobación de archivos

El siguiente archivo representa un archivo de texto simple para algunas pruebas:

Test1.txt

```
1 1:Andreas
2 2:Marcus
3 3:Tom
4 4:Steve
```

Este es el constructo que se debe usar para tomar un curso de acción si los comandos `if` son verídicos y otro si es falso.

ifelse1.sh

```
01 echo "Checking..."
02 grep Hardy Test1.txt
03 if [ $? -ne 0 ]
04 then
05     echo 'Hardy not found in File Test1.txt!'
06 else
07     echo 'Hardy found in File Test1.txt!'
08 fi
09 echo
10 echo "...done."
```

```
Test
Test:>
Test:>
Test:>
Test:>
Test:>. ifelse1.sh
Checking...
Hardy not found in File Test1.txt!
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

```

02 grep Tom Test1.txt
03 if [ $? -ne 0 ]
04 then
05     echo 'Tom not found in File Test1.txt!'
06 else
07     echo 'Tom found in File Test1.txt!'
08 fi
09 echo
10 echo "...done."

```

Otro ejemplo If-Then-Else sobre buscar archivos

1.2 Comprobación de argumentos de línea de comando

En lugar de establecer una variable y luego ejecutar una secuencia de comandos, con frecuencia es más elegante colocar los valores para las variables en la línea de comando.

Usamos los parámetros posicionales

\$1, \$2, ..., \$N

para este propósito.

\$#

se refiere a la cantidad de argumentos de línea de comando.

\$0

se refiere al nombre del guion.

El siguiente es un ejemplo simple:

[ifelse3.sh](#)

```

01 echo "Checking..."
02 grep $1 Test1.txt
03 if [ $? -ne 0 ]
04 then
05     echo $1 'not found in File Test1.txt!'
06 else
07     echo $1 'found in File Test1.txt!'
08 fi
09 echo
10 echo "...done."

```

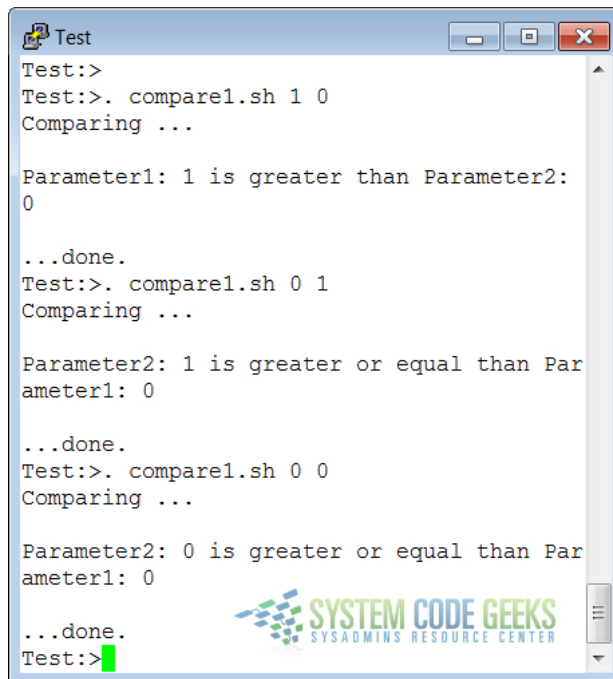


El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

```
04 echo "Comparing ..."  
05 echo  
06 if [ $value1 -le $value2 ]  
07 then  
08     echo "Parameter2:" $value2 "is greater or equal than Parameter1:" $value1  
09 else  
10     echo "Parameter1:" $value1 "is greater than Parameter2:" $value2  
11 fi  
12 echo  
13 echo "...done."
```



```
Test:  
Test:>. compare1.sh 1 0  
Comparing ...  
  
Parameter1: 1 is greater than Parameter2:  
0  
  
...done.  
Test:>. compare1.sh 0 1  
Comparing ...  
  
Parameter2: 1 is greater or equal than Par  
ameter1: 0  
  
...done.  
Test:>. compare1.sh 0 0  
Comparing ...  
  
Parameter2: 0 is greater or equal than Par  
ameter1: 0  
  
...done.  
Test:>
```

Un ejemplo If-Then-Else sobre la comparación de la entrada

1.3 Probando la cantidad de argumentos

El siguiente ejemplo muestra cómo cambiar el script anterior para que imprima un mensaje si se dan más o menos de 2 argumentos:

[compare2.sh](#)

```
01 #!/bin/bash  
02 if [ ! $# == 2 ]  
03 then  
04     echo "Usage: Command Value1 Value2"  
05     return 1  
06 fi  
07  
08 value1="$1"  
09 value2="$2"  
10 echo "Comparing ..."  
11 echo  
12 if [ $value1 -le $value2 ]  
13 then  
14     echo "Parameter2:" $value2 "is greater or equal than Parameter1:" $value1  
15 else  
16     echo "Parameter1:" $value1 "is greater than Parameter2:" $value2  
17 fi  
18 echo  
19 echo "...done."
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

```

Test:>
Test:>. compare2.sh 1 0
Comparing ...

Parameter1: 1 is greater than Parameter2:
0

...done.
Test:>. compare2.sh 1 2
Comparing ...

Parameter2: 2 is greater or equal than Par
ameter1: 1

...done.
Test:>. compare2.sh 1
Usage: Command Value1 Value2
Test:>
Test:>. compare2.sh 1 1
Comparing ...

Parameter2: 1 is greater or equal than Par
ameter1: 1

...done.
Test:>

```



Otro ejemplo If-Then-Else sobre la comparación de la entrada

El primer argumento se conoce como

\$1

, el segundo como

\$2

y así sucesivamente. El número total de argumentos se almacena en

\$#

1.4 Prueba de que existe un archivo

Esta prueba se realiza en muchos guiones, porque no sirve de nada comenzar muchos programas si sabes que no van a funcionar:

[fileexists.sh](#)

```

01 #!/bin/bash
02 # This script gives information about a file.
03 FILENAME="$1"
04 echo "Properties for $FILENAME:"
05 if [ -f $FILENAME ]
06 then
07     echo "Size is $(ls -lh $FILENAME | awk '{ print $5 }')"
08     echo "Type is $(file $FILENAME | cut -d":" -f2 -)"
09     echo "Inode number is $(ls -li $FILENAME | cut -d" " -f1 -)"
10 else
11     echo "File does not exist."
12 fi

```

```

Test
Test:>
Test:>
Test:>
Test:>. fileexists.sh Test1.txt
Properties for Test1.txt:
Size is 25
Type is ASCII text
Inode number is 5284205
Test:>
Test:>

```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

tenga en cuenta que se hace referencia al archivo comando una variable; en este caso, es el primer argumento del guión. Alternativamente, cuando no se dan argumentos, las ubicaciones de los archivos generalmente se almacenan en variables al comienzo de un script, y su contenido se refiere a estas variables. Por lo tanto, cuando desee cambiar el nombre de un archivo en un script, solo necesita hacerlo una vez.

2. Construcciones If-Then-Elif-Else

Esta es la forma completa de la declaración if:

```
01 if TEST-COMMANDS;
02 then
03     CONSEQUENT-COMMANDS;
04 elif
05     MORE-TEST-COMMANDS;
06 then
07     MORE-CONSEQUENT-COMMANDS;
08 else
09     ALTERNATE-CONSEQUENT-COMMANDS;
10 fi
```

La

TEST-COMMANDS

lista se ejecuta, y si su estado de retorno es cero, la

CONSEQUENT-COMMANDS

lista se ejecuta.

Si

TEST-COMMANDS

devuelve un estado distinto de cero, cada lista elif se ejecuta sucesivamente, y si su estado de salida es cero,

MORE-CONSEQUENT-COMMANDS

se ejecuta el correspondiente y el comando finaliza.

Si a lo demás le sigue una

ALTERNATE-CONSEQUENT-COMMANDS

lista, y el comando final en la cláusula final if o elif tiene un estado de salida distinto de cero, entonces

ALTERNATE-CONSEQUENT-COMMANDS

se ejecuta.

El estado de retorno es el estado de salida del último comando ejecutado, o cero si ninguna condición es verdadera.

Aquí hay un ejemplo simple:

[compare3.sh](#)

```
01 #!/bin/bash
02 value1="$1"
03 value2="$2"
04 echo "Comparing ..."
05 echo
06 if [ $value1 -eq $value2 ]
07 then
08     echo "Parameter2:" $value2 "is equal with Parameter1:" $value1
09 elif [ $value1 -gt $value2 ]
10 then
11     echo "Parameter1:" $value1 "is greater than Parameter2:" $value2
12 else
13     echo "Parameter1:" $value1 "is smaller than Parameter2:" $value2
14 fi
15 echo
16 echo "...done."
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

```

Test:>. compare3.sh 1 2
Comparing ...

Parameter1: 1 is smaller than Parameter2:
2

...done.
Test:>. compare3.sh 1 1
Comparing ...

Parameter2: 1 is equal with Parameter1: 1

...done.
Test:>
Test:>. compare3.sh 2 1
Comparing ...

Parameter1: 2 is greater than Parameter2:
1

...done.
Test:>

```

Otro ejemplo If-Then-Else sobre la comparación de la entrada

3. Anidado Si las declaraciones

Dentro de la declaración if, puede usar otra instrucción if. Puede usar tantos niveles de if anidados como pueda lógicamente administrar.

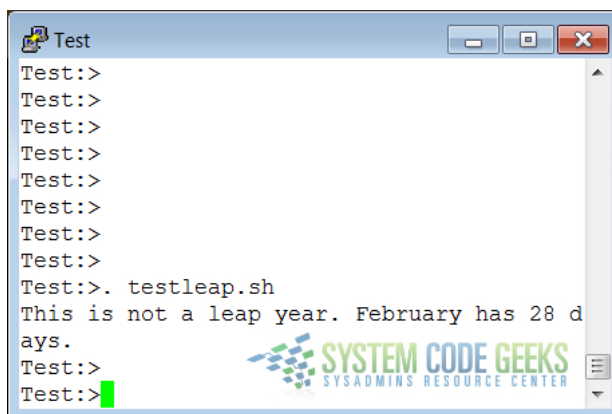
Este es un ejemplo de prueba de años bisiestos:

[testleap.sh](#)

```

01 #!/bin/bash
02 # This script will test if we're in a leap year or not.
03 year=`date +%Y`
04
05 if [ `${year} % 400` -eq "0" ]
06 then
07     echo "This is a leap year. February has 29 days."
08 elif [ `${year} % 4` -eq 0 ]
09 then
10     if [ `${year} % 100` -ne 0 ]
11     then
12         echo "This is a leap year, February has 29 days."
13     else
14         echo "This is not a leap year. February has 28 days."
15     fi
16 else
17     echo "This is not a leap year. February has 28 days."
18 fi

```



```

Test:
Test:
Test:
Test:
Test:
Test:
Test:
Test:
Test:
Test:>. testleap.sh
This is not a leap year. February has 28 d
ays.
Test:
Test:

```

Un ejemplo de If-Then-Else sobre comprobación, si el año actual es bisiesto

En este ejemplo, usamos los corchetes dobles para probar una expresión aritmética. Esto es equivalente a la declaración de let.

Te quedarás atrapado usando corchetes aquí, si intentas algo así

```
`${year} % 400`
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

compare4.sh

```

01 #!/bin/bash
02 value1="$1"
03 value2="$2"
04 echo "Comparing ..."
05 echo
06 if [ $value1 -lt $value2 ] || [ $value1 -eq $value2 ]
07 then
08     echo "Parameter1:" $value2 "is lower or equal than Parameter1:" $value1
09 else
10     echo "Parameter1:" $value1 "is greater than Parameter2:" $value2
11 fi
12 echo
13 echo "...done."

```

5. Uso de la declaración de salida y If

El estado de salida se usa con más frecuencia si la entrada solicitada por el usuario es incorrecta, si una declaración no se ejecutó correctamente o si se produjo algún otro error.

La declaración de salida toma un argumento opcional. Este argumento es el código de estado de salida entero, que se devuelve al padre y se almacena en la

```
$?
```

variable.

Un argumento cero significa que la secuencia de comandos se ejecutó correctamente. Cualquier otro valor puede ser utilizado por los programadores para devolver diferentes mensajes al padre, de modo que se puedan tomar diferentes acciones según la falla o el éxito del proceso hijo. Si no se da ningún argumento al comando de salida, el shell padre usa el valor actual de la

```
$?
```

variable.

A continuación se muestra un ejemplo con un guión penguin.sh ligeramente adaptada, que envía su estado de salida de nuevo a los padres,

```
feed.sh
```

```
:
```

penguin.sh

```

01 #!/bin/bash
02 # This script lets you present different menus to Tux. He will only be happy
03 # when given a fish. We've also added a dolphin and (presumably) a camel.
04 if [ "$menu" == "fish" ]
05 then
06     if [ "$animal" == "penguin" ]
07     then
08         echo "HMMMMMM fish... Tux happy!"
09     elif [ "$animal" == "dolphin" ]
10     then
11         echo "Pweetpeettreetpeterdepweet!"
12     else
13         echo "*prrrrrrrt*"
14     fi
15 else
16     if [ "$animal" == "penguin" ]
17     then
18         echo "Tux don't like that. Tux wants fish!"
19         return 1
20     elif [ "$animal" == "dolphin" ]
21     then
22         echo "Pweepwishpeeterdepweet!"
23         return 2
24     else
25         echo "Will you read this sign?!"
26         return 3
27     fi
28 fi

```

Este script se invoca en el siguiente, que por lo tanto exporta sus variables menu y animal:

feed.sh

```

01 #!/bin/bash
02 # This script acts upon the exit status given by penguin.sh
03 export menu="$1"
04 export animal="$2"
05 feed="penguin.sh"
06 $feed $menu $animal
07
08 case $? in
09 1)
10     echo "Guard: You'd better give'm a fish... less they get violent..."

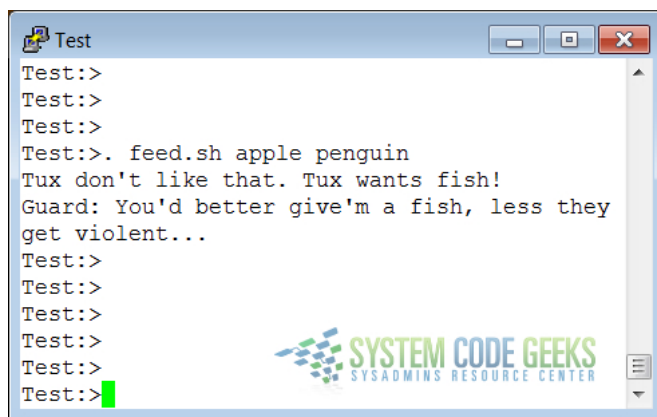
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis



Un ejemplo de If-Then-Else sobre verificar la comida de un animal dado

Como puede ver, los códigos de estado de salida se pueden elegir libremente. Los comandos existentes generalmente tienen una serie de códigos definidos.

6. Uso de declaraciones de casos

Anidado si las declaraciones pueden ser agradables, pero tan pronto como se enfrentan con un par de posibles acciones diferentes, tienden a confundir. Para los condicionales más complejos, use la sintaxis del caso:

```

1
2
3
4         ...
5 esac      CASE) COMMAND-LIST;;

```

Cada caso es una expresión que coincide con un patrón. Los comandos en el

COMMAND-LIST

para la primera coincidencia se ejecutan. El símbolo "|" se usa para separar múltiples patrones, y el operador ")" termina una lista de patrones.

Cada caso más sus comandos correspondientes se llaman cláusula. Cada cláusula debe terminarse con ";".

Cada declaración de caso finaliza con la declaración esac.

Aquí hay un ejemplo simple:

testcase.sh

```
01 case "$1" in
02 start)
03   echo "Processes will be started..."
04   ;;
05 stop)
06   echo "Processes will be stopped..."
07   ;;
08 status)
09   echo "Current State of the Processes is..."
10   ;;
11 restart)
12   echo "Processes will be restarted..."
13   ;;
14   *)
15   echo $"Usage: Script {start|stop|restart|status}"
16   esac
```



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

Descárgalo gratis

```
Test:~>. testcase.sh start
Processes will be started...
Test:~>
Test:~>. testcase.sh stop
Processes will be stopped...
Test:~>
Test:~>. testcase.sh restart
Processes will be restarted...
Test:~>
Test:~>. testcase.sh status
Current State of the Processes is...
Test:~>
Test:~>. testcase.sh test
Usage: Script {start|stop|restart|status}
Test:~>
Test:~>
Test:~>
```

Un ejemplo de menú If-Then-Else

Una aplicación clásica de tal ejemplo es un script. Este tipo de scripts a menudo hacen uso de declaraciones de casos para iniciar, detener y consultar servicios del sistema.

7. Resumen

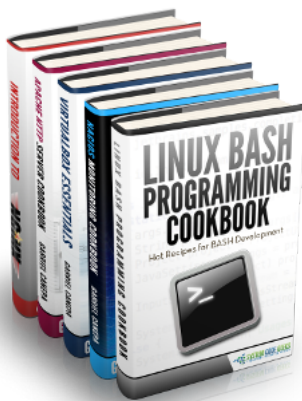
En este capítulo, aprendimos a construir condiciones complejas en nuestros scripts para que se puedan llevar a cabo diferentes acciones ante el éxito o el fracaso de un comando. Las acciones se pueden determinar usando la instrucción `if`.

Las definiciones más complejas de las condiciones se suelen incluir en una declaración de caso.

Tras una prueba de condición exitosa, la secuencia de comandos puede informar explícitamente al padre utilizando el estado de salida 0.

En caso de falla, cualquier otro número puede ser devuelto. En función del código de retorno, el programa principal puede tomar la acción adecuada.

¿Desea saber cómo desarrollar sus habilidades para convertirse en un administrador de sistemas Rockstar?



¡Suscríbete a nuestro boletín para comenzar a rocking
¡Para comenzar, te ofrecemos nuestros eBooks más vendidos
ahora mismo!

- GRATIS!
1. Introducción a NGINX
 2. Libro de cocina del servidor Apache HTTP
 3. VirtualBox Essentials
 4. Nagios Monitoring Cookbook
 5. Linux BASH Programming Cookbook
 6. Tutorial de la base de datos PostgreSQL

y muchos más

Dirección de correo electrónico:

Etiquetado con:



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

¡Sé el primero en comentar!Notificar de 

Start the discussion

BASE DE CONOCIMIENTOS

[Cursos](#)[Minilibros](#)

LA RED CODE GEEKS

[.NET Code Geeks](#)[Java Code Geeks](#)[Geeks del Código del Sistema](#)[Geeks de código web](#)

SALÓN DE LA FAMA

[Tutorial de configuración de Apache \(CentOS / Ubuntu Linux\)](#)[Apache habilita el tutorial SSL / TLS](#)[Ejemplo de mod_rewrite de Apache: Redireccionamiento y reescritura de URL](#)[Ejemplo de configuración de host virtual basado en nombre de Apache \(y establecimiento de límites de uso de ancho de banda\)](#)[Ejemplo de reescritura de URL de Apache: Reescritura y redirección de URL con mod_rewrite](#)[Cómo instalar el servidor web Apache \(instalación CentOS / Ubuntu Linux\)](#)[Linux sed Ejemplos](#)[Guía de configuración de Nginx \(Ubuntu 12.04 LTS\)](#)[Guía de configuración Nginx SSL \(Ubuntu 12.04 LTS\)](#)[Guía de proxying de Nginx Websockets \(Ubuntu 12.04 LTS\)](#)

ACERCA DE GEEKS DEL CÓDIGO DEL SISTEMA

SCGs (System Code Geeks) es una comunidad en línea independiente e el mejor centro de recursos para desarrolladores de sistemas operativos: arquitecto técnico, al líder del equipo técnico (desarrollador senior), al c proyecto y a los desarrolladores junior por igual. Los SCG sirven al dese sistema operativo, al ingeniero del sistema operativo y a las comunidad noticias diarias escritas por expertos de dominio, artículos, tutoriales, re anuncios, fragmentos de código y proyectos de código abierto.

RENUNCIA

Todas las marcas comerciales y marcas registradas que aparecen en Sy son propiedad de sus respectivos dueños.

Código del sistema Geeks y todo el contenido copyright © 2010-2017, Exelixis Media PC | Términos de uso | Política de privacidad | Contacto



El libro de cocina de programación Linux BASH

- Use el shell y todos los comandos BASH de Linux
- Aprende las palabras clave, la sintaxis y otras características básicas
- Configuración de permisos y administración de usuarios

[Descárgalo gratis](#)