Bourne Again Shell

Parte II: Comandos Estructurados - Selecciones

Dr. Mario M. Berón

Universidad Nacional de San Luis

if-then

Formato

```
if comando
then
 comandos
fi
```

Funcionamiento

El comando **if** funciona de la siguiente manera:

- Se ejecuta el comando que se encuentra definido en la línea del **if**.
- Si el estado de salida de ese comando es cero (o sea el comando se ejecutó exitosamente) entonces se ejecutan los comandos listados en la parte **then**. En otro caso dichos comandos no se ejecutan.

if-then

Ejemplo

```
#!/bin/bash
# Prueba de la sentencia if
if date
then
 echo "Funcionó"
fi


$ ./test1
Sat Sep 29 14:09:24 EDT 2007
Funcionó
$
```

if-then

Ejemplo

```
#!/bin/bash
# Prueba de un comando erróneo
if asdfg
then
 echo "No funciona"
fi
echo "Estamos fuera de la sentencia if"


$ ./test2
./test2: line 3: asdfg: command not found
Estamos fuera de la sentencia if
$
```

if-then

Bloques de Comandos

La sección **then** puede tener más de un comando. El intérprete trata a esos comandos como un bloque. Dicho bloque se ejecuta si el comando que se encuentra en la línea de la sentencia **if** retorna como estado de salida un cero o saltándolas en otro caso.

if-then

Ejemplo

```
#!/bin/bash
# Prueba de múltiples comandos en la sección del "then"
testuser=rich
if grep $testuser /etc/passwd
then
echo Los archivos de bash para el usuario $testuser son:
ls -a /home/$testuser/.b*
fi

$ ./test3
rich:x:500:500:Rich Blum:/home/rich:/bin/bash
Los archivos de bash para el usuario rich son:
/home/rich/.bash history /home/rich/.bash profile
/home/rich/.bash logout /home/rich/.bashrc
$
```

if-then

Ejemplo

```
#!/bin/bash
# Prueba de múltiples comandos en la sección del "then"
testuser=rich
if grep $testuser /etc/passwd
then
echo Los archivos de bash para el usuario $testuser son:
ls -a /home/$testuser/.b*
fi

$ ./test3
$
```

if-then

Formato Alternativo del if-then

Cuando se coloca un punto y coma al final del comando a evaluar (condición) es posible incluir el **then** en la misma línea que el **if**.

```
if comando; then
   comandos
fi
```

if-then-else

Formato

```
if comando
then
 comandos
else
 comandos
fi
```

Funcionamiento

- El comando que se encuentra en la línea de la sentencia **if** se ejecuta.
- Si el estado de la salida es cero se ejecuta el bloque de sentencias asociadas con el **then**. En otro caso se ejecuta el bloque de sentencias asociados con el **else**.

if-then-else

Ejemplo

```
#!/bin/bash
# Prueba del else
testuser=badtest
if grep $testuser /etc/passwd
then
 echo Los archivos para el usuario $testuser son:
 ls -a /home/$testuser/.b*
else
 echo "El nombre de usuario $testuser no existe en el sistema"
fi
$ ./test4
El nombre de usuario badtest no existe en el sistema
$
```

if-then-elif

Formato

```
if comando1
then
 comandos
elif comando2
then
 comandos
fi
```

Funcionamiento

if-then-elif

Formato

```
if comando1
then
 Conjunto de comandos 1
elif comando2
then
 Conjunto de comandos 2
elif comando3
then
 Conjunto de comandos 3
fi
```

Nota

Puede haber tantos if anidados como el usuario lo desee.

if-then-elif

El Comando test

Provee una forma de evaluar condiciones. Si la condición evaluada es verdadera el comando test retorna como resultado un cero. Si la condición evaluada es falsa este comando retorna como resultado un uno.

Formato

```
test condición
```

if-then-elif

test con if

```
if test condición
then
 comandos
fi
```

test con if Formato Alternativo

```
if [ condición ]
then
 comandos
fi
```

Los corchetes tienen que tener un espacio antes y después de la condición.

Condiciones

Condiciones

El comando test puede evaluar:

1. Comparaciones Numéricas.
2. Comparaciones de String.
3. Comparaciones de Archivos.

Condiciones

Comparaciones Numéricas

1. n1 -eq n2. Verifica si n1 es igual a n2.
2. n1 -ge n2. Verfica si n1 es mayor o igual que n2.
3. n1 -gt n2. Verifica si n1 es mayor que n2.
4. n1 -le n2. Verifica si n1 es menor o igual que n2.
5. n1 -lt n2. Verifica si n1 es menor que n2.
6. n1 -ne n2. Verifica si n1 no es igual que n2.

Condiciones

Ejemplo

```
#!/bin/bash
# Uso de compraraciones numéricas
val1=10
val2=11
if [ $val1 -gt 5 ]
then
 echo "El valor de test $val1 es mayor que 5"
fi
if [ $val1 -eq $val2 ]
then
 echo "Los valores son iguales"
else
 echo "Los valores son diferentes"
fi
```

Condiciones

Comparaciones de String

1. str1 = str2. Verifica si str1 es la misma cadena que str2.
2. str1 != str2. Verifica si str1 no es la misma cadena que str2.
3. str1 < str2. Verifica si str1 es menor que str2.
4. str1 > str2. Verifica si str1 es mayor que str2.
5. -n str1. Verifica si str1 tiene una longitud mayor que cero.
6. -z str1. Verifica si str1 tiene una longitud de cero.

Condiciones

Comparaciones de String-Ejemplo

```
#!/bin/bash
# testing string equality
testuser=baduser
if [ $USER != $testuser ]
then
 echo "No es $testuser"
else
 echo "Bienvenido $testuser"
fi

$ ./test8
No es baduser
$
```

Condiciones

Orden de los Strings

1. Los símbolos mayor y menor deben ser precedidos por un \ para evitar ser interpretados como una redirección.
2. Los símbolos mayor y menor no son los mismos que los utilizados por el comando *sort*.

Condiciones

El Problema de la Redirección

```
#!/bin/bash
# Mal uso de compraciones de string
val1=baseball
val2=hockey
if [ $val1 \> $val2 ]
then
 echo "$val1 es mayor que $val2"
else
 echo "$val1 es menor que $val2"
fi
$ ./badtest
baseball es mayor que  hockey
$ ls -l hockey
-rw-r--r-- 1 rich rich 0 Sep 30 19:08 hockey
```

Condiciones

El Problema del Sort

```
# Prueba del orden de strings
val1=Testing
val2=testing
if [ $val1 \> $val2 ]
then
 echo "$val1 es mayor que $val2"
else
 echo "$val1 es mayor que $val2"
fi
$ ./test9b
Testing es menor que testing
$ sort testfile
testing
Testing
```

Condiciones

Comparaciones de Archivos Posibles

A continuación se muestran las posibles comparaciones que se pueden realizar:

- -d file. Verfica si el archivo *file* existe y si es un directorio.
- -e file. Verfica si el archivo *file*.
- -f file. Verfica si el archivo *file* y es un archivo.
- -r file. Verfica si el archivo *file* and es legible.
- -s file. Verfica si el archivo *file* and no está vacío.
- -w file. Verfica si el archivo *file* and se puede escribir.

Condiciones

Comparaciones de Archivos Posibles

A continuación se muestran las posibles comparaciones que se pueden realizar:

- -x file. Verfica si el archivo *file* existe y es ejecutable.
- -O file. Verfica si el archivo *file* existe y es propiedad del usuario corriente.
- -G file. Verfica si el archivo *file* existe y el grupo por defecto es el mismo que el grupo del usuario corriente.
- file1 -nt file2. Verfica si el archivo *file1* es más nuevo que *file2*.
- file1 -ot file2. Verfica si el archivo *file1* más antiguo que *file2*.

Condiciones

Ejemplo

```
#!/bin/bash
if [ -d $HOME ]
then
 echo "Su directorio HOME existe"
 cd $HOME
 ls -a
else
 echo "Hay un problema con su directorio HOME"
fi
```

Condiciones Compuestas

Lógica Booleana

La sentencia if permite el uso de lógica booleana para combinar tests.

```
[ condición1 ] && [ condición2 ]
[ condición1 ] || [ condición2 ]
```

Condiciones Compuestas

Ejemplo

```
if [ -d $HOME ] && [ -w $HOME/testing ]
then
 echo "El archivo existe y puede escribirlo"
else
 echo "No puedo escribir el archivo"
fi
```

Características Avanzadas

Características Avanzadas

1. Doble parentesis para expresiones matemáticas.
2. Doble corchetes para funciones de manipulación de string avanzadas.

Características Avanzadas

Características Avanzadas: Doble Paréntesis

((expresión))

Donde *expresión* puede ser cualquier asignación matemática o expresión de comparación.

Características Avanzadas

Características Avanzadas: Doble Paréntesis

1. val++ post-incremento
2. val– post-decremento
3. ++val pre-incremento
4. val pre-decremento
5. ! negación lógica
6. $\sim$ negación bit a bit
7. ** exponenciación
8. << desplazamiento a izquierda a nivel de bits
9. >> desplazamiento a derecha a nivel de bits
10. & AND a nivel de bits
11. | OR a nivel de bits
12. && AND lógico

Características Avanzadas

Características Avanzadas: Doble Corchete

[[expresión]]

Este tipo de comparación usa las mismas comparaciones permitidas por el comando *test*. Sin embargo, esta opción permite hacer comparaciones utilizando expresiones regulares.

Características Avanzadas

Ejemplo

```
if [[ $USER == r* ]]
then
 echo "Hola $USER"
else
 echo "Lo siento, no lo conozco"
fi
```

El Comando case

Formato

```
case variable in
 patrón1 | patron2) comandos1;;
 patrón3) comandos2;;
 *) default comandos;;
esac
```

El comando *case* compara la variable especificada contra los diferentes patrones. Si hubo concordancia, el intérprete ejecuta los comandos especificados por el patrón.

El Comando case

Formato

```
var=1
case $var in
 1) echo "Opción Uno" ;;
 2) echo "Opción Dos" ;;
 *) echo "Ninguna de las opciones" ;;
esac
```