Bourne Again Shell

Parte II: Comandos Estructurados-Iteraciones

Dr. Mario M. Berón

Universidad Nacional de San Luis

for

Formato

for var in lista do comandos done

Funcionamiento

La variable *var* toma cada elemento de la *lista*. En la primer iteración *var* toma el primer elemento de la *lista*. En la segunda el segundo elemento de la *lista* y así siguiendo. La cantidad de veces que se ejecutan los *comandos* ubicados dentro de la iteración es igual a la longitud de la *lista* recibida como parámetro.

for

Formas de Especificar los Elementos de la Lista

- Colocar los valores de la lista directamente en el comando.

```
for test in Alabama Alaska Arizona Arkansas California
do
 echo El próximo estado es: $test
done
```

Después de la finalización de la iteración la variable *test* puede ser usada en el resto del script. Además dicha variable queda con el último valor de la lista siempre y cuando el programador no lo haya cambiado.

for

Formas de Especificar los Elementos de la Lista

- Colocar los valores en una variable.

```
#!/bin/bash
# Uso de una variable para almacenar la lista
list="Alabama Alaska Arizona Arkansas Colorado"
list=$list" Connecticut"
for state in $list
do
 echo "Ha visitado $state?"
done
```

for

Formas de Especificar los Elementos de la Lista

- Usar los valores retornados por un comando.

```
#!/bin/bash
# Lectura de valores desde un archivo
file="states"
for state in ‘cat $file‘
do
 echo "Visite: $state"
done
```

for

Formas de Especificar los Elementos de la Lista

- Especificación de los varlores por medio de la lectura de un directorio usando comodines.

```
#!/bin/bash
# Iterar por todos los archivos de una carpeta
for file in /home/rich/test/*
do
 if [ -d "$file" ]
 then
  echo "$file es una carpeta"
 elif [ -f "$file" ]
 then
  echo "$file es un archivo"
 fi
done
```

for

Formas de Especificar los Elementos de la Lista

- Especificación de valores por carpetas y listas.

```
#!/bin/bash
# Iteración a través de carpetas y listas
for file in /home/rich/.b* /home/rich/badtest
do
 if [ -d "$file" ]
 then
  echo "$file es una carpeta"
 elif [ -f "$file" ]
 then
  echo "$file es un archivo"
 else
  echo "$file no existe"
 fi
done
```

for al Estilo C

Formato

```
for (( Asiganción a Variable;
       Condición ;
       Progreso de la Iteración ))
```

Formato

```
for (( a = 1; a < 10; a++ ))
```

Formato

- La asignación del valor de la variable puede contener espacios.
- La variable en la condición no está precedida por el signo dolar.
- La ecuación para el progreso de la iteración no usa el comando *expr*.

for al Estilo C

Ejemplo

```
#!/bin/bash
# Verficación de la iteración al estilo C
for (( i=1; i <= 10; i++ ))
do
 echo "El próximo número es: $i"
done
```

for al Estilo C

Ejemplo

```
#!/bin/bash
# Variables múltiples
for (( a=1, b=10; a <= 10; a++, b-- ))
do
 echo "$a - $b"
done
```

while

Formato

```
while comando test
do
 otros comandos
done
```

Funcionamiento

Los *otros comandos* se ejecutan mientras el comando *test comando* retorne como estado de salida cero.

while

Ejemplo

```
#!/bin/bash
# while comando test
var1=10
while [ $var1 -gt 0 ]
do
 echo $var1
 var1=$[ $var1 - 1 ]
done
```

while

Múltiples Comandos test

```
#!/bin/bash
# Verificación de una iteración while
# con múltiples comandos
var1=10
while echo $var1
      [ $var1 -ge 0 ]
do
 echo "Se está dentro de la iteración"
 var1=$[ $var1 - 1 ]
done
```

Until

Formato

```
until comando test
do
 otros comandos
done
```

Funcionamiento

Siempre que estado de salida el comando *test* sea distinto de cero el intérprete ejecuta los comandos dentro del *until* (*otros comandos*).

Until

Ejemplo

```
#!/bin/bash
# Uso del comando until
var1=100
until [ $var1 -eq 0 ]
do
 echo $var1
 var1=$[ $var1 - 25 ]
done
```

Until

Ejemplo de Múltiples Comandos Test

```
#!/bin/bash
# uso del comando until
var1=100
until echo $var1
      [ $var1 -eq 0 ]
do
 echo Dentro de la iteración: $var1
 var1=$[ $var1 - 25 ]
done
```

Control del Loop

Break y Continue

Son dos comandos que permiten controlar lo que sucede dentro del loop.

Break

Formato

```
break
```

```
break n
```

Funcionamiento

1. break: posibilita salir desde cualquier tipo de itreación. Luego de la ejecución del break el control se transfiere al próximo comando después de la iteración.
2. break n: tiene la misma función que break excepto que en lugar de salir de la iteración corriente sale de la iteración de nivel de anidamiento indicado por *n*. El valor por defecto de *n* es 1.

Break

Ejemplo de break

```
#!/bin/bash
for var1 in 1 2 3 4 5 6 7 8 9 10
do
 if [ $var1 -eq 5 ]
 then
  break
 fi
 echo "Numero de iteración: $var1"
done
echo "La iteración for ha finalizado"
```

Break

Ejemplo de break n

```
#!/bin/bash
# Uso del break para salir a un ciclo externo
for (( a = 1; a < 4; a++ ))
do
 echo "Ciclo externo: $a"
 for (( b = 1; b < 100; b++ ))
 do
  if [ $b -gt 4 ]
  then
   break 2
  fi
  echo " Ciclo interno: $b"
 done
done
```

Continue

Formato

```
continue

continue n
```

Funcionamiento

1. continue: Para el procesamiento de los comandos dentro de la iteración y transfiere el control a la evaluación de la condición de la iteración.
2. continue n: Tiene la misma funcionalidad que *continue* excepto que transfiere el control a la iteración de nivel *n*.

Continue

Ejemplo: conitnue

```
#!/bin/bash
# Uso del comando continue
for (( var1 = 1; var1 < 15; var1++ ))
do
 if [ $var1 -gt 5 ] && [ $var1 -lt 10 ]
 then
  continue
 fi
 echo "Número de Iteración: $var1"
done
```

Continue

Ejemplo: conitnue n

```
#!/bin/bash
# Se continúa en el ciclo externo
for (( a = 1; a <= 5; a++ ))
do
 echo "Iteración $a:"
 for (( b = 1; b < 3; b++ ))
 do
 if [ $a -gt 2 ] && [ $a -lt 4 ]
 then
  continue 2
 fi
 var3=$[ $a * $b ]
 echo " El resultado de $a * $b es $var3"
 done
done
```