Códigos de Retorno y Estatus de Salida

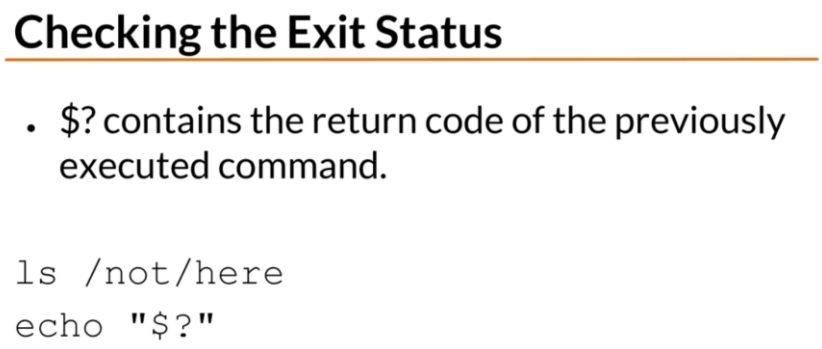
Cada vez que ejecutamos un comando este nos devuelve un “Estado de Salida”. Este “Estado de Salida”, que a veces es llamado “Código de Retorno” o “Código de Salida”, es un número entero que va de 0 a 255.

- Por convención, los comandos que se ejecutan con éxito devuelven un Código de Salida 0.
- Si se encuentra algún tipo de error, se devuelve un Código de Salida distinto de 0.

Estos Códigos de Salida pueden ser utilizados en nuestro script para la comprobación de errores. Puede ser una comprobación simple, como comprobar que el Código de Salida sea 0, o puede ser más compleja, como la comprobación de un Código de Salida específico.

Si queremos averiguar qué significan los distintos Códigos de Salida, tenemos que consultar la documentación del comando en cuestión. Podemos utilizar el comando “man” y el comando “info” para leer la documentación de la mayoría de los comandos de nuestro sistema.

La variable especial $? Contiene el Codigo de Salida del comando ejecutado anteriormente. En el siguiente script Shell, el comando “ls” es llamado con una ruta a un archivo que no existe:



Inmediatamente después de que el comando “ls” es ejecutado, el Código de Salida de ese comando es mostrado en la consola usando el comando “echo $?”. Esto mostrará un 2 en la pantalla.

Recordemos que los Códigos de Salida distintos de 0 indican algún tipo de error. Si el archivo “/not/here” existiera y “ls” fuera capaz de mostrar información sobre ese archivo con éxito, el Código de Salida habría sido 0.

Podemos utilizar el Código de Salida de un comando para tomar una decisión o realizar una acción diferente basada en ese código. En el siguiente ejemplo de un fragmento de script Shell, utilizamos el comando “ping” para probar nuestra conectividad de red a www.google.com:

```
HOST="google.com"
ping -c 1 $HOST
if [ "$?" -eq "0" ]
then
  echo "$HOST reachable."
else
  echo "$HOST unreachable."
fi
```

La opción -c para el comando “ping” simplemente indica que se envíe un solo paquete. Una vez ejecutado el comando “ping”, comprobamos el Código de Salida. Si el código es 0, entonces imprimimos en pantalla que google.com es accesible. Si el código es diferente de 0, entonces imprimimos en la pantalla que google.com es inalcanzable.

En el siguiente ejemplo sólo se comprueba si se produce un error:

```
HOST="google.com"

ping -c 1 $HOST

if [ "$?" -ne "0" ]
then
  echo "$HOST unreachable."
fi
```

En la sentencia “if”, buscamos un Código de Salida distinto de 0. Si “$?” no es igual a cero, entonces hacemos un “echo” indicando que el host es inalcanzable. Si el comando “ping” tiene éxito y devuelve un 0, entonces la sentencia “echo” no se ejecuta.

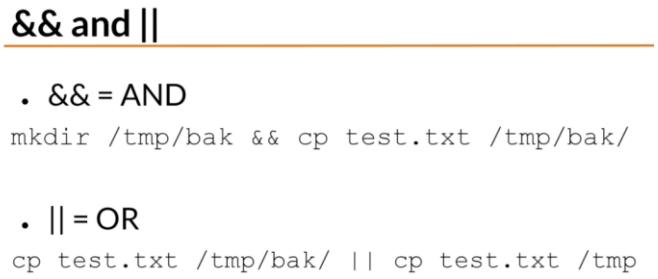
Podemos asignar el Código de Salida de un comando a una variable y luego usar esa variable más adelante:

```
HOST="google.com"
ping -c 1 $HOST
RETURN_CODE=$?

if [ "$RETURN_CODE" -ne "0" ]
then
  echo "$HOST unreachable."
fi
```

En este ejemplo, el valor de $? Se asigna a la variable RETURN_CODE. Así, la variable RETURN_CODE contiene el Código de Salida.

Además de utilizar sentencias "if" con los Códigos de Salida, podemos utilizar AND y OR:



- El doble ampersand (&&) representa AND, mientras que el doble pipe (||) representa O.
- Podemos encadenar varios comandos con AND u OR.
- El comando que sigue a un doble ampersand (&&) sólo se ejecutará si el comando anterior se cumple. Dicho de otra manera, el comando que sigue a un doble ampersand sólo se ejecutará si el comando anterior resulta en un Código de Salida en 0.

En el ejemplo anterior se ejecuta la instrucción "mkdir /tmp/bak". Si devuelve un Código de Salida 0, entonces se ejecutará el comando "cp". En este ejemplo, podemos ver que no hay necesidad de que el comando "cp" sea ejecutado si el directorio no pudo ser creado.

El comando que sigue al doble pipe (||) solo se ejecutará si el comando anterior falla. Si el primer comando devuelve un Código de Salida 0, entonces se ejecuta el siguiente comando.

En el ejemplo "OR", se ejecuta el primer comando "cp", que es "cp test.txt /tmp/bak". Si se ejecuta con éxito, el siguiente comando "cp" NO se ejecutará. En este ejemplo, si el primer comando "cp" tiene éxito, entonces hemos creado con éxito una copia del archivo "test.txt". Por lo tanto, no es necesario otra copia de seguridad del archivo test.txt.

Si el primer comando falla en la instrucción del doble pipe (||), entonces se ejecuta el segundo. En este ejemplo, si "test.txt" no puede copiarse en "/tmp/back". Entonces se intentará copiarlo en "/tmp". Con un "OR", sólo uno de los dos comandos se ejecutará con éxito.

Volvamos a un ejemplo anterior:

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST && echo "$HOST reachable."
```

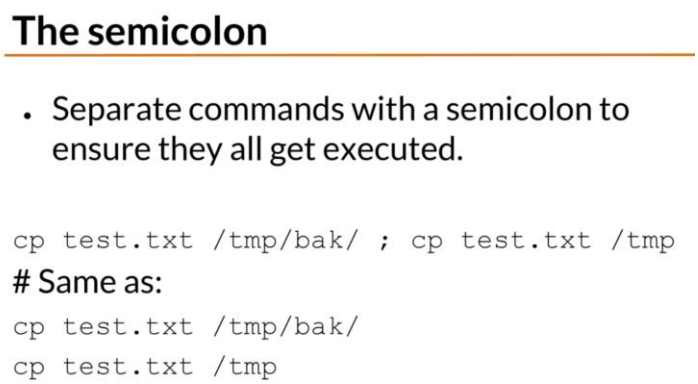
En este ejemplo, en lugar de utilizar "if", utilizamos un "AND". En este ejemplo, si el comando "ping" nos da un Código de Salida 0, se mostrará en pantalla "google.com reachable".

En el siguiente ejemplo estamos utilizando “OR” en lugar de “if”:

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST || echo "$HOST unreachable."
```

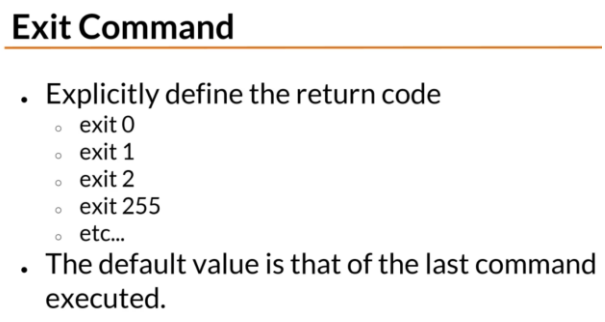
En este ejemplo, si el comando “ping” falla, se mostrará en pantalla “google.com unreachable”. Si el comando “ping” tiene éxito, la sentencia “echo” no se ejecutará.

Si queremos encadenar varios comandos en una sola línea, podemos hacerlo separándolos con punto y coma (;):



El punto y coma no comprueba el Código de Salida del comando anterior. El comando que sigue a un punto y coma siempre se ejecutará, sin importar si el comando anterior falló o tuvo éxito. En esencia, separar comandos con un punto y coma es lo mismo que poner esos comandos en líneas separadas.

No sólo los comandos predefinidos devuelven un Código de Salida, sino también los scripts Shell. Podemos controlar el Código de Salida de nuestro script usando el comando “exit”:



Simplemente usamos el comando “exit” en nuestro script y lo acompañamos con un número del 0 al 255. Si no especificamos un Código de Salida con el comando “exit”, entonces el Código de Salida ejecutado previamente será usado como Código de Salida del script. Esto también se cumple si no incluimos el comando “exit” en nuestro script.

Podemos utilizar el comando “exit” en cualquier parte de nuestro script, pero es importante tener en cuenta que siempre que se alcance el comando “exit”, nuestro script dejará de ejecutarse.

En el siguiente ejemplo, estamos controlando el Código de Salida de nuestro script con el comando “exit”:

```
#!/bin/bash
HOST="google.com"
ping -c 1 $HOST
if [ "$?" -ne "0" ]
then
  echo "$HOST unreachable."
  exit 1
fi
exit 0
```

Si el comando “ping” tiene éxito, se recibe un Código de Salida 0. Esto significa que la prueba en la sentencia “if” es falsa y el bloque de código no se ejecutará. Entonces se ejecutará la línea “exit 0”, esto detiene la ejecución del script y devuelve un 0 como Código de Salida.

Si el comando “ping” falla, entonces se recibe un Código de Salida distinto de 0. Esto hace que la sentencia “if” sea verdadera y se ejecuten los comandos “echo” y “exit 1”. El comando “exit 1” detendrá la ejecución del script y devolverá un Código de Salida 1.

Ahora nuestro script puede ser llamado por otro script y su Código de Salida puede ser examinado como cualquier otro comando.

Ya que controlamos los Códigos de Salida, podemos hacer que tengan un significado importante. Tal vez, un Código de Salida. De 1 significa que un tipo de error ocurrió, mientras que, un código de retorno de 2 significa que un tipo diferente de erro ocurrió.

Si tenemos scripts que están ejecutando otros scripts, entonces podemos programar de acuerdo a estos Códigos de Salida si lo necesitamos o lo queremos.