Recordatorio de Bash

Administración de Servidores

Pablo García-Sánchez

Departamento de Ingeniería Informática
Universidad de Cádiz
Introducción a Bash

Curso 2018 – 2019

Indice

Sección 1 | Introducción

Primer ejemplo en Bash

```
#!/bin/bash
# Ejemplico
echo Hello World!
```

Ejecución

```
pgarcia@bash: ./myscript.sh
bash: ./myscript.sh: Permission denied
pgarcia@bash: ls -l myscript.sh
-rw-r--r-- 18 pgarcia users 4096 Feb 17 09:12 myscript.sh
pgarcia@bash: chmod 755 myscript.sh
ls -l myscript.sh
pgarcia@bash: -rwxr-xr-x 18 pgarcia users 4096 Feb 17 09:12 myscript.
./myscript.sh
Hello World!
```

El Shebang

- El signo de exclamación y hash ( ) se denomina Shebang. Muestra la ruta al intérprete (o programa) que debe utilizarse para ejecutar (o interpretar) el resto de las líneas del archivo de texto.
- Para los scripts Bash será el camino a Bash, pero hay muchos otros tipos de scripts y cada uno tiene su propio intérprete.
- Importante: El shebang debe estar en primera línea del archivo (la línea 2 no servirá, incluso si la primera línea está en blanco). Tampoco debe haber espacios antes del # o entre el ! y el camino hacia el intérprete.
- Es posible no usarlo y seguir ejecutando el script, pero es imprudente. Si estás en una terminal y ejecuta el shell de Bash y ejecuta un script sin Shebang, entonces Bash asumirá que es un script de Bash. Así que esto sólo funcionará asumiendo que el usuario que ejecuta el script lo está ejecutando en un shell de Bash y hay una variedad de razones por las que esto puede no ser el caso, lo cual es peligroso.

Sección 2 | Variables

Argumentos por línea de comandos

```
#!/bin/bash
# A simple copy script
cp $1 $2
# Let's verify the copy worked
echo Details for $2
ls -lh $2
```

Otras variables especiales

- $0 - El nombre del script Bash.
- $1 - $9 - Los primeros 9 argumentos para el script Bash. (Como se mencionó anteriormente.)
- $# Cuántos argumentos se pasaron al script Bash.
- $@ - Todos los argumentos suministrados al script Bash.
- $? - El estado de salida del proceso ejecutado más recientemente.
- $$- El ID de proceso del script actual.
- $USER - El nombre de usuario del usuario que ejecuta el script.
- $HOSTNAME - El nombre de host de la máquina en la que se está ejecutando el script.
- $SECONDS - El número de segundos desde que se inició el script.
- $RANDOM - Devuelve un número aleatorio diferente cada vez que se hace referencia a él.
- $LINENO - Devuelve el número de línea actual en el guión Bash.

Asignar variables

```
#!/bin/bash
# Ejemplo de asignacion de variables
myvariable=Hola
anothervar=Pablo
echo $myvariable $anothervar
echo
sampledir=/etc
ls $sampledir
```

Comillas dobles y comillas simples

```
pgarcia@bash: myvar='Hello World'
pgarcia@bash: echo $myvar
Hello World
pgarcia@bash: newvar="More␣$myvar"
pgarcia@bash: echo $newvar
More Hello World
pgarcia@bash: newvar='More $myvar'
pgarcia@bash: echo $newvar
More $myvar
```

Substitución de comandos

```
myvar=$( ls /etc | wc -l )
echo Hay $myvar entradas en el directorio /etc
```

Si la salida tiene varias líneas, entonces las nuevas líneas simplemente se eliminan y toda la salida se convierte en una sola línea.

Exportar variables con export

script1.sh

```
#!/bin/bash
# Demostrar el ambito de variable 1
var1=blah
var2=foo
# Comprobemos el valor actual
echo $0 :: var1 : $var1, var2 : $var2
export var1
./script2.sh
# Comprobamos como estan ahora
echo $0 :: var1 : $var1, var2 : $var2
```

script2.sh

```
#!/bin/bash
# Demostrar el ambito 2
echo $0 :: var1 : $var1, var2 : $var2
# Cambiemos los valores
var1=flop
var2=bleh
```

Salida

```
pgarcia@bash: ./script1.sh
script1.sh :: var1 : blah, var2 : foo
script2.sh :: var1 : blah, var2 :
script1.sh :: var1 : blah, var2 : foo
```

- Cuando exportamos una variable le decimos a Bash que cada vez que se crea un nuevo proceso (para ejecutar otro script o algo así) haga una copia de la variable y la entregue al nuevo proceso. Así que aunque las variables tendrán el mismo nombre, existen en procesos separados y por lo tanto no están relacionadas entre sí.
- La exportación de variables es un proceso unidireccional. El proceso original puede pasar variables al nuevo proceso, pero cualquier cosa que el proceso haga con la copia de las variables no tiene impacto en las variables originales.

Sección 3 | Entradas

Usar read

```
#!/bin/bash
# Preguntar el nombre
echo Hola, con quien hablo?
read varname
echo Un placer conocerte, $varname
```

Usar read con prompt (p) y silent (s)

```
#!/bin/bash
# Preguntar el login
read -p 'Username: ' uservar
read -sp 'Password: ' passvar
echo
echo Gracias, $uservar ahora tenemos tus datos
```

Leyendo de STDIN

```
#!/bin/bash
# Ejemplo de uso de STDIN
echo Sumario de ventas:
echo ==================================
echo
# cortamos usando espacio como delimitador
# y cojo el campo 2 y 3
cat /dev/stdin | cut -d' ' -f 2,3 | sort
```

Sección 4 | Aritmética

Let

```
#!/bin/bash
# Aritmetica usando let
let a=5+4
echo $a # 9
let "a = 5 + 4"
echo $a # 9
let a++
echo $a # 10
let "a = 4 * 5"
echo $a # 20
let "a = $1 + 30"
echo $a # 30 + primer parametro de la linea de c
```

Expr

example.sh

```
#!/bin/bash
# Aritmetica basica usando expr
expr 5 + 4
expr "5 + 4"
expr 5+4
expr 5 \* $1
expr 11 % 2
a=$( expr 10 - 3 )
echo $a # 7
```

Salida

```
pgarcia@bash: ./example.sh 12
9
5 + 4
5+4 #OJO CON LOS ESPACIOS
60
1
7
```

Longitud de una variable

```
#!/bin/bash
# Muestra la longitud de una variable.
a='Hello World'
echo ${#a} # 11
b=4953
echo ${#b} # 4
```

Sección 5 | If

If

```
#!/bin/bash
# Sentencia if basica
if [ $1 -gt 100 ]
then
   echo Vaya numero mas grande.
   pwd
fi
date
```

El comando test []

Operator	Description
! EXPRESSION	The EXPRESSION is false.
-n STRING	The length of STRING is greater than zero.
-z STRING	The lengh of STRING is zero (ie it is empty).
STRING1 = STRING2	STRING1 is equal to STRING2
STRING1 != STRING2	STRING1 is not equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is numerically equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is numerically greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is numerically less than INTEGER2
-d FILE	FILE exists and is a directory.
-e FILE	FILE exists.
-r FILE	FILE exists and the read permission is granted.
-s FILE	FILE exists and it's size is greater than zero (ie. it is not empty).
-w FILE	FILE exists and the write permission is granted.
-x FILE	FILE exists and the execute permission is granted.

OJO: Al ser un comando devuelve 0 si es TRUE y 1 si es FALSE

Ejemplos del comando test

```
pgarcia@bash :test 001 = 1
pgarcia@bash :echo $?
1
pgarcia@bash :test 001 -eq 1
pgarcia@bash :echo $?
0
pgarcia@bash :touch myfile
pgarcia@bash :test -s myfile
pgarcia@bash :echo $?
1
pgarcia@bash :ls /etc > myfile
pgarcia@bash :test -s myfile
pgarcia@bash :echo $?
0
```

Ifs anidados

```
#!/bin/bash
# Ifs anidados
if [ $1 -gt 100 ]
  then
  echo Que numero mas grande.
  if (( $1 % 2 == 0 ))
    then
    echo Tambien es numero par.
  fi
fi
```

elif vs else

```
#!/bin/bash
# elif statements
if [ $1 -ge 18 ]
  then
  echo Puedes ir a la fiesta.
elif [ $2 == 'yes' ] #Si tienes justificante
  then
  echo Puedes ir a la fiesta pero venir antes de medianoche.
else
  echo No puedes ir a la fiesta.
fi
```

Operaciones booleanas

```
#!/bin/bash
# Ejemplo AND
if [ -r $1 ] && [ -s $1 ]
then
  echo Este fichero es util.
fi
```

Case

```
#!/bin/bash
# case example
case $1 in
   start)
      echo starting
      ;;
   stop)
      echo stoping
      ;;
   restart)
      echo restarting
      ;;
   *)
      echo don\'t know
   ;;
esac
```

Ejemplo más complejo

```
#!/bin/bash
# Print a message about disk useage.
space_free=$( df -h | awk '{ print $5 }' | sort -n | tail -n 1 |
case $space_free in
  [1-5]*)
    echo Mucho espacio libre
    ;;
  [6-7]*)
    echo Podria haber un problema en el futuro
    ;;
  8*)
    echo Deberias empezar a borrar ficheros inutiles
    ;;
  9*)
    echo Houston, vamos a tener un problema pronto
    ;;
    *)
    echo Aqui falla algo
    ;;
esac
```

Sección 6 | Bucles

While

```
#!/bin/bash
# Bucle while basico
counter=1
while [ $counter -le 10 ] #until tambien sirve
do
   echo $counter
   ((counter++))
done
echo Terminado
```

For

```
#!/bin/bash
# Bucle for basico
names='Stan Kyle Cartman'
for name in $names
do
  echo $name
done
echo Listo
```

For (rangos)

```
#!/bin/bash
# Bucle for basico con rango
for value in {1..5} #otro ejemplo: {10..0..2}
do
   echo $value
done
echo Listo
```

Ejemplo de for sobre ficheros

```
#!/bin/bash
# Copiar ficheros html a extension php
for value in $1/*.html
do
  cp $value $1/$( basename -s .html $value ).php
done
```

Sección 7 | Funciones

Funciones y variables locales

```
#!/bin/bash
# Experimentos con el ambito de las variables
varchange () {
local var1='local 1'
  echo Hola $1
  echo -Dentro de la funcion: var1 is $var1 : var2 is $var2
  var1='changed again'
  var2='2 changed again'
}
var1='global 1'
var2='global 2'
echo Antes de la llamada  : var1 is $var1 : var2 is $var2
varchange Pablo
echo Despues de la llamada: var1 is $var1 : var2 is $var2
```

Salida

```
Antes de la llamada  : var1 is global 1 : var2 is global 2
Hola Pablo
-Dentro de la funcion: var1 is local 1 : var2 is global 2
Despues de la llamada: var1 is global 1 : var2 is 2 changed again
```

Referencias

- Ryan Chadwick
  `https://ryanstutorials.net/bash-scripting-tutorial`