Tema

1.5

Sistema Operativos (SSOO)

Shell Scripts II

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Índice

- Shell scripts

Shell Script: variables

- Los valores se almacenan como cadenas de texto.
- No es necesario declarar las variables, con asignarle un valor es suficiente.
- Los operadores matemáticos que convierten las variables en número para realizar cálculos.
- Para consultar el valor de una variable se antepone el símbolo **$** al nombre.

```
#!/bin/bash
cadena = “¡Hola Gente!”  #esto es un comentario
echo $cadena
```

- No existe el casting de los tipos de las variables, por lo que se podrá asignar a una misma variables diferentes valores sin importar su tipo.
- Se recomienda usar el mismo tipo de datos a una variable dentro de un mismo script.
- El carácter de escape de la bash **** mantiene el carácter siguiente.

Shell Script: uso de comillas

- Las cadenas de textos deben ir acompañadas de comillas simples o dobles.
- El uso de comillas dobles "…" permite referenciar otras variables dentro de la cadena, e.g. cadena = "Tu edad es **$edad**"
- El uso de comillas simples permite mostrar una cadena sin la posibilidad de referenciar otras variables dentro de la misma cadena.
  - cadena = 'Tu edad es $edad'

Ejercicio:

Mostrar la siguiente cadena de texto:

El valor de la variable 'edad' es "31"

Shell Script: variables entre procesos

- El comando **export** pone una variable en el entorno de forma que sea accesible por los procesos hijos.
- Si el proceso hijo modifica el valor de la variable cadena, no modificará el valor original del proceso padre. Verificarlo cambiando el valor de cadena desde un proceso hijo.

Observa este ejemplo:

```
$ cadena=libro
$ bash # Ejecuta una nueva shell (proceso hijo)
$ echo $cadena # No debería mostrar nada cadena
$ exit # Regresa al proceso padre
$ export cadena
$ bash
$ echo $cadena
libro # El resultado es...
```

Shell Script: variables

Hay dos tipos de variables de entorno:

– Variables locales

– Variables del entorno

Las Variables del entorno se establecen por el sistema y se pueden encontrar utilizando el comando **env**.

Las variables de entorno contiene valores especiales.

Las variables del entorno se definen en **/etc/profile**,

/etc/profile.d/ y **~/.bash_profile**.

Estos ficheros son de inicialización y son leídos cuando se invoca la bash shell.

Cuando la login shell sale, la bash lee **~/.bash_logout**

Shell Script: variables

HOME: argumento por defecto (directorio home) del comando **cd**.

PATH: el path de búsqueda de comandos. Es una lista de directorios separados por ':' en los que se busca cuando se teclea cualquier comando.

Normalmente, introducimos los comandos de la siguiente manera:

– $ ./trash.sh

Estableciendo **PATH=$PATH**:. Nuestro directorio de trabajo se incluye en path de búsqueda de comando y simplemente podremos introducir:

– $ trash.sh

Shell Script: variables

- LOGNAME: contiene el nombre de usuario
- HOSTNAME: contiene el nombre de la máquina
- MACHTYPE: sistema hardware
- UID: contiene el id del usuario que no puede ser modificado
- SHLVL: contiene el nivel de anidamiento de la shell
- PS1: secuencia de caracteres monstrados antes del prompt
  - \t hora
  - \d fecha
  - \w directorio actual
  - \W última parte del directorio actual
  - \u nombre de usuario
  - \$ caracter del prompt

Shell Script: otras variables

- $#: número argumentos
- $*: todos los argumentos de la shell
- $@: semejante a la variable anterior
- $-: opciones suministradas a la shell
- $?: devolver valor de la última orden ejecutada
- $!: identificación del proceso de la última orden que comenzó con &

Shell Script: Exit

Exit se puede utilizar para finalizar la ejecución de un script o para devolver un valor, el cuál estará disponible al proceso padre del script.

- Cuando un script termina con **exit** sin parámetros, el estado de salida será el del último comando ejecutado en el script.
- Cuando se ejecuta el script con el comando exit con parámetros, la sintaxis es: exit nnn, donde nnn=0-255 y el estado de salida es nnn

Shell Script: Operadores

+ suma, - resta, * multiplicación, / división, ** exponenciación

% módulo

Prueba lo siguiente:

$ a=(5+2)*3

$ echo $a

$ b=2**3

$ echo $a+$b

Shell Script: Leer de teclado

Read permite solicitar un valor de entrada para almacenarlo en una variable.

Opciones

read –s (no hace echo de la entrada)

read –nN (acepta sólo N caracteres de entrada)

read –p “mensaje” (muestra un mensaje)

read –tT (acepta una entrada por un tiempo máximo de T segundos)

Ejemplo (copiar.sh)

```
#!/bin/bash
echo -n “Introduzca nombre de fichero a copiar: ”
read fichero
cp -i $fichero
echo “El fichero $fichero ha sido copiado!“
```

Shell Script: Leer de teclado

Ejemplo:

$ read –s –n1 -p “si (S) o no (N)?” respuesta

si (S) o no (N) ? S

$ echo $respuesta

S

Shell Script: Ejecutar comandos

El símbolo ` tiene un uso diferente de ´. Se utiliza para sustitución de instrucciones. Es decir si dentro de un script aparece el texto `comando` entonces se ejecutará lo orden que está entre las `

```
$ LISTA=`ls`
$ echo $LISTA # Lista los archivos
fichero.sh, scr01.sh
```

Otra forma de realizar la sustitución de comandos: **$(comando)**

```
$ LISTA=$(ls)
$ echo $LISTA
hola.sh leer.sh
ls $( pwd )
ls $( echo /bin )
```

Shell Script: Estructura de control

Condicional: la forma más básica es:

– if [*expresión*];

then

instrucciones

elif [*expresión*];

then

instrucciones

else

instrucciones

fi

– Las secciones elif(else if) y else son opcionales

Shell Script: Expresiones

Una expresión puede ser: comparación de cadenas, comparación numérica, operadores de fichero y operadores lógicos y se representa mediante [expresión]:

Comparación de cadenas:

=

!=

-n evalúa si la longitud de la cadena es superior a 0

-z evalúa si la longitud de la cadena es igual a 0

Ejemplos:

[s1 = s2] (true si s1 es igual a s2, sino false)

[s1 != s2] (true si s1 no es igual a s2, sino false)

[s1] (true si s1 no está vacía, sino false)

[-n s1] (true si s1 tiene longitud mayor que 0, sino false)

[-z s2] (true si s2 tiene longitud 0, sino false)

Shell Script: Expresiones

Comparación numérica:

-eq

-ge

-le

-ne

-gt

-lt

Ejemplos:

[n1 -eq n2]

[n1 -ge n2]

[n1 -le n2]

[n1 -ne n2]

[n1 -gt n2]

[n1 -lt n2]

1

```
#!/bin/bash
echo -n “Introduzca su nombre de usuario: ”
read login
if [ “$login” = “$USER” ]; then
  echo “Hola, $login. Cómo está hoy?”
else
  echo “Tú no eres $login!!!”
fi
```

Shell Script: Expresiones

Comparación numérica:

- -eq
- -ge
- -le
- -ne
- -gt
- -lt

Ejemplos:

- [n1 -eq n2]
- [n1 -ge n2]
- [n1 -le n2]
- [n1 -ne n2]
- [n1 -gt n2]
- [n1 -lt n2]

```
#!/bin/bash
echo -n “Introduzca un número 1 < x < 10: "
read num
if [ “$num” -lt 10 ]; then
  if [ “$num” -gt 1 ]; then
    echo “$num*$num=$(($num*$num))”
  else
    echo “¡Número fuera de rango!”
  fi
else
  echo “¡Número fuera de rango!”
fi
```

2

Shell Script: Expresiones

Operadores de archivos:

-d verifica si el path dado es un directorio

-f verifica si el path dado es un archivo

-s verifica si el path dado en un link simbólico

-e verifica si el fichero existe

-s verifica si el fichero tiene un tamaño mayor a 0

-r verifica si el fichero tiene permiso de lectura

-w verifica si el fichero tiene permiso de escritura

-x verifica si el fichero tiene permiso de ejecución

Ejemplos:

```
[ -d nombre_fichero]
[ -f nombre_fichero ]
[ -e nombre_fichero ]
[ -s nombre_fichero ]
[ -r nombre_fichero ]
[ -w nombre_fichero ]
[ -x nombre_fichero ]
```

```
#!/bin/bash
if [ -f /etc/fstab ]; then
  cp /etc/fstab .
  echo “Hecho.”
else
  echo “Archivo /etc/fstab no existe.”
  exit 1
fi
```

Shell Script: Expresiones

Operadores lógicos:

- **! NOT**
- **-a AND**
- **-o OR**
- **– && AND**
- **– || OR**

```
#!/bin/bash
echo -n “Introduzca un número entre 1 < x < 10:”
read num
if [ “$num” -gt 1 –a “$num” -lt 10 ];
then
  echo “$num*$num=$(($num*$num))”
else
  echo “Número introducido incorrecto !”
fi
```

4

```
#!/bin/bash
echo -n “Introduzca un número 1 < x < 10: "
read num
if[ “$number” -gt 1 ] && [ “$number” -lt 10 ];
then
  echo “$num*$num=$(($num*$num))”
else
  echo “Número introducido incorrecto !”
fi
```

5

Universidad Francisco de Vitoria UFV Madrid

Shell Script: parámetros

Los parámetros posicionales se asignan desde la shell cuando se invoca. Parámetro posicional “N” se referencia como “${N}”, o “$N” cuando “N” lo forma un sólo dígito

Parámetros especiales

$# número de parámetros pasados

$0 devuelve el nombre del shell script que se está ejecutando y su ubicación en el sistema de archivos

$* devuelve en una cadena de caracteres todos los parámetros pasados al script

$@ devuelve un array con los parámetros pasados al script

6

```
#!/bin/bash
echo “$#; $0; $1; $2; $*; $@”
$ parametros.sh estudiante1 estudiante2
2; ./parametros.sh; estudiante1; estudiante2; estudiante1 estudiante2; estudiante1 estudiante2
```

Shell Script: case

```
case $var in
val1)
instrucciones;;
val2)
instrucciones;;
*)
instrucciones;;
esac
```

```
#!/bin/bash
echo -n “Introduzca un número entre 1 < x < 10: ”
read x
case $x in
1) echo “Valor de x es 1.”;;
2) echo “Valor de x es 2.”;;
3) echo “Valor de x es 3.”;;
4) echo “Valor de x es 4.”;;
5) echo “Valor de x es 5.”;;
6) echo “Valor de x es 6.”;;
7) echo “Valor de x es 7.”;;
8) echo “Valor de x es 8.”;;
9) echo “Valor de x es 9.”;;
0 | 10) echo “Número incorrecto.”;;
*) echo “Valor no reconocido.”;;
esac
```

Shell Script: for

for

```
for var in lista
do
statements
done
```

```
#!/bin/bash
let sum=0
for num in 1 2 3 4 5
do
let “sum = $sum + $num”
done
echo $sum
```

7

8

```
#!/bin/bash
for x in papel lapiz boli; do
echo “El valor de la variable x es: $x”
sleep 1
done
```

Shell Script: for

for

for var in lista

do

statements

done

```
#!/bin/bash
lista="antonio luis maria pepa"
for x in $lista
do
echo "El valor de la variable x es: $x"
sleep 1
done
```

9

```
#!/bin/bash
for x in "papel A4" "lapiz STADTLER" "boli BIC"; do
echo "El valor de la variable x es: $x"
sleep 1
done
```

10

Shell Script: for

```
#!/bin/bash
# Lista todos los ficheros del directorio /bin
for x in /bin
do
ls -l “$x”
done
```

11

```
#!/bin/bash
# Lista todos los ficheros del directorio
actual
for x in *
do
ls -l “$x”
sleep 1
done
```

12

```
#!/bin/bash
read –p “Introduzca el nombre de un directorio: ”
directorio
echo “enlaces simbólicos en el directorio
$directorio “
for fichero in $( find $directorio -type l )
do
echo "$fichero"
done
```

13

Shell Script: arrays

Crear un array

```
mascota[0]=perro
mascota[1]=gato
mascota[2]=pez
pet=( perro gato pez )
```

Longitud máxima de un array son 1024 elementos.

Para extraer una entrada del array ${array[i]}

```
$ echo ${mascota[0]}
perro
$ echo ${mascota[2]}
pez
```

Shell Script: arrays

Para extraer todos los elementos se utiliza un asterisco:

```
echo ${array[*]}
```

Para saber cuántos elementos hay en el array:

```
echo ${#array[*]}
```

Podemos combinar los arrays con bucles utilizando **for**:

```
for x in ${array[*]}
do
echo ${array[$x]}
done
```

Shell Script: arrays

Estructura tipo C alternativa para **for**

```
for (( EXPR1 ; EXPR2 ; EXPR3 ))
do
instrucciones
done
```

```
#!/bin/bash
echo “Introduzca un número: ”; read x
let sum=0
for (( i=1 ; $i<$x ; i=$i+1 )) ; do
let “sum = $sum + $i”
done
echo “La suma de los primeros $x números es: $sum”
```

14

Shell Script: while

while *expresion_evalua_a_true*

do

instrucciones

done

```
#!/bin/bash
echo –n “Introduzca un número: ”; read x
let sum=0; let i=1
while [ $i –le $x ]; do
let “sum = $sum + $i”
let “i = $i + 1”
done
echo “La suma de los primeros $x números es: $sum”
```

15

Shell Script: until

until *expresion_evalua_a_true*

do

instrucciones

done

```
#!/bin/bash
echo “Introduzca un número: ”; read x
echo ;
until [ “$x” -le 0 ]; do
echo $x
x=$(($x –1))
sleep 1
done
echo ; echo FIN
```

16

Shell Script: funciones

```
#!/bin/bash
function check() {
    if [ -e "/home/$1" ]
       then
       return 0
    else
       return 1
    fi
}
```

17

```
echo “Introduzca el nombre del archivo: ” ; read x
if check $x
then
  echo “$x existe !”
else
   echo “$x no existe !”
fi
```

Shell Script: hacer debug

Bash ofrece dos formas de depurar los shell scripts

-v : muestra cada línea completa del script antes de ser ejecutada

-x : muestra cada línea abreviada del script antes de ser ejecutada

Uso: #!/bin/bash –v, o #!/bin/bash –x

```
#!/bin/bash –x
echo –n “Introduzca un número: ”; read x
let sum=0
for (( i=1 ; $i<$x ; i=$i+1 )) ; do
let “sum = $sum + $i”
done
echo “La suma de los $x primeros números es: $sum”
```

Bibliografía

- **CARRETERO**, Jesús, GARCÍA, Félix, DE MIGUEL, Pedro, PÉREZ, Fernando. Sistemas Operativos: una visión aplicada. McGraw-Hill, 2001.
- **STALLINGS**, William. Sistemas operativos: aspectos internos y principios de diseño. 5ª Edición. Editorial Pearson Educación. 2005. ISBN: 978-84-205-4462-5.
- **TANENBAUM**, Andrew S. Sistemas operativos modernos. 3ª Edición. Editorial Prentice Hall. 2009. ISBN: 978-607- 442-046-3.

marlon.cardenas@ufv.es

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior