UT 013.
INTRODUCCIÓN AL SHELL SCRIPTING

Sistemas informáticos
CFGS DAW

Álvaro
Maceda
a.macedaarranz@edu.gva.es

2022/2023

Versión:230331.1052

Licencia

Nomenclatura

A lo largo de esta unidad se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE

UT 013. INTRODUCCIÓN AL SHELL SCRIPTING

1. ESTRUCTURA DE CONTROL S

1.1 Loo ps

Los bucles son otra estructura básica de programación. En Bash podemos usar los bucles `for` y `while` para implementar esta estructura.

1.1.1 Bucle For

Los bucles `for` recorren una lista de valores. Por ejemplo:

```
para i en 1 2 3 4 5
hacer
  echo "En bucle... número $i"
done

Bucle ... número 1
Bucle ... número 2
Bucle ... número 3
Bucle ... número 4
Bucle ... número 5
```

Puede utilizar patrones glob para el bucle `for`. El patrón glob será reemplazado por los archivos que coincidan con ese patrón, pero si el globbing no consigue coincidir con ningún archivo/directorio, se conservará el carácter globbing original:

```
para item en asereje archivo*
sarandonga do
  echo "El artículo es
$artículo" done

El elemento es asereje
El elemento es
file00.txt El elemento
es file01.txt El
```

También es posible utilizar la sustitución de comandos en bucles `for`:

```
contador=0
for file in $(ls -1)
do
  contador=$((contador+1))
  echo "El fichero $contador es
$fichero" done

Archivo 1 es
... El
fichero 2 es
```

Separador de campo interno (IFS)

Si utiliza una cadena como parámetro para el bucle for, la cadena se dividirá en "palabras", y cada palabra se tomará como una entrada diferente para el bucle:

```
data="Esto se ejecutará cinco
veces" for item in $data
do
  echo "El item es $item"
done

El artículo es
This El artículo
es will El
artículo es run El
artículo es five
```

El comportamiento es el mismo si utilizamos la cadena directamente. Esto ocurre porque las cadenas se dividen utilizando un carácter especial, `IFS`. `IFS` significa "Internal Field Separator" y es una variable especial utilizada en programación shell para especificar el delimitador que separa los campos de una cadena.

Por defecto, la variable IFS está configurada con caracteres de espacio en blanco (es decir, espacio, tabulador y nueva línea), lo que significa que una cadena se dividirá en campos siempre que se encuentre alguno de estos caracteres. Sin embargo, puede cambiar el valor de `IFS` para especificar un carácter delimitador o un conjunto de caracteres diferente. Por ejemplo:

```
IFS=":|"
data="different:fields|one:variable"
for item in $data; do
  echo "El artículo es
$artículo" done

El artículo es
diferente El artículo
es campos El artículo
es uno
```

Bucle en C

Existe una sintaxis especial para ejecutar bucles for con contadores similar a la de los lenguajes de programación comunes:

```
for ((i = 0 ; i < 3 ; i++)); do
  echo $i
hecho

0
1
2
```

Gamas

Utilizando la función Bash Sequence Expression `{START..END..INCREMENT}` podemos generar rangos de enteros o caracteres, definiendo un inicio y un final. Esta función se utiliza a menudo junto con los bucles for:

```
para i en {1..5..2}
hacer
  echo "Número: $i"
done

Número: 1
Número: 3
Número: 5
```

1.1.2 Bucle while

En los scripts de Bash, un bucle `while es` una sentencia de flujo de control que permite ejecutar un bloque de código repetidamente mientras cierta condición es verdadera. La sintaxis general de un bucle while en Bash es la siguiente:

```
while [ condition ] # Aquí también puede utilizar la sintaxis de
doble corchete do
  # bloque de código a ejecutar
done
```

Veamos un ejemplo. El siguiente script imprime los números del 1 al 5 utilizando un bucle while:

```
n=1
while [ $n -le 5 ]; do
  echo $n
  n=$((n+1))
hecho
```

Bucles infinitos

También puedes crear bucles infinitos en bash usando la sintaxis while con una condición que evalúe siempre a true. Puedes hacerlo con `true` y `:`. En caso de que crees un bucle infinito, necesitarás uno de los comandos exit, break o continue para salir de ese bucle. Por ejemplo:

```
mientras : # También puedes usar
while true do
  read -p "Introduzca un valor: "
  input if [ "$input" = "exit" ];
  then
    romper
  fi
  echo "Has introducido:
```

El programa imprimirá los valores hasta que el usuario introduzca `exit`.

1.1.3 Tratamiento de ficheros con bucles

En Bash, puedes leer archivos con bucles usando varios comandos como `while` y `for`. Con `while` puede utilizar la redirección en combinación con el comando de `lectura` para procesar un archivo línea por línea:

```
filename='lo que sea.txt'
while read $LINE; do
  echo "La línea es: $LINE"
done < $nombrearchivo
```

También puede redirigir la salida de un comando a un bucle while:

```
cat fichero.txt | while read línea; do
  echo "$línea"
hecho
```

Del mismo modo, puede utilizar el bucle for para leer un archivo utilizando subcomandos:

```
for línea in $(cat /ruta/a/archivo.txt); do
  echo "$línea"
hecho
```

1.2 Exit, break y continue

En las secuencias de comandos del shell, exit, break y continue son comandos de flujo de control que permiten modificar el comportamiento de bucles y secuencias de comandos.

El comando `exit` se utiliza para salir de un shell. Este comando finaliza la sesión actual del intérprete de comandos Bash y devuelve al usuario al intérprete de comandos principal o al indicador del sistema operativo. Se puede utilizar con un argumento numérico que representa un código de estado de salida. El código de estado de salida se utiliza para indicar el éxito o el fracaso del comando o script ejecutado. Por convención, un valor de `0` representa el éxito, y cualquier valor distinto de cero representa un error o fallo.

```
if [ number -lt 0 ]; then
  exit 1 # Salir con código de estado de error
fi
echo "El número es >= 0, puede continuar"
```

`break` se utiliza para salir de un bucle. Cuando se llama a break dentro de un bucle, el bucle termina inmediatamente y el control se transfiere a la siguiente sentencia después del bucle. Esto es útil para situaciones en las que se desea salir de un bucle antes de tiempo basándose en alguna condición:

```
while true; do
  read -p "Adivina el número:" NÚMERO
  if [ $NÚMERO -eq $GUESS ]; then
    echo "Has encontrado el
    número" break
  fi
hech
```

El comando `continue` se utiliza para saltar la iteración actual de un bucle y pasar a la siguiente iteración. Esto resulta útil en situaciones en las que se desea saltar ciertas iteraciones de un bucle basándose en alguna condición.

```
echo "Imprimiendo sólo ficheros en el directorio
actual:" for fichero in *; do
  if [[ ! -f $archivo ]]; then
    continue
  fi
  echo $archivo
done
```

2. ARGUMENTO S

Hasta ahora, todo lo que hemos puesto en un script se podía hacer directamente desde la consola.

Sin embargo, cuando trabajamos con scripts podemos utilizar algo adicional, que son los argumentos de script. En Bash

los argumentos son valores que se pasan al script cuando se ejecuta. Estos argumentos permiten personalizar el comportamiento del script y hacerlo más flexible.

Para pasar argumentos a un script Bash, puedes especificarlos después del nombre del script, separados por espacios. Por ejemplo, si tienes un script llamado `mi_script.sh` y quieres pasarle dos argumentos, lo ejecutarías así:

```
./mi_script.sh -foo argumento2
```

En este ejemplo, `-foo` y `argument2` son los dos argumentos que se pasan al script.

Dentro del script Bash, puedes acceder a estos argumentos usando variables especiales, comenzando con $1 para el primer argumento, $2 para el segundo argumento, y así sucesivamente. Por ejemplo, para imprimir el primer argumento, puede utilizar el comando echo de la siguiente manera:

```
echo "El primer argumento es: $1"
echo "El segundo argumento es:
```

En el ejemplo anterior, el script imprimirá:

```
El primer argumento es -foo
El segundo argumento es: argument2
```

También puede utilizar la variable especial `$#` para obtener el número total de argumentos pasados al script, y `$0` para obtener el nombre del propio script.

3. MATERIA COMPLEMENTARIA L

Bucles en Bash

https://ryanstutorials.net/bash-scripting-tutorial/bash-loops.php

https://linuxhandbook.com/bash-loops/

Lectura de archivos con bucles

https://www.shellscript.sh/loops.html

Parámetros

https://tecadmin.net/tutorial/bash-scripting/bash-command-arguments/