Programación en Shell

Material elaborado por:

L.I. Eduardo Iván Ortega Alarcón

eduardo.ortega@unam.mx

Departamento de Supercómputo, DGTIC, UNAM.

Descripción y conceptos

- La programación en shell se utiliza para automatizar procedimientos en un entorno GNU/Linux.
- Por medio de comandos, se pueden obtener datos del sistema así como de los usuarios y procesarla para poder presentar información relevante.
- También puede ser utilizada para otros fines.
- Ayuda mucho a la administración de sistemas.
  - Ahorra tiempo.
  - Reduce la posibilidad de error.

DGTIC

Scripts en Shell

- Lenguaje interpretado.
- No se considera un lenguaje de programación.
- Utiliza un intérprete de órdenes que ejecutará las instrucciones.
- Cada instrucción se coloca en una línea y se van ejecutando secuencialmente.
- Puede utilizarse el metacaracter ; para escribir más de un comando en una línea.

DGTIC

Ejecución

- Los scripts de shell pueden ejecutarse de las siguientes formas:
  1. Asignando el permiso de ejecución al script y llamándolo directamente con su ruta relativa o absoluta:

```
$ chmod u+x ./script.sh
$ ./script.sh
```

Es recomendable escribir la línea hash-bang al inicio del script.

```
#!/bin/bash
```

DGTIC

Ejecución

2. Enviando al shell como argumento un script.

```
$ bash ./script.sh
```

3. Utilizando el comando punto.

   El comando punto se utiliza para ejecutar las instrucciones contenidas en el script dentro del mismo shell donde se está trabajando.

```
$ . ./script.sh
```

DGTIC

Evaluación de expresiones

- Comando test.
  - Compara números enteros, cadenas y verifica archivos.

  - Si la expresión es verdadera, regresa 0.
  - Si es falsa, regresa algo distinto a 0.

  - Sintaxis:

```
$ test [expresion]
```

DGTIC

Evaluación de expresiones

- Opciones para **comparación de números enteros** con test.

Opción	Uso	La expresión es verdadera cuando:
`-eq`	`test num1 -eq num2`	num1 es igual a num2.
`-ne`	`test num1 -ne num2`	num1 es diferente de num2.
`-gt`	`test num1 -gt num2`	num1 es mayor que num2.
`-lt`	`test num1 -lt num2`	num1 es menor que num2.
`-ge`	`test num1 -ge num2`	num1 es mayor o igual que num2.
`-le`	`test num1 -le num2`	num1 es menor o igual que num2.

DGTIC

Evaluación de expresiones

- Opciones para **comparación de cadenas** con test.

Opción	Uso	La expresión es verdadera cuando:
=	`test cad1 = cad2`	cad1 es igual a cad2.
!=	`test cad1 != cad2`	cad1 es diferente de cad2.
-z	`test -z cad1`	cad1 tiene una longitud de 0 caracteres (es decir, existe y esta vacía).
	`test cad1`	cad1 no es vacía.

DGTIC

Evaluación de expresiones

- Opciones para **verificación de archivos** con test.

Opción	Uso	La expresión es verdadera cuando:
`-e`	`test -e archivo`	archivo existe.
`-f`	`test -f archivo`	archivo existe y es de tipo ordinario.
`-d`	`test -d archivo`	archivo existe y es de tipo directorio.
`-b`	`test -b archivo`	archivo existe y es de tipo bloque.
`-c`	`test -c archivo`	archivo existe y es de tipo caracter.
`-p`	`test -p archivo`	archivo existe y es de tipo tubería.
`-S`	`test -S archivo`	archivo existe y es de tipo socket.
`-h`	`test -h archivo`	archivo existe y es de tipo liga simbólica.

DGTIC

Evaluación de expresiones

- Opciones para **verificación de archivos** con test.

Opción	Uso	La expresión es verdadera cuando:
`-r`	`test -r archivo`	archivo existe y se puede leer.
`-w`	`test -w archivo`	archivo existe y se puede escribir.
`-x`	`test -x archivo`	archivo existe y se puede ejecutar.
`-ef`	`test file1 -ef file2`	file1 y file2 tienen el mismo inodo.
`-nt`	`test file1 -nt file2`	file1 fue modificado más recientemente que file2.
`-ot`	`test file1 -ot file2`	file2 fue modificado más recientemente que file1.

DGTIC

Evaluación de expresiones

- Opciones para **comparaciones lógicas** con test.

Opción	Uso	La expresión es verdadera cuando:
`-a`	`test exp1 -a exp2`	exp1 y exp2 son verdaderas.
`-o`	`test exp1 -o exp2`	exp1 o exp2 son verdaderas. Si la primera expresión es verdadera, ni siquiera revisa la segunda.
`!`	`test ! exp`	exp es falsa.

- Las expresiones pueden agruparse con paréntesis.
- Recuerda escapar los paréntesis con \

DGTIC

Estructuras de control

- Instrucción if.
- Sintaxis:

```
if [ condición ]
then
   Comando 1
   Comando 2
   …
elif [ condición ]
then
   Comando 1
   Comando 2
   …
else
   Comando 1
   Comando 2
   …
fi
```

- Nota: Es importante dejar un espacio entre los corchetes y la condición. Es parte de la sintaxis. Aquí se ilustra con rojo:

```
if [ condición ]
```

- Sintaxis simple:

```
if [ condición ]
then
   Comando 1
   Comando 2
   …
fi
```

- Sintaxis con else:

```
if [ condición ]
then
   Comando 1
   Comando 2
   …
else
   Comando 1
   Comando 2
   …
fi
```

DGTIC

Estructuras de control

- Instrucción case.
- Sintaxis:

```
case $variable in
patron)
   Comando 1
   Comando 2
   …
;;
patron)
   Comando 1
   Comando 2
   …
;;
patron)
   Comando 1
   Comando 2
   …
;;
esac
```

- Se puede utilizar el carácter pipe para que se acepten 2 o más caracteres, ejemplo:

```
case $palabra in
saludo|s)
   echo Hola mundo!
;;
despedida|d)
   echo Adiós mundo cruel
;;
*)
   echo Meh.
;;
esac
```

- Como se aprecia en el ejemplo, se puede utilizar un asterisco para realizar algo cuando la variable no coincide con ningún patrón.

DGTIC

Estructuras de control iterativas

- Instrucción for.
- Sintaxis:

```
for var in lista palabras
do
   Comando 1
   Comando 2
   …
done
```

- Sintaxis

```
for (( exp1;exp2;exp3 ))
do
   Comando 1
   Comando 2
   …
done
```

DGTIC

Estructuras de control iterativas

- Instrucción while.
- Sintaxis:

```
while [ condición ]
do
   Comando 1
   Comando 2
   …
done
```

- Instrucción until.
- Sintaxis

```
until [ condición ]
do
   Comando 1
   Comando 2
   …
done
```

DGTIC

Condiciones

- Es importante mencionar que en cualquier momento donde se escribe una condición (como en el `if` o en el `while`), pueden ingresarse cualquiera de las opciones del comando test. Ejemplo:

```
if [ -f “$archivo“ –a “$numero“ –gt 0 ]
then
   echo El archivo $archivo existe y es ordinario
   echo Además el número $numero es mayor que cero
fi
```

DGTIC