Primer Script

(página 267)

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

# Primer "bash script"
# Podría traducirse como "guion de bash"
# o "secuencia de comandos de bash"

QUE='Mundo'

echo "Hola $QUE!" # Otro comentario ;-)
```

Funciones

(página 281)

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

foo=1

bar() {
    local foo=2
    echo $foo

    return
}

function baz {
    local foo=3
    echo $foo

    return
}

echo $foo
bar
baz
echo $foo
```

Control de Flujo: 'if'

(página 288)

```
if [ expresión_para_test ]; then
    comandos
elif [ expresión_para_test ]; then
    comandos
else
    comandos
fi
```

Expresiones de Enteros para `'test'`

Expresión	Es verdadero si
`entero1 -eq entero2`	`entero1 es igual a entero2.`
`entero1 -ne entero2`	`entero1 no es igual a entero2.`
`entero1 -le entero2`	`entero1 es menor o igual a entero2.`
`entero1 -lt entero2`	`entero1 es menor que entero2.`
`entero1 -ge entero2`	`entero1 es mayor o igual a entero2.`
`entero1 -gt entero2`	`entero1 es mayor que entero2.`

Expresiones de Archivos para `'test'`

Expresión	Es verdadero si
archivo1 -ef archivo2	archivo1 y archivo2 tienen los mismos números de inodo (los dos nombres de archivo se refieren al mismo archivo por enlace duro).
archivo1 -nt archivo2	archivo1 es más nuevo que archivo2.
archivo1 -ot archivo2	archivo1 es más antiguo que archivo2.
-b archivo	archivo existe y es un archivo con bloqueo especial (dispositivo).
-c archivo	archivo existe y es un archivo de caracter especial (dispositivo).
-d archivo	archivo existe y es un directorio.
-e archivo	archivo existe.
-f archivo	archivo existe y es un archivo normal.
-g archivo	archivo exite y tiene establecida una ID de grupo.
-G archivo	archivo exite y su propietario es el ID de grupo efectivo.
-k archivo	archivo exite y tiene establecido su "sticky bit"
-L archivo	archivo existe y es un enlace simbólico.
-O archivo	archivo existe y su propietario es el ID de usuario efectivo.
-p archivo	archivo existe y es un entubado con nombre.
-r archivo	archivo existe y es legible (tiene permisos de lectura para el usuario efectivo).
-s archivo	archivo existe y tiene una longitud mayor que cero.
-S archivo	archivo existe y es una conexión de red
-t fd	fd es un descritor de archivo dirigido de/hacia el terminal. Puede usarse para determinar que entrada/salida/error estándar está siendo redirigido.
-u archivo	archivo existe y es setuid
-w archivo	archivo existe es es editable (tiene permisos de escritura para el usuario efectivo).
-x archivo	archivo existe y es ejecutable (tiene permisos de ejecución/búsqueda para el usuario efectivo).

Expresiones de Archivos para `'test'`

Expresión	Es verdadero si
archivo1 -ef archivo2	archivo1 y archivo2 tienen los mismos números de inodo (los dos nombres de archivo se refieren al mismo archivo por enlace duro).
archivo1 -nt archivo2	archivo1 es más nuevo que archivo2.
archivo1 -ot archivo2	archivo1 es más antiguo que archivo2.
-b archivo	archivo existe y es un archivo con bloqueo especial (dispositivo).
-c archivo	archivo existe y es un archivo de caracter especial (dispositivo).
-d archivo	**archivo existe y es un directorio.**
-e archivo	archivo existe.
-f archivo	**archivo existe y es un archivo normal.**
-g archivo	archivo exite y tiene establecida una ID de grupo.
-G archivo	archivo exite y su propietario es el ID de grupo efectivo.
-k archivo	archivo exite y tiene establecido su "sticky bit"
-L archivo	**archivo existe y es un enlace simbólico.**
-O archivo	archivo existe y su propietario es el ID de usuario efectivo.
-p archivo	archivo existe y es un entubado con nombre.
-r archivo	**archivo existe y es legible (tiene permisos de lectura para el usuario efectivo).**
-s archivo	**archivo existe y tiene una longitud mayor que cero.**
-S archivo	archivo existe y es una conexión de red
-t fd	fd es un descritor de archivo dirigido de/hacia el terminal. Puede usarse para determinar que entrada/salida/error estándar está siendo redirigido.
-u archivo	archivo existe y es setuid
-w archivo	**archivo existe es es editable (tiene permisos de escritura para el usuario efectivo).**
-x archivo	**archivo existe y es ejecutable (tiene permisos de ejecución/búsqueda para el usuario efectivo).**

Expresiones de Cadenas de Caracteres para `'test'`

Expresión	Es verdadero si
`cadena`	`cadena no es nula.`
`-n cadena`	`La longitud de cadena es mayor que cero.`
`-z cadena`	`La longitud de cadena es cero.`
`cadena1 = cadena2` `cadena1 == cadena2`	`cadena1 y cadena2 son iguales. Pueden usarse signos igual simples o dobles, pero es preferible usar dobles signos igual.`
`cadena1 != cadena2`	`cadena1 y cadena2 no son iguales.`
`cadena1 > cadena2`	`cadena1 se ordena detrás de cadena2.`
`cadena1 < cadena2`	`cadena1 se ordena antes de cadena2.`

[[ ]]

Evalúa si un resultado es verdadero o falso. Soporta todas las expresiones de `'test'` y añade una nueva expresión de cadenas: `cadena1 =~ regex`

(( ))

Es útil para operar con enteros. Se usa para realizar pruebas de veracidad aritmética. Una prueba de veracidad aritmética es verdadera si el resultado de la evaluación aritmética no es cero.

Operadores Lógicos

Operación	test	[[ ]] y (( ))
AND	-a	&&
OR	-o	||
NOT	!	!

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

echo -n "Ingrese un entero:  "
read entero

if [ $entero -eq 5 ]; then
    echo "Igual a 5"
else
    echo "Distinto de 5"
fi
```

Control de flujo: 'while / until'

(página 309)

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

contador=1

while [[ $contador -le 10 ]]; do
    echo $contador
    contador=$((contador + 1))
    if [[ $contador -eq 6 ]]; then
        break
    fi
done

echo "Fin de script"
```

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

contador=1

until [[ $contador -gt 5 ]]; do
    echo $contador
    contador=$((contador + 1))
done

echo "Fin de script"
```

Control de Flujo: 'case'

(página 323)

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

read -p "Ingrese un entero [1-3]: " ENTERO

case $ENTERO in
    1)
        echo "Uno"
    ;;
    2|3)
        echo "Dos o Tres"
    ;;
    *)
        echo "Error!"
        exit 1
    ;;
esac

echo "Fin de script"
```

Control de flujo: `'for'`

(página 338)

```
for variable [in palabras]; do
    comandos
done
```

```
for (( inicializar; condición; actualizar )); do
    comandos
done
```