Scripts

2° parte

1.
Variables

Concepto – Declaración – Captura de comandos y operaciones

Una variable es un espacio donde se guarda (y se recupera) datos que se utilizan en un programa

Cuando escribimos código, las variables se utilizan para:

- *Guardar datos y estados.*
- *Asignar valores de una variable a otra.*
- *Representar valores en una expresión matemática.*
- *Mostrar valores por pantalla.*

Declaración de variables

```
#!/bin/bash

mensaje=”Hola Mundo”
echo $mensaje
mensaje=52
echo “Faltan $mensaje para el verano”
```

- No dejar espacios ni antes ni después del **=**
- Los nombres de las variables son case sensitive
- Para espacios o caracteres especiales en una variable, usar barra inversa para anular el comportamiento especial de un caracter o bien comillas dobles.
- Para usar el valor de una variable hay que anteponer $ antes del nombre

Declaración de variables: captura de un valor

```
#!/bin/bash

fecha=`date`
actual=$(pwd)
echo “Hoy $fecha”
echo “Directorio $actual”
```

```
#!/bin/bash

numero=50
suma=`expr $1 + $2`
total=$(($suma+$numero))
Echo “Total: $total”
```

- Un comando con sus parámetros (puede ser la llamada a otro script) debe ir encerrado entre acentos graves o precedido por $ entre ().
- El resultado del comando queda almacenado dentro de la variable, aunque consista en varias líneas de texto.

Declaración de variables: read

```
#!/bin/bash

echo “Ingrese una opción: ”
read opcion
echo “Opción elegida: $opcion ”
```

≈

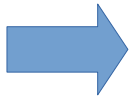
```
#!/bin/bash

read -p “Ingrese una opción: ” opcion
echo “Opción elegida: $opcion ”
```

- Lee una línea de la entrada estándar y asigna el valor ingresado a la variable que siga al comando read.
- Con la opción –p puede especificarse un mensaje que se presentará como prompt antes de empezar a leer.

Variable no inicializada

```
echo “Introduzca su nombre”
read nombre
if [ $nombre = "Antonio" ] then
  echo “Bienvenido Jefe”
else
  echo Hola $nombre
fi
```



```
echo “Introduzca su nombre”
read nombre
if [ “$nombre” = "Antonio" ] then
  echo “Bienvenido Jefe”
else
  echo Hola $nombre
fi
```

- Si en el código se hace referencia a una variable que no inicializada, en general no ocurrirá ningún error, sólo será sustituida por una cadena vacía.
- Esto puede generar un error en tiempo de ejecución si al sustituir la variable por “nada” la sintaxis del comando en el que aparece deja de ser correcta.
- Se soluciona encerrando la variable entre comillas dobles, no se estropea la sintaxis de la comparación

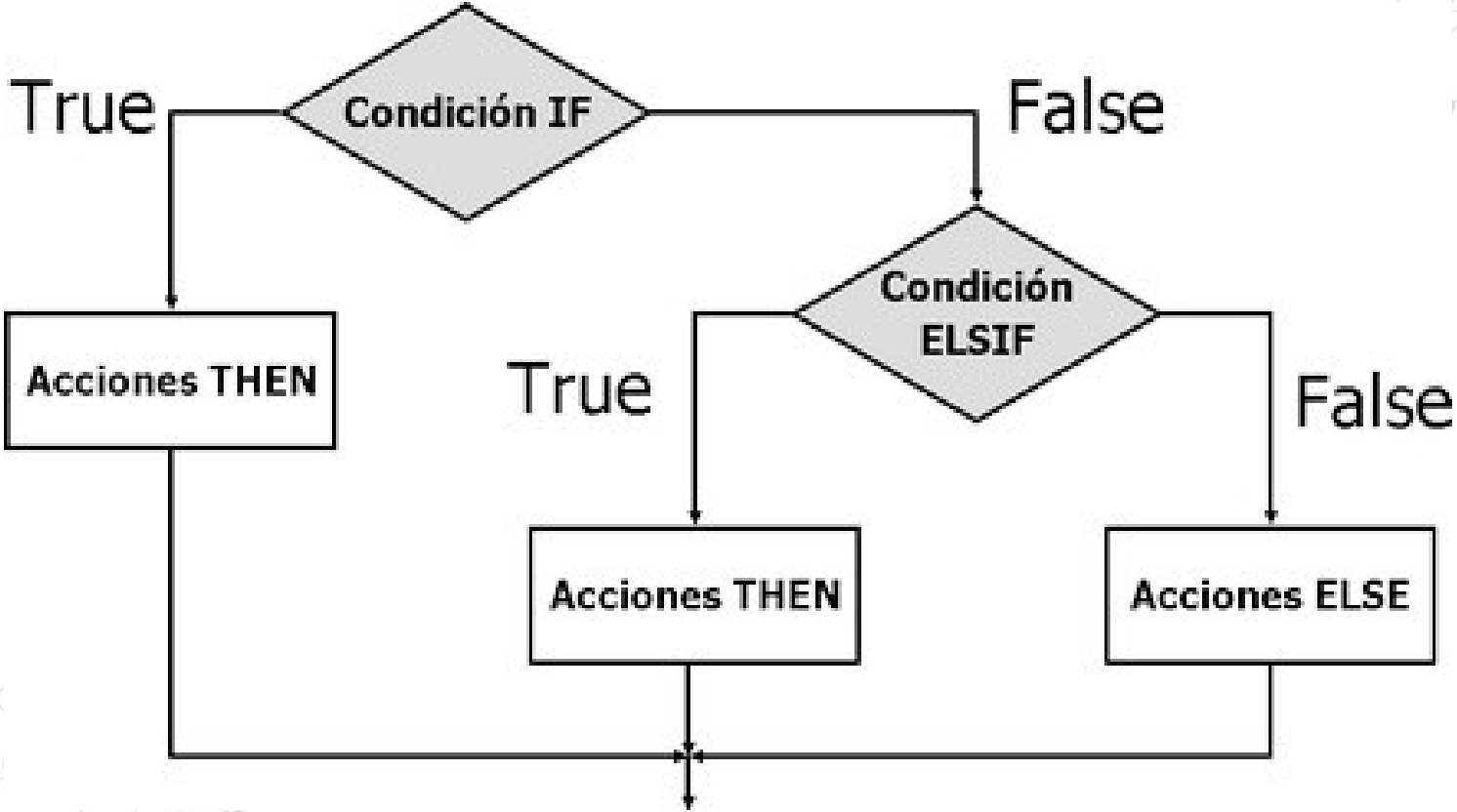
2.
Estructuras

Estructuras para controlar el flujo de ejecución de un script

Estructuras

estructura	funcionamiento
if / else	Ejecuta una serie de comandos dependiendo si una condición se cumple o no.
case	Ejecuta una o varias listas de comandos dependiendo del valor de una variable
while	Ejecuta una serie de comandos mientras una condición se cumpla.
until	Ejecuta una serie de comandos hasta que una condición se cumpla.
for	Ejecuta una serie de comandos un número determinado de veces.

Diagrama de flujo if / else



if / else

ESTRUCTURA

```
if condición1 then
   sentencias ...
[elif condición2 then
   sentencias ...]
[else
   sentencias ...]
fi
```

EJEMPLO

```
read opcion
if [ $opcion -eq 1 ]; then
    echo "Buenos días"
elif [ $opcion -eq 2 ]; then
    echo "Buenas tardes"
elif [ $opcion -eq 3 ]; then
    echo "Buenas noches"
else
    echo "Adiós"
fi
```

Ejemplo

```
#!/bin/bash

pin="1234"
echo "Ingrese su pin"
read clave

if [ "$clave" = "$pin" ];  then
        echo "Pin correcto"
        echo "Acceso permitido"
else
        echo "Pin incorrecto"
fi
```

```
#!/bin/bash


if [ -d "$1" ]; then
  echo "Es un directorio"
elif [ -f "$1" ]; then
  echo "Es un archivo"
else
  echo "No es directorio ni archivo"
fi
```

case

ESTRUCTURA

```
case expresión in
    caso_1
     sentencias;;
    caso_2
     sentencias;;
. . .
esac
```

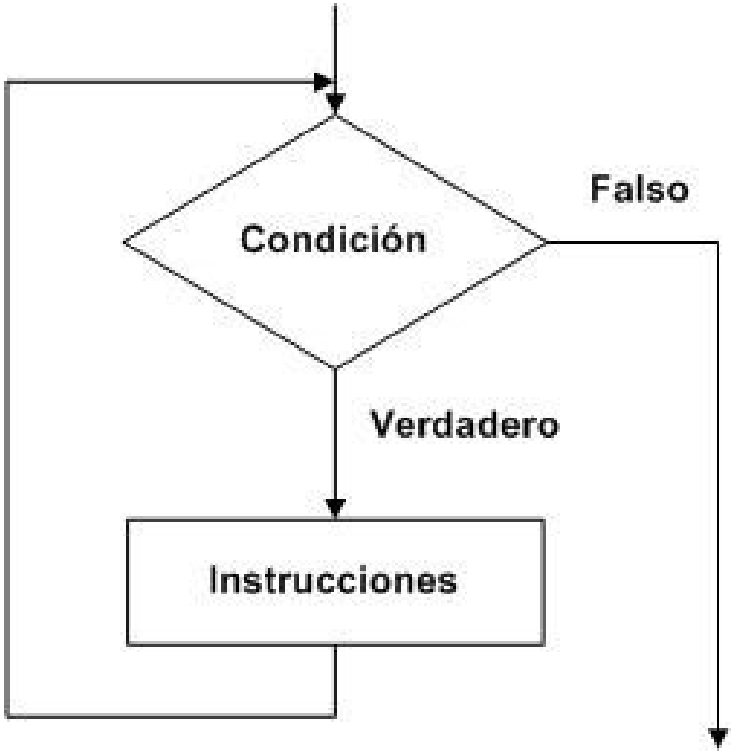
EJEMPLO

```
read opcion
case $opcion in
1) echo "Buenos días" ;;
2) echo "Buenos tardes" ;;
3) echo "Buenos noches" ;;
*) echo "Adiós" ;;
esac
```

Ejemplo case: menú

```
#!/bin/bash
clear
opcion=0
until [ $opcion -eq 4 ]
do
echo "-------MENU------"
echo "1-Mostrar fecha y hora"
echo "2-Mostrar calendario 2020"
echo "3-Mostrar mis procesos "
echo "4-Salir"
echo  "------>Ingrese opcion: "
read opcion
    case $opcion in
        1) date ;;
        2) cal 2020 ;;
        3) ps x ;;
        4) echo "Fin del programa" ;;
        *) echo "Opcion incorrecta" ;;
    esac
done
```

Diagrama de flujo while



while

ESTRUCTURA

```
while condición
do
    Sentencia1
    Sentencia2
      . . .
    Sentencia n
done
```

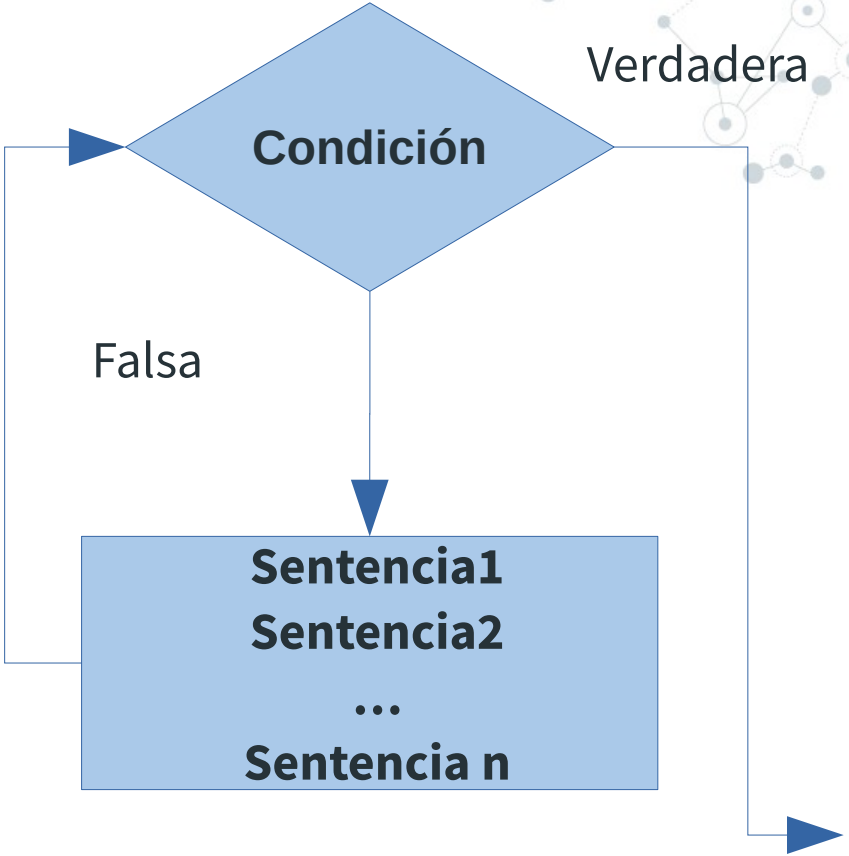
EJEMPLO

```
#!/bin/bash
echo "La suma finaliza al ingresar 0"
numero=1
suma=0
while [ $numero -ne 0 ]
do
      echo -n "Ingrese un numero: "
      read  numero
      suma=$(($suma+$numero))
done
echo "La suma es:" $suma
```

until

ESTRUCTURA

```
until condición
do
    sentencia1
    sentencia2
      . . .
done
```



Ejemplos until

```
#!/bin/bash
        contador=0
        until [  $contador -eq 10 ]; do
            echo “CONTADOR $contador”
           contador=$(($contador + 1))
        done
```

for

ESTRUCTURA

```
for var in 1 2 3 .. N
do
    sentencias
      . . .
done
```

EJEMPLO

```
#!/bin/bash
lista="rojo blanco azul verde
amarillo"
echo "Lista de Colores"

for nombre in $lista
do
    echo $nombre
done
```

Ejemplos: Bucle finito

```
#!/bin/bash

for i in 1 2 3 4 5 6 7 8 9
do
    echo "Bucle N° $i"
done
```

Ejemplos: Tabla de multiplicar

```
#!/bin/bash

echo "Ingrese un número: "
read n
echo
echo "**Tabla de multiplicar de $n **"
for i in 1 2 3 4 5 6 7 8 9 10
do
   producto=`expr $n "*" $i`
   echo "$n x $i= " $producto
done
```


¿Preguntas?