Los Scripts son archivos de textos interpretados. Para estos ejemplos utilizaremos el editor nano. Si no lo tienes instalado y usas Debian o Ubuntu ejecuta el siguiente comando:

```
$ sudo apt-get install nano
```

Ya instalado podemos continuar. Abrimos una terminal y escribimos.

```
$nano primero
```

Al abrir el editor nano escribimos dentro de él:

```
#!/bin/bash
# Ejemplo 1: Primer script !
## Ejemplo 1
echo "Primer script!"
```

Con la combinación de teclas Ctrl+o guardamos y con Ctrl+x salimos del editor. Lo importante es darle permisos de ejecución al archivo de esta forma

```
$chmod 755 ejemplo
```

Entonces lo ejecutamos para ver que es lo que nos devuelve el scripts para esto se ejecuta así

```
$./primero
```

Esto es lo que aparece

```
Primer script!
```

Variables

Para definir variables dentro de un script.

```
fijo1=10
fijo2=20
echo $fijo1
echo $fijo2
echo "El número fijo de la primera variable es $fijo1"
echo "El número fijo de la segunda variable es $fijo2"
echo "Las dos Variables son $fijo1 $fijo2"
```

Le damos permisos de ejecución y lo ejecutamos:

```
$./variables
```

Salida:

```
10
20
El número fijo de la primera variable es 10
El número fijo de la segunda variable es 20
Las dos Variables son 10 20
```

Agregar argumentos por línea de comandos

```
echo "Ejecutando= $0"
echo "Primer Argumento = $1"
echo "Segundo Argumento = $2"
echo "Tercer Argumento = $3"
echo "Todos los argumentos: $*"
```

NOTA: la variable $0 guarda el nombre del programa.

Damos permisos al fichero y lo ejecutamos de esta forma
$./argumentos uno dos tres

```
Ejecucion de ./argumentos
Primer Argumento = uno
Segundo Argumento = dos
Tercer Argumento = tres
Todos los argumentos: uno dos tres
```

Ingreso de datos

Para el ingreso de datos incluimos el comando read. Entonces:

```
echo ""
echo " EJEMPLO DE INGRESO DE DATOS"
echo -n "Pon el día de tu nacimiento: "
read dia
echo -n "Pon el mes de tu nacimiento: "
read mes
echo -n "Pon el Año de tu nacimiento:"
read numero
echo "La Fecha de tu nacimiento"
echo "El día: $dia"
echo "El mes : $mes"
echo "El Año : $numero"
echo ""
echo "La fecha completa es:"
echo "El día $dia en el mes de $mes del año $numero "
```

Lo guardamos
Le damos permisos
Lo ejecutamos ./ingreso

La salida es:

```
EJEMPLO DE INGRESO DE DATOS
Pon el día de tu nacimiento: 30
Pon el mes de tu nacimiento: abril
Pon el Año de tu nacimiento:1976
La Fecha de tu nacimiento
El día: 30
El mes : abril
El Año : 1976

La fecha completa es:
El día 30 en el mes de abril del año 1976
```

Evaluar expresiones

En este ejemplo sumaremos 2 expresiones. Entonces:

```
echo ""
echo " EJEMPLO DE SUMA DE DATOS"
echo " Suma de 2 valores "
echo -n "Pon el primer valor a sumar: "
read numero1
echo -n "Pon el segundo valor a sumar: "
read numero2
numero3=$[numero1+numero2]
echo ""
echo "Los dos números a sumar serán $numero1 más el $numero2"
echo "El total es = $numero3 "
```

Guardamos, damos permisos y ejecutamos

```
EJEMPLO DE SUMA DE DATOS
Suma de 2 valores
Pon el primer valor a sumar: 20
Pon el segundo valor a sumar: 30
```

```
Los dos números a sumar serán 20 más el 30
El total es = 50
```

** Se pueden ocupar operadores distintos como
+ - suma o resta
! negación
* / % multiplicación, división y resto
y muchas mas

Estructuras condicionales

En este ejemplo mostraremos distintos mensajes según la una condición establecida.

Creamos el archivo:
```
$nano condiciones
```

Escribimos el siguiente código:

```
echo ""
echo " EJEMPLO DE CONDICIÓN PARA COMPARAR SI UN NÚMERO ES IGUAL 40"
echo ""
echo -n " Ingresa un valor para compararlo con 40 = "
read valor
if [ $valor = 40 ]
then
  echo ""
  echo " El valor que ingresaste es igual 40 "
  echo ""
else
  echo ""
  echo " El valor que ingresaste es no es igual a 40"
  echo ""
fi
```

Guardamos, damos permisos y ejecutamos
./condiciones

Entonces nos devuelve.

```
EJEMPLO DE CONDICIÓN PARA COMPARAR SI UN NÚMERO ES IGUAL 40
Ingresa un valor para comparlo con 40 = 40
El valor que ingresaste es igual 40
```

Bash como programa tiene algunos argumentos útiles y propios que se usan con frecuencia en la elaboración de Scripts en los condicionales vinculados a la determinación de elementos sobre los archivos, variables, cadenas de palabras o cadenas de pruebas, los mas comunes son:

1. Argumentos de Archivos

-d ----------> Archivo existe y es un directorio
-c ----------> Archivo existe y es de caracteres
-e ----------> Archivo existe
-h ----------> Archivo existe y es un vínculo simbólico
-s ----------> Archivo existe y no está vacío
-f ----------> Archivo existe y es normal
-r ----------> Tienes permiso de lectura del archivo
-w ----------> Tienes permiso de escritura en el archivo
-x ----------> Tienes permiso de ejecución del archivo

-O ----------> Eres propietario del archivo
-G ------- ---> Perteneces al grupo que tiene acceso al archivo
-n ----------> Variable existe y no es nula
Archivo1 nt Archivo2 ---------> Archivo1 es mas nuevo que Archivo2
Archivo1 -ot Archivo2 --------> Archivo1 es mas viejo que Archivo2

2. Argumentos de cadenas

-z ------------------------> La cadena está vacía
-n ------------------------> La cadena no está vacía
cadena1 = cadena2 ---------> Si las cadenas son iguales
cadena1 != cadena2 --------> Si las cadenas son diferentes
cadena1 <> Si la cadena 1 va antes en el orden lexicográfico
cadena1 >cadena2 ----------> Si la cadena 1 va después en el orden lexicográfico

Ejemplo con argumentos para cadenas:
Escribe el siguiente script y comprueba la salida. Modifica el script para que la salida sea "Son iguales".

```
#!/bin/bash
VAR1=Pablo
VAR2=Pedro
if [ "$VAR1" = "$VAR2" ]; then
echo Son iguales
else
echo Son diferentes
fi
```

Ejemplo con argumentos para ficheros:
Escribe el siguiente script y comprueba la salida.

```
#!/bin/bash
DIR=~/fotos
if [ ! -d "$DIR" ]; then
mkdir "$DIR"
if [ $? -eq 0 ]; then
echo "$DIR" ha sido creado..."
else
echo "Se produce un error al crear "$DIR"
fi
else
echo "Se usará "$DIR" existente"
fi
```

NOTA: La variable $? da 0 si no ha habido errores en la ejecución.

3. Argumentos de números:

-eq (n1 igual a n2)
-ge (n1 mayor o igual a n2)
-le (n1 menor o igual a n2)
-ne (n1 no igual a n2)
-gt (n1 mayor que n2)
-lt (n1 menor que n2)

Ejemplos de expresiones de comparación numérica:
– [n1 -eq n2]
– [n1 -ge n2]
– [n1 -le n2]

– [n1 -ne n2]
– [n1 -gt n2]
– [n1 -lt n2