



Scripting en Bash



¿Qué es un script?

Definición

Un script es un archivo que contiene una secuencia de comandos que, al ejecutarse, automatiza tareas repetitivas.

Extensiones comunes

En Windows: .bat o .cmd. En Linux: .sh

Características de los Scripts

Automatización

Reutilización

Simplicidad

Flexibilidad

Elementos de un script

Shebang

Indica el intérprete que se utilizará para ejecutar los comandos del script.

Estructura: `#!/(ubicación interprete)`

Ejemplo: `#!/bin/bash`

Comentarios

Inician con el símbolo `"#"` y sirven para explicar el funcionamiento del script.

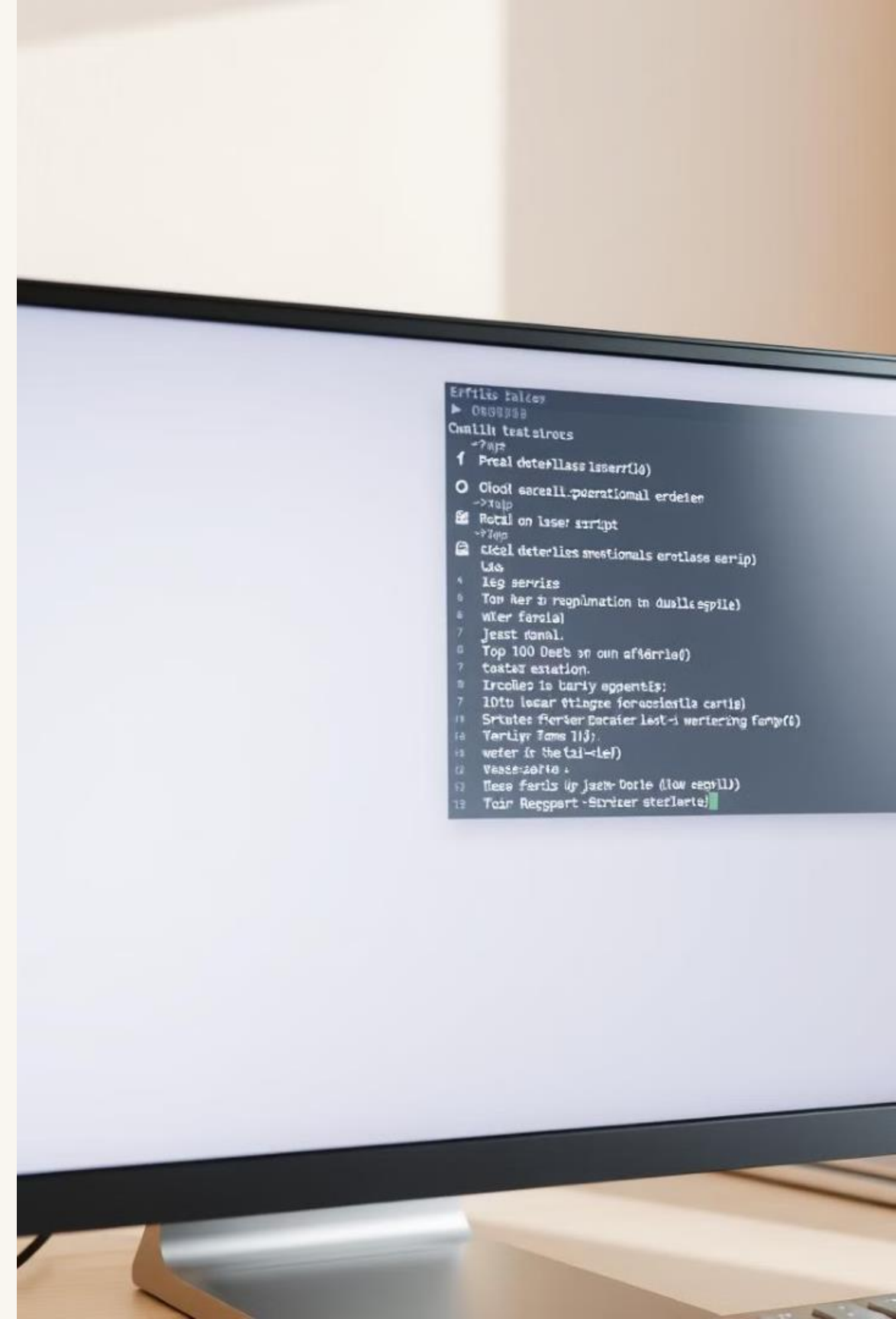
Comandos

Son las instrucciones que se ejecutan cuando el script se ejecuta.

Ejemplo: `echo Hola Mundo`

Importancia del Shebang

- 1 Ejecución directa
- 2 Portabilidad
- 3 Flexibilidad



Creación de un script

Paso 1

Crear un archivo con extensión ".sh". Por ejemplo: "hola.sh"

Podemos utilizar

- nano
- vi
- emacs

Paso 3

Dar permisos de ejecución al archivo con el comando:
`chmod +x ./hola.sh`

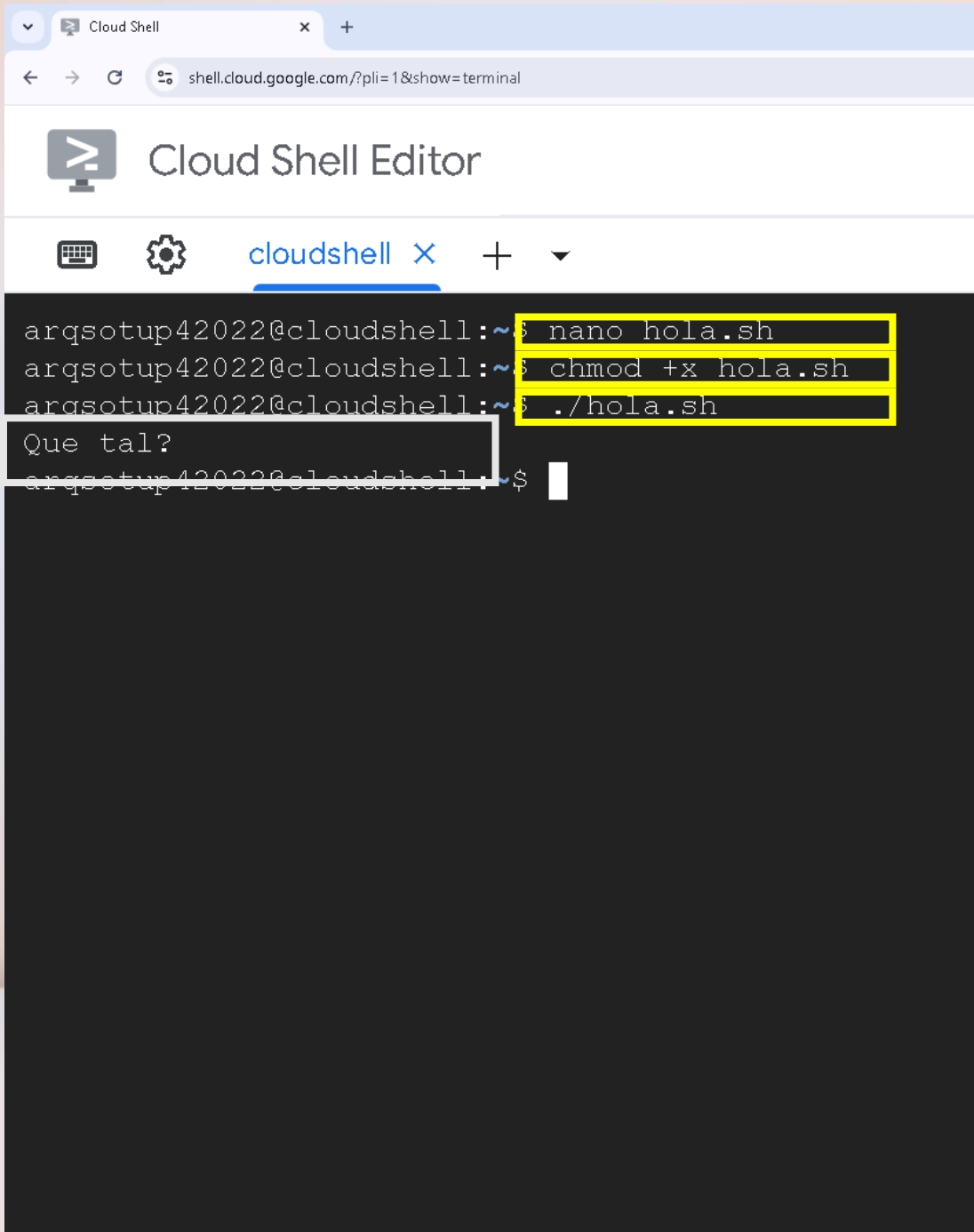
Paso 2

Escribir los comandos del script en el archivo. Ejemplo:

```
#!/bin/bash
# Primer programa
echo Que tal?
```

Paso 4

Ejecutar el script: `./hola.sh`



The screenshot shows a Cloud Shell terminal window. The terminal output is as follows:

```
argsotup42022@cloudshell:~$ nano hola.sh
argsotup42022@cloudshell:~$ chmod +x hola.sh
argsotup42022@cloudshell:~$ ./hola.sh
Que tal?
argsotup42022@cloudshell:~$
```

The commands `nano hola.sh`, `chmod +x hola.sh`, and `./hola.sh` are highlighted with yellow boxes. The output of the script is "Que tal?".

Variables en Bash



Espacios de Memoria

Las variables son espacios en memoria donde se almacenan datos que se utilizan en el script.



Declaración

nombre_variable=valor_variable



Acceso

Se accede al valor de la variable usando el símbolo dólar seguido del nombre: ***\$nombre***



Variables en Bash - Condiciones

Sólo puede contener caracteres alfanuméricos y guiones bajos “_”

El primer carácter debe ser una letra del alfabeto o “_”.

Algunos nombres son reservados por el sistema como variables de entorno y no se deben utilizar (Ej: PATH).

Es case sensitive
(distingue mayúsculas y minúsculas).

miVariable no es igual **a mivariable**

No pueden contener espacios.




```

19  rloy nala controlat:
18  chey:
16  print eants(aring,'Decday) - (apt pooker))
10  <-      - uhor une codes
11  uinious.ply raare iorgole andejnts round tutt the fatyle.
13  uintole.phy from. Potvact in highstatle procinte fracline: Rith Soction;
12  uinistaceSthaloge follier usl. del ty vent.
23  uinious.ply raare Trrnifjion wist fromy to tonu.
21  uintove.phy from. Parrrtable Mustecortionl; ( - sotrie;
23  uiniule.alyccthbiration; - resseues:
    uiniscanglylcable loo:
    )

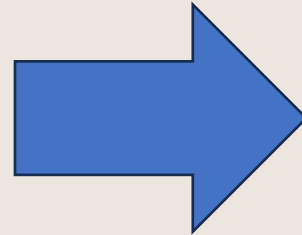
```

Ejemplo práctico

```

#!/bin/bash
a_imprimir='Hola mundo'
echo $a_imprimir
a_imprimir=5.5
echo $a_imprimir

```



```

arqsotup42022@cloudshell:~$ cat ejemplo_variable.sh
#!/bin/bash
a_imprimir='Hola mundo'
echo $a_imprimir
a_imprimir=5.5
echo $a_imprimir

arqsotup42022@cloudshell:~$ ./ejemplo_variable.sh
Hola mundo
5.5
arqsotup42022@cloudshell:~$

```



Operaciones Aritméticas

Expansión Aritmética

`$((expresión))`

Suma y resta

```
echo $((num + 2))
```

Potencia

```
echo $((num ** 2))
```

Multiplicación, división, módulo

```
echo $((num * 2))
```

```
echo $((num / 2))
```

```
echo $((num % 2))
```

Incremento y decremento (++ y --)

```
echo $((num++))
```

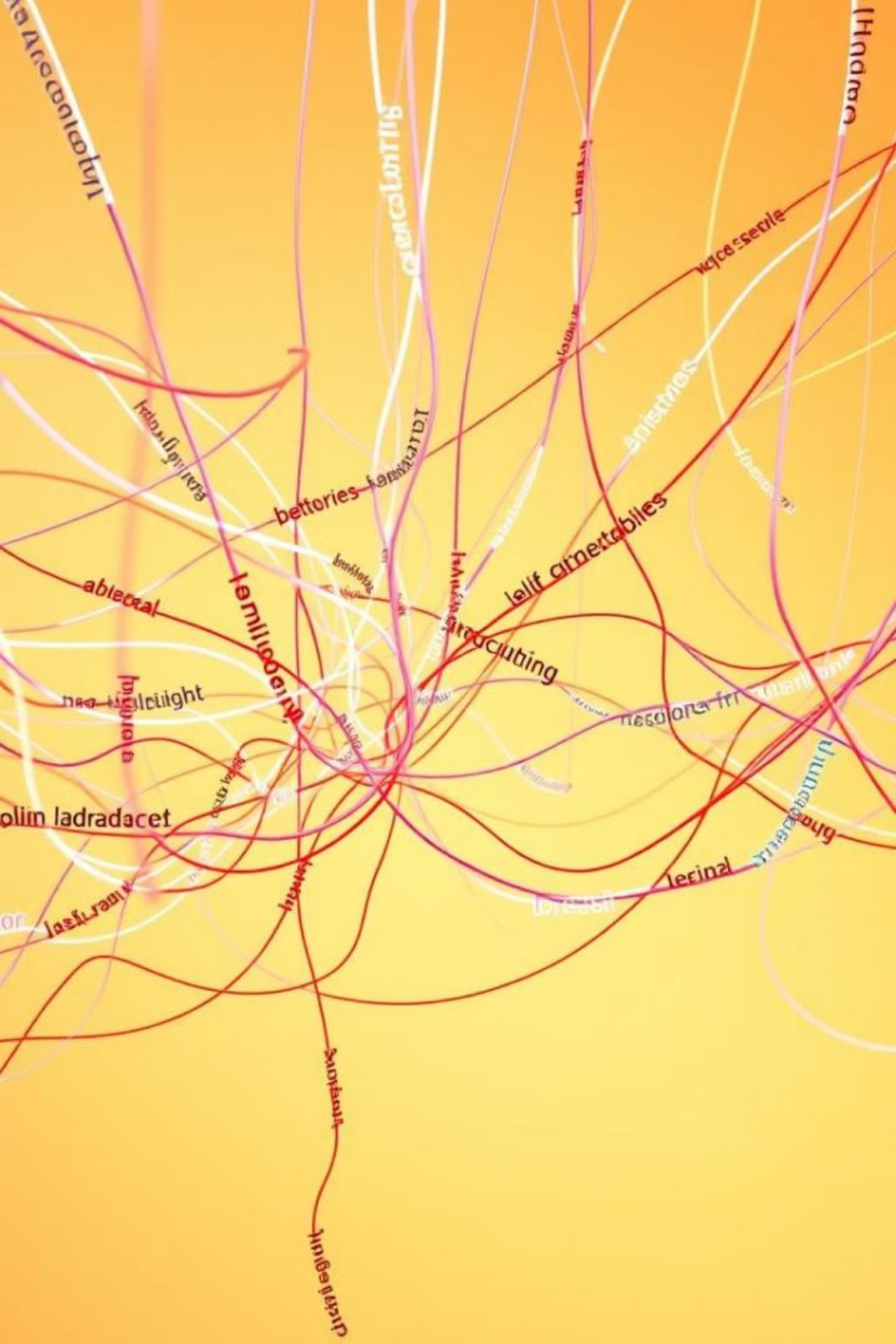
```
echo $((++num))
```



Ejemplo con ++

```
argsotup42022@cloudshell:~$ cat operaciones.sh
#!/bin/bash

num=15
echo Valor original: $num
echo Primero aplico num++: $((num++))
echo El resultado es: $num
echo Despues aplico ++num: $((++num))
echo El resultado es: $num
argsotup42022@cloudshell:~$
argsotup42022@cloudshell:~$ ./operaciones.sh
Valor original: 15
Primero aplico num++: 15
El resultado es: 16
Despues aplico ++num: 17
El resultado es: 17
argsotup42022@cloudshell:~$
```

Manipulación de cadenas de texto (strings) en Bash

Extraer subcadenas

Sintaxis

`${cadena:posicion:longitud}`

Ejemplo

```
echo ${cadena:0}
echo ${cadena:0:1}
echo ${cadena:7}
echo ${cadena:7:3}
echo ${cadena:7:-3}
echo ${cadena: -4} (espacio antes de -4)
echo ${cadena: -4:2} (espacio antes de -4)
```

Cadena a manipular

abcABC123ABCabc

Salida

abcABC123ABCabc

a

23ABCabc

23A

23ABC

Cabc

Ca

```
for tet (reatave substriant, y00);  
fort rirmove-:ls sublsrinnentrines,y0);  
  
for tet (reatave slates-repariant y00)
```

Borrar subcadenas buscando patrones

Ejemplo: cadena=abcABC123ABCabc

Borrado corto

`${cadena#subcadena}`

~~abc~~ABC123ABCabc

`echo ${cadena#a*C} : 123ABCabc`

Borrado largo

`${cadena##subcadena}`

~~abcABC123ABC~~abc

`echo ${cadena##a*C} : abc`



Reemplazar subcadenas

Ejemplo: cadena=abcABC123ABCabc

Sustituye primer coincidencia

`${cadena/buscar/reemplazar}`

abcABC123ABCabc

`echo ${cadena/abc/xyz} : xyzABC123ABCabc`

Sustituye todas las coincidencias

`${cadena//buscar/reemplazar}`

abcABC123ABCabc

`echo ${cadena//abc/xyz} : xyzABC123ABCxyz`

Seguimos manipulando strings

1

Convertir a minúsculas

`${cadena,,}`

Ejemplo: cadena="Hola, Que Tal?"



"hola, que tal?"

2

Convertir a mayúsculas

`${cadena^^}`



"HOLA, QUE TAL?"

Código:

```
argsotup42022@cloudshell:~$ cat mayuminu.sh
#!/bin/bash
x="Hola, Que Tal?"
echo $x # Imprime el original
y=${x,,}
echo $y # Pasado a minúsculas
z=${y^^}
echo $z # Pasado a mayúsculas
argsotup42022@cloudshell:~$ ./mayuminu.sh
Hola, Que Tal?
hola, que tal?
HOLA, QUE TAL?
argsotup42022@cloudshell:~$
```

El comando read

Ejemplo 1 – Mostrar un mensaje al usuario

```
read -p "Introduce tu nombre: " nombre  
echo "Hola, $nombre!"
```

Ejemplo 2 – Ocultar entrada de usuario

```
read -s -p "Introduce tu contraseña: " contraseña  
echo "Contraseña introducida: $contraseña"
```

Ejemplo 3 – Definir un tiempo límite

```
read -t 5 -p "Introduce un número en 5 segundos: " numero  
echo "El número introducido es: $numero"
```

