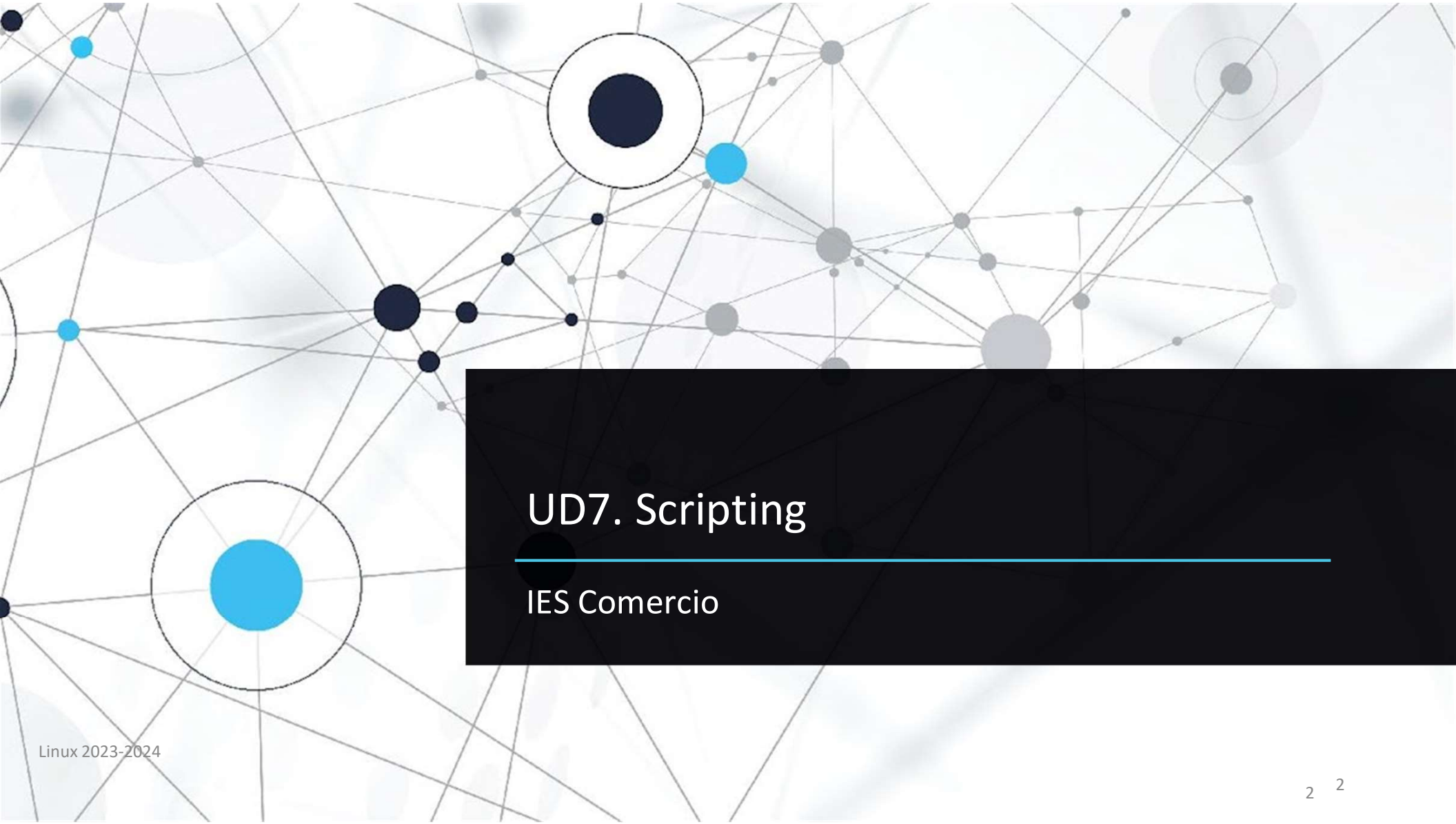


UD7. Scripting

IES Comercio

SCRIPTING

INTRODUCCIÓN

¿Qué es un script?

Un script es un **archivo de texto** que contiene una serie de **comandos** que pueden ser **ejecutados en secuencia***.

Estos comandos pueden ser:

- simples (creación de un directorio, copia de archivos)
- complejos (procesamiento de datos, automatización de tareas repetitivas)

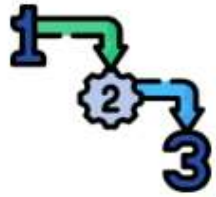
- del SO (gestor de arranque)
- del usuario

Modos de ejecución

❖VARIOS COMANDOS

1. EJECUCIÓN SECUENCIAL

```
comando1 ; comando2 ; comando3
```



2. EJECUCIÓN CONDICIONADA

```
comando1 && comando2
comando3 || comando4
```

Modos de ejecución

❖EN FUNCIÓN DEL PLANO DE EJECUCIÓN

1. SECUENCIA EN PRIMER PLANO (FOREGROUND)

```
comando1 ; comando2 ; comando3 ; …
```

2. SECUENCIA EN SEGUNDO PLANO (BACKGROUND)

```
comando4 &
```

¿Para qué se usa el scripting?

Los usuarios pueden realizar fácilmente tareas que de otra manera serían **tediosas** o requerirían **muchos pasos** manuales:

- ✓ automatizar tareas
- ✓ simplificar procesos repetitivos
- ✓ realizar operaciones complejas de manera eficiente

Ejemplos de scripting

1. Automatización de copias de seguridad periódicas

2. Procesamiento/análisis de archivos de registro (xej: buscar de ciertos patrones y generar informes automáticamente)

3. Configuración del sistema
(xej: configurar automáticamente el entorno de trabajo de un usuario, instalando programas, configurando variables de entorno, etc.)

SCRIPTING

PASOS PARA EJECUTAR UN SCRIPT

1. Escribe tu script

- extensión **.sh**
  (xej: **mi_script.sh**)

- línea de shebang en la parte superior del script
  (xej Bash: **#!/bin/bash**)

Editores en CLI

```
nano nombre_del_archivo
```

```
vi nombre_del_archivo
```

```
vim nombre_del_archivo
```

```
emacs nombre_del_archivo
```

2. Dale permiso de ejecución

```
$ chmod +x nombre_script.sh
```

```
chmod +x mi_script.sh
```

3. Ejecuta tu script

a. Ejecución directa desde la línea de comandos

```
$ ruta*/nombre_script.sh
```

```
./mi_script.sh
```

b. Especificando el intérprete manualmente

```
$ sh nombre_script.sh
```

```
sh mi_script.sh
```

```
$ bash nombre_script.sh
```

```
bash mi_script.sh
```

*Si el directorio actual ‘que contiene el script' no está en el PATH, para ejecutar el script hay que indicar donde se encuentra

c. Programación tareas

```
0 2 * * * /ruta/nombre_script.sh
```

```
at now + 5 /ruta/nombre_script.sh
```

d. Usando el comando source

```
$ source nombre_script.sh
```

```
$ . nombre_script.sh
```

e. Programación de tareas con systemd

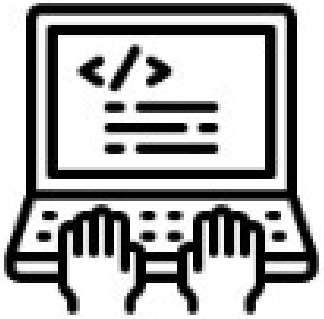
/etc/systemd/system/mi_script.service

```
[Unit]
Description=Mi script de ejemplo

[Service]
ExecStart=/ruta/a/mi_script.sh

[Install]
WantedBy=multi-user.target
```

```
$ sudo systemctl enable mi_script.service
```

```
$ sudo systemctl start mi_script.service
```

SCRIPTING



Programación del Script

Conceptos básicos

1. Shebang (#!)
2. Variables
3. Operadores
4. Estructuras de control
5. Funciones

Conceptos básicos:

1.Shebang (#!):

Es la primera línea de un script y especifica qué intérprete de comandos debe ser usado para ejecutar el script. Por ejemplo, **#!/bin/bash** indica que el script debe ser ejecutado usando Bash.

2.Variables:

Son contenedores para almacenar datos. En Bash, las variables se definen y se accede a ellas sin necesidad de especificar su tipo.

3.Estructuras de control:

Incluyen las estructuras de decisión (if/else) y las estructuras de bucle (for, while) que permiten controlar el flujo de ejecución del script.

4.Funciones:

Son bloques de código que realizan una tarea específica. Las funciones en Bash se definen utilizando la sintaxis **nombre_de_la_funcion() { ... }** y pueden ser invocadas en cualquier parte del script.

SCRIPTING

Shebang

Conceptos básicos

1. **Shebang (#!)**
2. Variables
3. Operadores
4. Estructuras de control
5. Funciones

SHEBANG

Primea línea del script donde le informamos al sistema operativo cual es el **intérprete de comandos** que tiene que utilizar para trabajar con nuestro script

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

Por ejemplo, **#!/bin/bash** indica que el script debe ser ejecutado usando Bash.

```bash
#!/bin/bash

edad=18

if [ $edad -ge 18 ]; then
    echo "Eres mayor de edad."
else
    echo "Eres menor de edad."
fi
```

```
#!/bin/bash

for i in {1..5}; do
    echo "Número: $i"
done
```

```
#!/bin/bash

saludar() {
    echo "¡Hola, $1!"
}

saludar "Juan"
```

SCRIPTING

VARIABLES

Conceptos básicos

1. Shebang (#!)
2. **Variables**
3. Operadores
4. Estructuras de control
5. Funciones

VARIABLES

Permiten **almacenar** y **manipular** datos dentro de un script.

En Bash, las variables se definen y se accede a ellas sin necesidad de especificar su tipo.

Declaración de variables

En Bash, las variables se declaran asignándoles un valor

```
nombre="Juan“
edad=25
```

Acceso a variables

Para acceder al valor de una variable, se utiliza el símbolo de dólar (**$**) seguido del nombre de la variable.

Al acceder a una variable, no se incluye el signo de igual (**=**) utilizado en la declaración

```
echo "El nombre es: $nombre"
echo "La edad es: $edad"
```

Variables especiales

Contienen información sobre el **entorno de ejecución** del script

$0: El nombre del script.
$#: El número de argumentos pasados al script.
$1, **$2**, ...: Los argumentos pasados al script.
$@: Todos los argumentos pasados al script como una sola cadena.
$?: El código de salida del último comando ejecutado.
$$: El PID (Identificador de Proceso) del script.

Modificación de variables

Las variables en Bash son **mutables**, lo que significa que su valor puede ser cambiado durante la ejecución del script.

Para modificar el valor de una variable, simplemente se le asigna un nuevo valor.

```
nombre="María"
edad=30
nombre="Pedro"
edad=35
```

Variables de entorno

- Las variables de entorno son variables que están disponibles para **todos los procesos hijos de un script**.

- Definición:
  - dentro del script, usando el comando **export**.

```
export PATH="$PATH:/ruta/a/nuevo/directorio"
```

  - heredadas del entorno en el que se ejecuta el script.

Variables de solo lectura

- Su valor no puede ser modificado una vez que se ha asignado.
- Se utiliza el comando **readonly**.

```
readonly nombre="Juan"
```

Variables predefinidas

- $HOSTNAME
- $USER
- $DISPLAY
- $EDITOR
- $HOME
- $MANPATH
- $OLDPWD
- $PATH
- $PWD

1. Crea un script que imprima "¡Hola, Mundo!" en la terminal.

```
#!/bin/bash

# Imprimir mensaje
echo "¡Hola, Mundo!"
```

2. Crea un script que solicite al usuario su nombre y luego imprima un saludo personalizado.

```
#!/bin/bash

# Solicitar al usuario su nombre
echo "Por favor, introduce tu nombre:"
read nombre

# Imprimir saludo personalizado
echo "¡Hola, $nombre! Bienvenido."
```

3. Crea un script que muestre todos los archivos y directorios en el directorio actual

```
#!/bin/bash

# Mostrar archivos y directorios en el directorio actual
echo "Archivos y directorios en el directorio actual:"
ls
```

Actividad

P23 – Scripting I - Variables