1. **¿Qué es el Shell Scripting? ¿A qué tipos de tareas están orientados los script? ¿Los scripts deben compilarse? ¿Por qué?**

Un Shell Script es un programa que está creado con instrucciones que son ejecutadas por un Shell (CLI o intérprete de comandos) de Unix o Linux. El código no es compilado ni precompilado, se va ejecutando línea por línea efectuando lo que cada instrucción le indica. Necesita un programa que entienda los comandos y estructuras que contiene y esto se suele poner en la primera línea del programa. Por ejemplo #!/bin/bash significa que le pasaremos al BASH las lineas del fichero de Script. Dado que el BASH es el intérprete de comandos más famoso de Linux, los Script que se crean para este entorno también se pueden llamar Bash Script.

Principalmente sirve para automatizar tareas y para realizar procesos más complejos de los que un solo comando puede efectuar. Aunque los comando se pueden enlazar mediante tuberías o XARGS, a veces necesitamos tomar decisiones condicionales o recorrer elementos mediante bucles. Aquí es donde necesitamos organizarlo todo en un Shell Script.

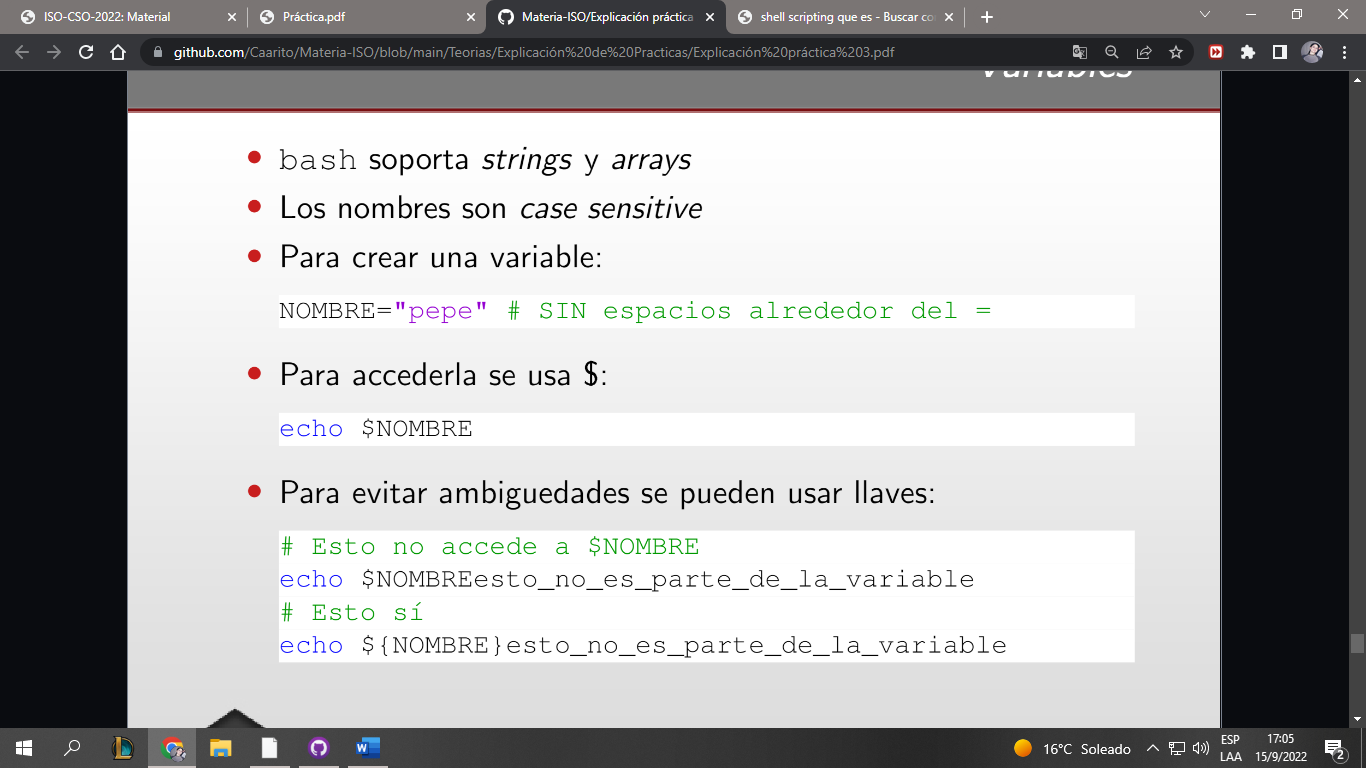
El propio sistema de Linux tiene programadas multitud de tareas con sus Script del sistema, desde la rotación de logs, actualización del arranque, gestión de servidores, niveles de ejecución etc.

1. **Investigar la funcionalidad de los comandos echo y read**

echo: imprime el texto.

read: lee una línea desde entrada estándar.

1. **¿Como se indican los comentarios dentro de un script?** Con #
2. **¿Cómo se declaran y se hace referencia a variables dentro de un script?**



**3. Crear dentro del directorio personal del usuario logueado un directorio llamado practicashell-script y dentro de él un archivo llamado mostrar.sh cuyo contenido sea el siguiente:**

**#!/bin/bash**

**# Comentarios acerca de lo que hace el script**

**# Siempre comento mis scripts, si no hoy lo hago**

**# y mañana ya no me acuerdo de lo que quise hacer**

**echo "Introduzca su nombre y apellido:"**

**read nombre apellido**

**echo "Fecha y hora actual:"**

**date**

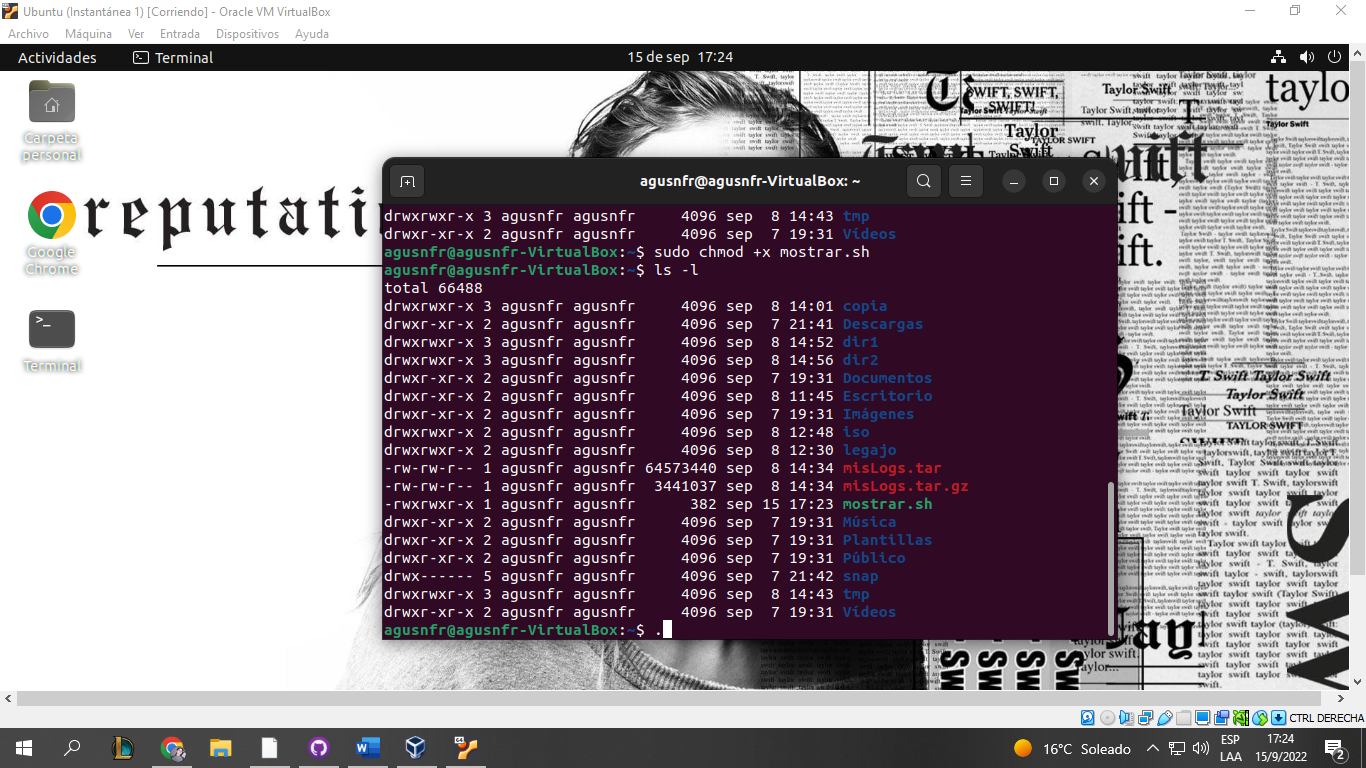
**echo "Su apellido y nombre es: “ 🡪** falta una comilla aca

**echo "$apellido $nombre"**

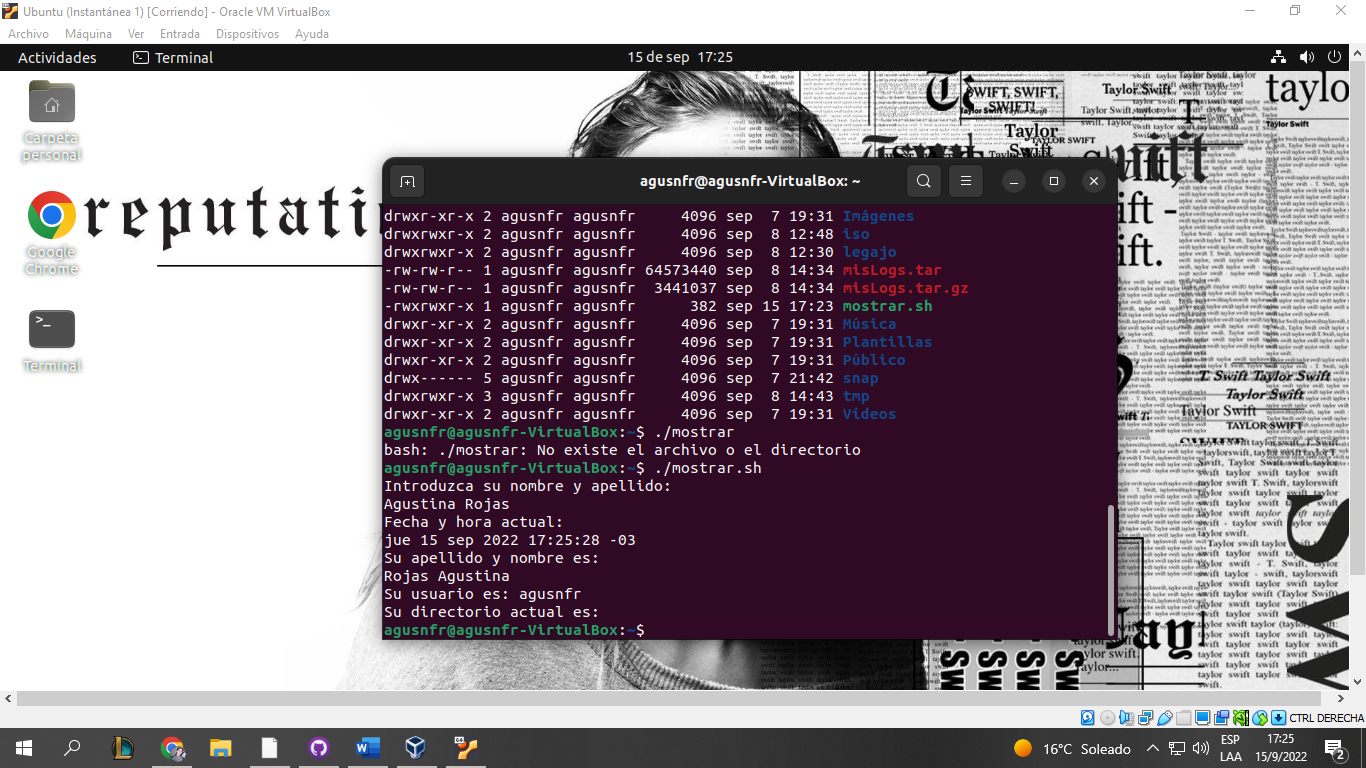
**echo "Su usuario es: `whoami`"**

**echo "Su directorio actual es:"**

1. **Asignar al archivo creado los permisos necesarios de manera que pueda ejecutarlo**



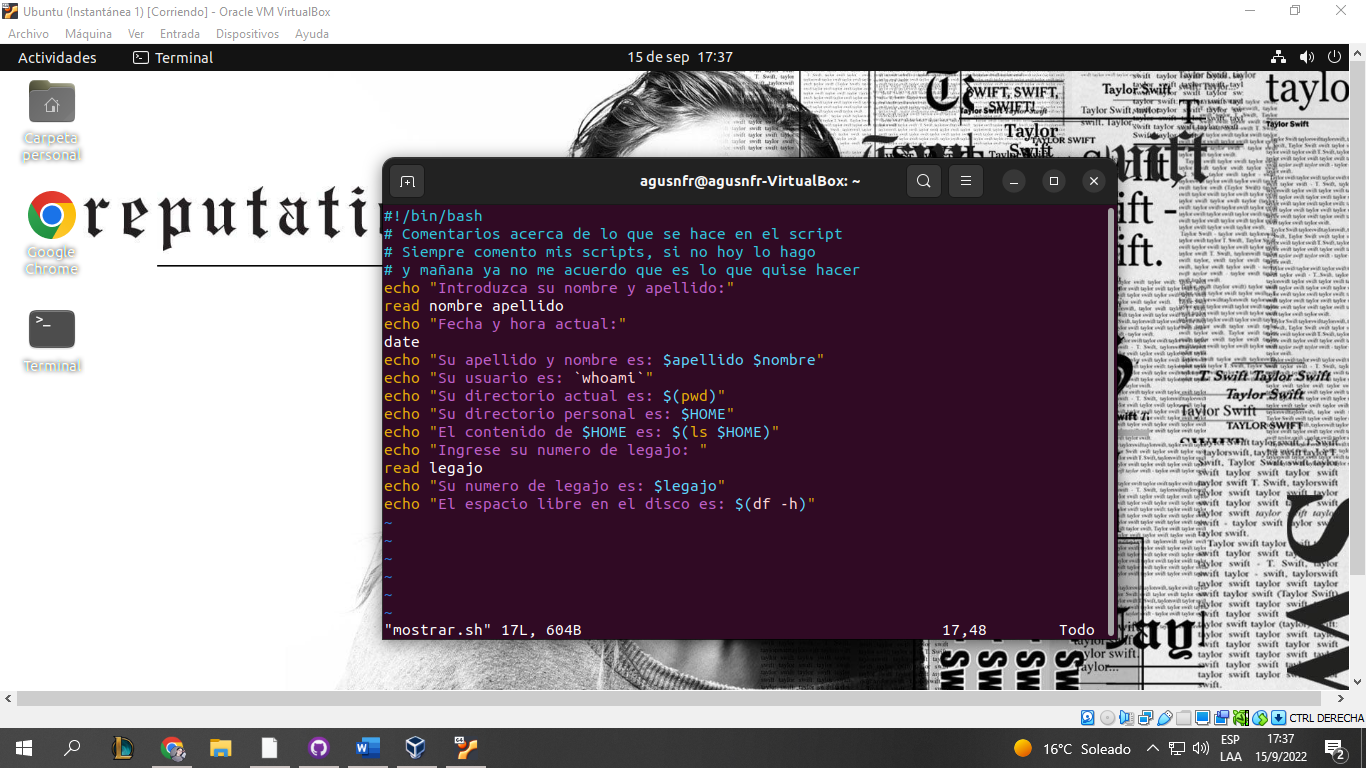
1. **Ejecutar el archivo creado de la siguiente manera: ./mostrar**
2. **¿Qué resultado visualiza?**



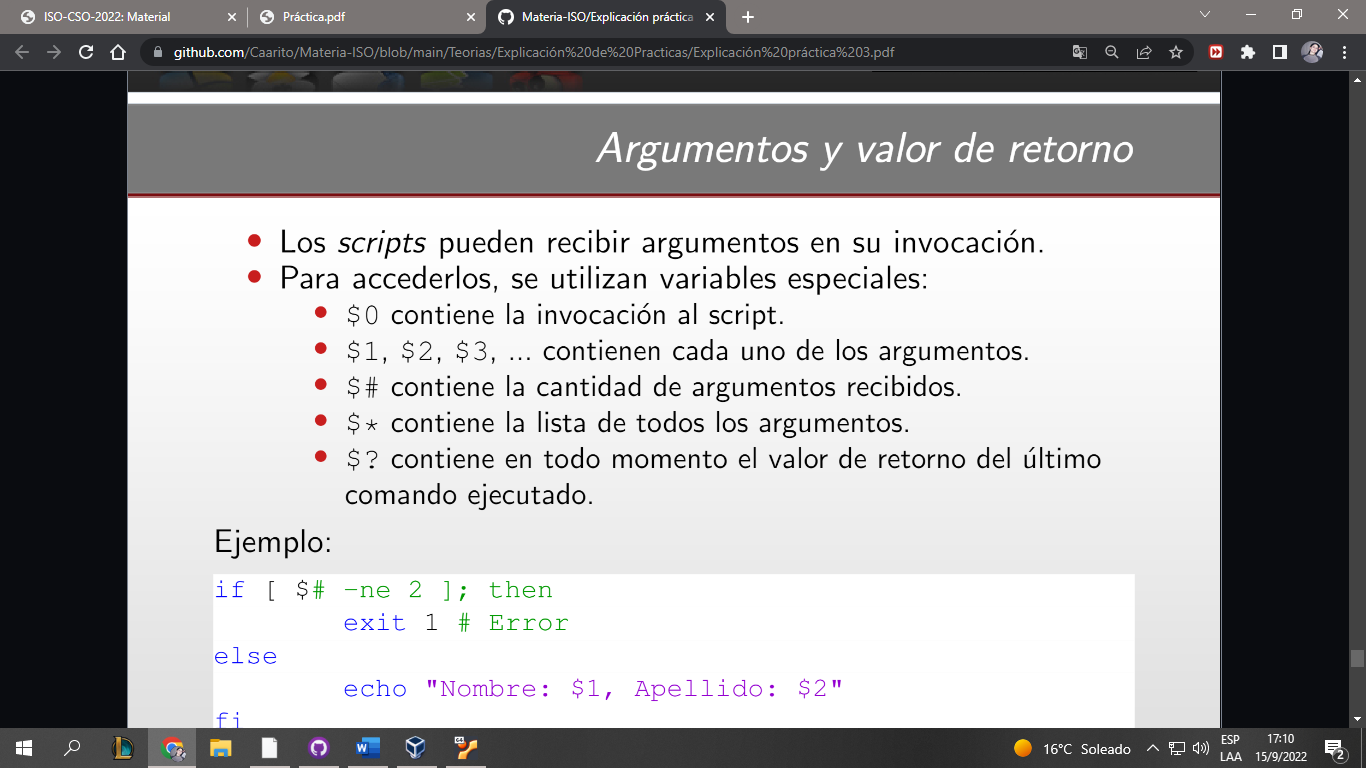
1. **Las backquotes (`) entre el comando whoami ilustran el uso de la sustitución de comandos. ¿Qué significa esto?**

Permite utilizar la salida de un comando como si fuese una cadena de texto normal, es equivalente a hacer $(whoami)

1. **Realizar modificaciones al script anteriormente creado de manera de poder mostrar distintos resultados (cuál es su directorio personal, el contenido de un directorio en particular, el espacio libre en disco, etc.). Pida que se introduzcan por teclado (entrada estándar) otros datos.**

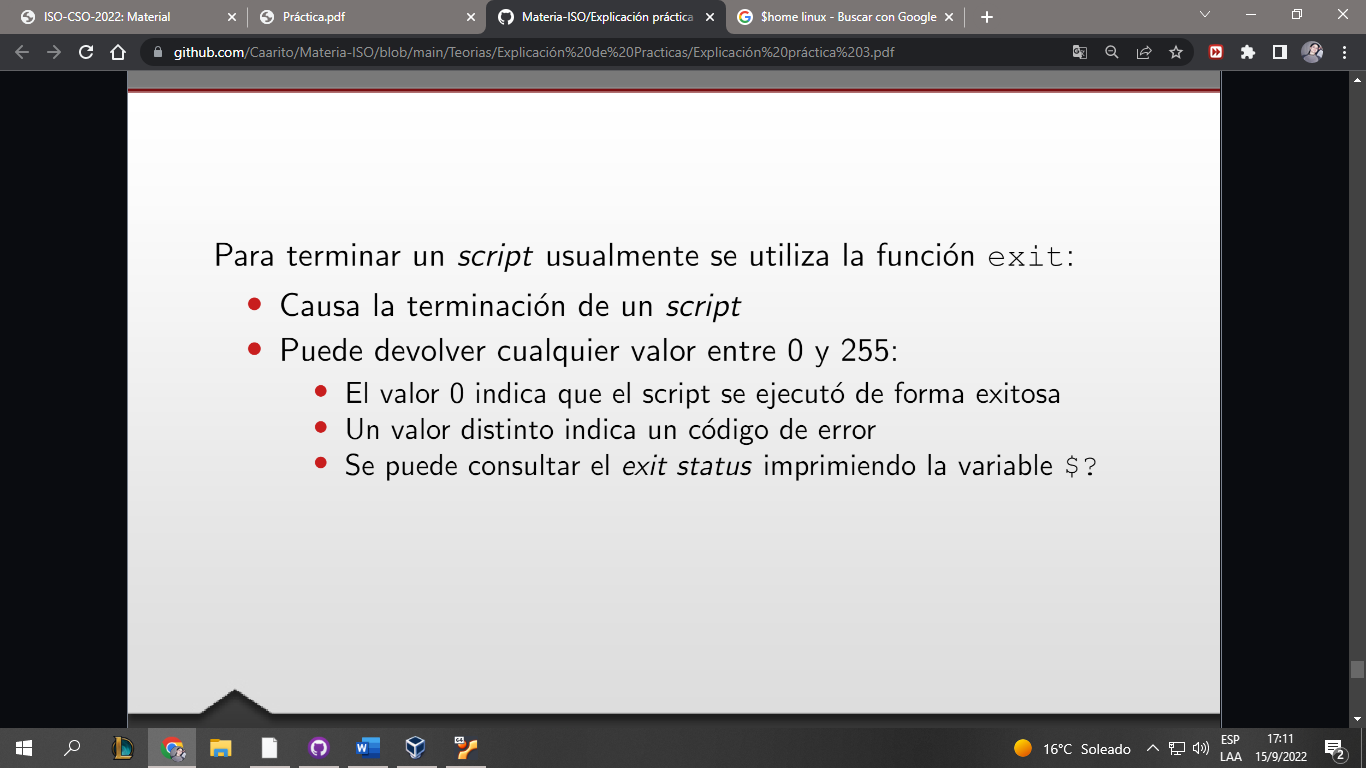


**4. Parametrización: ¿Cómo se acceden a los parámetros enviados al script al momento de su invocación? ¿Qué información contienen las variables $#, $\*, $? Y $HOME dentro de un script?**



$HOME es el directorio de trabajo por defecto del usuario.

**5. ¿Cual es la funcionalidad de comando exit? ¿Qué valores recibe como parámetro y cual es su significado?**



**6. El comando expr permite la evaluación de expresiones. Su sintaxis es: expr arg1 op arg2, donde arg1 y arg2 representan argumentos y op la operación de la expresión. Investigar que tipo de operaciones se pueden utilizar.**

Este comando devuelve en su salida estándar el resultado de las expresiones aritméticas pasadas como argumentos. Su sintaxis es expr expresión.

Todos los elementos de la expresión deben ir separados por al menos un espacio, y ciertos operadores aritméticos llevan como prefijo una barra invertida para evitar toda confusión con los caracteres especiales del shell.

Operaciones básicas: suma, resta, multiplicación, división y módulo en números enteros.

Otras: evaluación de expresiones regulares, operaciones de cadena como subcadena, longitud de cadenas, etc.

Los operadores aritméticos son:

+: suma

-: resta

\\*: multiplicación

/: división entera

%: resto de la división entera o módulo

\( y \): paréntesis

Se utiliza generalmente una sustitución de comandos para asignar el resultado del comando expr a una variable. Se obtiene por ejemplo:

[agusnfr]$ expr 2 + 3

5

[agusnfr]$ expr 2 - 3

-1

[agusnfr]$ expr 2 + 3 \\* 4

14

[agusnfr]$ expr \( 2 + 3 \) \\* 4

20

[agusnfr]$ resultado=$(expr 9 / 2)

[agusnfr]$ echo $resultado

4

[agusnfr]$ expr $resultado % 3

1

<https://linuxhint.com/linux-expr-command/> mas info ahí

**7. El comando “test expresión” permite evaluar expresiones y generar un valor de retorno, true o false. Este comando puede ser reemplazado por el uso de corchetes de la siguiente manera [ expresión ]. Investigar que tipo de expresiones pueden ser usadas con el comando test. Tenga en cuenta operaciones para: evaluación de archivos, evaluación de cadenas de caracteres y evaluaciones numéricas.**

Comando test

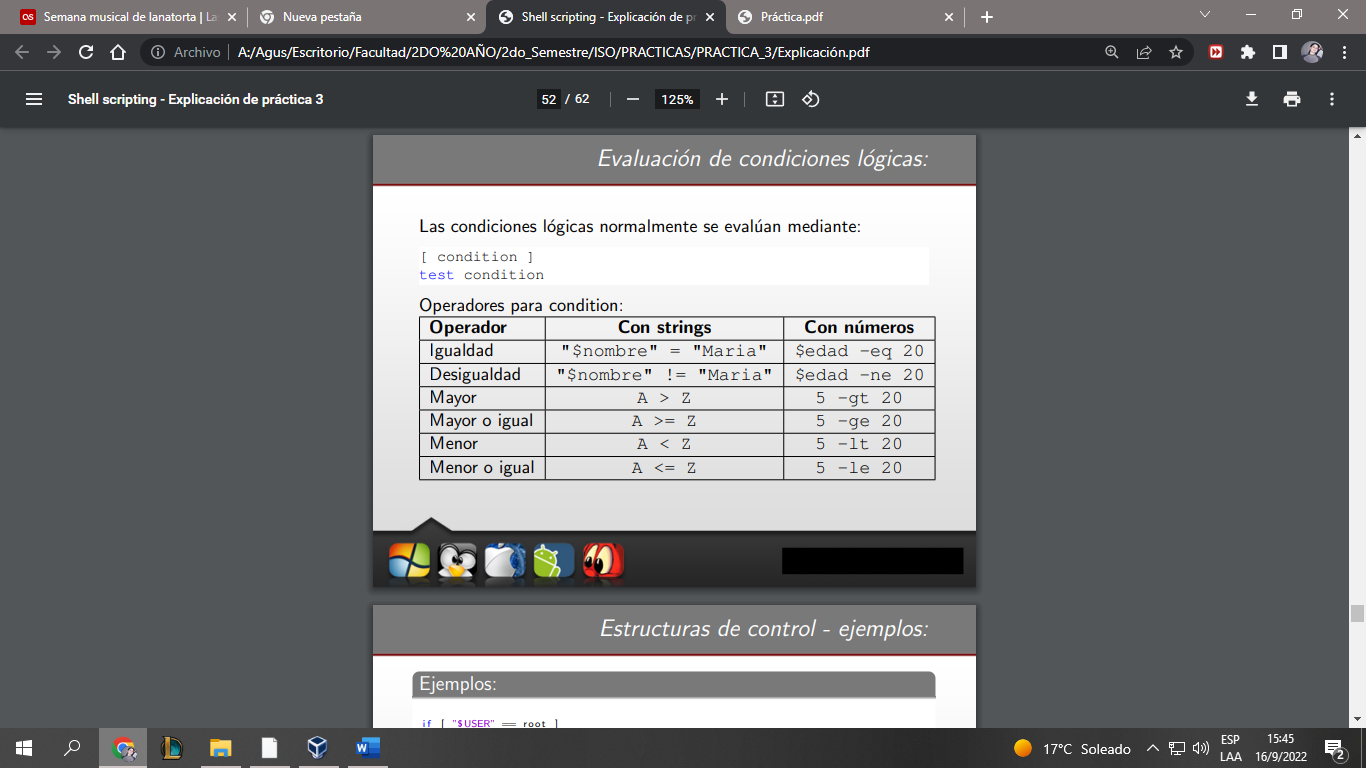
El comando test permite efectuar una serie de pruebas sobre los archivos, las cadenas de caracteres, los valores aritméticos y el entorno de usuario.

Este comando tiene un código de retorno igual a cero cuando el test es positivo, y diferente de cero en caso contrario; esto permite utilizarlo en encadenamientos de comandos con ejecución condicional (&& y ||) o en las estructuras de control que veremos más adelante.

El comando test posee dos sintaxis: test expresión y [ expresión ], donde "expresión" representa el test que se debe efectuar.

Los espacios detrás del corchete de apertura y antes del corchete de cierre son obligatorios en la sintaxis [ expresión ]. En general, todos los elementos de sintaxis del comando test deben ir separados por al menos un espacio.

El resto de la sección presenta los principales operadores que componen las expresiones de test del comando.



<https://francisconi.org/linux/comandos/test>

**8. Estructuras de control. Investigue la sintaxis de las siguientes estructuras de control incluidas en shell scripting:**

* **if**
* **case**
* **while**
* **for**
* **select**

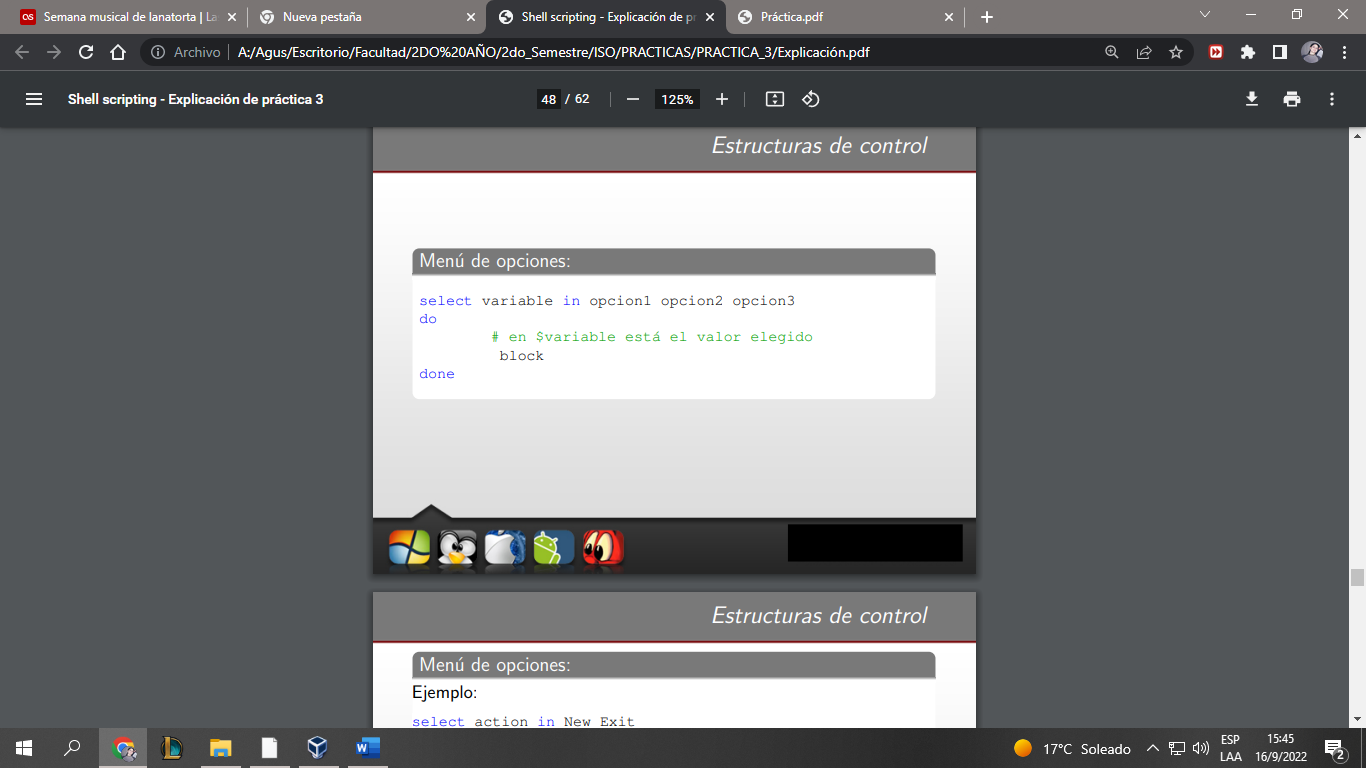
**IF**



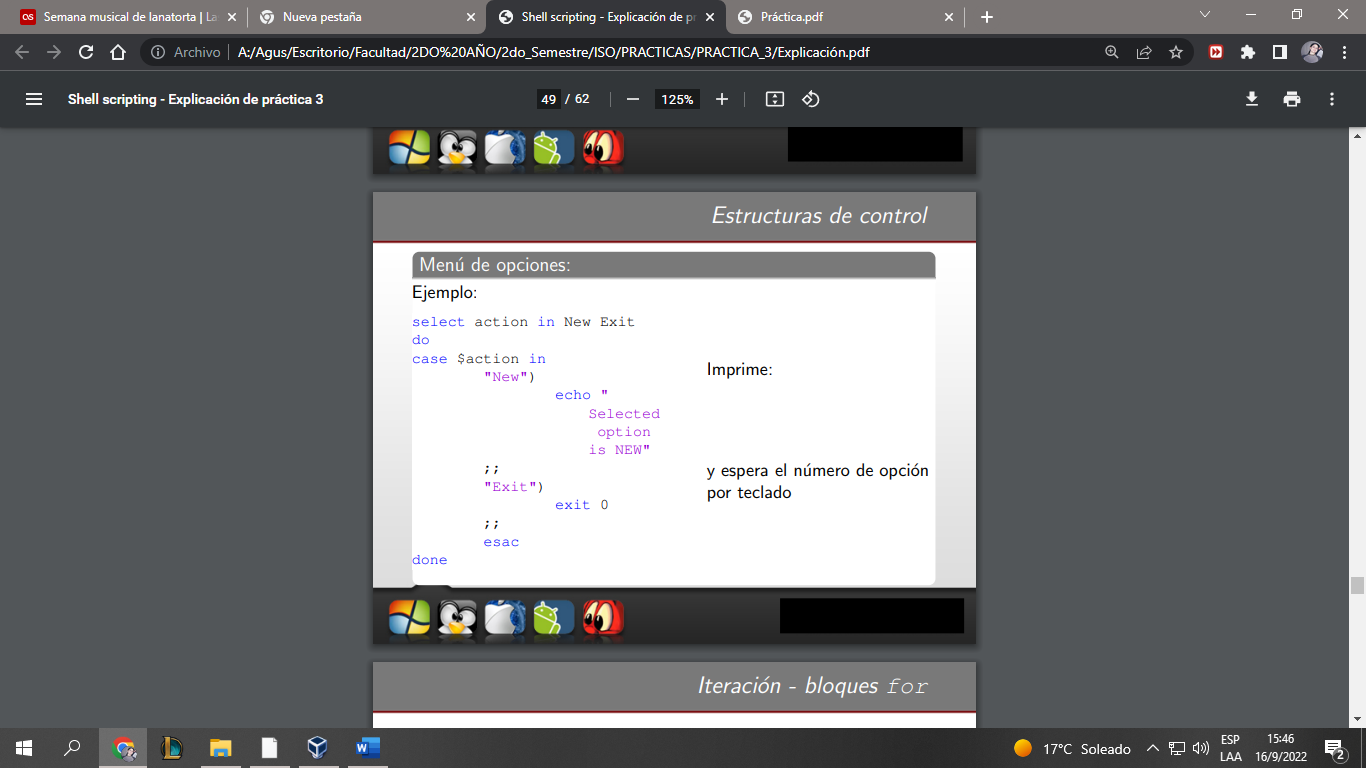
**CASE**



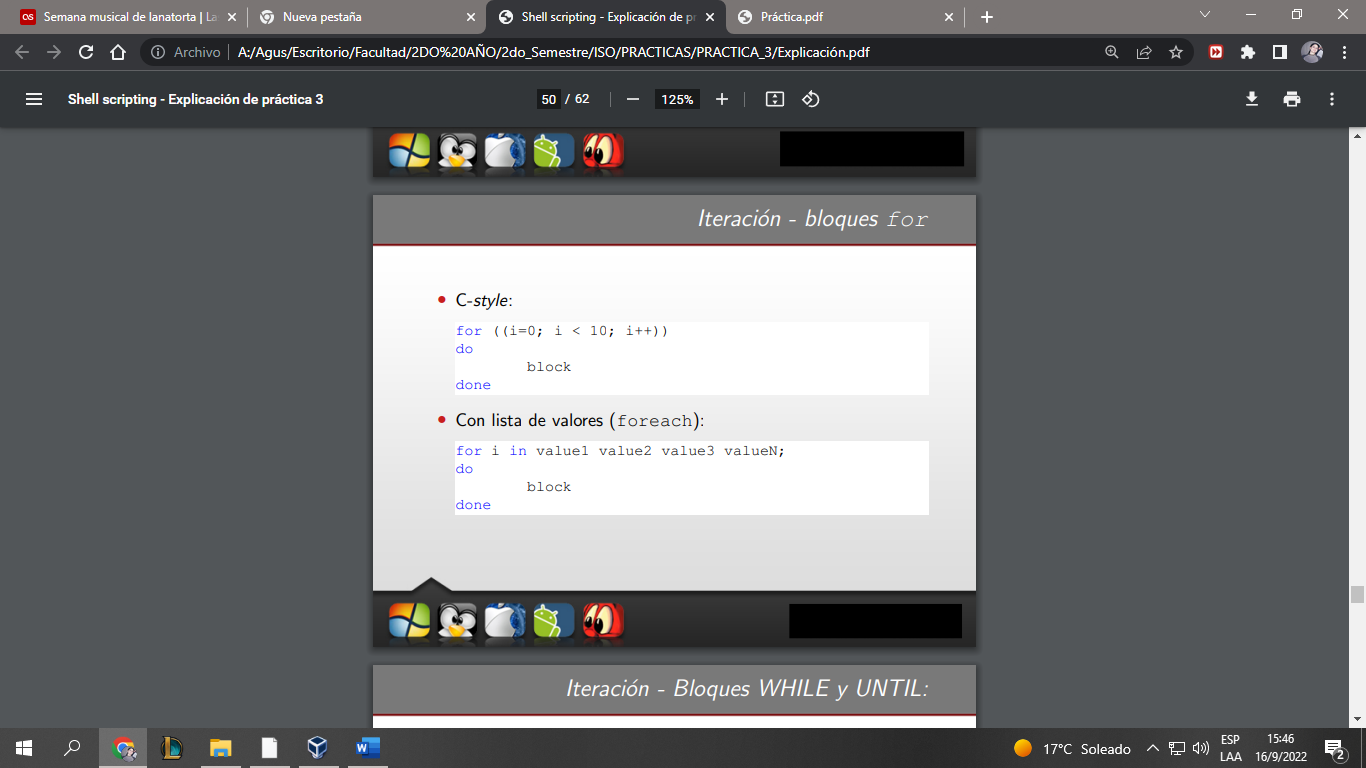
SELECT



<https://noviello.it/es/como-usar-select-en-bash-en-linux/>



**FOR**



**WHILE**



**UNTIL**



**9. ¿Qué acciones realizan las sentencias break y continue dentro de un bucle? ¿Qué parámetros reciben?**

break [n] corta la ejecución de n niveles de loops.

continue [n] salta a la siguiente iteración del enésimo loop que contiene esta instrucción.

**10. ¿Qué tipo de variables existen? ¿Es shell script fuertemente tipado? ¿Se pueden definir arreglos? ¿Cómo?**

Las variables son de tipo string o arreglos.

El shell es un lenguaje débilmente tipado (las variables en un idioma están fuertemente tipadas si tienen tipos explícitos; las variables sin tipos débiles son lenguajes débilmente tipados) y son diferentes de C. La definición de variables en la programación de shell no requiere un tipo, y no existe un concepto de tipo.

Si se puede definir arreglos.

