ISO 3

Objetivo

El objetivo de esta práctica es que el alumno desarrolle habilidades concernientes a Shell Scripting.

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

deben compilarse? ¿Por qué?

Es la escritura y el uso de archivos de script. Escritos en el lenguaje que da el shell del sistema.

2. Investigar la funcionalidad de los comandos echo y read.

Echo imprime un mensaje que se escribe

Y read espera una entrada de teclado para guardarla en una variable

(a) ¿Cómo se indican los comentarios dentro de un script? Con #######

(b) ¿Cómo se declaran y se hace referencia a variables dentro de un script?

Se hace referencia con $ y se declara con nombre=”lo que sea que le quieras poner”

3. Crear dentro del directorio personal del usuario logueado un directorio llamado práctica shell-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 ## entrada de consola
echo "Fecha y hora actual:" ##imprime en la consola
date ## imprime la fecha
echo "Su apellido y nombre es:
echo "$apellido $nombre"
echo "Su usuario es: `whoami`"
echo "Su directorio actual es:"
Pwd ## dice en qué directorio se encuentra el usuario
```

(a) Asignar al archivo creado los permisos necesarios de manera que pueda ejecutarlo

Para dar permisos uso chmod 766 y el nombre del archivo

(b) Ejecutar el archivo creado de la siguiente manera: ./mostrar

(c) ¿Qué resultado visualiza?

Yyyyyyy

(d) 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. • Permite guardarlo en variables o utilizarlos directamente.

(e) 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?
"El nombre del script es: $0"
"El primer parámetro es: $1"
"El número de parámetros es: $#"
"Todos los parámetros como una cadena: $*"
"Todos los parámetros individualmente: $@"
"El código de salida del último comando: $?"
"El directorio HOME del usuario: $HOME"

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

Causa la terminación de un script
• Puede devolver cualquier valor entre 0 y 255:
• El valor 0 indica que el script se ejecutó de forma exitosa
• Un valor distinto indica un código de error
• Se puede consultar el exit status imprimiendo la variable $?

Este hace que se salga del script

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.

El comando **expr** sirve para realizar operaciones matemáticas estas pueden ser tanto sumas, restas, divisiones, módulo(resto de una división), para comparaciones

expr 1+2 o 1+2

expr 1*/2 o 2/2

expr 3%2

expr 1=1 o expr 1!=1 devuelve un 1 o 0

expr 2 /<3 o 2/>3

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.

Esta verifica ficheros y compara valores
Comparación de valores

Operador	Con cadenas	Con numeros

Igualdad	"$nombre" = "Maria"	$edad -eq 20
Desigualdad	"$nombre" != "Maria"	$edad -ne 20
Mayor	A > Z	5 -gt 20
Mayor o igual	A >= Z	5 -ge 20
Menor	A < Z	5 -lt 20
Menor o igual	A <= Z	5 -le 20

Evaluación de archivos

Operador	**Ejemplo**
Existe	-e algo
Fichero regular	-f algo
Fichero es un directorio	-d algo
Comprobar si un archivo tiene permisos de ejecución	-x archivo
Comprueba si un archivo no está vacío	-s archivo
Comprueba si un archivo tiene permisos de escritura	-w archivo
Comprueba si un archivo tiene permisos de lectura	-r archivo

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

If

```
If (algo=algo)
then
        algo
else
        algo
```

case

```
case
```

while

for

Select

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

La sentencia **break** se utiliza para **salir inmediatamente de un bucle**. Cuando se ejecuta break, el control del programa salta a la primera línea de código después del bucle, finalizando su ejecución de manera abrupta.
Opcionalmente, puedes especificar un número n que indica cuántos niveles de bucle deseas romper
La sentencia **continue** se utiliza para **saltar el resto de las instrucciones en la iteración actual** del bucle y pasar a la siguiente iteración. A diferencia de break, el bucle no se termina, solo se omiten las instrucciones restantes en la iteración en curso.
Al igual que break, puede recibir un número n que indica cuántos niveles de bucle debe omitir.

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

Strings y Arreglos ().
Shell Script no es un lenguaje fuertemente tipado. Es un lenguaje **de tipado débil**, lo que significa que las variables no tienen un tipo fijo y pueden almacenar cualquier tipo de dato (texto, números, etc.) sin necesidad de declaración explícita de tipo.
Si, se escribe nombre de la variable= (“algo” “algo” “algo”).
Por ejemplo :

```
mi_arreglo=("uno" "dos" "tres")
arreglo_a=() # Se crea vacıo
arreglo_b=(1 2 3 5 8 13 21) # Inicializado
```

O de forma individual para cada posición :

```
mi_arreglo[0]="uno"
mi_arreglo[1]="dos"
mi_arreglo[2]="tres"
```

11. ¿Pueden definirse funciones dentro de un script? ¿Cómo? ¿Cómo se maneja el pasaje de parámetros de una función a la otra?

Si

```
nombre_funcion() {
        # código de la función
}
```

Por ejemplo

```
saludar() {
        echo "Hola, $1"
}
```

El pasaje de parámetros se realiza utilizando variables posicionales ($1, $2, etc.) dentro de la función.
Por ejemplo

```
multiplicar() {
        resultado=$(( $1 * $2 ))
        echo "El resultado es: $resultado"
}
multiplicar 5 3
```