

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 scripts? ¿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 - <u>eq</u> 20
Desigualdad	"\$nombre" != "Maria"	\$edad - <u>ne</u> 20
Mayor	A > Z	5 - <u>gt</u> 20
Mayor o igual	A >= Z	5 - <u>ge</u> 20
Menor	A < Z	5 - <u>lt</u> 20
Menor o igual	A <= Z	5 - <u>le</u> 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 vacío
```

```
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
```

12. Evaluación de expresiones:

(a) Realizar un script que le solicite al usuario 2 números, los lea de la entrada Standard e imprima la multiplicación, suma, resta y cual es el mayor de los números leídos.

(b) Modificar el script creado en el inciso anterior para que los números sean recibidos como parámetros. El script debe controlar que los dos parámetros sean enviados.

(c) Realizar una calculadora que ejecute las 4 operaciones básicas: +, -, *, %. Esta calculadora debe funcionar recibiendo la operación y los números como parámetros

13. Uso de las estructuras de control:

(a) Realizar un script que visualice por pantalla los números del 1 al 100 así como sus cuadrados.

(b) Crear un script que muestre 3 opciones al usuario: Listar, DondeEstoy y QuienEsta. Según la opción elegida se le debe mostrar:

Listar: lista el contenido del directoria actual.

DondeEstoy: muestra el directorio donde me encuentro ubicado.

QuienEsta: muestra los usuarios conectados al sistema.

(c) Crear un script que reciba como parámetro el nombre de un archivo e informe si el mismo existe o no, y en caso afirmativo indique si es un directorio o un archivo. En caso de que no exista el archivo/directorio cree un directorio con el nombre recibido como parámetro.

14. Renombrando Archivos: haga un script que renombre solo archivos de un directorio pasado

como parámetro agregándole una CADENA, contemplando las opciones:

“-a CADENA”: renombra el fichero concatenando CADENA al final del nombre del archivo

“-b CADENA”: renombra el fichero concantenado CADENA al principio del nombre del archivo

Ejemplo:

Si tengo los siguientes archivos: /tmp/a /tmp/b

Al ejecutar: ./renombra /tmp/ -a EJ

Obtendré como resultado: /tmp/aEJ /tmp/bEJ

Y si ejecuto: ./renombra /tmp/ -b EJ

El resultado será: /tmp/EJa /tmp/EJb

15. Comando cut. El comando cut nos permite procesar la líneas de la entrada que reciba (archivo, entrada estándar, resultado de otro comando, etc) y cortar columnas o campos, siendo posible indicar cuál es el delimitador de las mismas. Investigue los parámetros que puede recibir este comando y cite ejemplos de uso.

16. Realizar un script que reciba como parámetro una extensión y haga un reporte con 2 columnas, el nombre de usuario y la cantidad de archivos que posee con esa extensión. Se debe guardar el resultado en un archivo llamado reporte.txt

17. Escribir un script que al ejecutarse imprima en pantalla los nombre de los archivos que se

encuentran en el directorio actual, intercambiando minúsculas por mayúsculas, además de eliminar la letra a (mayúscula o minúscula). Ejemplo, directorio actual:

IsO

pepE

Maria

Si ejecuto: ./ejercicio17

Obtendré como resultado:

iSo

PEPe

mRI

Ayuda: Investigar el comando tr

18. Crear un script que verifique cada 10 segundos si un usuario se ha logueado en el sistema

(el nombre del usuario será pasado por parámetro). Cuando el usuario finalmente se loguee, el programa deberá mostrar el mensaje "Usuario XXX logueado en el sistema" y salir.

19. Escribir un Programa de "Menu de Comandos Amigable con el Usuario" llamado menú, el cual, al ser invocado, mostrará un menú con la selección para cada uno de los scripts creados

en esta práctica. Las instrucciones de cómo proceder deben mostrarse junto con el menú.

El menú deberá iniciarse y permanecer activo hasta que se seleccione Salir. Por ejemplo:

MENU DE COMANDOS

03. Ejercicio 3

12. Evaluar Expresiones

13. Probar estructuras de control

...

Ingrese la opción a ejecutar: 03

20. Realice un script que simule el comportamiento de una estructura de PILA e implemente las siguientes funciones aplicables sobre una estructura global definida en el script:

push: Recibe un parámetro y lo agrega en la pila pop: Saca un elemento de la pila

length: Devuelve la longitud de la pila print: Imprime todos elementos de la pila

21. Dentro del mismo script y utilizando las funciones implementadas:

Agregue 10 elementos a la pila

Saque 3 de ellos

Imprima la longitud de la cola

Luego imprima la totalidad de los elementos que en ella se encuentran.

22. Dada la siguiente declaración al comienzo de un script: num=(10 3 5 7 9 3 5 4) (la cantidad de elementos del arreglo puede variar). Implemente la función productoria dentro de este script, cuya tarea sea multiplicar todos los números del arreglo

23. Implemente un script que recorra un arreglo compuesto por números e imprima en pantalla

sólo los números pares y que cuente sólo los números impares y los informe en pantalla al finalizar el recorrido.

24. Dada la definición de 2 vectores del mismo tamaño y cuyas longitudes no se conocen.

vector1=(1 .. N)

vector2=(7 .. N)

Por ejemplo:

vector1=(1 80 65 35 2)

y

vector2=(5 98 3 41 8).

Complete este script de manera tal de implementar la suma elemento a elemento entre ambos vectores y que la misma sea impresa en pantalla de la siguiente manera:

La suma de los elementos de la posición 0 de los vectores es 6

La suma de los elementos de la posición 1 de los vectores es 178

...

La suma de los elementos de la posición 4 de los vectores es 10

25. Realice un script que agregue en un arreglo todos los nombres de los usuarios del sistema

pertenecientes al grupo “users”. Adicionalmente el script puede recibir como parametro:

“-b n”: Retorna el elemento de la posición n del arreglo si el mismo existe. Caso contrario, un mensaje de error.

“-l”: Devuelve la longitud del arreglo

“-i”: Imprime todos los elementos del arreglo en pantalla

26. Escriba un script que reciba una cantidad desconocida de parámetros al momento de su invocación (debe validar que al menos se reciba uno). Cada parámetro representa la ruta absoluta

de un archivo o directorio en el sistema. El script deberá iterar por todos los parámetros recibidos, y solo para aquellos parámetros que se encuentren en posiciones impares (el primero, el tercero, el q verificar si el archivo o directorio existen en el sistema, imprimiendo en pantalla que tipo

de objeto es (archivo o directorio). Además, deberá informar la cantidad de archivos o directorios inexistentes en el sistema.

27. Realice un script que implemente a través de la utilización de funciones las operaciones básicas sobre arreglos:

inicializar: Crea un arreglo llamado array vacío

agregar_elem <parametro1>: Agrega al final del arreglo el parámetro recibido

eliminar_elem <parametro1>: Elimina del arreglo el elemento que se encuentra en la posición recibida como parámetro. Debe validar que se reciba una posición válida

longitud: Imprime la longitud del arreglo en pantalla

imprimir: Imprime todos los elementos del arreglo en pantalla

inicializar_Con_Valores <parametro1><parametro2>: Crea un arreglo con longitud <parametro1>y en todas las posiciones asigna el valor <parametro2>

28. Realice un script que reciba como parámetro el nombre de un directorio. Deberá validar que

el mismo exista y de no existir causar la terminación del script con código de error 4. Si el directorio existe deberá contar por separado la cantidad de archivos que en él se encuentran para los cuales el usuario que ejecuta el script tiene permiso de lectura y escritura, e informar

dichos valores en pantalla. En caso de encontrar subdirectorios, no deberán procesarse, y tampoco deberán ser tenidos en cuenta para la suma a informar.

29. Implemente un script que agregue a un arreglo todos los archivos del directorio /home cuya

terminación sea .doc. Adicionalmente, implemente las siguientes funciones que le permitan acceder a la estructura creada:

verArchivo <nombre_de_archivo>: Imprime el archivo en pantalla si el mismo se encuentra en el arreglo. Caso contrario imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 5

cantidadArchivos: Imprime la cantidad de archivos del /home con terminación .doc

borrarArchivo <nombre_de_archivo>: Consulta al usuario si quiere eliminar el archivo lógicamente. Si el usuario responde Si, elimina el elemento solo del arreglo. Si el

usuario responde No, elimina el archivo del arreglo y también del FileSystem. Debe validar que el archivo exista en el arreglo. En caso de no existir, imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno 10

30. Realice un script que mueva todos los programas del directorio actual (archivos ejecutables)

hacia el subdirectorio “bin” del directorio HOME del usuario actualmente logueado. El script debe imprimir en pantalla los nombres de los que mueve, e indicar cuántos ha movido, o que no ha movido ninguno. Si el directorio “bin” no existe,deberá ser creado.