# UNIVERSIDAD DE COLIMA

## Facultad de Telemática
## Ingeniería en software

# Práctica 2. Ejercicios de Scripts

Materia: Administración de Sistemas Linux

Docente: Juan Manuel Ramírez Alcaraz

Ismael Meza Rodríguez

6°K

30 de abril del 2020

# Índice

Índice **2**

Introducción **3**

**Desarrollo** **3**

Script para saber quien soy 3
Variables de entorno 4
Variables de usuario 4
Sustitución de comando 5
Cálculo matemático 6
Condicionales 6
Usuario existente 7
Múltiples condicionales 8
Múltiples pruebas 9
Comparaciones numéricas 9
Comparación de cadenas 10
Comparación de ficheros 11

**Referencias** **11**

# Introducción

Durante esta práctica se buscará realizar unos ejecutables de consola para realizar unos ejercicios propuestos por el profesor con el cual podremos poner en práctica los conocimientos adquiridos a partir de la lectura de un recurso donde vemos más a fondo las funciones disponibles así como nuevas sintaxis y uso de parámetros.

# Desarrollo

## Ejercicio 1

Elabora un script que reciba como parámetros nombres de personas e imprima un saludo para cada uno de ellos. Si no se introducen parámetros se deberá mostrar un mensaje que indique que se debe introducir al menos un nombre. Prueba como se puede introducir como parámetro un nombre compuesto de dos o más palabras.

Para hacer este ejercicio debemos de hacer uso de un ciclo for para hacer el recorrido de los parámetros que recorra el parámetro *$** ya que contempla los parámetros contandolos como uno solo los que van entrecomillados. También hacemos uso de *$#* para poder saber cuantos parámetros fueron pasados al script.

Creamos el archivo con *nano "1.- Saluda.sh"* y pegamos el siguiente código:

```
#!/bin/sh

echo "Números de parámetros: $#"

echo "Parametros pasados: $*"

# Si no se pasan parámetros entonces se marca un mensaje

if [ $# = 0 ];

then

   echo "Debes pasar al menos un nombre."

else
```

```
# Si se pasaron parámetros entonces se recorren uno por uno y se les saluda

   for nombre in $*; do

       echo "Hola $nombre"

   done

fi
```

Lo ejecutamos de la siguiente manera `. 1.-\ Saluda.sh Ismael Juan Raul Jose` lo cual no regresa la siguiente salida.

```
meza@meza:~/Documents/Scripts$ . 1.-\ Saluda.sh Ismael Juan Raul Jose
Números de parametros: 4
Parametros pasados: Ismael Juan Raul Jose
Hola Ismael
Hola Juan
Hola Raul
Hola Jose
meza@meza:~/Documents/Scripts$
```

## Ejercicio 2

Elabora un script semejante al anterior, pero ahora los parámetros deben ser rutas absolutas que se deberán agregar a la variable de ambiente PATH. Imprimir la variable. La modificación de la variable debe ser temporal, es decir, antes de finalizar el script la variable debe regresar a su valor inicial.

Para este ejercicio debemos de crear una variable temporal para guardar el valor de la variable PATH para poder modificarla y regresar a su estado anterior, también debemos de hacer uso de concatenación e igualmente usar de nuevo un ciclo for para recorrer los directorios.

Crearemos el archivo con `nano 2.-\ Agrega\ a\ Path.sh` y copiaremos el siguiente código dentro:

```
#!/bin/sh

#Guardado del valor de la variable PATH

temp=$PATH

echo "Variable PATH antes de ser modificada -> $PATH"
```

```
echo "Números de parámetros: $#"

echo "Parametros pasados: $*"

# Si no se pasan parámetros se marca un mensaje

if [ $# = 0 ];

then

   echo "Debes pasar al menos un directorio."

else

# Si se pasaron parámetros entonces se recorren y se agregan al PATH
agregando : antes de otro directorio

   for directorio in $*; do

       PATH="$PATH:$directorio"

   done

fi

echo "Variable PATH modificada -> $PATH"

# Se regresa el valor original a la variable PATH

PATH=$temp

echo "Variable PATH después de ser regresada a su valor inicial ->
$PATH"
```

Para correrlo usamos *. 2.-\ Agrega\ a\ Path.sh /home/meza /ora/ /ruta/* lo cual nos dará la siguiente salida:

```
meza@meza:~/Documents/Scripts$ . 2.-\ Agrega\ a\ Path.sh /home/meza /ora/ /ruta/
Variable PATH antes de ser modificada -> /home/meza/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/gam
es:/usr/games
Números de parametros: 3
Parametros pasados: /home/meza /ora/ /ruta/
Variable PATH modificada -> /home/meza/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
:/home/meza:/ora/:/ruta/
Variable PATH después de ser regresada a su vaor inicial -> /home/meza/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:
/bin:/usr/local/games:/usr/games
meza@meza:~/Documents/Scripts$
```

## Ejercicio 3

Elabora un script que lea el nombre de un usuario y verifique si se encuentra registrado en el sistema. Se debe imprimir un mensaje que indique si es usuario registrado o no. el script debe iterar solicitando un nuevo nombre de usuario hasta que la cadena que se introduzca sea “fin”. Si es necesario, crear nuevos usuarios para probar el script.

Para este ejercicio debemos de hacer uso de la sentencia *read* para que el usuario pueda hacer inputs en la consola, asimismo debemos usar un ciclo while para que se repita la acción de pedir el nombre de usuario, también debemos hacer uso de la sentencia *if* para poder saber si existe un usuario haciendo uso del comando grep con el archivo */etc/passwd*.

Crearemos el archivo con nano ***3.-\ Usuarios.sh*** y le copiaremos el siguiente codigo:

```
#!/bin/sh

usuario='null'

# Ciclo necesario para que se pida el usuario mientra no sea 'fin'

while [ "$usuario" != 'fin' ]; do

    echo "Introduzca un usuario (introduzca fin sin quiere finalizar el
programa):"

   # Se hace que se ingrese un nombre de usuario

   read usuario

   # Se busca si está presente en /etc/passwd con el uso de grep

   if grep $usuario /etc/passwd; then

       echo "El usuario $usuario si existe"

   else

       echo "El usuario $usuario no existe"

   fi

done
```

Para ejecutarlo usamos `. 3.-\ Usuarios.sh` lo cual nos dará la siguiente salida:

```
meza@meza:~/Documents/Scripts$ . 3.-\ Usuarios.sh
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
meza
meza:x:1000:1000:Ismael Meza Rodríguez,,,:/home/meza:/bin/bash
El usuario meza si existe
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
root
root:x:0:0:root:/root:/bin/bash
El usuario root si existe
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
meziwis
meziwis:x:1001:1001::/home/meziwis:
El usuario meziwis si existe
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
hola
El usuario hola no existe
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
xd
El usuario xd no existe
Introduzca un usuario (introduzca fin sin quiere finalizar el programa):
fin
El usuario fin no existe
meza@meza:~/Documents/Scripts$
```

## Ejercicio 4

Elabora un script que lea dos números y un operador (+, -, *, /) y que imprima el resultado de la operación seleccionada. Para el caso de la resta se debe imprimir también el valor absoluto del resultado (restar siempre el número menor del mayor). El proceso se debe repetir hasta que se introduzca el valor 0 en cada uno de los números y el valor “q” en el operador (se deben cumplir las tres condiciones).

Para hacer este ejercicio vamos a hacer uso de la sentencia while donde podremos pedir los números y el operador tantas veces lo requiera, también haremos uso de una sentencia *if* para saber cómo ejecutar la resta y también la sentencia *read* para pedir los datos.

Para crear el archivo usamos `nano 4.-\ Artimetica.sh` y le copiamos el codigo siguiente:

```
#!/bin/sh

num1=-1
```

```
num2=-1

op='?'

# Ciclo necesario para que se pida al usuario los valores mientras no
sean

# num1 = 0, num2 = 0, op = q

until [ $num1 = 0 -a $num2 = 0 -a $op = 'q' ]; do

   # Se hace la solicitud de los datos

   echo "Introduzca el primer número: "

   read num1

   echo "Introduzca el segundo número: "

   read num2

   echo "Introduzca el operador (+, -, *, /): "

   read op

   # Se hace la operación

   expresion="$num1$op$num2"

   if [ $num1 -lt $num2 -a $op == '-' ];then

       expresion="$num2$op$num1"

   fi

   if [ $op = 'q' ];then

       echo "Presiona ENTER para continuar";

       read asd

       clear

       continue

   fi

   echo "El valor es $((expresion))"
```

```
    echo "Presiona ENTER para continuar";

    read asd

    clear

done

echo "Finalizado correctamente."
```

Para ejecutarlo usamos `. 4.-\ Artimetica.sh` y nos arroja la siguiente salida:

```
Introduzca el primer número:
10
Introduzca el segundo número:
2
Introduzca el operador (+, -, *, /):
/
El valor es 5
Presiona ENTER para continuar
```

## Ejercicio 5

Crea un archivo de texto que contenga un conjunto de números cualesquiera y elabora un script que lea dichos números y que calcule e imprima el máximo, el mínimo, el promedio y la desviación estándar. El nombre del archivo deberá ser introducido como un parámetro.

Para este ejercicio haremos uso de ciclos for, condicionales, parámetros especiales, extensión aritmética y algo de razonamiento.

Para la desviación estándar usaremos la siguiente fórmula

$$\mathrm{DE} = \sqrt{\frac{\sum |x - \mu|^2}{N}}$$

Para crear el archivo usaremos `nano 5.-\ Numeros.sh` y pegaremos el siguiente codigo dentro:

```
#!/bin/sh

nums=`cat $*`

total_num=0

suma=0

min=9999999

max=-9999999

promedio=0

sumad=0

xd=""

desv=0

# Se iteran todos los números

for num in $nums; do

   # Cada iteración se aumenta el número total de números

   total_num=$((total_num + 1))

   # Cada iteración se suman los números

   suma=$((suma + num))

   # Si el número es menor que el mínimo entonces se reemplaza

   if [ $num -lt $min ];then

       min=$num

   fi

   # Si el número es mayor que el máximo entonces se reemplaza

   if [ $num -gt $max ];then

       max=$num

   fi
```

```
done

for num in $nums; do

   xd=$((num - promedio))

   xd=$((xd * xd))

   sumad=$((sumad + xd))

done

desv=$((sumad / total_num))

desv=$(echo "sqrt($desv)" | bc)

promedio=$((suma / total_num))

echo "Lista de números $nums"

echo "Máximo: $max"

echo "Mínimo: $min"

echo "Promedio: $promedio"

echo "Desviación estándar: $desv"
```

Dentro del mismo directorio crearemos un archivo con *nano Numeros.txt* y le copiaremos lo siguiente:

```
12 8 4 2 4 6 10 2 20 21 25 35 50 43

60 23 37 39 12 1 30 2 40 2 10 23 12

40 32 23 52 43 21 11 3 19 3 10 32 11
```

Para ejecutarlo usaremos *. 5.-\ Numeros.sh Numeros.txt* lo cual no arroja lo siguiente:

```
meza@meza:~/Documents/Scripts$ . 5.-\ Numeros.sh Numeros.txt
Lista de numeros 12 8 4 2 4 6 10 2 20 21 25 35 50 43
60 23 37 39 12 1 30 2 40 2 10 23 12
40 32 23 52 43 21 11 3 19 3 10 32 11
Máximo: 60
Mínimo: 1
Promedio: 20
Desviación estándar: 26
meza@meza:~/Documents/Scripts$
```

# Conclusiones

Esta practica vino de lujo para poder practicar este lenguaje de programación ya que la sintaxis es diferente a otros lenguajes de programación tradicionales, también te pone a pensar cómo automatizar algunas tareas que con la interfaz gráfica sería demasiado tardado.

Los códigos estarán en el siguiente repositorio de GitHub}

https://github.com/MezaDios/Practica-2-Bash

# Referencias

http://trajano.us.es/~fjfj/shell/shellscript.htm