Semana 16:

Programación de Shell Scripts en Linux

Sistemas Operativos

Objetivos de la Sesión

- Aplicar comandos básicos de Linux.
- Entender el funcionamiento de los Scripts en Linux.
- Utilizar las diferentes sentencias en un script en Linux.

Las variables

- Principalmente, Bash utiliza variables de tipo de cadena de caracteres. Esto lo diferencia de los lenguajes de programación donde las variables tienen distintos tipos.
- Las variables son una herramienta que nos va a permitir **guardar valores numéricos o cadenas de caracteres en memoria, pudiendo acceder a esta mediante el nombre que le asignemos**.
- La declaración de la variable se efectuará en el momento en el que le asignamos un valor.

Las variables – ejemplo

- **Creando script.**

```
[root@linux ~]# cat script
#!/bin/bash
clear
echo "Ingrese su edad:"
read z
a=2020
b="que tenga buen día"
echo "Ahora sé que usted tiene $z  años en el $a, $b"
```

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
Ingrese su edad:
34
Ahora sé que usted tiene 34  años en el 2020, que tenga buen día
```

Las variables – parámetros posicionales

- Los parámetros posicionales son los encargados de recibir los argumentos de un script. Sus nombres son 1,2,3,etc; con lo que para acceder a ellos utilizamos el símbolo $ de la forma $1,$2,$3,etc. Además, el parámetro posicional 0 almacena el nombre del script donde se ejecuta.
  - **Creando script.**

```
[root@linux ~]# cat script01
#!/bin/bash
echo "Esto es una prueba de varibles"
echo $0
echo $1
echo $2
echo $3
```

Las variables – parámetros posicionales

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script01 10 11 12
Esto es una prueba de varibles
./script01
10
11
12
[root@linux ~]# ./script01 Tecsup Universitario Alianza
Esto es una prueba de varibles
./script01
Tecsup
Universitario
Alianza
```

Las variables – sustitución de comandos

- La sustitución de comandos nos permite usar la salida de un comando como si fuera el valor de una variable.
- La sintaxis de la sustitución de comandos es:

$ (**comando**)

- **Creando script.**

```
[root@linux ~]# cat script02
#!/bin/bash
directorios_etc=$(find /etc/sysconfig -type d)
echo "Los directorios que se encuentran en "/etc/sysconfig" son los siguientes:"
echo $directorios_etc
```

Las variables – sustitución de comandos

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script02
Los directorios que se encuentran en /etc/sysconfig son los siguientes:
/etc/sysconfig /etc/sysconfig/cbq /etc/sysconfig/console /etc/sysconfig/modules
/etc/sysconfig/network-scripts
```

- **Creando script.**

```
[root@linux ~]# cat script03
#!/bin/bash
permisos=$(ls -ld /etc)
echo "Los permisos del directorio "/etc" son:"
echo $permisos
```

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script03
Los permisos del directorio /etc son:
drwxr-xr-x. 145 root root 8192 Jul 10 15:54 /etc
```

Los arrays / tablas

- Los arrays nos van a permitir crear **estructuras de datos en la que vamos a poder almacenar 1024 variables**, se declaran mediante los corchetes [ ] o asignando los valores de forma directa, siendo el 0 la primera posición y 1023 la última.
  - **Creando script.**

```
[root@linux ~]# cat script04
#!/bin/bash
equipo[0]="Perú"
equipo[1]="Argentina"
equipo[2]="Brasil"
equipo[3]="Colombia"
equipo[4]="Ecuador"

echo "El país que juega mejor al fútbol es: "
echo ${equipo[0]}
```

Los arrays / tablas

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script04
El país que juega mejor al fútbol es:
Perú
```

- También es posible listar todos los elementos de una vez mediante el símbolo asterisco (*) :

```
[root@linux ~]# cat script05
#!/bin/bash
equipo[0]="Perú"
equipo[1]="Argentina"
equipo[2]="Brasil"
equipo[3]="Colombia"
equipo[4]="Ecuador"

echo "Las mejores selecciones de fútbol son: "
echo ${equipo[*]}
```

```
[root@linux ~]# ./script05
Las mejores selecciones de fútbol son:
Perú Argentina Brasil Colombia Ecuador
```

Los arrays / tablas

- Otra forma de declarar los array es la siguiente.
  - **Creando script.**

```
[root@linux ~]# cat script06
#!/bin/bash
ciudades=(Lima Quito Bogotá Santiago)
echo ${ciudades[0]}
```

  - **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script06
Lima
```

Los condicionales

- Permiten **realizar acciones solamente bajo las condiciones que nosotros indiquemos.**
- El condicional **if** es el más conocido y usado, su estructura es muy básica:

```
if [condición 1]
then
<comandos a ejecutar si se cumple la condición 1>
else
<comandos a ejecutar si no se cumple condición 1>
fi
```

Los condicionales

- **Creando script.**

```
[root@linux ~]# cat script07
#!/bin/bash
echo "Ingresar el número uno:"
read z
if [ $z -eq 1 ]
then
echo "El número ingresado fue 1"
else
echo "Usted no ingresó el número 1"
fi
```

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script07
Ingresar el número uno:
1
El número ingresado fue 1
[root@linux ~]# ./script07
Ingresar el número uno:
4
Usted no ingresó el número 1
```

Los condicionales

- **Creando script.**

```
[root@linux ~]# cat script08
#!/bin/bash
clear
echo "Ingresar un número del 1 al 3:"
read z
if [ $z -eq 1 ]
then
echo "El número ingresado fue 1"
elif [ $z -eq 2 ]
then
echo "El número ingresado fue 2"
elif [ $z -eq 3 ]
then
echo "El número ingresado fue 3"
else
echo "El número ingresado no fue ni 1, 2 o 3"
fi
```

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
Ingresar un número del 1 al 3:
1
El número ingresado fue 1
```

```
Ingresar un número del 1 al 3:
5
El número ingresado no fue 1 ni 2 ni 3
```

Los condicionales

- **Creando script.**

```
[root@linux ~]# cat script09
#!/bin/bash
if [ $1 == "-d" ];
then
    df -hT /
fi
if [ $1 == "-m" ];
then
   free -h
fi
if [ $1 == "-u" ]
then
   w
fi
if [ $1 == "-c" ];
then
    uptime
fi
if [ $1 == "-h" ];
then
        echo "Mostrar información del sistema"
        echo ""
        echo "Ayuda:"
        echo "-d: Espacio en Disco"
        echo "-m: Memoria disponible"
        echo "-u: Usuarios del sistema"
        echo "-c: Carga del sistema"
        echo "-h: Muestra ayuda de este sistema"
fi
```

Los condicionales

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script09 -h
Mostrar información del sistema

Ayuda:
-d: Espacio en Disco
-m: Memoria disponible
-u: Usuarios del sistema
-c: Carga del sistema
-h: Muestra ayuda de este sistema
[root@linux ~]# ./script09 -d
Filesystem              Type  Size  Used Avail Use% Mounted on
/dev/mapper/centos-root xfs   6.2G  4.0G  2.3G  64% /
```

Los bucles

- Es importante tener cuidado y controlar que **la condición del bucle se vaya a cumplir en algún momento**, sino caeremos en un bucle infinito y tendremos que interrumpir la ejecución del script.
- Los bucles **for** en los shell scripts se basan en una o varias listas de elementos, que se asignan a una variable sucesivamente.

```
for var in lista
do
<comandos a ejecutar>
done
```

Operadores de ficheros

- Nos permitirán hacer operaciones con ficheros del sistema (archivos y/o directorios).

  -d : es un directorio
  -f : es un archivo
  -e : existe el fichero
  -r : es leíble
  -s : no está vacío
  -w : es editable
  -x : es ejecutable
  -O : eres el dueño del fichero
  -G : el grupo del fichero es igual al tuyo
  -nt : fichero1 es más reciente que fichero2
  -ot : fichero 1 es más antiguo que fichero2

Operadores de ficheros – ejemplos de uso

- **Creando script.**

```
[root@linux ~]# cat script10
#!/bin/bash
echo "Ingrese ruta de directorio"
read x
if [ -f $x ]
then
echo "Sí es un archivo"
else
echo "Puso la ruta de un directorio"
fi
```

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[root@linux ~]# ./script10
Ingrese ruta de directorio
/etc/resolv.conf
Sí es un archivo
[root@linux ~]# ./script10
Ingrese ruta de directorio
/root
Puso la ruta de un directorio
```

Operadores de ficheros – ejemplos de uso

- **Creando script.**

```
[tecsup@linux ~]$ cat script11
#!/bin/bash
echo "Ingrese ruta de un archivo"
read x
if [ -w $x ] && [ -x $x ]
then
echo "El archivo es editable y ejecutable"
elif [ -w $x ]
then
echo "El archivo es editable"
elif [ -r $x ]
then
echo "El archivo solo tiene permisos de lectura"
else
echo "El archivo no tiene ningún permiso"
fi
```

- **Validando los permisos de los archivos**

```
[tecsup@linux ~]$ ls -l lectura
-r--------. 1 tecsup tecsup 0 Jul 11 14:05 lectura
[tecsup@linux ~]$ ls -l escritura
-rw-rw-r--. 1 tecsup tecsup 0 Jul 11 14:04 escritura
[tecsup@linux ~]$ ls -l ejecutable
-rwx------. 1 tecsup tecsup 0 Jul 11 14:05 ejecutable
```

Operadores de ficheros – ejemplos de uso

- **Ejecutando script (previamente se le dio permiso de ejecución al propietario).**

```
[tecsup@linux ~]$ ./script11
Ingrese ruta de un archivo
/home/tecsup/lectura
El archivo solo tiene permisos de lectura
[tecsup@linux ~]$ ./script11
Ingrese ruta de un archivo
/home/tecsup/escritura
El archivo es editable
[tecsup@linux ~]$ ./script11
Ingrese ruta de un archivo
/home/tecsup/ejecutable
El archivo es editable y ejecutable
```

Funciones

- Se pueden reutilizar dentro de nuestro script.

```
[root@linux ~]# cat function01
#!/bin/bash
function limpiar {
        clear
}
function usuario_logeado {
        whoami
}
function mensaje {
        echo "La memoria disponible del equipo se muestra a continuación"
}
function memoria_disponible {
        free -h
}
limpiar
usuario_logeado
mensaje
memoria_disponible
```

Funciones

- Ejecutando script.

```
root
La memoria disponible del equipo se muestra a continuación
              total        used        free      shared  buff/cache   available
Mem:           972M        680M         71M         18M        220M        124M
Swap:          819M         23M        796M
```

Bucle while

- Con el bucle **while** creamos una iteración condicional

```
while condición
do
<comandos a ejecutar>
done
```

FIN DE LA UNIDAD

TECSUP
Pasión por la Tecnología

Bibliografía

- Adelstein, Torn (2007). Administración de Sistemas Operativos Linux. Madrid: Anaya Multimedia (005.43L/A23)
- Alegría Loainaz, Iñaki (2005). Linux Administración del Sistema y la Red. Madrid: Pearson Educación (005.43L/A37)
- Negus, Christopher (2013). Linux, Bible. Albany NY: A.De Boeck (005.43L/N36)