Semana 15:

Programación de Shell Scripts en Linux

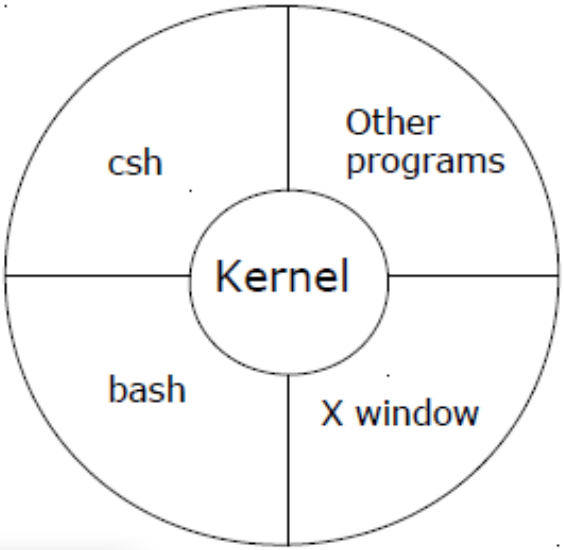
Sistemas Operativos

Capacidades de la Sesión

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

SHELL de Linux

- Es la interfaz entre el usuario final y el sistema operativo linux.
- Una plataforma linux tiene múltiples versiones pero solo utilizamos una de ellas por defecto.
- Para saber la versión de Shell que utiliza puede ayudarse ejecutando el siguiente comando "**echo $SHELL**"




```
[root@linux ~]# echo $SHELL
/bin/bash
```

SHELL de Linux

- La versión más popular de Shell ,usada en la actualidad, es **bash**.
- **bash** no es únicamente una excelente shell por CLI sino que también es un lenguaje de scripting en sí mismo.
- El shell scripting permite utilizar las capacidades de la shell para automatizar multitud de tareas que, de otra forma, requerirían múltiples comandos introducidos de forma manual.

Creacion de Scripts

- Necesitaremos conocer una herramienta de edición de textos (vi, vim, nano, etc).

- Crearemos un archivo de nombre “script01” dentro del directorio home de root; con el siguiente contenido.

Primero, tienes que indicar dónde se encuentra el intérprete de comandos.

```
[root@linux ~]# vim script01
[root@linux ~]# pwd
/root
[root@linux ~]# ls
script01
[root@linux ~]# cat script01
#!/bin/bash
echo "Esto es una prueba para TECSUP"
```

- Veremos que el archivo creado no tiene permisos de ejecución.

```
[root@linux ~]# ls -l
total 4
-rw-r--r--. 1 root root 50 Jul  4 12:55 script01
```

Creacion de Scripts (cont.)

- Procederemos a darle permisos de lectura para el propietario del archivo.

```
[root@linux ~]# chmod 744 script01
[root@linux ~]# ls -l
total 4
-rwxr--r--. 1 root root 50 Jul  4 12:55 script01
```

- Procederemos a ejecutar el archivo “ejecutable”. Para esto, anteponemos **./** delante del nombre del archivo**.**

```
[root@linux ~]# ./script01
Esto es una prueba para TECSUP
```

- Si quisiéramos crear un script que mueva todo los temporales a un directorio (“/borrado”) y luego los borre; ¿Cómo lo haríamos?

TECSUP
Pasión por la Tecnología

Creacion de Scripts (cont.)

- Recordar que los temporales se alojan en “/tmp”. Por lo tanto, crearemos un archivo (script02) con el siguiente contenido.

```
[root@linux ~]# cat script02
#!/bin/bash
mkdir /borrado
mv /tmp/* /borrado/
rm -rf /borrado/*
[root@linux ~]# pwd
/root
```

- Si listamos el directorio “/tmp” veremos que tiene contenido.

```
[root@linux ~]# ls /tmp
systemd-private-32e5ef9e99104326b3e02e18ae93cffa-bolt.service-qEIfwq
systemd-private-32e5ef9e99104326b3e02e18ae93cffa-colord.service-ReQ4Qt
systemd-private-32e5ef9e99104326b3e02e18ae93cffa-cups.service-qOgQeS
systemd-private-32e5ef9e99104326b3e02e18ae93cffa-fwupd.service-fX5v7q
systemd-private-32e5ef9e99104326b3e02e18ae93cffa-rtkit-daemon.service-KBeW6S
```

Creacion de Scripts (cont.)

- Procederemos a darle permiso de ejecución al propietario y ejecutaremos el script.

```
[root@linux ~]# chmod 744 script02
[root@linux ~]# ls -l
total 8
-rwxr--r--. 1 root root 50 Jul  4 12:55 script01
-rwxr--r--. 1 root root 65 Jul  4 13:46 script02
[root@linux ~]# ./script02
```

- Ahora, si listamos el directorio “/tmp” veremos que limpió todo el contenido.

```
[root@linux ~]# ls /tmp
[root@linux ~]# ls /borrado/
```

Creacion de Scripts – Declarando variables

- Como en cualquier lenguaje de programación, en shell scripting se pueden utilizar variables.

- Todos los valores son almacenados como cadenas de texto pero también hay operadores matemáticos que convierten las variables en números para el cálculo.

- No es necesario declarar una variable, simplemente asignándole un valor a su referencia será suficiente para crearla.

Creacion de Scripts – Declarando variables (cont.)

- Crearemos un script, en el cual declararemos una variable y le asignaremos una cadena de texto. Luego, recuperaremos la variable anteponiendo el símbolo **$** delante de su nombre.

```
[root@linux ~]# pwd
/root
[root@linux ~]# cat script03
#!/bin/bash
a="Esto es una prueba de variables"
echo $a
[root@linux ~]# chmod 744 script03
```

- Luego, ejecutamos el script.

```
[root@linux ~]# ./script03
Esto es una prueba de variables
```

Creacion de Scripts – comando “read”

- El comando read nos permite solicitar un valor de entrada para almacenarlo en una variable.

```
[root@linux ~]# pwd
/root
[root@linux ~]# cat script04
#!/bin/bash
clear
echo "¿Cómo te llamas?:"
echo "Ingresa tu nombre:"
read x
echo "Ahora sé que tu nombre es $x"
[root@linux ~]# chmod 744 script04
```

- Luego, ejecutamos el script.

```
¿Cómo te llamas?:
Ingresa tu nombre:
Roberto Rodríguez
Ahora sé que tu nombre es Roberto Rodríguez
```

Creacion de Scripts – comando “expr”

- El comando **expr** nos puede servir para realizar en shell script operaciones aritméticas. Puedes realizar sumas, restas multiplicaciones y divisiones enteras.

```
[root@linux ~]# pwd
/root
[root@linux ~]# cat script05
#!/bin/bash
clear
echo "Ingresar primer valor:"
read a
echo "Ingresar segundo valor:"
read b
c=`expr $a + $b`
d=`expr $a - $b`
echo "La suma de los 2 valores es: $c"
echo "La resta de los 2 valores es: $d"
[root@linux ~]# chmod 744 script05
```

Creacion de Scripts – comando “expr”

- Ejecutando el script.

```
Ingresar primer valor:
100
Ingresar segundo valor:
20
La suma de los 2 valores es: 120
La resta de los 2 valores es: 80
[root@linux ~]#
```

Condicionales: if ,elif,else

- La sentencia condicional tiene el siguiente formato:

```
if condición
then
    sentencias-then-if
elif condición
then
    sentencias-then-elif
else
    sentencias-else
fi
```

-lt	Menor que
-le	Menor igual que
-gt	Mayor que
-ge	Mayor igual que
-eq	Igual
-ne	distinto

Condicionales: if ,elif,else - ejemplos

- Creamos un script con el siguiente contenido.

```
[root@linux ~]# cat script06
#!/bin/bash
echo "Ingrese un número:"
read a
echo "Ingrese otro número:"
read b
if [ $a = $b ]
then
        echo "Los números ingresados son iguales"
else
        echo "Los números ingresados son diferentes"
fi
[root@linux ~]# chmod 744 script06
```

Condicionales: if ,elif,else - ejemplos

- Creamos otro script con el siguiente contenido.

```
[root@linux ~]# cat script07
#!/bin/bash
echo "Ingrese un número:"
read a
echo "Ingrese otro número:"
read b
c=`expr $a - $b`
if [ $c -gt 0 ]
then
        echo "El primer número es mayor que el segundo"
elif [ $a = $b ]
then
        echo "Los números ingresados son iguales"
else
        echo "El segundo número es mayor que el primero"
fi
```

Bucle: for

- En cada iteración la **variable** toma un valor de **SERIE**, que en caso de no contener elementos hará que no se ejecute nada y se devuelva un valor 0. En caso de que se ejecuten comandos, el resultado devuelto tras el bucle es el del último comando ejecutado.

```
for var [in lista]
do
    .....
   lista de órdenes
    .....
done
```

Bucle: for - ejemplos

- Creando un script sencillo, el cual imprimirá en consola el valor definido para la lista de variables.

```
[root@linux ~]# cat script08
#!/bin/bash
for i in 1 2 3 4 5;
do
        echo $i
done
[root@linux ~]# chmod 744 script08
[root@linux ~]# ./script08
1
2
3
4
5
```

Bucle: for - ejemplos

- Crearemos otro script, el cual cargará en una lista de variables las rutas de directorios ubicados en “/etc/sysconfig/”. Luego por cada directorio imprimirá ,en consola, la ruta absoluta y los permisos que tiene configurados.

```
[root@linux ~]# cat script09
#!/bin/bash
Archivos=$(find /etc/sysconfig/ -type d);
for i in $Archivos;do
        permisos=$(ls -ld $i);
        echo "La ruta del directorio alojado es $i";
        echo "los permisos que tiene el archivo son $permisos";
done;
[root@linux ~]# chmod 744 script09
```

Sentencia “case”

- El conjunto de palabras **case** y **esac** conforman un selector en función de un resultado.

```
case $var in
  val1)
   instrucciones;;
  val2)
    instrucciones;;
  *)
    instrucciones;;
  esac
```

Sentencia “case” - ejemplo

- Creamos un script, usando la sentencia **case**. El script tendrá el siguiente contenido.

```
#!/bin/bash
echo "Por favor ingrese un número"
read x
case $x in
0) echo "cero"
;;
1) echo "uno"
;;
2) echo "dos"
;;
3) echo "tres"
;;
*) echo "El número es mayor a 3"
;;
esac
[root@linux ~]# chmod 744 script10
```

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)