



INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

SISTEMAS OPERATIVOS



INSTITUTO POLITÉCNICO NACIONAL

Práctica 1: Introducción al shell: comandos básicos y scripts

ESCOM®

○ Alumnos:

- 1) Arcos Hermida Aldo Alejandro (N° lista 5)
- 2) Chávez Becerra Bianca Cristina (N° lista 9)
- 3) Islas Osorio Enrique (N° lista 20)
- 4) Juárez Cabrera Jessica (N° lista 21)
- 5) Palmerin García Diego (N° lista 28)

○ Grupo: 4CM1

○ Materia: Sistemas operativos

○ Profesor: Araujo Díaz David

Objetivo

Aprender el uso de los comandos básicos de intérprete de comandos de LINUX/UNIX, así como la ejecución de scripts.

Descripción

Los usuarios pueden interactuar con el sistema operativo por medio de un intérprete de comandos (Shell) o una interfaz gráfica de usuario (GUI). El intérprete de comandos es un programa del sistema que proporciona una interfaz de línea de comandos, mediante la cual el usuario indica al sistema operativo lo que se quiere hacer.

Existen diferentes intérpretes de comandos: Bourne shell (sh), C shell (csh), Bourne again shell (bash), korn shell (ksh), tenex c shell (Tcsh) y Zero shell (zsh). En la mayoría de estos el indicador del sistema consiste en el nombre del equipo, seguido por dos puntos (:), el directorio actual seguido por un carácter que indica el tipo de usuario conectado. Los usuarios pueden ser de tipo administrador (root) representado por un # o usuario normal denotado por \$.

Consola

La interfaz de línea de comandos o consola (en inglés, *command-line interface*, CLI) es un tipo de interfaz de usuario de computadora que permite a los usuarios dar instrucciones a algún programa informático o al sistema operativo por medio de una línea de texto simple.

En su forma más simple, una CLI consiste en un espacio donde se pueden escribir órdenes las cuales se las manda a un módulo interpretador de órdenes (Shell) que analiza la secuencia de caracteres recibida y, si la sintaxis de la orden es correcta, ejecuta la orden dentro del contexto del programa o del sistema operativo donde se encuentra.

Shell

El shell es un programa que te permite ejecutar otros programas. También se dice que es un intérprete de comandos, que al fin de cuentas esos comandos son programas que realizan alguna acción en el equipo.

Bash

Es una popular interfaz de usuario de línea de comandos, específicamente un shell de Unix; así como un lenguaje de scripting. Bash fue originalmente escrito por Brian Fox para el sistema operativo GNU.

Bash es un intérprete de órdenes que generalmente se ejecuta en una ventana de texto donde el usuario escribe órdenes en modo texto. Bash también puede leer y ejecutar órdenes desde un archivo, llamado guión o 'script'. Al igual que todos los intérpretes de Unix, es compatible con variables y estructuras de control para pruebas de condición e iteración.

Scripting

Los `#!` es la sintaxis que se utiliza en los scripts de shell para indicar al intérprete que ejecute el script en los sistemas operativos Unix / Linux. Además especifica que el script debe ejecutarse usando bash como intérprete, y el intérprete bash se puede encontrar en el directorio `/bin`.

```
#!/bin/bash
```

Para poder ejecutar un script en Unix/Linux necesitamos cambiar el permiso que inicialmente tiene nuestro archivo. Para eso podemos ejecutar el siguiente comando.

```
$ chmod 777 hello-world.sh
```

Obviamente, depende de cual sea la funcionalidad del script debemos de otorgarle los permisos al usuario. Pero para fines prácticos le colocamos `777` para indicarle que tiene permiso de leer, escribir y ejecutar el usuario, los grupos y todos.

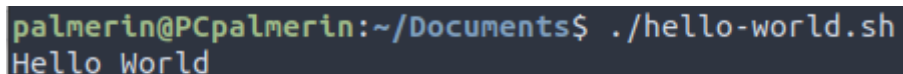
Ahora si podemos ejecutar nuestro archivo de la siguiente manera:

```
./hello-world.sh
```

Echo

```
#!/bin/bash  
echo "Hello world"
```

Salida

A terminal window showing the command `palmerin@PCpalmerin:~/Documents$./hello-world.sh` and its output `Hello World`.

```
palmerin@PCpalmerin:~/Documents$ ./hello-world.sh  
Hello World
```

IF statement

Básicamente la estructura de control If funciona como el comando `test`, por lo tanto cualquier operación lógica que queramos realizar tendrá que hacer utilizando los operadores que con este viene.

```
#!/bin/bash  
# Condiciones if  
if [ $1 -gt 500 ]  
then  
    echo Es un número grande  
elif [ $1 -gt 100 ]  
then
```

```
    echo Es un número medio
else
    echo Es un número chico
fi
```

El signo \$1 hace referencia al argumento con el que ejecutamos nuestro script, es decir:

```
./condiciones.sh <numero>
```

Entonces dependiendo de qué número hayamos ingresado \$1 tomará ese valor.

Cabe aclarar que lo que se pone dentro de los corchetes es otra forma de escribir el comando de test. Si hubiéramos escrito lo mismo pero con la otra sintaxis también se ejecutará sin problemas.

Salida

```
palmerin@PCpalmerin:~/Documents$ ./if-statement.sh 100
Es un número chico
```

```
palmerin@PCpalmerin:~/Documents$ ./if-statement.sh 1000
Es un número grande
```

```
palmerin@PCpalmerin:~/Documents$ ./if-statement.sh 300
Es un número medio
```

```
#!/bin/bash
# Condiciones if
if [[ $1 -gt 500 ]]
then
    echo Es un numero grande
elif test $1 -gt 100
then
    echo Es un numero medio
else
    echo Es un numero chico
fi
```

Algunos ejemplos de operadores lógicos disponibles son:

Operator	Description
! EXPRESSION	The EXPRESSION is false.
-n STRING	The length of STRING is greater than zero.
-z STRING	The length of STRING is zero (ie it is empty).
STRING1 = STRING2	STRING1 is equal to STRING2
STRING1 != STRING2	STRING1 is not equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is numerically equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is numerically greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is numerically less than INTEGER2
-d FILE	FILE exists and is a directory.
-e FILE	FILE exists.
-r FILE	FILE exists and the read permission is granted.
-s FILE	FILE exists and its size is greater than zero (ie. it is not empty).
-w FILE	FILE exists and the write permission is granted.
-x FILE	FILE exists and the execute permission is granted.

Si se quiere realizar una operación lógica como lo sería and(&&) o or(||). Solo tendríamos que hacerlo de la siguiente manera

```
if [ $1 -gt 500 ] || [ $1 -lt 1000 ]
```

```
palmerin@PCpalmerin:~/Documents$ ./if-statement.sh 600
Numero medio
```

While y For

Al igual que los if statement while tiene que tener una condición de paro para eso tenemos que usar el comando de test.

```
#!/bin/bash
# Estructuras de iteracion
i=0
while [ $i -lt 10 ]
do
    echo Number: $i
    ((i++))
done
```

Salida

```
palmerin@PCpalmerin:~/Documents$ ./whileFor.sh
Number: 0
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
```

El uso más común para el while, es para leer ya sea archivos o data stream. Un ejemplo de esto es:

```
#!/bin/bash
file=/etc/passwd

while read -r line; do
    echo $line
done < "$file"
```

Este programa imprimirá todo lo que haya en el archivo passwd ubicado en el directorio de etc, línea por línea.

```
palmerin@PCpalmerin:~/Documents$ ./whileFor.sh
#!/bin/bash
echo "Hello World"
```

La estructura for es muy similar, por ejemplo: el siguiente programa imprimirá los numeros del 0 al 3

```
#!/bin/bash
for i in {0..3}
do
    echo "Number: $i"
done
```

```
palmerin@PCpalmerin:~/Documents$ ./whileFor.sh
Number: 0
Number: 1
Number: 2
Number: 3
```

Si queremos ir incrementando con otro valor lo podemos hacer de la siguiente manera:

```
#!/bin/bash
for i in {0..20..5}
do
    echo "Number: $i"
done
```

```
palmerin@PCpalmerin:~/Documents$ ./whileFor.sh
Number: 0
Number: 5
Number: 10
Number: 15
Number: 20
```

Comandos

Existen una gran variedad de comandos. Algunos pueden variar en su sintaxis según la distribución que se esté utilizando. En este caso se usó Ubuntu para probar los comandos más básicos, pero muy importantes, pues también son de los más utilizados

cal (calendario) – muestra el calendario

```
bcris@LAPTOP-ENT89IS0:~$ cal
    February 2022
Su Mo Tu We Th Fr Sa
           1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28
```

date

```
bcris@LAPTOP-ENT89IS0:~$ date  
Mon Feb 21 00:19:51 CST 2022
```

pwd: Muestra la ubicación del directorio actual:

```
pwd
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop  
root@aldo-ThinkPad-L490:/home/aldo/Desktop# pwd  
/home/aldo/Desktop  
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

ls: Imprime el contenido de un directorio:

```
ls
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop  
root@aldo-ThinkPad-L490:/home/aldo/Desktop# ls  
ESCOM  
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

mkdir: Creamos la carpeta "practica1":

```
mkdir practica1
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop  
root@aldo-ThinkPad-L490:/home/aldo/Desktop# mkdir practica1  
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

cd: Nos movemos al directorio "practica1":

```
cd practica1
```



```
root@aldo-ThinkPad-L490: /home/aldo/Desktop/prac...
root@aldo-ThinkPad-L490:/home/aldo/Desktop# cd practica1
root@aldo-ThinkPad-L490:/home/aldo/Desktop/practica1# pwd
/home/aldo/Desktop/practica1
root@aldo-ThinkPad-L490:/home/aldo/Desktop/practica1#
```

cd: Regresamos al directorio anterior inmediato:

```
cd -
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop/practica1# cd -
/home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

nano: creamos el fichero “prueba.txt, guardamos y salimos del editor nano:

```
nano prueba.txt
//ctrl + o
//ctrl + z
```

```
GNU nano 4.8      prueba.txt      Modified
hola mundo
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# nano prueba.txt

Use "fg" to return to nano.

[1]+  Stopped                  nano prueba.txt
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

cat: Ver el contenido de “prueba.txt”:

```
cat prueba.txt
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# cat prueba.txt
hola mundo
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

cp: Copia el archivo "prueba.txt" a la carpeta "practica1":

```
cp prueba.txt /home/aldo/Desktop/practica1
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# cp prueba.txt /home/aldo/Desktop/practica1
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

touch: Creamos un archivo .c:

```
touch hola.c
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# touch hola.c
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

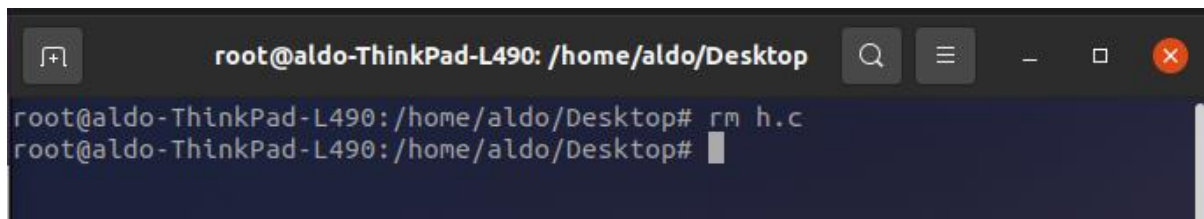
mv: Renombramos el archivo "hola.c" a "h.c":

```
mv hola.c h.c
```

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# mv hola.c h.c
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

rm: Borramos el archivo "h.c":

```
rm h.c
```

A terminal window titled 'root@aldo-ThinkPad-L490: /home/aldo/Desktop'. The window has a dark background and a light-colored title bar with standard Linux window controls (minimize, maximize, close). The terminal shows two lines of text: 'root@aldo-ThinkPad-L490:/home/aldo/Desktop# rm h.c' and 'root@aldo-ThinkPad-L490:/home/aldo/Desktop#'. The cursor is at the end of the second line.

```
root@aldo-ThinkPad-L490: /home/aldo/Desktop
root@aldo-ThinkPad-L490:/home/aldo/Desktop# rm h.c
root@aldo-ThinkPad-L490:/home/aldo/Desktop#
```

Conclusiones

La presente práctica propició un acercamiento al intérprete de comandos de LINUX/UNIX. Se identificaron algunos comandos básicos y en cada caso se implementaron en el shell, visualizando la funcionalidad de cada uno de ellos. Además, se introdujo la creación y ejecución de scripts, los cuales pueden ser de gran utilidad para automatizar tareas. Ahora en conclusión cada uno de los comandos mencionados son de vital importancia en un momento dado para la ejecución de un servidor, montarlo o desmontarlo ya que desde la creación de un usuario es importante hasta los permisos y los distintos comandos que se necesitan para poder llevar a cabo las distintas operaciones que se requieren para poder enlazar nuestro servidor a nuestro Sistema operativo.

Referencias

- GeeksforGeeks. (2021, 18 febrero). *Essential Linux/Unix Commands*. Recuperado 22 de febrero de 2022, de <https://www.geeksforgeeks.org/essential-linuxunix-commands/>
- Jurado, C. L. (2021, 18 febrero). *¿Qué es la «shell» de Linux?* CCM. Recuperado 22 de febrero de 2022, de <https://es.ccm.net/contents/316-linux-shell>
- Silberschatz, A., Baer, G. & Gagne, G. (2018) *Operating Systems Concepts*. Wiley
- Valois, M. A. (2021, 27 julio). *Comandos básicos Linux: es hora de dar tus primeros pasos*. Blog HostGator México. Recuperado 22 de febrero de 2022, de <https://www.hostgator.mx/blog/comandos-basicos-linux/>