TUTORIAL DE COMANDOS Y SCRIPTS EN LINUX

Los comandos en Linux son útiles para moverse por el sistema operativo y poder realizar tareas de forma rápida y eficaz. En este tutorial se presenta un recopilatorio de los comandos más importantes y básicos de Linux, basándose en el manejo de directorios y archivos. Posteriormente se presenta también la posibilidad de acceder a la ayuda detallada de los diferentes comandos con el "***man***" *de Linux* y una descripción de la definición y realización de un script.

Estructura de Directorios

La estructura de directorios en la que se organiza un sistema Linux es en forma de un único árbol de directorios y de forma jerárquica. Cada disco o partición tiene su propia organización lógica y al mismo tiempo pertenece a la misma estructura lógica de todo el sistema.

Cada distribución Linux hace la modificación de un estándar a la hora de estructurar el árbol de directorios, para adaptarlo a sus propias necesidades. De todas formas el estándar de la estructura de directorios es el siguiente:

Figura 1: Estructura de directorios

Manejo de directorios y archivos:

En el sistema de archivos de Linux, los directorios se organizan a manera de árbol (un directorio puede contener otros directorios distintos) a partir de un directorio llamado directorio raíz y que se denota por **'/'**. Cada directorio puede contener otros directorios o nombres de archivos.

Linux distingue mayúsculas y minúsculas, por lo que los ficheros "texto1.txt" y "Texto1.txt" son ficheros distintos. Un fichero o comúnmente denominado archivo es información de un mismo tipo, localizada en algún dispositivo de almacenamiento. Cada archivo puede tener diversos nombres en diversos directorios, cada nombre es un enlace a la información del archivo.

A continuación se describen los principales comandos para la gestión de directorios y ficheros, clasificados según su principal función.

Comandos de ayuda

Siempre que no se sabe cómo funciona o para que sirve un comando, hay que documentarse antes de usarlo y para ello tenemos estos comandos.

- **man comando:** muestra el manual del comando que le indiquemos.
- **comando help :** muestra una ayuda de los comandos
- **whatis comando:** muestra una descripción del comando
- **whereis comando:** muestra la ruta donde se encuentran los archivos relacionados al programa que permite ejecutar el comando.

Comando para cambiar de usuario

El comando de consola linux **su** (switch user) se utiliza para cambiar de usuario cuando estamos dentro de la consola de linux.

Sintaxis:
su nombreUsuario

Si usamos el comando linux **su** sin usuario, nos logueará como root por defecto, pidiéndonos el password previamente. (Súper Usuario)

Comandos para gestionar directorios

- **Navegar**

Para navegar utilizamos el comando **cd** (Change directory – Cambiar Directorio).

Sintaxis:
cd <nombre_directorio>

A la hora de movernos por los directorios usando el comando cd se pueden usar rutas relativas (Una ruta relativa es aquella que se construye a partir del directorio en el que nos encontramos) o absolutas.

Para ver los directorios existentes en la ubicación puede utilizar el comando: **ls**

Ejemplo:
Si se tiene la siguiente estructura de directorios:
/dir1/dir2/dir3

Si se está en el *dir2* y se quiere ir al *dir3, el comando usado sería*:
cd dir3

Si ahora queremos ir al *dir1*, el comando usado sería:
cd ../../

Cuando se usa *../* se baja un nivel en la estructura de directorios. En el ejemplo de arriba bajaríamos primero al dir2 y luego al dir1.

- Si se utiliza únicamente el comando ***cd***, nos lleva a nuestro directorio *home, del usuario correspondiente*. En el directorio home se encuentran típicamente carpetas como: Escritorio, Documentos, Imágenes, Descargas entre otras.

- Si se utiliza el comando ***cd /*** nos lleva al directorio raíz.

Un ejemplo de lo anteriormente planteado se puede ver en la siguiente figura:

```
daniel@debian:/$ cd
daniel@debian:~$ ls
Carpeta sin título  Documentos  Imágenes  Plantillas  Vídeos
Descargas           Escritorio  Música    Público
daniel@debian:~$ cd /
daniel@debian:/$ ls
bin   dev  home        lib    lost+found  mnt  proc  run   srv  tmp  var
boot  etc  initrd.img  lib64  media       opt  root  sbin  sys  usr  vmlinuz
```

- **Crear**

Crear directorios

Para crear un directorio usamos el siguiente comando dentro de la consola:

Sintaxis:
mkdir NombreNuevodirectorio

Ejemplo:

mkdir tierra

Crea un directorio con el nombre *tierra* donde se esté ubicado.

Para crear un directorio en una ubicación específica, esta es seleccionada a partir de **'/ '**

mkdir /home/tu_cuenta/<nombre_dir_esp>/nombre_dir_nuevo

Ejemplo:

mkdir /home/estudiante/tierra

Crea el directorio *'tierra'* en la raíz de la cuenta *'estudiante'*.

Para crear directorios dentro de otros, usamos el siguiente comando:
mkdir -p nuevo_dir_raiz/subdir1/subdir2

Ejemplo:

mkdir -p Tierra/Satélites/Jupiter

En este ejemplo se ha creado el directorio *Tierra*. Dentro de Tierra crea el directorio *Satélites*, y dentro de este el directorio *Jupiter*.

Crear archivos

Para crear un archivo usamos el siguiente comando dentro de la consola:
Sintaxis:
touch NombreNuevoArchivo

Ejemplo:

touch fichero

Crea un archivo nuevo denominado *fichero*. Si el archivo existe actualiza la hora de modificación.

- **Copiar**

Para copiar un directorio de un lugar a otro se utiliza el comando dentro de la consola:
Sintaxis
cp <ruta_origen> <ruta_destino>

Si se desea copiar un directorio y su contenido en ella se debe agregar –r, quedando el comando así:

cp –r <ruta_origen> <ruta_destino>

Ejemplo:

cp -r home/estudiante/docs home/estudiante/docs_importantes

En el ejemplo se copia el directorio docs *en el* directorio docs_importa*ntes* con todo su contenido.

- **Mover**

Para mover un directorio de un lugar a otro se utiliza el siguiente comando dentro de la consola:

Sintaxis:

mv <ruta_origen> <ruta_destino>

Ejemplo:

mv docs/presupuesto misdocs/

En este ejemplo movemos el directorio p*resupuesto* ubicado en docs al directorio *misdocs.*

- **Borrar**

Para borrar un archivo o directorio se utiliza el comando dentro de la consola:

Sintaxis:

rm <nombre_directorio>

El comando *rm* acepta también la opción *-r* (borrado recursivo) para poder eliminar directorios enteros.

Ejemplo:

rm –r copia_seguridad

En el ejemplo se borra totalmente el directorio c*opia_seguridad*.

Para borrar un directorio también se puede utilizar el siguiente comando:

Sintaxis:

rmdir <nombre_directorio>

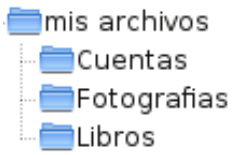
La orden rmdir sirve para eliminar un directorio cuyo nombre será el que le pasemos como argumento. Como condición para el borrado, el directorio deberá estar vacío, si no, nos mostrará un mensaje que describe que el directorio no se puede borrar.

Ejemplo:

rmdir misDocumentos

En el ejemplo se borra totalmente el directorio misDocumentos.

Ejemplo:



Ubicados dentro del directorio "mis archivos" al ejecutar el siguiente comando:

rmdir "mis archivos"

Muestra un mensaje que describe que el directorio no se puede borrar porque no se encuentra vacío

- **Listar**

Para ver los directorios existentes en la ubicación deseada se puede utilizar el siguiente comando:
ls

Utilice el comando **ls -l** para ver una descripción detallada de los permisos de cada archivo.

- **Conocer en qué directorio estamos**

Para conocer en qué directorio estamos utilizamos el comando **pwd**. Esto muestra toda ruta hasta el directorio en que nos encontramos.

Ejemplo:

Pwd

Resultado: **/home/usuario**

Comandos para gestionar archivos

- **Mover**

Para mover un directorio o archivo de un lugar a otro se utiliza el siguiente comando:

Sintaxis
mv <ruta_origen> <ruta_destino>

Ejemplo:
mv docs/Prueba.pdf misdocs/

En este ejemplo movemos el archivo *Prueba.pdf* ubicado en el directorio docs al *directorio misdocs.*

mv prueba.txt /home/mi_dir

En este ejemplo movemos el archivo p*rueba.txt* al directorio mi_dir.

Si se desea renombrar un archivo podemos utilizar **mv**.

Ejemplo:
mv Prueba.txt NuevoArchivo.txt

Con este comando también podemos renombrarlo, pasaría de llamarse *Prueba.txt* a *NuevoArchivo.txt.*

- **Copiar**

Para copiar un archivo de un lugar a otro se utiliza el siguiente comando:

Sintaxis:
cp <archivo> <dirección_donde_pegar>

Ejemplo:
Supongamos que *Prueba.txt* esta en /home/Estudiante/Prueba.txt y nosotros en "/":

cd /home/Estudiante/

Para desplazarnos hacia el directorio

cp Prueba.txt /home/Midirectorio/

Se haría una copia del archivo *Prueba.txt* en el directorio *Midirectorio*.

- **Borrar:**
  Para eliminar archivos usamos el comando "rm", el cual debe usarse con cuidado ya que una vez se borren archivos con este comando no podrán ser recuperados.

  Ejemplo:
  rm -i Prueba.txt

  Pregunta si realmente deseamos eliminar el archivo Prueba.txt.

  Ejemplo:
  rm -rf directorio

  La opción **-r** borra todos los archivos y directorios de forma recursiva. Por otra parte, **-f** borra todo sin pedir confirmación. Estas opciones pueden combinarse causando un borrado recursivo y sin confirmación del directorio que se especifique.

- **Permisos**
  En Linux se pueden dar diferentes tipos de permisos a los archivos y permisos que tenemos en nuestro sistema. Para ello cada archivo tiene 3 tipos de permisos para cada propietario, grupo y el resto que no pertenecen a los dos anteriores.

  Los tipos de permisos que podemos encontrar son los siguientes:

  r Permiso de lectura, a un archivo permite leer el contenido, a un directorio permite listar el contenido de dicho directorio.

  w Permiso de escritura, a un archivo permite modificar el contenido, a un directorio permite crear y borrar contenidos dentro del directorio.

  x Permiso de ejecución, a un archivo permite ejecutar ese archivo, a un directorio permite acceder al directorio.

  Si listamos el contenido de un directorio con **ls -l** podremos ver cuáles son los permisos de cada directorio.

  Ejemplo:

  -rw-r--r-- 1 lostscene lost 4960 Jul 11 12:12 prueba.txt

  Del comando anterior, cada campo significa:

  - indica que se trata de un archivo, si fuera un directorio mostraría el "d" y si fuera un enlace simbólico "l".

rw- permisos para el propietario del archivo que en este caso se llama *lostscene*. Los permisos que tendría son lectura y escritura, no tendría ejecución porque aparece el caracter "-".
r-- esos tres caracteres son los permisos para el grupo al que pertenece el archivo que en este caso se llama *lost*. Y tiene únicamente permiso de lectura.
r--esos tres caracteres son los permisos para aquellos que no son el propietario ni el grupo al que pertenece el archivo. Y tiene únicamente el permiso de lectura.

Para cambiar los permisos de un archivo usamos el comando **chmod**. Este comando se puede usar de varias formas.

1. Forma Octal: Consiste en que cada permiso tiene un valor, los valores son los siguientes:

   - Permiso de lectura (r) : vale **4**
   - Permiso de escritura (w) : vale **2**
   - Permiso de ejecución (x) : vale **1**

Para dar permisos con la forma octal lo que se hace es sumar lo que vale cada uno de los permisos que vamos a dar.

Ejemplo: Si para el archivo anterior le queremos dar todos los permisos al propietario, permiso de lectura al grupo y al resto ningún permiso lo hacemos colocando el siguiente comando:

chmod 740 prueba.txt

Primero se coloca **chmod** luego los permisos y después al directorio o archivo al cual lo queramos aplicar. Ponemos 740 porque 7 es la suma de los valores de los permisos que le damos al propietario, 4 al grupo y 0 al resto:

- Permisos para el propietario: 4 (valor del permiso de lectura) + 2 (valor del permiso de escritura) + 1 (valor del permiso de ejecución) = 7.
- Permisos para el grupo: 4 (valor del permiso de lectura) + 0 (porque no le queremos dar el permiso de escritura) + 0 (porque no le queremos dar el permiso de ejecución) = 4.
- Permisos para el resto: 0 (porque no le queremos dar el permiso de lectura) + 0 (porque no le queremos dar el permiso de escritura) + 0 (porque no le queremos dar el permiso de ejecución) = 0.

Si por ejemplo quisiéramos dar todos los permisos al usuario, grupo y al resto sería:

chmod 777 prueba.txt

2. La otra forma de dar permisos es indicar al comando **chmod** a quien se lo queremos dar y qué permiso. A quien se lo queremos dar se indica de la siguiente forma:
   - **u:** propietario del archivo
   - **g:** grupo al que pertenece el archivo
   - **o**: aquellos que no son el propietario
   - **a:** a todos

Los permisos se indican de la siguiente forma:
- **r:** permiso de lectura
- **w:** permiso de escritura
- **x:** permiso de ejecución

Para dar, quitar o igualar permisos se indican con los siguientes caracteres:
+ se añade un permiso
- se quita un permiso
= se igualan los permisos

Ejemplo:
Si queremos quitar al grupo el permiso de ejecución se colocaría:
chmod g-x prueba.txt

Si queremos dar todos los permisos al propietario sería:
chmod u+rwx prueba.txt

Si por ejemplo al grupo le queremos dar el permiso de lectura y al resto quitarle el permiso de ejecución sería:
chmod g+w,o-x prueba.txt

- **Listar**

  Para listar los archivos existentes en un directorio usamos el comando ls.

  Sintaxis:
  ls nombre_directorio

  Ejemplo:

  ls documentos/

  Para ver qué archivos hay en el directorio d*ocumentos*:

  ls -a

  Muestra todos los archivos del directorio actual.

  ls -l -h *.for

Muestra todos los atributos (-l) de todos (*) los archivos que terminan con .for, además muestra el tamaño (-h) en Bytes, KBytes, etc.

- **Manual de Linux**

  Este comando se utiliza para llamar al Manual de Linux y preguntarle sobre un comando en concreto y sus opciones de uso.

  Sintaxis:
  man NOMBRECOMANDO

  Si el *NOMBRECOMANDO* existe, se abrirá la aplicación *MAN* con toda la información referente a ese comando, todas sus opciones y explicaciones.

  Una vez se encuentra en la descripción provista por la aplicación *man*, se puede salir pulsando la tecla ESC y luego escribiendo: **q**., o simplemente escribiendo q. Algunos comandos aceptan el parámetro *--help* para mostrarnos información de sus opciones. Podemos usar este comando para no tener que llamar a *man*.ç

  NOMBRECOMANDO --help

  Ejemplo:
  man chmod

  Con lo anterior se nos desplegarán las páginas del manual del comando "chmod", donde se describe a detalle la función del mismo, así como los parámetros que acepta y el cómo utilizarlo.

- **Comprimir y descomprimir archivos**

  Para comprimir y descomprimir archivos están los comandos *gzip* y *bzip2*. Estos dos comandos solo permiten comprimir y descomprimir un archivo a la vez y no pueden comprimir y descomprimir directorios.

  gzip
  gzip se complementa con el comando gunzip el cual permite descomprimir archivos comprimidos con gzip.

  Sintaxis:
  gzip <nombre_documento>

Ejemplo:

gzip Documento

Crea el archivo *Documento.gz* y el archivo original desaparece.

Ahora para descomprimirlo se ejecuta el siguiente comando:

gzip -d Documento.gz

O también:

gunzip Documento.gz

Descomprime el archivo Documento.gz y crearía el archivo Documento.

Al usar cualquiera de los dos comandos al comprimir se genera un archivo .gz y desaparece el archivo de origen, al descomprimirlo se crea el archivo comprimido y desaparece el archivo .gz.

bzip2
bzip2 se complementa con el comando bunzip2 el cual descomprime archivos comprimidos con *bzip2*.

Sintaxis:
bzip2 <nombre_documento>

Ejemplo:

bzip2 Documento

Esto crearía el archivo D*ocumento.bz2* y el archivo Documento desaparece.

Ahora para descomprimirlo se ejecuta el siguiente comando:

bzip2 -d Documento.bz2

O también:

bunzip2 Documento.bz2

Esto descomprimiría el archivo *Documento.bz2* y crearía el archivo Documento.

- **Empaquetar Directorios:**

  Empaquetar es agrupar en un solo fichero varios archivos y/o directorios, en cambio, comprimir significa aplicar un algoritmo que hará que varios archivos ocupen menos espacio en el disco.

  El comando **tar** permite crear a partir de varios archivos un único archivo que los contiene. También permite empaquetar en un único archivo uno o varios directorios.

  Ejemplo:
  Si se quiere hacer un paquete de la siguiente estructura de directorios directorio1/directorio2/directorio3 hay que ejecutar el siguiente comando:

  tar vcf directorioEmpaquetado.tar directorio1

  Esto coloca en el archivo *directorioEmpaquetado.tar* toda la estructura de directorios a partir del *directorio1* con todos sus archivos.

  A diferencia de **gzip** y **bzip2** no desaparece el archivo o directorios que empaquetemos.

  Ahora si queremos desempaquetar el archivo que acabamos de crear hay que ejecutar el siguiente comando:

  tar vxf directorioEmpaquetado.tar

  Con esta orden desempaquetaríamos el archivo y nos generaría de nuevo la estructura de directorios. Al desempaquetar el archivo .tar no se borra el archivo .tar.

  Si se quiere hacer con gzip para empaquetarlo y comprimirlo hay que utilizar el siguiente comando:

  tar cfvz directorioComprimido.tar.gz directorio

  Ahora para desempaquetarlo y descomprimirlo se realiza de la siguiente forma:

  tar xfvz directorioComprimido.tar.gz

  Si se quiere desempaquetarlo y descomprimirlo a un directorio concreto se hace de la siguiente forma:

  tar zxvf directorioComprimido.tar.gz -C directorioDestino

  Ahora para hacer lo mismo pero comprimiéndolo con bzip2 hay que utilizar el siguiente comando:

tar jfvc directorioComprimido.tar.bz2 directorio

Ahora para desempaquetarlo y descomprimirlo se hace de la siguiente forma:

tar jfvx directorioComprimido.tar.bz2

Si se quiere desempaquetarlo y descomprimirlo a un directorio concreto se hace de la siguiente forma:

tar jxvf directorioComprimido.tar.bz2 -C directorioDestino

SCRIPTS

- **¿Qué es un Script?**

El shell es un intérprete de órdenes; los intérpretes de órdenes de Linux son auténticos lenguajes de programación. Como tales, incorporan sentencias de control de flujo, sentencias de asignación, funciones, etc. Los programas de shell no necesitan ser compilados como ocurre en otros lenguajes, en este caso, el propio shell los ejecuta línea a línea. A estos programas se les conoce con el nombre de *shell scripts* y son los equivalentes a los archivos por lotes de otros sistemas operativos.

- **¿Cómo se hace un Script?**

Para realizar un Script en Linux, básicamente debemos realizar:

1. Creación

Para crear un script en Linux solo debemos crear un archivo de texto con un editor de texto gráfico como gedit o un editor en terminal como vim, nano o emacs. Este archivo contendrá las órdenes que el Shell va a ir interpretando y ejecutando. La extensión del archivo debe ser .sh .

Ejemplo:
HolaMundo.sh

En la primera línea del script se debe indicar que shell se va a usar (/bin/bash/ , /usr/bin/perl , etc.).

Ejemplo:

#! /bin/bash

"#!" Se conoce con el nombre de Sha Bang. Su función es indicarle al sistema que se trata de un conjunto de comandos para que sean interpretados. En realidad, es un número mágico de dos bytes. El número mágico es un marcador especial para indicar el tipo de archivo, en este caso, indica que se trata de un script de shell ejecutable.

Para introducir comentarios se debe poner **#** por cada línea de un comentario. Para mostrar comentarios que luego veremos por pantalla, se utilizará el comando:
echo <texto>

2. Asignar Permisos de Ejecución

Posteriormente a su creación se debe asignar permisos de ejecución al archivo creado, utilizando la orden **chmod**, como ya se explicó anteriormente en la sesión *Permisos* de un *Archivo*.

3. Ejecutar Script

Para ejecutar un script se utiliza el comando:
sh <NombreScript>

Cuando se ejecuta un script es posible que se necesite el paso de parámetros. En el caso de los scripts, los parámetros se encuentran en variables especiales que identificamos como $1, $2, $3, etc. El nombre de la variable nos indica la ubicación del parámetro, la variable $0 indica por defecto el nombre del archivo.

Ejemplo:

Ejemplo.sh

```
#!/bin/bash
echo "El nombre del programa es $0"
echo "El primer parámetro recibido es $1"
echo "El segundo parámetro recibido es $2"
```

De tal forma que si ejecutamos el Ejemplo.sh así:

sh Ejemplo.sh Hola Mundo

Al ejecutar el script mostrara:

```
El nombre del programa es Ejemplo.sh
El primer parámetro recibido es Hola
El segundo parámetro recibido es Mundo
```