Comandos Linux

Como estrategia, vamos a priorizar los comandos en base a su funcionalidad.

Entendiendo el prompt

El **prompt** en la shell de **linux** es la información o el símbolo que se encuentra antes del cursor, es decir donde empezamos a escribir los comandos que ingresamos a la shell.

Nos muestra información útil para situarnos en el contexto de la shell. Sigue la siguiente estructura:

- Usuario con el que ejecutamos el comando actual (En el ejemplo el usuario kali)
- Equipo donde ejecutamos el comando actual (En el ejemplo kali)
- Ruta en la que nos encontramos situados (En el ejemplo en /)
    - En caso de que la ruta actual esté dentro del home del usuario actual (/home/kali) la ruta puede verse acortada por el símbolo ~.
        - IMPORTANTE: ~ se utiliza como sinónimo del home del usuario (/home/kali para el usuario kali por ejemplo)

```
┌──(kali㉿kali)-[/]
└─$
```

Ruta relativa vs Ruta absoluta

Muchos de los comandos que veremos más adelante necesitan que le especifiquemos rutas de ficheros para su funcionamiento. Estas rutas pueden ser absolutas o relativas en función de qué directorio tomemos como referencia (punto de partida):

- Ruta absoluta: Cuando hablamos de ruta absoluta nos referimos a la ruta donde se encuentra el fichero desde el inicio del sistema de ficheros. Es decir, desde /.

```
/home/kali/Descargas/fichero.txt
```

- Ruta relativa: Cuando hablamos de ruta relativa nos referimos a la ruta donde se encuentra el fichero desde la carpeta donde nos encontramos en el terminal.

```
Si el terminal se encuentra en /home

kali/Descargas/fichero.txt

Si el terminal se encuentra en /home/kali

Descargas/fichero.txt

Si el terminal se encuentra en /home/kali/Descargas

fichero.txt
```

Para formar las rutas relativas tenemos caracteres "especiales" que nos facilitan la vida:

- . : Hace referencia al directorio actual
- .. : Hace referencia al directorio padre
- ~ : Hace referencia al home del usuario

Siguiendo el ejemplo anterior, si la shell se encuentra en /home/kali todas las rutas siguientes son equivalentes:

```
/home/kali/Descargas/fichero.txt (ruta absoluta)

Descargas/fichero.txt (ruta relativa)

~/Descargas/fichero.txt (ruta relativa desde el home del usuario kali ~)
```

Si la shell se encontrase en /home/kali/Escritorio por ejemplo, sería necesario volver a la carpeta /home/kali para llegar a Descargas. Por tanto:

```
../Descargas/fichero.txt

# .. -> un paso para atrás, hacia el directorio padre
```

Comandos para desplazarnos por la terminal

pwd

Muestra el nombre del directorio. Con pwd (abreviatura de print working directory) la consola muestra el nombre del directorio de trabajo (en curso).

Sintaxis:

```
$> pwd
/home/kali/Descargas
```

ls

Muestra el contenido del directorio como una lista. La orden ls equivale a list y se utiliza para mostrar el contenido de un fichero (los nombres de todos sus archivos y carpetas).

Sintaxis:

ls

ls [ruta_directorio]

```
$> ls [OPCIONES] DIRECTORIO

$> ls /home/kali/Descargas

$> ls | grep lo que queramos buscar
```

Además, ls tiene 2 parámetros especialmente interesantes:

- -l para mostrar los resultados en modo lista detallada (para ver los permisos asociados y el tamaño)
- -a para mostrar también ficheros ocultos (aquellos cuyo nombre empiezan por un punto)

ls -la

ls -la [ruta_directorio]

Sintaxis:

```
$> ls -la

$> ls -la /home/kali/Descargas
```

cd

Navega por el árbol de ficheros. El comando cd es la abreviatura de change directory y se utiliza para navegar por el directorio.

Sintaxis:

cd [directorio_destino]

```
$> cd Descargas
```

cd [directorio_padre]

```
$> cd ..
```

whoami

Muestra el nombre de usuario propio. Utiliza el comando whoami para obtener tu propio nombre de usuario.

Sintaxis:

```
$> whoami
$> root
```

history

Esta función sirve de ayuda al introducir órdenes en la consola y permite ejecutar de nuevo un comando introducido con anterioridad seleccionándolo con ayuda de las flechas del teclado y confirmándolo con la tecla Enter. Utilizando la orden sin opciones ni argumentos se obtiene la lista completa de comandos numerada.

Sintaxis:

```
$> history
$> history | grep PALABRA CLAVE
```

tree

Presenta los directorios en forma de árbol. Mientras que ls muestra el contenido de los directorios como lista, el comando tree, siguiendo este esquema sintáctico, muestra la jerarquía completa del directorio en forma de árbol.

Sintaxis:

```
$> tree
```

grep

Explora archivos de texto. Con la orden grep (global regular expression print) se pueden explorar archivos de texto tales como archivos de registro. Como patrón de búsqueda se pueden utilizar secuencias de caracteres o expresiones regulares.

Sintaxis:

Para buscar una palabra en un archivo de texto:

```
$> grep búsqueda archivo
$> cat archivo | grep búsqueda
$> grep -r búsqueda carpeta
```

Encontrar una palabra sin tener en cuenta las mayúsculas y minúsculas:

```
$> grep -i búsqueda archivo
```

Buscar múltiples palabras clave:

```
$> grep búsqueda1 archivo | grep búsqueda2 archivo
$> grep -e búsqueda1 -e búsqueda2 archivo
```

Cantidad en la búsquedas :

```
$> grep -c -i búsqueda1 archivo
$> grep -i búsqueda1 | wc -l
```

locate

Localizar un archivo. El uso del argumento **-i** junto con este comando hará que no distinga entre mayúsculas y minúsculas.

Sintaxis:

```
$> locate -i escuela*nota -->  buscará cualquier archivo que contenga la palabra «escuela» y «nota», ya sea en mayúsculas o minúsculas.
```

find

Explora el sistema de archivos. find es un programa de líneas de comandos cuya función es buscar archivos en el sistema.

Sintaxis:

```
$> find /home/ -name notas.txt buscará un archivo llamado **notas.txt** dentro del directorio de inicio y sus subdirectorios.
```

groups

Muestra información sobre grupos. Con groups el terminal enumera los grupos a los que pertenece una determinada cuenta de usuario.

Sintaxis:

```
$> groups nombre del usuario
$> cat /etc/group | grep nombre del grupo
```

less

Muestra el contenido de un archivo de texto en la consola.

Sintaxis:

```
$> less nombre del fichero
```

tail

Muestra las 10 últimas líneas de un archivo.

Sintaxis:

```
$> tail -n nombredearchivo.ext
```

Comandos para la gestión de archivos

touch

Permite crear un fichero vacío con el nombre que se pase como parámetro.

Sintaxis:

touch [nombre_del_fichero]

```
$> touch fichero.txt

$> touch /home/kali/fichero.txt
```

cat

Muestra el contenido de un fichero directamente en la terminal:

Sintaxis:

```
$> cat fichero.txt
Soy el contenido de fichero.txt

$>
```

rm

Borra archivo o directorio. El programa rm (remove) borra archivos o directorios de forma irreversible.

Sintaxis:

rm [nombre_del_fichero]

```
$> rm fichero.txt
$> rm [OPCIONES] ARCHIVO1 ARCHIVO2
```

Si se le añade el parámetro -r también permite borrar directorios (recursivamente) con o sin otros ficheros dentro.

Sintaxis:

rm -r [nombre_del_fichero] (--recursive)

```
$> rm -r Descargas/
```

mkdir

Permite crear un directorio (carpeta).

Sintaxis:

mkdir [nombre_del_directorio]

```
$> mkdir Ejercicio

# Se pueden añadir más directorio a crear en la misma línea

$> mkdir Ejercicio1 Ejercicio2 Ejercicio3
```

rmdir

Permite borrar un directorio (carpeta) siempre y cuando se encuentre vacío (sin ficheros dentro).

Sintaxis:

rmdir [nombre_del_directorio]

```
$> rmdir Ejercicio
```

cp

Permite copiar un fichero o un directorio (origen) a una nueva ruta (destino).

Sintaxis:

cp [fichero_origen] [ruta_destino]

```
$> cp fichero.txt /home/kali/Ejercicio
```

mv

Permite mover un fichero o un directorio (origen) a una nueva ruta (destino).
mv puede utilizarse también para cambiar el nombre de los archivos.

Sintaxis:

mv [fichero_origen] [ruta_destino]

```
$> mv fichero.txt /home/kali/Ejercicio
$> mv fichero.txt nuevonombre.txt
```

nano

Editor de textos.

Sintaxis:

```
$> nano nombre del archivo
```

echo

Muestra una cadena en la salida estándar.

Sintaxis:

```
$> echo Esto es un ejemplo
$> Esto es un ejemplo
```

Comandos para la gestión de permisos

chmod (change mode)

Permite modificar los permisos asociados a un fichero. Para ello, especificaremos los valores en **decimal** para cada uno de los 3 bloques de permisos:

- Permisos para el creador
- Permisos para el grupo
- Permisos para el resto de usuarios

Recordatorio: Los permisos para cada uno de los grupos son 3 (rwx). Para calcular el valor decimal que queremos asignar, primero generamos el número binario, poniendo a 1 los permisos que queremos otorgar.

- Por ejemplo
  - 111 -> rwx (lectura, escritura y ejecución)
  - 101 -> r-x (lectura y ejecución)
  - 100 -> r-- (sólo lectura)
  - etc

Estos valores tendremos que pasarselos como número decimal al comando chmod para modificar los permisos.

- Por ejemplo
  - 111 -> 7
  - 101 -> 5
  - 100 -> 4

Sintaxis:

chmod 750 [nombre_fichero]

```
$> chmod 750 fichero.txt
```

chown

Permite modificar el creador del fichero.

Sintaxis:

chown [usuario] [nombre_fichero]

```
$> chown root fichero.txt
```

chgrp

Permite modificar el grupo asociado al fichero.

Sintaxis:

chgrp [grupo] [nombre_fichero]

```
$> chgrp administradores fichero.txt
```

su

Permite cambiar de un usuario a otro.

Sintaxis:

su [usuario]

```
$> su root
Password: ....
```

sudo

Permite a los usuarios no root ejecutar comandos que normalmente requieren privilegios. Ejecuta programas con los permisos de otros usuarios. La instrucción sudo (substitute user do) puede anteponerse a una llamada al sistema para ejecutarla con los derechos de un usuario diferente de forma segura. Generalmente se requiere la contraseña del usuario que realiza la invocación. Cuando se introduce el comando sin indicar ningún nombre de usuario, se utiliza el superusuario root como usuario

Sintaxis:

sudo [usuario]

```
$> sudo su
$> su nombre
```

Ejecutar un fichero

./

Permite ejecutar un fichero del tipo .bin, .sh y .run

Sintaxis:

./ [fichero]

```
$> ./comando.sh
```

ftp

Transfiere archivos por FTP.

Sintaxis:

```
$> ftp IP
```

ping

Comprueba la conexión de red.

Sintaxis:

```
$> ping 8.8.8.8
```

wget

Descargar archivos de Internet

Sintaxis:

```
$> wget seguido del enlace de descarga.
```