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 /hom/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

Nos muestra la ruta actual en la que nos encontramos dentro del sistema de ficheros

Sintaxis:

pwd

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

ls

Permite listar los ficheros o directorios que contiene la ruta actual en la que me encuentro. También permite listar los ficheros de otras rutas si se le pasa como parámetro.

Sintaxis:

ls

ls [ruta_directorio]

```
$> ls

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

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]

```
$> ls -la

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

cd

Permite movernos entre los directorios del sistema de ficheros.

Sintaxis:

cd [directorio_destino]

```
$> cd Descargas
```

cd [directorio_padre]

```
$> cd ..
```

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:

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

$>
```

rm

Permite borrar el fichero que se le pasa como parámetro.

Sintaxis:

rm [nombre_del_fichero]

```
$> rm fichero.txt
```

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

Sintaxis:

rm -r [nombre_del_fichero]

```
$> 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).

Sintaxis:

mv [fichero_origen] [ruta_destino]

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

Comandos para la gestión de permisos

chmod

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

Sintaxis:

sudo [usuario]

```
$> sudo su
```

Ejecutar un fichero

./

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

Sintaxis:

./ [fichero]

```
./comando.sh
```