MANUAL 2

SHELL SCRIPTS

COMANDOS BÁSICOS

Contenido

1. Introducción

Existe un conjunto de comandos que todo usuario debe conocer para poder manejarse en un sistema Linux. La mayoría de estos comandos están relacionados con el manejo de archivos y directorios

2. Comandos

Un comando típico de Linux suele estar formado por el nombre de un **programa** y de **opciones** y **argumentos**:

Sintaxis:

```
comando [opciones] [argumentos]
```

Ejemplo:

```
$ wc –l miArchivo
```

El nombre del programa (`wc`, comando para contar palabras) hace referencia a un programa que el shell localiza y ejecuta. Las opciones, que suelen comenzar con un guión, afectan al comportamiento del programa. En el ejemplo anterior, la opción `–l` indica a `wc` que cuente líneas y no palabras. El argumento `miArchivo` especifica el archivo que `wc` debe leer y procesar.

Los comandos pueden tener varias opciones y argumentos.

Las opciones se pueden proporcionar de forma individual:

```
$ wc –l -w miArchivo              #Dos opciones individuales
```

O se pueden combinar detrás de un único guión (aunque algunos comandos no reconocen las opciones combinadas):

```
$ wc –lw miArchivo                #Igual que –l -w
```

También se permiten varios argumentos:

```
$ wc –l miArchivo1 miArchivo2     #Contar líneas de dos archivos
```

A continuación, se describen un conjunto de comandos básicos:

2.1. ls (listar archivos)

Sirve para listar archivos.

Sintaxis:

```
ls [opciones] [archivos]
```

Si se ejecuta ls sin argumentos, dará como resultado un listado de todos los archivos (incluyendo directorios) del directorio donde el usuario está posicionado

```
$ ls
```

O puede enumerar los archivos de directorios concretos:

```
$ ls dir1 dir2 dir3
```

Opciones más importantes

`-a`	Enumera todos los archivos, incluidos aquellos cuyo nombre empieza por un punto (archivos ocultos).
`-l`	Muestra información detallada de cada archivo (atributos del archivo)
`-s`	Antepone el tamaño del archivo en bloques. Muy útil para ordenar los archivos por tamaño: Ejemplo: ls –s | sort –n
`-R`	Lista los contenidos de todos los directorios recursivamente.
`-t`	Lista los archivos ordenados por el tiempo de modificación en vez de ordenarlos alfabéticamente.
`-S`	Ordena el listado por el tamaño de los archivos.

El comando `ls` devuelve como estado de salida un número diferente de 0 cuando ls intenta listar un archivo que no existe.

```
$ ls abc
ls: abc: No such file or directory
$ echo $?
1
```

2.2. cp (copiar archivos)

Se utiliza para copiar archivos.

Sintaxis:

```
cp [opciones] archivos (archivo | directorio)
```

En su forma más sencilla, copia archivo en archivo2 (si archivo2 no existe lo crea y si existe sobrescribe su contenido).

```
$ cp archivo archivo2
```

O copia varios archivos a un directorio

```
$ cp archivo1 archivo2 archivo3 directorio_destino
```

Opciones más importantes

`-p`	Además de copiar el contenido del archivo, copia también sus permisos, marcas de tiempo, etc.
`-i`	Modo interactivo. Pregunta antes de sobreescribir los archivos de destino
`-R`	Copia directorios recursivamente
`-a`	Copia directorios de forma recursiva (conservando todos los atributos de los archivos como permisos y marcas de tiempo)
`-u`	No copia un archivo (no directorio) si en el destino ya existe tal archivo, el cual tiene igual tiempo de modificación o más reciente

2.3. mv (mover o renombrar archivos)

Este comando se usa tanto para mover archivos, como para renombrarlo

Sintaxis:

```
mv [opciones] origen destino
```

Si se indica dos nombres, el comando mv cambiará el nombre de un archivo:

```
$ mv nombre-actual nombre-nuevo
```

Mueve archivos y directorios a un directorio de destino

```
$ mv archivo1 archivo2 dir3 directorio_destino
```

Opciones más importantes

`-i`	Modo interactivo. Pregunta antes de sobrescribir los archivos de destino.

2.4. cd (cambiar de directorio)

Este comando se usa para cambiar de directorio.

Sintaxis:

```
cd [directorio]
```

Ejemplo: Cambia al directorio `/home/juan/ejercicios`

```
$ cd /home/juan/ejercicios
```

Si no se indica un directorio, de forma predeterminada cd selecciona el directorio principal del usuario (el almacenado en la variable HOME)

```
$ cd
```

2.5. touch (cambiar fecha de acceso/modificación o crear archivo vacío)

Este comando se utiliza para cambiar la fecha de acceso y/o modificación a un archivo.

Sintaxis:

```
touch [opción] archivo
```

Si el argumento `archivo` corresponde al nombre de un archivo que no existe, `touch` creará el archivo con dicho nombre y sin ningún contenido.

Opciones más importantes

`-a`	Cambia solamente la hora de acceso.
`-m`	Cambia solamente la hora de modificación
`-d fecha`	Usa `fecha` en lugar de la fecha actual. El formato de fecha es el siguiente: MMDDHHMMAAAA. Por ejemplo, para representar el 7 de abril de 2019 a la 1:00 a.m., se escribirá: 040701002019. Si el año a usar es el año actual, se puede obviar, entonces el ejemplo anterior quedaría así: 04070100.

2.6. rm (eliminar archivos o directorios)

El comando rm permite eliminar archivos o directorios.

Sintaxis:

```
rm [opciones] archivos | directorios
```

Permite eliminar varios archivos indicando el nombre de cada uno de ellos:

```
$ rm archivo1 archivo2 archivo3
```

Permite eliminar directorios de forma recursiva

```
$ rm -r dir1 dir2
```

Opciones más importantes

`-i`	Modo interactivo. Pregunta antes de eliminar el archivo
`-r`	Elimina un directorio y su contenido de forma recursiva.

2.7. ln (crear enlaces a archivos)

El comando ln crea enlaces a un archivo

Sintaxis:

```
ln [opciones] origen destino
```

Un enlace es una referencia a otro archivo. De forma intuitiva, los enlaces asignan varios nombres al mismo archivo, lo que le permite existir en dos (o más) ubicaciones simultáneamente.

Hay dos tipos de enlaces: **simbólicos** y **físicos**

Un **enlace simbólico** hace referencia a otro archivo por su ruta, similar a un acceso directo de Windows. Para crear un enlace simbólico se utiliza la opción `-s`:

```
$ ln -s archivo enlaceSimbolico
```

Si se borra el archivo original, el enlace no será válido y apuntará a una ruta de archivo inexistente.

Un **enlace físico**, es simplemente un segundo nombre de un archivo físico en el disco. Si se borra el archivo original, el enlace sigue funcionando. Para crear un enlace físico:

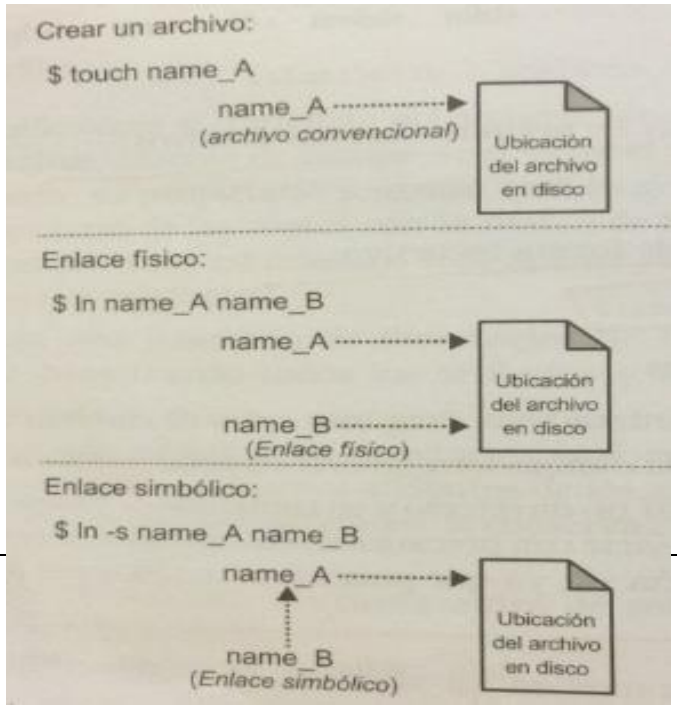
```
$ ln archivo enlaceFisico
```

Ejemplo:

Cuando se ejecuta `ls -l` en un directorio donde hay un enlace simbólico, éste se nota de la siguiente manera:

```
$ ls -l claves
lrwxrwxrwx    1 usuario usuario 11 Apr  8 13:33 claves -> /etc/passwd
```

La «l» al comienzo de la línea específica el tipo de archivo listado, en este caso, un enlace.



2.8. pwd (mostrar nombre del directorio actual)

Este comando imprime en pantalla el directorio donde el usuario está trabajando.

En lugar del comando pwd podemos utilizar también la variable de entorno PWD, ya que almacena la ruta del directorio actual en el que nos encontramos.

Ejemplo: Si se ejecuta el comando pwd en el directorio principal del usuario juan obtenemos:

```
$ pwd
/home/juan
```

2.9. basename (imprimir último componente de una ruta de archivo)

El comando basename imprime el último componente de una ruta de archivo.

```
$ basename /home/juan/trabajos/documento1.txt
documento1.txt
```

2.10. mkdir (crear directorios)

El comando mkdir crea directorios

Sintaxis:

```
mkdir [opciones] directorios
```

Permite crear uno o más directorios:

```
$ mkdir dir1 dir2 dir3
```

Opciones más importantes

-p	Crea los directorios padre que falten para cada argumento `directorio`.
-m modo	Crea el directorio con los permisos indicados

Ejemplo: Crea `/uno` y `/uno/dos` si no existen y, tras ello, `/uno/dos/tres`

```
$ mkdir –p /uno/dos/tres
```

Ejemplo: Crea el directorio `misCosas` con los permisos indicados

```
$ mkdir –m 0755 misCosas
```

2.11. rmdir (eliminar directorios vacíos)

El comando rmdir elimina directorios vacios

Sintaxis:

```
rmdir [opciones] directorios
```

Opciones más importantes

-p	Dada una ruta de directorio, elimina no solo el directorio indicado sino también los directorios principales especificados, que deben estar vacíos.

Para eliminar un directorio no vacío y su contenido:

```
$ rm –r directorio
```

2.12. man (ayuda en línea)

El comando man sirve para desplegar en pantalla las páginas de manual, que proporcionan ayuda en línea acerca de cualquier comando, función de programación, archivo de configuración, etc.

Ejemplo: Muestra la ayuda acerca del comando `ls`

```
$ man ls
```

IMPORTANTE: Si no conocemos el nombre del comando que queremos buscar, pero sabemos alguna palabra clave sobre lo que hace podemos utilizar el comando man de la siguiente forma:

```
$ man –k "texto que busco"
```

Ejemplo: Queremos buscar el nombre de un comando que liste directorios (sería el comando `ls` pero no nos acordamos del nombre)

```
$ man –k "list directory"

dir (1)          - list directory contents
ls (1)           - list directory contents
Ntfsls           - list directory contents on an NTFS filesystem
Vdir             - list directory contents
```

2.13. whoami (mostrar el nombre de usuario)

Su función consiste en presentar en pantalla el nombre de usuario del usuario que lo ejecuta. Puede ser distinto al nombre de inicio de sesión (el resultado de `logname`)

Ejemplo:

```
$ logname
juan
$ whoami
juan
$ su
Password: *******
# logname
juan
# whoami
root
```

2.14. date (mostrar fecha)

Este comando tiene dos funciones: una es la de mostrar en pantalla la fecha del sistema (en varios formatos, como veremos a continuación), la otra es la función de configurar la hora del sistema, pero para que esta funcionalidad se cumpla, se debe ejecutar el comando desde una sesión de root.

Sintaxis:

```
date [opción...] [+FORMAT]
```

FORMAT controla el formato con que se mostrará la fecha, algunas de las opciones de este argumento son:

%a	Día de la semana abreviado.
%A	Día de la semana completo.
%b	Nombre del mes abreviado.
%B	Nombre del mes completo.
%d	Día del mes.
%y	Año con dos dígitos
%m	Número de mes.
%H	Hora, en formato 24h.
%M	Minuto.
%S	Segundos.

Existen muchísimas más opciones de formato que puedes consultar a través de man.

Ejemplo: Salida estándar del comando date (ejecución sin argumentos)

```
$ date

Sun Apr  8 21:19:34 ART 2019
```

Ejemplo: Muestra el nombre del día de la semana completo, el día del mes y el nombre del mes completo

```
$ date +"%A %d %B"

Sunday 08 April
```

2.15. du (mostrar información espacio ocupado por un archivo)

Proporciona información sobre el espacio ocupado en disco por un determinado archivo o directorio.

Sintaxis:

```
du [opciones] [archivos | directorios]
```

Opciones más importantes

-h	Imprime los tamaños de forma más legible (Ejemplo: 1K, 20M, 1G)
-b	Muestra el tamaño en bytes
-k	Muestra el tamaño en kilobytes
-m	Muestra el tamaño en megabytes
-s	No muestra el tamaño de los subdirectorios

Al ejecutar el comando du sin opciones, nos indica el espacio de disco ocupado (indicado en número de bloques) por el directorio donde nos encontramos.

Ejemplo:

```
$ du
4     ./Imágenes
…
12    ./Documentos
12363 .
```

2.16. df (mostrar información de la utilización del espacio en disco)

Provee información sobre la utilización del espacio en disco para cada uno de los diferentes sistemas de archivos.

Sintaxis:

```
df [opciones] [sistema-de-archivo]
```

Si no se provee del argumento sistema-de-archivo, df informará acerca de todos los sistemas de archivos montados y en funcionamiento.

Opciones más importantes

-h	Imprime los tamaños de forma más legible (Ejemplo: 1K, 20M, 1G)
-k	Muestra los tamaños en bloques de 1024 bytes
-m	Muestra los tamaños en bloques de Megabytes

Ejemplo:

```
$ df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/hda2             2949060   2102856    696400  75% /
/dev/hda1               23302      2593     19506  12% /boot
/dev/hda4            10144728   5506796   4637932  54% /home
/dev/hdb2             3678764   3175268    503496  86% /u
```

2.17. cat (mostrar el contenido de un archivo)

El comando cat permite mostrar el contenido de uno o más archivos en pantalla

Sintaxis:

```
cat [opción] [archivo]
```

Las principales formas de utilizar el comando son:

Muestra el contenido de archivo por pantalla

```
$ cat archivo
```

Acepta lo que se introduce por teclado y lo almacena en `archivo` (si no existe archivo lo crea). Para terminar, se emplea `ctrl+d`

```
$ cat > archivo
```

2.18. less (paginar texto en pantalla)

Su función es paginar texto en pantalla. Muchas veces ocurre que cuando se ejecuta algún comando, la salida del mismo es demasiada información como para que se pueda leer en la pantalla del monitor, entonces se puede redireccionar esta salida con `less` para que permita al usuario leer sin mayores problemas, pudiendo avanzar o retroceder en el texto con las flechas de cursor del teclado. También se utiliza para visualizar archivos de texto almacenados en disco.

Si se oprime la barra espaciadora, el less avanzará un número de líneas igual al número de líneas por pantalla que posea la terminal que se esté usando.

Pulsando la tecla ENTER se va avanzando de una en una línea.

2.19. file (determinar el tipo de un archivo)

Determina el tipo de archivo

Sintaxis:

```
file [opción] archivo
```

Se trata de un comando muy útil cuando se necesita obtener información sobre archivos en función de su tipo (script de linux, directorios, ficheros de texto, etc.)

Opción útil:

-b Omite nombres de archivo (columna izquierda de la salida)

Ejemplo:

```
$ file miScript.sh
miScript.sh:    Bourne-Again shell script
```

Ejemplo:

```
$ file -b miScript.sh
Bourne-Again shell script
```

3. Permisos de usuario

Los permisos de un archivo cualquiera (inclusive los directorios) se agrupan en 3 grupos de 3 bits cada uno, como se muestra más abajo:

```
rwx    rwx    rwx
 |      |      |
 |      |    otros
 |    grupo
usuario
```

Cada grupo posee 3 bits:

r: Lectura

w: Escritura

x: Ejecución

Con las diferentes combinaciones, se puede configurar un archivo para que pueda ser leído y modificado por su dueño, y sólo leído por el grupo y los demás. Por ejemplo, el archivo `etc/passwd`:

Al hacer `ls -l /etc/passwd` obtenemos la salida:

```
[Permisos]      [Propietario]  [Grupo] [Tamaño]     [Fecha creac]     [nombre archivo]
-rw-r--r--     1 root     root     1882    abr  5 12:44 /etc/passwd
```

Este archivo es del usuario root, y del grupo del mismo nombre, solamente se puede modificar (bit «w» de escritura) por su usuario dueño, y leer por el grupo y los demás.

¿Y para qué sirve el bit de ejecución en los directorios? Los directorios no se ejecutan, y evidentemente, el bit «x» en los **directorios** existe, como se ha aclarado anteriormente, en estos casos, dicho bit tiene un significado especial.

- El bit de ejecución (x) en los directorios permite el poder ver la información acerca de los archivos que contienen.
- El bit de lectura (r) permite listar los contenidos de un directorio.
- El bit de escritura (w) permite crear y borrar archivos dentro de un directorio.

3.1. chmod (cambiar permisos de archivos)

Para cambiar los permisos de los archivos se usa el comando `chmod`. Este comando en concreto tiene varias sintaxis permitidas.

Sintaxis:

```
chmod [-R] modo-en-octal archivo
```

La opción -R permite cambiar recursivamente los permisos de todos los archivos dentro de un directorio.

El modo en octal es un número en base 8 (octal) que especifica el permiso.

```
---   =    000 (0 en octal)
--x   =    001 (1 en octal)
-w-   =    010 (2 en octal)
-wx   =    011 (3 en octal)
r--   =    100 (4 en octal)
r-x   =    101 (5 en octal)
rw-   =    110 (6 en octal)
rwx   =    111 (7 en octal)
```

Ejemplo: Asigna todos los permisos al archivo `archivo.txt`

```
$ chmod 777 archivo.txt
```

Ejemplo: Asigna permisos de lectura y escritura (no de ejecución) a todos los archivos y directorios del directorio donde ejecutamos el comando.

```
$ chmod 666 *
```

Ejemplo: Asigna permisos al archivo `miScript.sh`. Los permisos asignados son de lectura, escritura y ejecución para el propietario, de lectura y escritura para el grupo y de lectura solo para el resto de usuarios.

```
$ chmod 764 miScript.sh
```

Otra forma de trabajar con el comando chmod es de la siguiente forma:

Sintaxis:

```
chmod [usuario] operación permiso archivo
```

donde:

usuario Indica a quien afecta el cambio de permiso. Puede tomar los siguientes valores:

u: usuario dueño del fichero

g: grupo de usuarios del dueño del fichero

o: todos los otros usuarios

a: todos los tipos de usuario (dueño, grupo y otros)

operación Puede ser:

+: activa permisos

-: desactiva permisos

=: establece esos permisos exactamente

permiso Indica el permiso que se activa o desactiva. Será una combinación de las letras: r, w, x

archivo Nombre del archivo cuyos modos de acceso se quieren cambiar

Ejemplo: Activa los permisos de ejecución al usuario y al grupo sobre el archivo `miScript.sh`:

```
$ chmod ug+x miScript.sh
```

Ejemplo: Desactiva el permiso de lectura y ejecución de todos los archivos y subdirectorios del directorio `/home/usuario/prueba` para el grupo y los otros:

```
$ chmod -R go-rx /home/usuario/prueba
```

3.2. chown (cambiar propietario de archivos)

Permite cambiar el propietario de archivos y directorios.

Sintaxis:

```
chown [opciones] usuario archivos
```

Opciones más importantes

-R	Cambia recursivamente el propietario en una jerarquía de directorios

Ejemplo: Establece al usuario `antonio` como propietario del archivo `f1.txt`

```
$ chown antonio f1.txt
```

Ejemplo: Para establecer además el grupo

```
$ chown antonio:profesores f1.txt
```

3.3. chgrp (cambiar grupo de archivos)

Permite cambiar el grupo al que pertenece un archivo.

Sintaxis:

```
chgrp [opciones] grupo archivos
```

Opciones más importantes

-R	Cambia recursivamente la propiedad grupo en una jerarquía de directorios

Ejemplo: Cambia el grupo del directorio `miCarpeta` a `profesores`

```
$ chgrp profesores miCarpeta
```

Ejemplo: Cambia el grupo del directorio `miCarpeta` y de todo su contenido a `profesores`

```
$ chgrp –R profesores miCarpeta
```

4. Comprimir y empaquetar archivos

4.1. tar (empaquetar varios archivos en uno y, opcionalmente, comprimir)

El comando tar permite empaquetar varios archivos y directorios en un mismo archivo y, opcionalmente, comprimirlo.

Sintaxis:

```
tar [opciones] [archivos]
```

Opciones más importantes

-c	Crea un nuevo archivo empaquetado. Debe indicar los archivos y directorios de entrada en la línea de comandos.
-r	Añade archivos a un archivo empaquetado existente.
-t	Enumera los archivos que contiene el archivo empaquetado
-x	Extrae los archivos del archivo empaquetado
-f file	Lee/escribe el archivo empaquetado file
-u	Añade archivos nuevos o modificados a un archivo comprimido existente.
-z	Usa compresión gzip
-j	Usa compresión bzip2
-v	Muestra información de las acciones del compresor

Ejemplo: Empaqueta (sin comprimir) el directorio `carpetaB` con el nombre `paquete.tar`

```
$ tar -cvf paquete.tar carpetaB/
```

Ejemplo: Desempaqueta el archivo `paquete.tar`

```
$ tar -xvf paquete.tar
```

Ejemplo: Empaqueta y comprime los archivos `f1.txt, f2.txt y f3.txt` con el nombre `comprimido.tar.gz`

```
$ tar -czvf comprimido.tar.gz f1.txt f2.txt f3.txt
```

Ejemplo: Descomprime el archivo comprimido.tar.gz

```
$ tar -xzvf comprimido.tar.gz
```

4.2. gzip y gunzip

`gzip` y `gunzip` comprimen y descomprimen archivos en formato Zip GNU. Los archivos comprimdos tienen el sufijo `.gz`

Sintaxis:

```
gzip [opciones] archivos
```

```
gunzip [opciones] archivos
```

Ejemplo: Comprime el archivo `miDocumento` para crear `miDocumento.gz`. Se elimina el archivo original

```
$ gzip miDocumento
```

Ejemplo: Descomprime `miDocumento.gz` para crear `miDocumento`. Se elimina el archivo `.gz` original.

```
$ gunzip miDocumento.gz
```