

## TEMA 2: GESTIÓN DE ARCHIVOS

Este tema cubre la gestión básica del sistema de archivos (archivos, directorios y links) de GNU/LINUX mediante las órdenes disponibles en el intérprete de comandos o shell.

### EL SISTEMA DE ARCHIVOS DE LINUX

Todos los sistemas operativos estructuran sus objetos en una estructura jerárquica, que consiste en un árbol de objetos que contiene información u otros objetos. Esta estructura en forma de árbol permite una organización clara de los objetos y permite contener distintos objetos en distintas localizaciones con el mismo nombre.

#### DIRECTORIOS Y FICHEROS

Un directorio es un objeto destinado a contener otros objetos, mientras que un archivo es un objeto destinado a contener información. El origen del sistema de archivos de LINUX se encuentra en el directorio `root` o `"/`

#### ESTRUCTURA BÁSICA DE DIRECTORIOS DE LINUX

A continuación se detallan los directorios más importantes que podemos encontrar en la mayoría de distribuciones LINUX.

##### **/bin /sbin**

Estos directorios contienen programas ejecutables que forman parte del sistema operativo. El directorio `/sbin` contiene las utilidades del sistema de archivos, particiones e inicio del sistema. El directorio `/bin` contiene el resto de comandos de la consola y utilidades varias.

##### **/boot**

Este directorio contiene la información necesaria para poder arrancar el sistema operativo. Entre otros archivos, aquí se encuentran los núcleos del sistema que se pueden iniciar y la configuración de gestor de arranque.

##### **/dev**

Este directorio contiene archivos de dispositivos que permiten la comunicación con los distintos elementos hardware que tengamos instalados en el sistema; discos duros (como `/dev/hda` o `/dev/sda`), particiones de discos duros (como `/dev/hda3` o `/dev/sda6`), unidades de CD-ROM (como `/dev/scd0`), disqueteras (como `/dev/fd0`), impresoras (como `/dev/lp0`), puertos serie (como `/dev/ttyS0` o `/dev/cua0`), puerto PS2 (como `/dev/psaux`), tarjetas de sonido (como `/dev/audio`), etc.

##### **/etc**

Este directorio contiene todos los archivos de configuración de nuestro sistema GNU/LINUX. Este directorio posee distintos subdirectorios que se utilizan para la configuración de los distintos elementos o servicios del sistema operativo.

##### **/lib**

Contiene las librerías que son necesarias durante el inicio del sistema operativo. La ventaja de usar librerías reside en que no es necesario integrar su código en los programas que las usan, reduciendo así el tamaño de los ejecutables. Cuando un programa necesita alguna de sus funciones, se carga la librería en memoria y puede

ser usada por cualquier otro programa que la necesite, sin necesidad de volver a cargarla en memoria.

Las librerías de GNU/LINUX son equivalentes a los archivos DLL de Windows.

### **/mnt**

Este directorio es típico de las distribuciones RedHat, y puede no estar presente en otras distribuciones. Su misión consiste en agrupar en un mismo lugar los puntos de montaje de diversos dispositivos. Este directorio contiene un subdirectorio adicional para cada una de las particiones o dispositivos disponibles en el sistema. Cuando accedemos a estos subdirectorios estamos accediendo a los dispositivos.

### **/home**

Este directorio contiene los directorios personales "home" de todos los usuarios del sistema (menos el `root`). Los usuarios convencionales únicamente pueden escribir en su directorio "home".

### **/root**

Este es el directorio personal del usuario `root` o súper usuario. Contiene básicamente la misma información que los directorios personales de los usuarios del sistema, pero orientada única y exclusivamente al usuario `root`.

### **/var**

Su nombre procede de variable, y esa es la naturaleza de la información que contienen sus subdirectorios y archivos, como colas de impresión (`/var/spool/lpd`), correo electrónico, o archivos de registro creados por los distintos procesos del sistema.

### **/usr**

Su nombre proviene de `user` y contiene una réplica de otros directorios del sistema operativo orientados a usuarios en lugar de al propio sistema operativo.

#### **/usr/X11R6**

Contiene todos los elementos que componen el entorno gráfico X Windows; binarios (`/usr/X11R6/bin`), librerías (`/usr/X11R6/lib`), manuales, etc.

#### **/usr/bin**

En este directorio se guardan los binarios o ejecutables de todas las aplicaciones orientadas al usuario.

#### **/usr/src**

Este directorio contiene el código fuente del núcleo del sistema GNU/LINUX y aplicaciones instaladas.

### **/tmp**

Este directorio contiene diversos archivos temporales que son usados por los programas del sistema operativo.

### **/proc**

Contiene los archivos de proceso. No son verdaderos archivos sino una forma de acceder a las propiedades de los distintos procesos que se están ejecutando en nuestro sistema. Para cada proceso en marcha existe un subdirectorio `/proc/<número de proceso>` con información relativa a ese proceso.

## I-NODES

Los i-nodes contienen la información de los objetos del sistema de archivos; donde están situados en el disco, la fecha de la última modificación y acceso, los permisos de usuario, etc.

## LINKS

Un link o enlace es un archivo especial que crea un atajo al archivo original situado en cualquier parte del sistema de archivos. Existen dos tipos de enlaces; los enlaces simbólicos y los enlaces hardware.

### LINKS SIMBÓLICOS

Un enlace simbólico es realmente un pequeño archivo que contiene un puntero al archivo apuntado. Cuando UNIX abre un enlace simbólico, lee el puntero y encuentra el archivo original. Los enlaces simbólicos pueden apuntar a sistemas de archivos diferentes, dispositivos diferentes o incluso a otros ordenadores conectados en red. La orden `ls -l` muestra una `l` en los links e informa a que archivo apunta el link. Cuando un archivo que posee un enlace simbólico se borra del sistema, el link no apunta a nada y recibe el nombre de enlace "stale" (la traducción al castellano sería "pasado de moda")

### LINKS HARDWARE

En un enlace hardware un mismo objeto (con un único i-node) posee dos o más referencias. Las referencias tienen diferentes nombres pero apuntan al mismo i-node, es decir al mismo objeto.

Los enlaces hardware tienen dos importantes limitaciones. La primera, como comparten i-node, el archivo y el enlace tienen que estar en el mismo sistema de archivo. La segunda, no pueden apuntar a directorios.

Los enlaces simbólicos son más fáciles de manejar y son más versátiles que los enlaces hardware. Por esta razón la mayoría de enlaces en sistemas UNIX son links simbólicos.

## COMANDOS DE GESTIÓN DE FICHEROS Y DIRECTORIOS

Una vez la jerarquía del sistema se ha definido, es necesario gestionar los objetos. Los objetos son constantemente creados, leídos, modificados, copiados, movidos y borrados. La gestión de estos objetos es una de las tareas más importantes del administrador del sistema. A continuación se presentan las órdenes básicas del intérprete de comandos que nos permiten gestionar los archivos y directorios.

### cd

**Sintaxis:** `cd directory`

La orden `cd` (change directory) cambia al directorio especificado en `directory`

### Ejemplos:

`$cd apache` cambia al directorio `apache` que se encuentra en el directorio actual.

`$cd /apache` cambia al directorio `apache` que se encuentra en el directorio raíz o `root` de la jerarquía de directorios del sistema operativo.

`$cd /apache/web` cambia al directorio `web` situado dentro de `/apache`.

`$cd ..` cambia al directorio anterior.

`$cd` (sin argumento) cambia al directorio personal del usuario actual.

`$cd ~alumno` cambia al directorio personal del usuario `alumno`.

## pwd

**Sintaxis:** `pwd`

La orden `pwd` (print working directory) muestra la ruta de acceso del directorio actual.

## cp

**Sintaxis:** `cp [options] file1 file2`  
`cp [options] files directory`

El primer comando, copia el archivo `file1` a `file2`. Si `file2` existe y el usuario tiene los permisos apropiados el archivo será remplazado. La segunda forma del comando `cp` copia uno o más archivos al directorio `directory`. Si el directorio no existe la orden mostrará un mensaje de error.

### Opciones:

- f (force) Fuerza a sobrescribir los archivos existentes en el destino.
- i (interactive) Pregunta antes de sobrescribir cualquier archivo.
- p Mantiene toda la información del archivo; propietario, grupo propietario, permisos, hora y fecha. Sin esta opción, el archivo o archivos copiados tendrán la fecha y hora actual, los permisos, propietario y grupo propietario por defecto.
- R (recursive) Si en `file1` se especifica un directorio, la opción `-r` o `-R` copia toda la jerarquía del directorio en el destino especificado.
- v (verbose) Muestra el nombre de cada archivo mientras se copia al destino especificado.

## mkdir

**Sintaxis:** `mkdir [options] directory`

Crea un directorio. El usuario tiene que poseer permisos de escritura en el directorio donde se creara el directorio.

### Opciones:

- p Crea los directorios intermedios si estos no existen.

## mv

**Sintaxis:** `mv [options] source target`

Mueve o renombra archivos y directorios. Si `target` no existe, `source` es renombrado. Si `target` existe, será sobrescrito. Si `target` es un directorio, `source` será movido dentro de ese directorio.

### Opciones:

- f Fuerza a no preguntar si el `target` existe, eliminando los mensajes de advertencia.
- i Fuerza a preguntar antes de mover cualquier archivo.

## rm

**Sintaxis:** `rm [options] files`

Elimina uno o más archivos del sistema. Para eliminar un archivo es imprescindible que el usuario tenga permiso de escritura en el directorio que contiene el archivo, pero no necesita permiso de escritura en el archivo. El comando `rm` también puede borrar directorios cuando se usan las opciones `-r` o `-R`.

**Opciones:**

- f Fuerza a no preguntar al borrar archivos sin permiso de escritura.
- i Fuerza a preguntar al borrar cada archivo.
- r Si `file` es un directorio, elimina recursivamente el contenido completo del directorio, incluidos los subdirectorios.

**rmdir**

**Sintaxis:** `rmdir [options] directory`

Borra directorios vacíos.

**Opciones**

- p Borra los directorios intermedios si estos están vacíos como resultado de la orden.

**touch**

**Sintaxis:** `touch [options] files`

Cambia la fecha del último acceso o/y modificación de `files`. Si no se especifica ninguna opción se actualizarán ambas fechas (acceso y modificación).

**Opciones:**

- a Actualiza únicamente la fecha del último acceso del archivo o archivos.
- m Actualiza únicamente la fecha de modificación del archivo o archivos.
- t No utiliza la fecha actual, sino el especificado a continuación mediante el formato de `[[CC]YY]MMDDhhmm[.ss]`.

**Ejemplos:**

Modifica la fecha del último acceso al 12 de enero de 2001 a las 18 horas, 45 minutos.

```
$touch -ta 200101121845 file
```

**ls**

**Sintaxis:** `ls [options] directory`

**Opciones:**

- A Lista todos los archivos, incluidos los ocultos (En UNIX los archivos ocultos son aquellos cuyo nombre empieza con un `."`), excepto los archivos `."` y `."`.
- l Lista los archivos en formato largo y muestra información detallada sobre ellos.
- R Lista de forma recursiva los contenidos de los subdirectorios.
- i Muestra el número de i-node de cada fichero.
- s Muestra el tamaño en KiloBytes junto a cada archivo.
- u Clasifica por fecha y hora del último acceso.
- t Clasifica por fecha y hora de la última modificación.

## ln

**Sintaxis:** `ln [options] file link`  
`ln [options] files directory`

Crea enlaces entre archivos. En la primera forma se crea un enlace llamado `link` que apunta al archivo `file`. En la segunda forma, se crea un enlace dentro del directorio `directory` para cada uno de los archivos especificados en `files`.

### Opciones:

- f Fuerza a sobrescribir los enlaces si existen previamente.
- i Pregunta antes de crear cada enlace.
- s Crea un enlace simbólico. Por defecto crea enlaces hardware.

## cat

**Sintaxis:** `cat [options] file`

Muestra el contenido del archivo `file`.

### Opciones:

- b Numera todas las líneas de salida que no están en blanco.
- n Numera todas las líneas de salida.
- s Reemplaza por una línea en blanco varias líneas en blanco adyacentes.

## METACARACTERES Y SELECCIÓN DE ARCHIVOS

Al manipular archivos con el intérprete de comandos, a menudo es necesario realizar una determinada operación con muchos archivos. Por ejemplo, en el desarrollo de un programa en C, es necesario usar la orden `touch` para forzar la compilación de todos los archivos.

Para realizar este tipo de operaciones de una manera rápida y simple, el intérprete de comandos dispone de varios metacaracteres. En vez de especificar el nombre de cada archivo, los metacaracteres substituyen parte del nombre del archivo. Los metacaracteres disponibles en GNU/LINUX se resumen en la siguiente tabla.

metacarácter	descripción
*	Sustituye cualquier número de caracteres, incluido cero caracteres. Por ejemplo, <code>x*</code> se corresponde con los archivos o directorios <code>x</code> , <code>xy</code> , <code>xyz</code> , <code>x.txt</code> , <code>xy.txt</code> ,...
?	Sustituye únicamente un carácter. Por ejemplo, <code>x?</code> se corresponde con <code>xx</code> , <code>xy</code> , <code>xz</code> , pero no con <code>x</code> o <code>xyz</code> .
[caracteres]	Sustituye un único carácter que este listado entre los corchetes. Por ejemplo, <code>x[yz]</code> se corresponde con <code>xy</code> y <code>xz</code> .
[!caracteres]	Sustituye un único carácter que <b>NO</b> este listado entre los corchetes. Por ejemplo, <code>x[!yz]</code> se corresponde con todos los archivos o directorios cuyo segundo carácter no es ni <code>y</code> ni <code>z</code> .
[a-z]	Sustituye un único carácter que este dentro del rango especificado entre los corchetes. Por ejemplo, <code>x[0-3]</code> se corresponde con <code>x0</code> , <code>x1</code> , <code>x2</code> y <code>x3</code> , pero no se corresponde con <code>xx</code> o <code>x4</code> .
[!a-z]	Sustituye un único carácter que <b>NO</b> este dentro del rango especificado entre los corchetes.