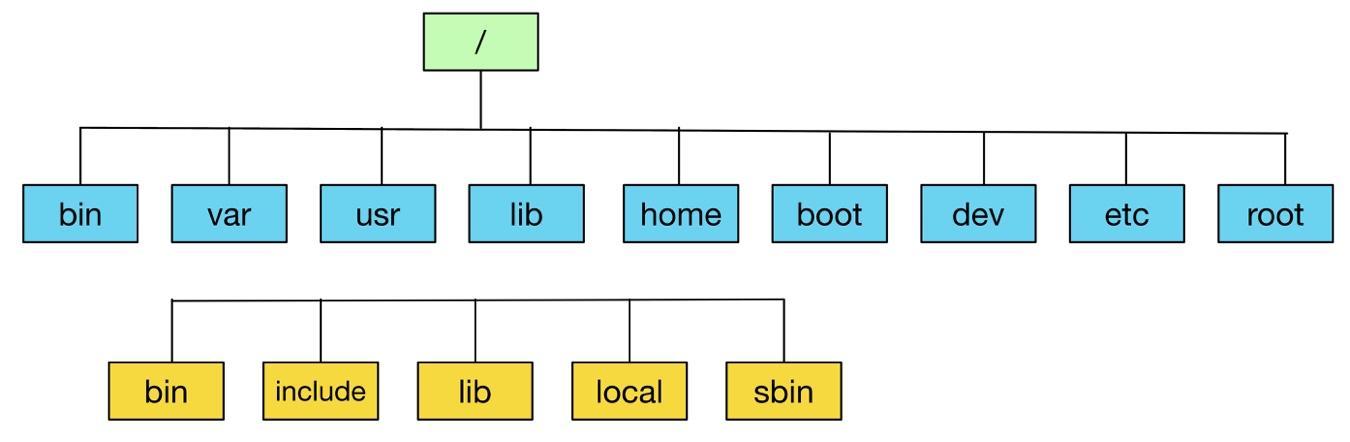
**ESTRUCTURA DE FICHEROS EN GNU / LINUX**

Según IONOS, en los sistemas **GNU/Linux**, la raíz del árbol de directorios es**/**, a partir de la cual se estructuran todas las carpetas y archivos del sistema. A continuación, se verán las carpetas más importantes y por último un pequeño esquema que resume la mayoría de las carpetas.

Fuente: https://www.uv.mx/personal/rcarrera/files/2015/12/Estructura-Linux.jpg

## ESTRUCTURA GENERAL

|  |  |
| --- | --- |
| **/** | La raíz del sistema de archivos. |
| **/boot/** | Ubicación que contiene los archivos de arranque del sistema. |
| **/etc/** | Carpeta que contiene la configuración central y específica del sistema, directorio donde se configuran servicios y aplicaciones para todos los usuarios del equipo. |
| **/home/** | Carpetas usuarios (distintos de root) que hay en el equipo. Aquí se encontrarán las *Carpetas Personales*, los archivos de los usuarios, sus datos, las configuraciones específicas de cada usuario para los programas, etc. |
| **/root/** | La carpeta personal del usuario root. |
| **/srv/** | Carpeta que contiene información de los servicios que la máquina *ofrece* a otras máquinas, ya sea mediante servicios como el del apache, como ficheros a través del NFS. |
| **/tmp/** | Carpeta temporal, se borra en cada arranque. |

## DATOS DE EJECUCIÓN

|  |  |
| --- | --- |
| **/run/** | Se encuentran datos volátiles de diferentes procesos, que se borran en cada arranque. |
| **/usr/** | Carpeta que contiene todo lo que el distribuidor (*vendor*) suministra ya sea mediante paquetería o en la instalación. |
| **/usr/bin/** | Binarios ejecutables que deben estar en el $PATH. |
| **/usr/lib/** | Librerías y utilidades necesarias para otros programas, pero que no deben ser lanzados por el usuario *directamente.* |
| **/usr/share/** | Recursos, no dependientes de arquitectura, para los programas, tales como: imágenes, documentación y otros. |

## DATOS DE EJECUCIÓN PERSISTENTES

|  |  |
| --- | --- |
| **/var/** | Datos de ejecución, pero persistentes, bases de datos, actualizaciones, etc. |
| **/var/cache/** | Caché persistente que pertenece al sistema. |
| **/var/lib/** | Datos persistentes del Sistema. |
| **/var/log/** | Archivos de registro del sistema, permanente. |
| **/var/spool/** | Datos persistentes de colas, tales como las de impresión o correo. |

## FICHEROS VIRTUALES DEL KERNEL (API)

|  |  |
| --- | --- |
| **/dev/** | Se trata de la carpeta que contiene todos los dispositivos hardware. |
| **/proc/** | Se trata de una carpeta que ofrece el kernel para poder interactuar con los procesos y proveer funcionalidad. Contiene la información acerca de los diferentes procesos que el SO administra. |
| **/sys/** | Se puede obtener información acerca del núcleo. |

## ENLACES DE COMPATIBILIDAD

|  |  |
| --- | --- |
| **/bin/, /sbin/, usr/sbin/** | Se trata de enlaces que apuntan a /usr/bin/ para ofrecer compatibilidad con aplicaciones antiguas. |
| **/lib/** | Lo mismo que el caso anterior pero sobre librería, que apuntan a: /usr/lib |
| **/var/run/** | Otro de los ajustes de compatibilidad, en este caso a /run/. |

## CARPETA /HOME/

Todas las carpetas vistas anteriormente, hacen referencia a elementos globales del sistema, sin embargo, cada usuario puede personalizar las aplicaciones y configurar su comportamiento para adaptarlo a sus necesidades. Esas configuraciones y datos personales se encuentran en una serie de directorios que serán repasados aquí.

|  |  |
| --- | --- |
| **~/.cache/** | Datos persistentes del caché del usuario. |
| **~/.config/** | Configuración y estado de las aplicaciones del usuario. |
| **~/.local/bin/** | Directorio propio del usuario que contiene ejecutables que pueden ser lanzados por el usuario. |
| **~/.local/share/** | Al igual que /usr/share/, contiene recursos como imágenes y documentos que están compartidos entre varias aplicaciones. |

## COMANDOS BÁSICOS DE LINUX

En esta categoría se incluyen todas las órdenes fundamentales que se utilizan para **administrar el terminal**, con las que se puede limpiar la ventana de la consola, recuperar comandos anteriores desde la historia o finalizar la sesión.

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **clear** | **Limpia la ventana del terminal**  Con el comando *clear* se borran todos los comandos de la sesión.  *clear*  El usuario recibe un terminal vacío con el prompt a la espera de órdenes. Los comandos introducidos anteriormente se guardan en el scrollback buffer.  También se puede limpiar la ventana de la consola con el atajo de teclado [CTRL] + [L] |
| **exit** | **Finaliza la sesión en el terminal**  Con el programa*exit* finaliza la sesión actual y se cierra la consola.  *exit*  También se puede utilizar el atajo de teclado [CTRL] + [D]. |
| **help** | **Muestra la lista de todos los comandos**  El comando *help* se utiliza para mostrar una lista de todas las órdenes del sistema.  *help*  Si se introduce *help* en combinación con un comando shell, se obtiene una breve descripción del comando en cuestión.  *help COMANDO* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **history** | **Muestra las entradas del historial**  En Bash, los últimos 500 comandos introducidos en el terminal se guardan en el historial. 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.  También se puede examinar el historial en función de palabras clave con el atajo de teclado [CTRL] + [R]. Utilizando la orden sin opciones ni argumentos se obtiene la lista completa de comandos numerada.  *history*  Para filtrar la salida se combina *history* con el programa *grep* (ver opciones de búsqueda) y una palabra clave utilizando la barra vertical.  *history | grep PALABRA CLAVE* |

## OPERACIONES EN EL DIRECTORIO

Algunos comandos básicos de Linux permiten llevar a cabo operaciones en los directorios del sistema, como puede ser crear ficheros, borrarlos y gestionarlos, así como navegar por el árbol del directorio. Entre los comandos más importantes de esta categoría se incluyen **cd, ls, mkdir o rmdir.**

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **cd** | **Navega por el árbol de ficheros**  El comando *cd* es la abreviatura de *change directory* y se utiliza para navegar por el directorio.  La sintaxis de esta orden sigue el esquema:  *cd [OPCIÓN] DIRECTORIO*  Si no se indica ningún directorio concreto, *cd* cambia automáticamente al directorio principal del usuario.  Si se acompaña de un guión (-), *cd* abre el directorio precedente. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **chroot** | **Ejecuta el programa en un nuevo directorio raíz**  La orden*chroot* (abreviatura de *change root*) se utiliza para ejecutar un comando en un directorio raíz diferente para, por ejemplo, aislar programas peligrosos del sistema de archivos restante, lo que se denomina *chroot jail*.  Este comando requiere derechos raíz y se orienta por este esquema:  *chroot DIRECTORIO COMANDO* |
| **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).  Esta es la sintaxis de la orden:  *ls [OPCIONES] DIRECTORIO*  Si no se añade a *ls* ningún directorio, el comando enumera el contenido del directorio en curso.  Con ayuda de diferentes opciones se puede definir qué información se ha de mostrar y cómo. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **mkdir** | **Crea un directorio**  El comando *mkdir* corresponde a *make directory*y permite a los usuarios de Linux crear directorios desde cero.  Para crear un directorio en el fichero en curso escribe la siguiente sintaxis:  *mkdir [OPCIÓN] DIRECTORIO*  Si lo que se necesita es crear varios ficheros a la vez, se escribe uno detrás de otro sin signos de puntuación y con espacio intermedio:  *mkdir [OPCIÓN] DIRECTORIO1 DIRECTORIO2*  Si se quiere crear un directorio nuevo en otro fichero diferente al actual, se ha de indicar la ruta absoluta o relativa al fichero:  mkdir /home/user/Desktop/test  *mkdir/Desktop/test*  En ambos ejemplos se crea el directorio *test* en el directorio *Desktop*. |
| **mkdirhier** | **Crea una jerarquía en el directorio**  Con *mkdirhier* se pueden crear jerarquías completas con un único comando:  *mkdirhier [OPCIÓN] /home/user/directorio1/directorio2/directorio3*  Si *directorio1* y *directorio2* ya existieran, entonces *mkdirhier*sólo crearía *directorio3*. En caso contrario, se crean los tres. |
| **pwd** | **Muestra el nombre del directorio**  Con *pwd* (abreviatura de *print working directory*) la consola muestra el nombre del directorio de trabajo (en curso).  La sintaxis del comando es:  *pwd [OPCIONES]* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **rmdir** | **Borra el directorio**  Si hiciera falta borrar un determinado directorio, se utiliza el comando *rmdir* (*remove directory*) según la siguiente sintaxis:  *rmdir [OPCIÓN] DIRECTORIO*  Con *rmdir,*sin embargo, solo se pueden borrar directorios vacíos. Si se quiere eliminar un fichero con todas sus carpetas y subcarpetas, se utiliza el comando*rm* (*remove*) con la opción *-r*.  **Cuidado:** *rmdir*no solicita confirmar el borrado. Los directorios seleccionados con el comando se borran definitivamente. |
| **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:  *tree [OPCIONES] [DIRECTORIO]* |

## OPERACIONES EN ARCHIVO

Los comandos Linux de esta categoría permiten llevar a cabo diversas operaciones desde el terminal que atañen a los archivos, y utilizando comandos básicos como ***cp*, *mv* y *rm****,* se pueden copiar, desplazar, renombrar o borrar archivos del sistema.

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **basename** | **Muestra el nombre del archivo**  Al comando *basename*se le indica una ruta a un archivo y devuelve su nombre sin la ruta.  La sintaxis del comando se compone de:  *basename [OPCIONES] Ruta/al/archivo [SUFIJO]*  Si en el terminal escribes, por ejemplo,*$ basename /home/user/imagen.jpg*, obtienes la siguiente respuesta:  *imagen.jpg*  Si se añade el sufijo de nuevo, el terminal no lo devuelve, como vemos aquí:  Entrada:*$ basename /home/user/imagen.jpg .jpg*  Salida: *imagen*  El comando puede ser ampliado a varios archivos con diferentes opciones. |
| **cat** | **Agrupa contenido de varios archivos**  El comando *cat* (de *concatenate*) nace como herramienta para enlazar archivos y puede emplearse como pager para mostrar el contenido de los archivos en el terminal.  Escribe *cat* con la siguiente sintaxis para leer y mostrar un archivo en la salida estándar:  *cat OPCIONES ARCHIVOS*  Si incluyes varios archivos, sepáralos mediantes espacios:  *cat OPCIONES ARCHIVO1 ARCHIVO2*  Para concatenar el contenido de varios archivos se utilizan los operadores de redirección >, < y |. Si utilizas el operador “mayor que” (>), se aúna el contenido de dos archivos en un tercero:  *cat archivo\_1.txt archivo\_2.txt > archivo\_3.txt* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **cmp** | **Compara archivos a nivel de byte**  *cmp*forma parte del paquete *diff,* y como este, también se utiliza para comparar el contenido de varios archivos, aunque a diferencia de como sucede en *diff,*aquí el cotejo tiene lugar byte por byte, lo que lo hace especialmente idóneo para archivos binarios*.*  Invoca a*cmp* según la siguiente sintaxis:  *cmp [OPCIONES] ARCHIVO1 ARCHIVO2*  Si *cmp*encuentra diferencias, entrega el número de bytes y el de la línea del primer archivo que difiera. |
| **comm** | **Compara archivos clasificados por líneas**  Utiliza el comando *comm* para comparar por líneas archivos previamente ordenados (con *sort*, p. ej.).  La invocación de este comando se fundamenta en la siguiente estructura sintáctica:  *comm [OPCIONES] ARCHIVO1 ARCHIVO2*  Sin opciones adicionales, el programa genera una salida en tres columnas: la primera contiene todas las líneas que solo aparecen en ARCHIVO1; la segunda, todas las líneas que solo aparecen en *ARCHIVO2,*y la tercera, todas las líneas que aparecen en ambos archivos.  El comando *comm* soporta tres opciones:      -1 = ocultar líneas únicas del *ARCHIVO1*      -2 = ocultar líneas únicas del *ARCHIVO2*      -3 = ocultar todas las líneas que aparecen en ambos archivos |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **cp** | **Copia archivos o directorios**  La orden*cp* (de *copy*) se utiliza para copiar archivos o ficheros y sigue la siguiente sintaxis:  cp *[OPCIONES] ORIGEN DESTINO*  El elemento ORIGEN es el que se ha de copiar y DESTINO se define a un archivo o un directorio donde se ha de alojar el elemento copiado. Si se define como destino a un archivo que ya existe, el archivo origen reescribe su contenido. También se puede crear un archivo de destino nuevo. Si se han de copiar varios archivos, entonces el destino ha de ser un directorio, del mismo modo que si se copia un directorio.  Copiar un archivo de origen en un archivo de destino en el directorio actual:  *cp [OPCIONES] ORIGEN DESTINO*  Ejemplo: *cp archivo.txt archivo\_copia.txt*  Copiar un archivo del directorio actual en un directorio de destino:  *cp [OPCIONES] ARCHIVO\_ORIGEN DIRECTORIO\_DESTINO*  Ejemplo: *cp archivo.txt home/user/documentos/2017*  Copiar varios archivos en un directorio:  *cp [OPCIONES] ARCHIVO\_ORIGEN1 ARCHIVO\_ORIGEN2 DIRECTORIO\_DESTINO*  Ejemplo: *cp archivo.txt archivo.odt home/user/documentos/2017*  Copiar un directorio desde el directorio actual en otro directorio diferente:  *cp  DIRECTORIO\_ORIGEN DIRECTORIO\_DESTINO*  Ejemplo: *cp directorio1 home/user/documentos/2017*  Si se tienen que copiar todos los directorios con el contenido completo se deben incluir en el copiado todos los subdirectorios con la opción -R. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **cut** | **Extrae el contenido de los archivos**  La orden*cut* permite extraer el contenido de las líneas de texto de un archivo (log o CSV) por columnas.  La sintaxis de este comando es:  *cut [OPCIONES] ARCHIVO*  Para indicar la posición exacta del fragmento que se tiene que extraer se utilizan las opciones *-b*(posición de byte),*-c*(posición del carácter), *-d*(carácter de separación) y *-f*(campo). |
| **diff** | **Compara archivos o directorios**  Con el programa *diff* se cotejan dos archivos. También permite averiguar si dos directorios contienen los mismos archivos. Se le invoca con esta estructura:  *diff [OPCIONES] ARCHIVO1 ARCHIVO2* |
| **dirname** | **Muestra la ruta al archivo**  *dirname* es lo contrario que *basename*: el comando permite extraer la parte de la ruta a un fichero sin mostrar su nombre.  Su sintaxis es:  *dirname [OPCIONES]*  Introduce *$ dirname /home/user/imagen.jpg*en el terminal para obtener la siguiente salida:  */home/user* |
| **file** | **Muestra el tipo de archivo**  Con *file* se puede obtener información sobre el tipo o formato de un archivo.  Sigue el siguiente esquema:  *file [OPCIONES] ARCHIVO* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **ln** | **Crea un enlace a un archivo o un directorio**  El programa *ln* (*link*) crea un vínculo (un archivo especial) que lleva a un archivo o un directorio. Con ello se genera otra entrada a este archivo que permite acceder a él a través de una ruta diferente. Para invocar *ln* el comando ha de incluir, como mínimo, la ruta al archivo de origen.  *ln [OPCIONES] ruta/al/archivo de origen*  En este caso, el enlace se crea en el directorio actual de trabajo sin cambiar de nombre. Otra alternativa consiste en indicar una ruta de destino y dar un nombre diferente al vínculo:  *ln [OPCIONES] ruta/al/archivo de origen ruta/al/enlace*  *ln* crea por defecto los llamados enlaces duros o hardlinks, que no son adecuados para crear vínculos a los directorios y tampoco se pueden utilizar más allá de las fronteras de partición. Es por esto que el comando suele usarse en combinación con la opción -s (*--symbolic*), con la cual se pueden generar enlaces simbólicos, incluso más allá de los límites del sistema de archivos. Los enlaces simbólicos se refieren siempre a la ruta “real” a un archivo y dependen de esta. |
| **lsof** | **Muestra los archivos abiertos en el terminal**  *lsof* equivale a *list open files* (muestra los archivos abiertos), un programa de apoyo que muestra en el terminal información sobre los archivos abiertos ordenados en función del PID (ID de proceso).  Para invocarlo se sigue el siguiente esquema:  *lsof [OPCIONES]*  Dado que todos los sistemas tipo UNIX como Linux lo gestionan todo en archivos *(«Everything is a file»*), la lista que devuelve *lsof* es muy larga. Por eso suelen usarse opciones que concretan el resultado. |
| **md5sum** | **Calcula sumas de control**  Con ayuda del comando *md5sum*se pueden calcular y comprobar sumas de control MD5*.* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **mv** | **Mueve archivos o directorios**  El programa *mv* (*move*) copia un archivo o un directorio y borra el original. Si esto tiene lugar dentro del mismo directorio, *mv* puede utilizarse también para cambiar el nombre de los archivos.  Para ello se sigue el siguiente esquema:  *mv [OPCIÓN] ORIGEN DESTINO*  Ejemplos de su aplicación:  Trasladar un archivo a otro directorio:  *mv [OPCIONES] ARCHIVO\_ORIGEN DIRECTORIO\_DESTINO*  Ejemplo: *mv archivo1.txt home/user/documentos/2017*  Trasladar varios archivos de origen a un directorio de destino:  *mv [OPCIONES] ARCHIVO\_ORIGEN1 ARCHIVO\_ORIGEN2 DIRECTORIO\_DESTINO*  Ejemplo: *mv archivo1.txt archivo2.txt home/user/documentos/2017*  Trasladar un subdirectorio del fichero actual a un directorio de destino:  *mv [OPCIONES] DIRECTORIO\_ANTIGUO DIRECTORIO\_NUEVO*  Ejemplo: *mv directorio1 home/user/documentos/2017*  Cambiar el nombre de un archivo en el directorio de trabajo:  mv *[OPCIONES] ARCHIVO\_ANTIGUO ARCHIVO\_NUEVO*  Ejemplo: *mv archivo1.txt archivo2.txt*  Cambiar el nombre de un subdirectorio en el directorio actual:  *mv [OPCIONES] DIRECTORIO\_ANTIGUO DIRECTORIO\_NUEVO*  Ejemplo: *mv directorio1 directorio2* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **paste** | **Agrupa el contenido de archivos en columnas**  De forma parecida a *cat*, el programa *paste* también muestra el contenido de los archivos en la salida estándar, pero mientras que *cat* los muestra de forma encadenada, *paste* los agrupa en columnas.  El esquema básico de esta función es:  *paste [OPCIONES] ARCHIVO1 ARCHIVO2*  En el modo estándar, los archivos indicados se agrupan de tal forma que todas las líneas cuyo número coincide se emiten en una misma línea en la salida. Así, cada línea de la salida contiene el contenido de todos los archivos de entrada.  Con la opción *-d* se puede configurar individualmente el tipo de separador que se usa. Por defecto suele utilizarse la tabulación.  Con la opción *-s* (de serial) se activa otro modo diferente en el cual todas las líneas del primer archivo de entrada se pegan en la primera línea de la salida. Los datos del resto de archivos de entrada se pegan a continuación en líneas de salida separadas. De esta forma, cada línea de la salida contiene únicamente el contenido de un archivo de entrada. |
| **rename** | **Cambia el nombre de archivos**  *rename*permite cambiar el nombre de archivos y carpetas con ayuda de expresiones regulares compatibles con perl (Regex). Al contrario de *mv*, *rename* se presta así para modificar los nombres de varios archivos parcial o totalmente.  La invocación de *rename* se realiza siguiendo el siguiente esquema:  *rename [OPCIONES] 'EXPRESIÓN\_REGULAR' ARCHIVOS*  En el caso de las sustituciones, las expresiones regulares siguen la siguiente sintaxis:  *s/PATRONDEBUSQUEDA/SUSTITUCIÓN/MODIFICADOR*  En el siguiente ejemplo todas las terminaciones *.html* se sustituyen por *.xhtml*:  *rename 's/\.html$/.xhtml/' \*.html* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **rm** | **Borra archivo o directorio**  El programa *rm* (*remove*) borra archivos o directorios de forma irreversible. Para ello se sigue la siguiente estructura:  *rm [OPCIONES] ARCHIVO*  o  *rm [OPCIONES] DIRECTORIO*  Para eliminar un directorio junto a todos sus subdirectorios se utiliza *rm* con la opción *-R* (*--recursive*):  *rm -R DIRECTORIO*  Cuando se ordena eliminar varios archivos o ficheros, estos se separan por espacios:  *rm [OPCIONES] ARCHIVO1 ARCHIVO2* |
| **shred** | **“Tritura” archivos**  *shred*es un comando que permite borrar archivos de forma permanente porque los reescribe de modo que, incluso contando con los mejores medios forenses, no se pueden restablecer.  Este comando se invoca siguiendo este esquema básico:  *shred [OPCIONES] ARCHIVO*  Utilizándolo con las siguientes opciones se eliminan archivos concretos:  *shred -fuz ARCHIVO*  La opción *-f* impulsa el proceso de eliminación, *-z* reescribe el contenido del archivo con ceros (la configuración estándar son datos aleatorios) y *-u* elimina el archivo triturado del sistema de forma similar al comando *rm*. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **sort** | **Ordena listas de archivos y salidas de programa**  Con el comando *sort* se ordenan las listas de archivos y las salidas de programa en líneas numérica y alfabéticamente.  Se invoca a *sort* siguiendo este esquema:  *sort [OPCIONES] ARCHIVO*  Este método se puede complementar con opciones como *-n* (ordenar numéricamente), *-R*(ordenar de forma aleatoria) o *-r* (ordenar en orden inverso). |
| **split** | **Divide archivos**  El comando *split* se utiliza para subdividir archivos. Sigue el siguiente esquema sintáctico:  *split [OPCIONES] [INPUT [PREFIJO]]*  INPUT equivale al archivo que se ha de subdividir y PREFIJO hace las veces de prefijo del nombre del nuevo subarchivo. A la hora de cambiar su nombre se sigue el siguiente esquema:  *PREFIJOaa, PREFIJOab, PREFIJOac …*  Si no se define ningún prefijo, *split* recurre al estándar *x*.  Con la opción *-b* (bytes) se puede definir el tamaño de las partes en que se divide el archivo. Esto puede hacerse en bytes (b), kilobytes (k) o megabytes (m).  Ejemplo:  *split -b 95m archivo.tgz split-archivo.tgz.*  Aquí*split* divide el archivo *archivo.tgz*en archivos de 95 megabytes y los nombra como sigue:  *split-archivo.tar.aa*  *split-archivo.tar.ab*  *split-archivo.tar.ac*  Los archivos que se han subdividido utilizando *split* pueden reunirse de nuevo con el comando *cat*.  *cat split-archivo.tar.\* > archivo.tar* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **stat** | **Muestra fecha y hora**  Con ayuda del comando *stat* (*status*) se puede obtener la fecha y la hora de los accesos y los cambios relativos a determinados archivos o directorios.  Este comando sigue esta sintaxis:  *stat [OPCIONES] ARCHIVO*  El formato de salida se puede configurar con opciones. |
| **touch** | **Cambia fecha y hora**  Para modificar la fecha y la hora de los accesos o los cambios que han tenido lugar en un archivo se utiliza el comando *touch.*Si se utiliza en un archivo que no existe, este se crea automáticamente, lo que hace que este comando también se utilice para generar archivos vacíos.  Para invocar a *touch*se tiene que seguir este esquema:  *touch [OPCIONES] ARCHIVO*  Si de lo que se trata es de cambiar la fecha y la hora de un archivo por una fecha determinada se utiliza la opción *-t* incluyendo la fecha en el formato *[AA]MMDDhhmm[.ss].*  Ejemplo*:*  *touch -t 1703231037 archivo.txt*  La fecha y la hora del acceso y los cambios respecto a este archivo se han fijado en el 23 de marzo de 2017 a las 10:37 horas. Si se prefiere, el cambio se puede limitar a la fecha de acceso o de modificación con las opciones *-a* y *-m*. Si se utiliza sin la opción *-t* se sella automáticamente con la fecha y la hora actual. |
| **uniq** | **Elimina duplicados en listas de archivos y salidas de programa**  El comando*uniq* suele usarse junto con *sort* para limpiar líneas dobles en los archivos ordenados.  En el siguiente ejemplo se enlaza el comando *sort* con el comando *uniq* con la barra vertical para seleccionar en primer lugar un archivo y a continuación para emitirlo sin líneas duplicadas:  *sort archivo.txt | uniq* |

## GESTIÓN DE LOS PERMISOS

Con Linux se pueden definir los **derechos de acceso y de posesión de archivos y ficheros** por parte de los usuarios fácilmente. Los comandos más importantes en cuanto a la gestión de permisos son chown y chmod. Los grupos se administran con la orden chgrp.

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **chattr** | **Gestiona los atributos de los archivos**  Con el programa *chattr* (abreviatura de *change attribute*) puedes asignar atributos a archivos o directorios. Diversos sistemas de archivos nativos de Linux (ext2, ext3, ext4, XFS, ReiserFS, JFS y OCFS2) soportan la configuración de atributos.  Utiliza *chattr* según la siguiente sintaxis para asignar un atributo:  *chattr [OPCIONES] +ATRIBUTO ARCHIVO*  Con el siguiente esquema se pueden eliminar los atributos asignados:  *chattr [OPCIONES] -ATRIBUTO ARCHIVO*  Añadiendo el atributo *-i* (de inmutable) se blinda un archivo o un directorio frente a cambios:  *chattr +i archivo.txt*  En la página del manual dedicada a *chattr* encuentras otros atributos y posibles opciones. |
| **chgrp** | **Administra los grupos de archivos y directorios**  El comando *chgrp* significa *change group* y se usa para cambiar el grupo de un archivo o un directorio, aunque se necesita estar en posesión de permisos de administrador para poder aplicarlo a un archivo o un fichero determinado. Además, solo se puede trabajar con los grupos a los cuales se pertenece.  *chgrp* se utiliza siguiendo la siguiente estructura:  *chgrp [OPCIONES] GRUPO ARCHIVO*  o  *chgrp [OPCIONES] GRUPO DIRECTORIO*  La opción *-R* incluye en la operación a las subcarpetas y los archivos contenidos en un directorio. |
| **COMANDO** | **DESCRIPCIÓN** |
| **chmod** | **Gestiona los permisos de acceso**  La llamada al sistema *chmod* (*change mode*) sirve para administrar los permisos en sistemas de archivos unixoides (ext2, ext3, ext4, reiser, xfs).  La sintaxis general de *chmod* es:  *chmod [OPCIONES] MODO ARCHIVO*  o  *chmod [OPCIONES] MODO DIRECTORIO*  *MODO* corresponde a la forma de asignar los permisos que se aplica. Para saber cómo se crean y qué hay que tener en cuenta, lee nuestro artículo sobre el [reparto de permisos de acceso](https://www.ionos.es/digitalguide/servidores/know-how/asignacion-de-permisos-de-acceso-con-chmod/) con *chmod*.  Con ayuda de la opción *-R* se pueden ampliar los derechos de forma recursiva a las subcarpetas y a los archivos contenidos en un fichero. |
| **chown** | **Administra derechos de propietario**  Normalmente, el creador de un archivo o un directorio se convierte automáticamente en su propietario (*owner*). La orden *chown* se deriva de *change owner* y permite configurar la propiedad de archivos y directorios.  Este comando se utiliza basándose en la siguiente composición:  *chown [OPCIONES] [USUARIO][:[GRUPO]] ARCHIVO*  o  *chown [OPCIONES] [USUARIO][:[ GRUPO]] DIRECTORIO*  Con objeto de definir derechos de propietario para usuarios o grupos se dispone de cuatro combinaciones posibles:  1) El propietario y el grupo de un fichero se definen según las indicaciones:  *chown [OPCIONES] nombre\_propietario:grupos\_archivos.txt*  2)El grupo se modifica en función de las indicaciones pero el usuario permanece invariable:  *chown [OPCIONES] :grupos\_archivos.txt* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **chown** | 3) Se define al propietario pero el grupo permanece invariable:  *chown [OPCIONES] propietario\_archivos.txt*  4) Se vuelve a definir al propietario y el grupo se define por el grupo estándar del usuario activo:  *chown [OPCIONES] nombre\_propietario: archivo.txt*  Los cambios se pueden extender a los subdirectorios con ayuda de la opción *-R*. |
| **lsattr** | **Muestra los atributos del archivo**  Si quieres ver qué atributos se definieron para un archivo o un directorio, puedes usar el comando *lsattr*(abreviatura de *list attributes*) según el esquema que sigue:  *lsattr [OPCIONES] ARCHIVO/DIRECTORIO* |

## ARCHIVAR Y COMPRIMIR

Linux facilita diversas tecnologías con el fin de comprimir y empaquetar archivos, aunque cabe tener en cuenta que no siempre un proceso de **archivado** implica una **compresión**. Por ejemplo, el programa para archivar ficheros *tar* suele utilizarse en combinación con programas de compresión como *gzip*, *bzip2* o *xz*.

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **tar** | **Escribe y extrae archivos en el fichero Tar**  El comando *tar* corresponde a *tape archiver*, un programa desarrollado en sus orígenes para grabar datos en unidades de cinta magnética y que hoy es uno de los programas más populares para archivar datos en Linux.  El programa permite escribir ficheros y directorios de forma secuencial en un archivo *tar* y recuperarlos a partir de él. A diferencia de lo que ocurre con el formato Zip propio de sistemas Windows, con *tar* los derechos de usuario de los ficheros guardados no se pierdenmdespués de descomprimirlos.  El programa de líneas de comandos *tar* se invoca según la siguiente sintaxis: |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **tar** | *tar [OPCIONES] ARCHIVOS*  Si quieres crear un archivo nuevo, utiliza *tar* con las opciones *-c* (crear nuevo archivo) y *-f* (escribir un archivo en el fichero indicado o extraerlo de él).  En el siguiente ejemplo los archivos *archivo1.txt* y *archivo2.txt* se escriben en el recién creado fichero *ejemplo.tar*:  *tar -cf ejemplo.tar archivo1.txt archivo2.txt*  Si lo que quieres es que el terminal te muestre el contenido de un archivo utiliza, *tar* con las opciones *-t* (mostrar contenido de un archivo), *-v* (salida detallada) y *-f* (ver arriba):  *tar -tvf ejemplo.tar*  En caso de que los archivos empaquetados se tengan que desempaquetar en la carpeta actual, *tar* se acompaña entonces de las opciones *-x* (extraer ficheros de un archivo) y *-f* (ver arriba).  *tar -xf ejemplo.tar*  Con *-j* (bzip2), *-J* (xz), *-z* (gzip) y *-Z* (compress), *tar* también ofrece opciones que permiten comprimir o descomprimir archivos durante el proceso de empaquetado o extracción invocando a otro programa.  En el ejemplo que sigue los archivos *archivo1.txt* y *archivo2.txt* se archivan en *ejemplo.tar.gz* y se comprimen con gzip.  *tar -czf ejemplo.tar.gz archivo\_1.txt archivo\_2.txt*  El siguiente comando extrae y descomprime todos los archivos guardados en *ejemplo.tar.gz*:  *tar -xzf ejemplo.tar.gz* |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **gzip / gunzip** | **Comprime o descomprime archivos con gzip**  En el caso de *gzip* (abreviatura de *GNU zip*), estamos ante un programa con el cual se pueden comprimir y descomprimir archivos cómodamente en el terminal.  Su sintaxis general sigue esta estructura:  *gzip [OPCIONES] ARCHIVOS*  Podrías utilizar *gzip* según este esquema para convertir el archivo *ejemplo.txt* al formato comprimido *ejemplo.txt.gz*:  *gzip ejemplo.txt*  Debes tener en cuenta que *gzip* borra por defecto el archivo original cuando lo empaqueta, algo que, por otro lado, puedes evitar con ayuda de la opción *-k*.  *gzip -k archivo.txt*  El programa también puede ser empleado sobre varios archivos al mismo tiempo: cada archivo inicial se convierte a un archivo *gz* independiente entonces.  El comando  *gzip ejemplo\_1.txt ejemplo\_2.txt ejemplo\_3.txt*  crea los archivos *ejemplo\_1.txt.gz*, *ejemplo\_2.txt.gz* y *ejemplo\_3.txt.gz*.  Para escribir varios archivos en un fichero comprimido común se utiliza *gzip* junto con el programa de archivado *tar*.  Si se trata de descomprimir un archivo *gz,* se combina el comando *gzip* con la opción *-d* o se utiliza el comando *gunzip*.  *gzip -d ejemplo.txt.gz*  *gunzip ejemplo.txt.gz*  Los archivos *gz* desempaquetados también se borran por defecto, pero es posible conservar el archivo *gz*, además del extraído, con la opción *-k*.  La compresión gzip se apoya en el algoritmo deflate, una combinación entre LZ77 y la codificación Huffman. En comparación con otros métodos de compresión, gzip se caracteriza por su velocidad, aunque su grado de compresión es más bajo. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **bzip2 / bunzip2** | **Comprime y descomprime archivos con bzip**  Una alternativa popular a *gzip* lo constituye el programa de líneas de comandos *bzip2*. Utilizando la misma sintaxis que aquel, este se basa sin embargo en un proceso de compresión en tres fases que posibilita un mayor grado de compresión.  Primero se somete a los archivos a la transformación reversible de Burrows-Wheeler por bloques y a continuación a la transformación Move to front. Por último tiene lugar la compresión en sí por el método de la codificación Huffman.  Los archivos que han sido comprimidos con *bzip2* llevan la terminación *bz2*. Utiliza *bzip* con este esquema para comprimir archivos:  *bzip2 [OPCIONES] ARCHIVOS*    *bzip2* también se puede aplicar en archivos *tar*.    La descompresión tiene lugar aquí de forma análoga a *bzip* con la opción *-d* o con el comando *bunzip2*.    A cambio de un alto grado de compresión los usuarios han de resignarse a un tiempo de ejecución más largo. |

|  |  |
| --- | --- |
| **COMANDO** | **DESCRIPCIÓN** |
| **xz** | **Comprime y descomprime archivos con *xz***  El comando *xz* convierte archivos en el formato de compresión de datos homónimo. Para invocar el programa se sigue el mismo esquema que con *gzip* y *bzip2*.  *xz [OPCIONES] ARCHIVOS*  Los archivos comprimidos con *xz* adoptan la terminación *.xz*. La descompresión tiene lugar también con la opción *-d* (como en *gzip* y *bzip*). De forma alternativa también se puede usar el comando *unxz*.  Al igual que los archivos *gz* y *bz2,* los archivos *xz* tampoco se archivan. Cuando se quiere escribir varios archivos en el mismo archivo comprimido *xz* también se ha de recurrir a la herramienta de archivado *tar*.  *xz* soporta diversos algoritmos de compresión, por defecto el Lempel-Ziv-Markow (LZMA/LZMA2). |

**BIBLIOGRAFÍA**

1. IONOS (2021, 29 enero). *Comandos de Linux: la lista fundamental*.IONOS Digitalguide. https://www.ionos.es/digitalguide/servidores/configuracion/comandos-de-linux-la-lista-fundamental/