Sistemas libres e introducción a Linux: comandos y permisos.

Linux

Introducción

GNU/Linux es un sistema operativo universal. El sistema operativo se define como un conjunto de programas que permiten interactuar con un pc y pueden ejecutar otros programas o aplicaciones.

En GNU/Linux, este sistema operativo se forma por un conjunto de programas que permiten controlar los diferentes elementos físicos del pc para ofrecerlos al usuario a la hora de ejecutar programas: escribir y leer datos de discos duros, utilizar dispositivos como impresoras o escáneres, etc.

Introducción

El núcleo que une todos estos programas, en este caso, se llama Linux; el resto del sistema fueron proyectos programados por o para el proyecto GNU, de ahí que para denominar al sistema operativo se utilice el término “GNU/Linux”, aunque de manera informal se suele llamar solamente Linux.

Este sistema operativo está basado en la filosofía de Unix: sistema operativo multitarea y multiusuario. Nadie es dueño de Linux; gran parte de su desarrollo lo han realizado y lo realizan voluntarios de todo el mundo de forma altruista.

Historia

Veamos el siguiente vídeo

https://www.youtube.com/watch?v=ShcR4Zfc6Dw

Distribuciones de GNU/Linux

Debido a que GNU/Linux es un sistema operativo completamente abierto, muchas organizaciones comenzaron a crear variantes de dicho sistema con unas ciertas características concretas orientadas a un grupo de usuarios específicos. Estas variantes recibieron el nombre de distribuciones GNU/Linux.

Existen muchos tipos de distribuciones, entre las que se pueden encontrar distribuciones que están soportadas comercialmente como Fedora (Red Hat), openSUSE(Novell) o Ubuntu (Canonical Ltd.), distribuciones mantenidas por la comunidad, como pueden ser Debian y Gentoo, o distribuciones que no están relacionadas con ninguna de las anteriores como puede ser Slackware.

Razones por las que usar Linux

A día de hoy existen múltiples distribuciones diseñadas para hacer la vida fácil al usuario particular, incluso distribuciones con una interfaz visual semejante a otros sistemas para que la curva de aprendizaje sea más corta.

Además de esto, hay que tener en cuenta que todas las aplicaciones son libres, por tanto no existen limitaciones a la hora de adquirir software.

Al ser multiusuario y multitarea, un solo equipo puede ser usado por múltiples usuarios a través de la exportación del display gráfico por red, que puede ser una buena solución empresarial para el ahorro de costes.

Razones por las que usar Linux

Al estar desarrollado por la comunidad y ser completamente gratuito, no es objetivo de organizaciones criminales u otros entes creadores de virus y malware a gran escala, como le puede suceder a otros sistemas operativos privados, además de contar con un soporte de una comunidad de millones de personas que aportan constantemente soluciones y parches a los problemas de seguridad de las distribuciones.

Acceso a los sistemas Linux

Existen diversos métodos para la obtención de distribuciones GNU/Linux. La más usual es la descarga de la imagen ISO de la distribución. Las más relevantes:

- Debian: https://www.debian.org/
- Ubuntu: https://ubuntu.com/

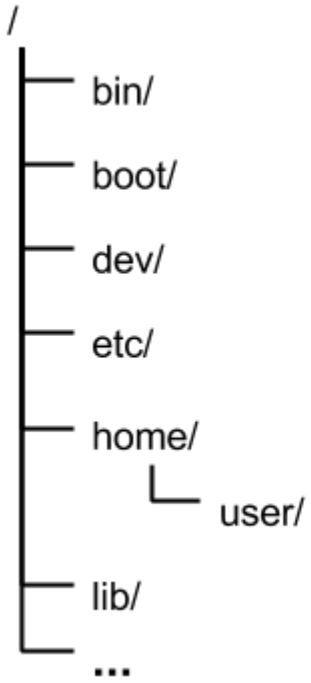
Estructura directorios Linux

Para comprender la estructura de directorios de Linux hay que analizar y conocer previamente el estándar FHS (Filesystem Hierarchy Standard). Esta norma determina cómo se definen los directorios principales y los contenidos del sistema operativo GNU/Linux, llegando a completarse en 1995.

Para entender cómo funciona Linux y su relación con los elementos físicos de almacenamiento, primero vamos a verlo desde el lado del software y a partir de ahí desarrollaremos la relación entre esto y los componentes físicos (particiones).

Estructura directorios Linux

Cada directorio que cuelga directamente del raíz (/) tiene un propósito concreto en el sistema.



Estructura directorios Linux

Veamos los tipos de archivos que tiene Linux:

- **Estáticos**: archivos que solo se pueden modificar por el usuario root, sin embargo, pueden ser leídos por cualquier otro usuario.
- **Dinámicos**: archivos que pueden ser modificados y leídos por cualquier usuario.
- **Compartidos**: archivos que se comparten en red o entre usuarios del mismo ordenador.
- **Restringidos**: Contiene ficheros que no se pueden compartir, solo son modificables por el administrador.

Estructura directorios Linux

FHS define algunos directorios con mucha precisión. Los más comunes definidos por FHS o utilizados por convención, son los siguientes:

/ : Los sistemas de ficheros Linux tienen su raíz en un mismo directorio conocido como sistema de ficheros raíz o directorio raíz. Todos los demás directorios se ramifican desde éste. Linux no utiliza letras de unidad sino que las particiones o discos extraíbles se montan en un punto dentro del sistema de ficheros raíz. Algunos directorios críticos deben residir siempre en la partición raíz pero otros pueden encontrarse en particiones independientes. No se debe confundir el directorio /root con el directorio raíz.

Estructura directorios Linux

/boot : Contiene ficheros estáticos y no compartibles relacionados con el arranque del ordenador.

/bin : Contiene algunos ficheros ejecutables que son accesibles para todos los usuarios y constituyen los comandos más importantes que pueden ejecutar los usuarios normales. Contiene ficheros estáticos. Sus ficheros son compartibles pero son tan importantes para el funcionamiento básico del ordenador que este directorio casi nunca se comparte.

Estructura directorios Linux

/sbin : Es similar a /bin pero contiene programas que sólo ejecuta el administrador. Es estático y en teoría compartible. En la práctica sin embargo, no tiene sentido compartirlo.

/lib : Contiene bibliotecas de programa que son código compartido por muchos programas y que se almacenan en ficheros independientes, para ahorrar RAM y espacio en disco. /lib/modules contiene módulos o drivers que se pueden cargar y descargar según necesitemos. Es estático y teóricamente compartible aunque en la práctica no se comparten.

Estructura directorios Linux

/usr : Aloja el grueso de los programas de un ordenador Linux. Tiene un contenido compartible y estático lo que permite montarlo en modo sólo lectura. Se puede compartir con otros sistemas Linux; muchos administradores separan /usr en una partición independiente aunque no es necesario. Contiene algunos subdirectorios similares a los del directorio raíz como /usr/bin y /usr/lib , que contienen programas y bibliotecas que no son totalmente críticos para el funcionamiento del ordenador.

Estructura directorios Linux

/opt : Es similar a /usr/local pero está pensado para los paquetes que no vienen con el SO como los procesadores de texto o juegos comerciales, que se guardan en sus propios subdirectorios. El contenido de /opt es estático y compartible. Se suele separar en su propia partición para convertirlo en un enlace simbólico a un subdirectorio de /usr/local.

/home : contiene los datos de los usuarios y es compartible y variable. Se considera opcional en FHS, pero, en la práctica lo opcional es el nombre. El directorio /home con mucha frecuencia reside en su propia partición.

Estructura directorios Linux

/etc : Contiene archivos de configuración del sistema específicos del Host de todo el sistema.

/root : Es el directorio home del usuario root. Como la cuenta de root es tan crítica y específica del sistema, este directorio variable no es realmente compartible.

/var : Contiene ficheros efímeros de varios tipos, de registro del sistema, de cola de impresión, de correo y news, etc. El contenido del directorio es variable pues algunos subdirectorios son compartibles y otros no. Se suele colocar /var en su propia partición, sobre todo si el sistema registra una gran actividad en /var.

Estructura directorios Linux

/tmp : Es donde se crean los archivos temporales y variables que necesitan los programas. La mayoría de las distribuciones limpian este directorio periódicamente en el inicio. Este directorio raramente se comparte pero se suele poner en una partición independiente para que los procesos no controlados no provoquen problemas en el sistema de ficheros al ocupar demasiado.

/mnt : La finalidad de este directorio es albergar el montaje de los dispositivos. Los medios montados en esta partición pueden ser estáticos o variables y, por norma general son compartibles.

Estructura directorios Linux

/media : Es una parte opcional del FHS como /mnt , pero que podría contener subdirectorios para tipos de medio específicos. Muchas distribuciones lo utilizan para los discos extraíbles.

/dev : contiene un gran número de ficheros que hacen de interfaces de hardware. Con los permisos apropiados se accede al hardware del dispositivo leyendo y escribiendo en el fichero de dispositivo asociado. El kernel permite que /dev sea un sistema de ficheros virtual creado automáticamente. El kernel y las herramientas de soporte crean sobre la marcha entradas en /dev para adaptarse a las necesidades de los drivers específicos.

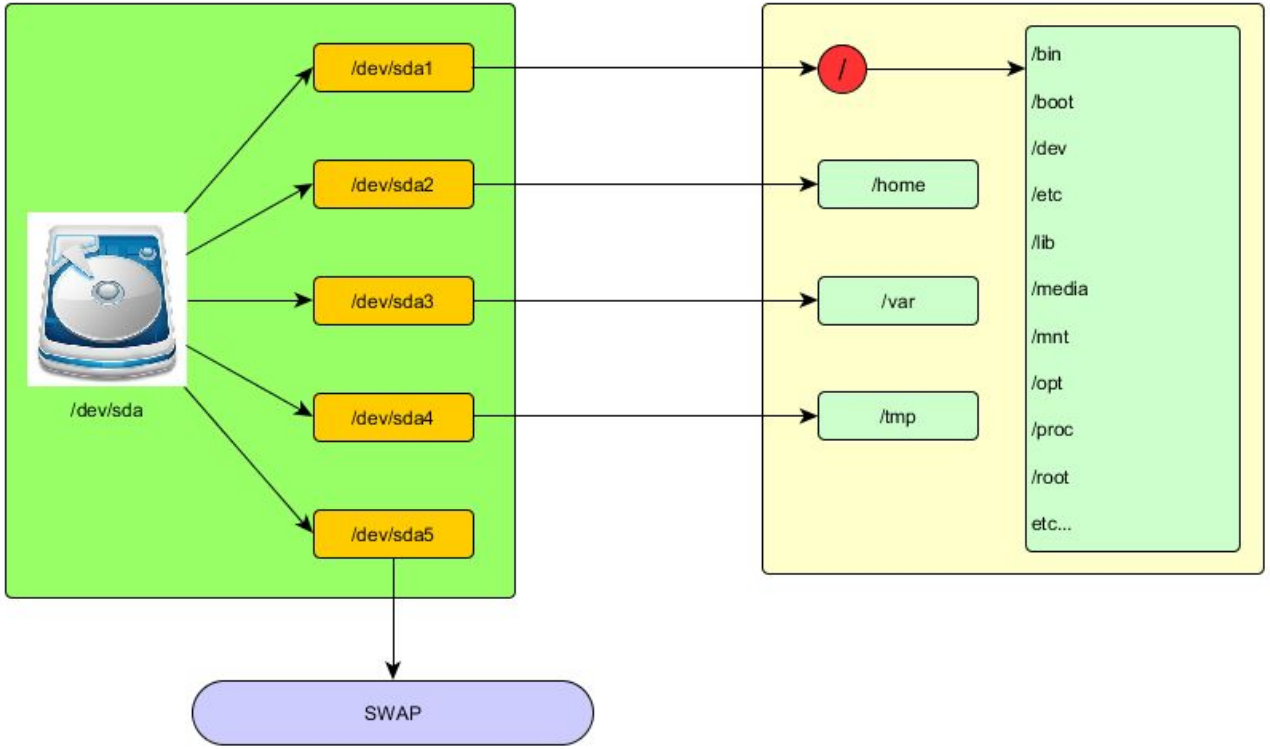
Estructura directorios Linux

/proc : Es un directorio inusual ya que no corresponde a un directorio o partición normal sino que se trata de un sistema de ficheros virtual que proporciona acceso a ciertos tipos de información del hardware dinámicamente. Esta información no se encuentra accesible a través de /dev.

Estructura directorios Linux

Tras este estudio del sistema de ficheros a nivel software, a continuación vamos a ilustrar con un ejemplo gráfico cómo se distribuyen los discos y particiones dentro del sistema de ficheros. El ejemplo consta de un disco SATA que tiene 5 particiones, y cada una de ellas albergará un directorio (o punto de montaje) de la estructura de directorios visto anteriormente:

Estructura directorios Linux



Estructura directorios Linux

Como se puede apreciar en la imagen, cada partición del disco duro está asociado a un directorio de la estructura de directorios única del sistema operativo, a diferencia de windows que cada partición del disco duro se le asigna una letra de unidad (C:, D:, etc) y en ella se contiene su propia estructura de directorios.

Estructura directorios Linux

Esta forma de componer la estructura de directorios tiene una serie de ventajas en cuanto a la integridad y seguridad del sistema operativo que son:

- **Soporte multi-operativo:** permite alojar varios sistemas operativos.
- **Elección del sistema de ficheros:** cada sistema de ficheros ofrece diferentes características eligiendolas según las necesidades que se tengan para esa partición.
- **Control y administración del espacio en disco:** Se puede controlar el acceso de los usuarios a las diferentes particiones.

Estructura directorios Linux

- **Protección de errores en el disco:** Al estar dividida la partición un error físico del disco probablemente afectará a una parte del sistema operativo, dejando la posibilidad de que se pueda seguir trabajando con él e intentar recuperar el sector dañado.
- **Seguridad:** puedes asegurar un sector de tu sistema de datos críticos montándolo en solo lectura. La ventaja de agregar esta característica a la partición y no a los ficheros es por cuestiones de redundancia.
- **Backup:** Las herramientas de copias de seguridad trabajan mejor en sistemas pequeños y aislados de tareas de escritura.

File Systems Linux

Ext significa "Extended file system", y fue el primero creado específicamente para Linux. Ha tenido cuatro grandes revisiones. "Ext" es la primera versión del sistema de archivos, introducida en 1992. Fue una actualización importante del sistema de archivos Minix utilizado en ese momento, pero carece de características importantes. Muchas distribuciones de Linux ya no soportan Ext.

File Systems Linux

Ext2 no es un sistema de archivos con journaling. Cuando se introdujo, fue el primer sistema de archivos que soportaba atributos de archivo extendidos y unidades de 2 terabytes. El hecho de que Ext2 no tenga un journaling significa que escribe menos en el disco, lo que lo hace útil para las memorias flash como las unidades USB. Sin embargo, los sistemas de archivos como exFAT y FAT32 tampoco utilizan el registro en el diario y son más compatibles con diferentes sistemas operativos, por lo que te recomendamos que evites Ext2 a menos que sepas que lo necesitas por alguna razón.

File Systems Linux

Ext3 es básicamente Ext2 con journaling. Ext3 fue diseñado para ser compatible con Ext2, permitiendo que las particiones se conviertan entre Ext2 y Ext3 sin necesidad de formatearlas. Ha existido durante más tiempo que Ext4, pero Ext4 ha existido desde 2008 y está ampliamente probado. En este momento, es mejor usar Ext4.

File Systems Linux

Ext4 también fue diseñado para ser compatible con versiones anteriores. Puedes montar un sistema de archivos Ext4 como Ext3, o montar un sistema de archivos Ext2 o Ext3 como Ext4. Incluye nuevas características que reducen la fragmentación de los archivos, permite volúmenes y archivos más grandes, y utiliza la asignación retardada para mejorar la vida de la memoria flash. Esta es la versión más moderna del sistema de archivos Ext y es la predeterminada en la mayoría de las distribuciones de Linux.

File Systems Linux

BtrFS, pronunciado "Butter" o "Better" FS, fue diseñado originalmente por Oracle. Sus siglas significan "B-Tree File System" y permite la agrupación de unidades, las instantáneas sobre la marcha, la compresión transparente y la desfragmentación en línea. BtrFS está diseñado para romper con la serie Ext de sistemas de archivos.

File Systems Linux

ZFS fue diseñado por Sun Microsystems para Solaris y ahora es propiedad de Oracle. ZFS admite muchas funciones avanzadas, como la agrupación de unidades, las instantáneas y la separación dinámica de discos. Cada archivo tiene una suma de comprobación, por lo que ZFS puede saber si un archivo está dañado o no. Sun ha creado ZFS bajo la licencia CDDL de Sun, lo que significa que no puede incluirse en el núcleo de Linux. Sin embargo, se puede instalar el soporte de ZFS en cualquier distribución de Linux.

File Systems Linux

XFS fue desarrollado por Silicon Graphics en 1994 para el sistema operativo SGI IRX, y fue portado a Linux en 2001. Es similar a Ext4 en algunos aspectos, ya que también utiliza la asignación retardada para ayudar con la fragmentación de archivos y no permite las instantáneas montadas. Puede ampliarse, pero no reducirse, sobre la marcha. XFS tiene un buen rendimiento cuando se trata de archivos grandes, pero tiene peor rendimiento que otros sistemas de archivos cuando se trata de muchos archivos pequeños. Puede ser útil para ciertos tipos de servidores que necesitan principalmente tratar con archivos grandes.

File Systems Linux

Swap es una opción al formatear una unidad, pero no es un sistema de archivos real. Se utiliza como memoria virtual y no tiene una estructura de sistema de archivos. No se puede montar para ver su contenido. El kernel de Linux utiliza la memoria Swap como "espacio de memoria virtual" para almacenar temporalmente los datos que no caben en la RAM. También se utiliza para hibernar. Mientras que Windows almacena su archivo de paginación como un archivo en su partición principal del sistema, Linux sólo reserva una partición vacía separada para el espacio de intercambio.

Shell

El shell es el entorno que hace de intermediario entre el usuario y los recursos del ordenador, como si fuera un entorno de programación en tiempo real para ejecutar tareas. La shell aparece cuando nos logueamos en el sistema en modo consola, o bien cuando en el entorno gráfico abrimos una terminal. La shell se compone de un prompt, que es un texto inicial que normalmente nos ofrece información útil como el usuario que está utilizando la shell, el hostname de la máquina o incluso el directorio sobre el que estamos posicionados en cada momento, y el cursor que recibe las ordenes de teclado.

Shell

Existen multitud de shells diferentes para poder interactuar con nuestro sistema operativo, aunque la más conocida y habitual en la mayoría de distribuciones es la shell bash.

Cada shell cuenta con sus propias características de uso, contando con atajos de teclado, visualización en vivo de ficheros, y atajos durante la navegación entre directorios.

Shell

A continuación vamos a citar las shells más famosas y una pequeña descripción de su procedencia:

- **bash** (Bourne Again Shell): Se basa en los principios de shell Bourne de Unix pero se ha extendido en varios aspectos. Es la shell por defecto para la mayoria de las cuentas de usuario y es la que se tratará con más detalle en este curso.
- **bsh**: El shell Bourne es la shell sobre la que está basada bash. Se conoce con el nombre de BSH. Su uso no es frecuente en Linux aunque el comando bsh suele ser un enlace simbólico a bash.

Shell

- **tcsh:** Este shell tcsh se basa en el anterior shell C (csh). Es una shell bastante popular en algunos círculos pero no hay distribuciones de Linux que lo traigan por defecto.
- **csh:** El original csh shell C no es muy utilizado en Linux pero si un usuario está familiarizado con csh, tcsh es un buen sustituto.
- **ksh:** El shell Korn (ksh) fue diseñado cogiendo las mejores opciones de la shell Bourne y el C shell. Tiene un pequeño pero dedicado numero de seguidores.
- **zsh:** El shell zsh Z (zsh) es la evolución de la shell Korn e incorpora características de esta última además de agregar otras.

Shell

Para usar inicialmente la shell bash, en nuestra máquina virtual abrimos una terminal, buscando en el menú “Terminal”. La shell bash cuenta con una serie de atajos de teclado que permiten la navegación más sencilla:

- Si al acceder a directorios pulsamos la tecla TAB se autocompleta en el caso de existir sólo una iteración o te muestra un desplegable de las diferentes iteraciones existentes.
- Las teclas del cursor arriba y abajo nos permiten navegar por el historial de comandos que hemos escrito anteriormente.

Shell

- Las teclas de cursor derecha e izquierda nos permiten navegar sobre lo que hemos escrito en la shell, también se puede utilizar la combinación Ctrl + Derecha o izquierda para saltar palabras o campos en vez de caracter a caracter.

- Si pulsamos la combinación de teclas Ctrl + r nos aparecerá un menú donde, si escribimos el comando que estamos buscando, nos aparecerá la primera iteración buscada. Si una vez encontrado, seguimos pulsando Ctrl +r, se seguirá realizando la búsqueda de forma inversa de esta misma iteración.

Shell

La shell, además de esto, se compone también de una serie de comandos internos y de variables de entorno que definirán parámetros en la configuración del sistema.

Variables de entorno: Las variables de entorno son muy utilizadas tanto por la shell como por otros programas para recuperar datos fundamentales del pc para su configuración.

- **SHELL**: Define la shell que estamos usando. El valor que toma es el binario que ejecuta la shell de bash, que se ubica en /bin/bash
- **USER**: Define el usuario que está actualmente logueado en la shell.

Shell

- **PATH**: Esta variable contiene un conjunto de rutas donde el sistema va a buscar los ejecutables. Para indicar varias rutas, cada una de ellas se separa de la otra con el carácter dos puntos “:”. De este modo, no es necesario indicarle al sistema en que ubicación se encuentra el programa para poder ejecutarlo.
- **PWD**: Esta variable muestra en qué directorio estamos posicionados en todo momento.
- **HOME**: Esta variable almacena el home del usuario con el que estamos logueados en la shell. El home del usuario es el espacio en la estructura de directorios donde se almacenan todos los datos de un usuario.

Shell

Comandos internos: Los comandos internos de la shell tienen la particularidad que no generan un IDentificador de Proceso (PID) adicional, sino que utilizan el de la propia shell:

- **alias:** Crea un nombre alternativo (alias) para un comando. Esto es útil para facilitar la entrada de comandos recurrentes que llevan varios argumentos.
- **echo**: Repite el texto o el contenido de una variable.
- **env**: Sin argumentos muestra las variables de entorno y el contenido de las mismas. Se puede utilizar también para ejecutar un comando con una variable de entorno modificada temporalmente.

Shell: Directorios y archivos

Una ruta señala la localización exacta de un archivo o directorio mediante una cadena de caracteres concreta. En líneas generales se compondrá de los nombres de los directorios que conforman el camino hasta nuestro archivo o directorio a lo largo del árbol de directorios, y finalmente estará el nombre del archivo o directorio al que se quiere llegar.

- Las rutas **absolutas** señalan la ubicación de un archivo o directorio desde el directorio raíz del sistema de archivos.
- Las rutas **relativas** señalan la ubicación de un archivo o directorio a partir de la posición actual en el sistema de archivos.

Shell: Directorios y archivos

A continuación veremos los comandos necesarios para la navegación y visualización de ficheros y directorios, así como la gestión de directorios y ficheros:

- touch
- cd
- mkdir
- rmdir
- rm
- ls

Shell: Permisos

Los ficheros y directorios tienen 3 tipos de identidad: La individual, la de grupo y la del resto del sistema. Por tanto, se puede indicar que un fichero o directorio en el sistema Linux tiene un perfil de propietario, un perfil de grupo y un perfil de sistema.

Cada uno de estos “perfiles” se le pueden asignar permisos. Los permisos se componen de tres elementos: **lectura**, **escritura** y **ejecución**. De este modo, la combinación de perfiles con sus respectivos permisos nos ofrecen una administración eficiente de los diferentes ficheros y directorios existentes en nuestro sistema.

Shell: Permisos

Existe una pequeña apreciación en el permiso de ejecución, ya que hay que discriminar su uso a ficheros y directorios, puesto que no realizan la misma función en ellos. Para los ficheros, el permiso de ejecución hace que el fichero sea ejecutable. Esto se suele hacer en shell scripts y otros ficheros especiales del sistema.

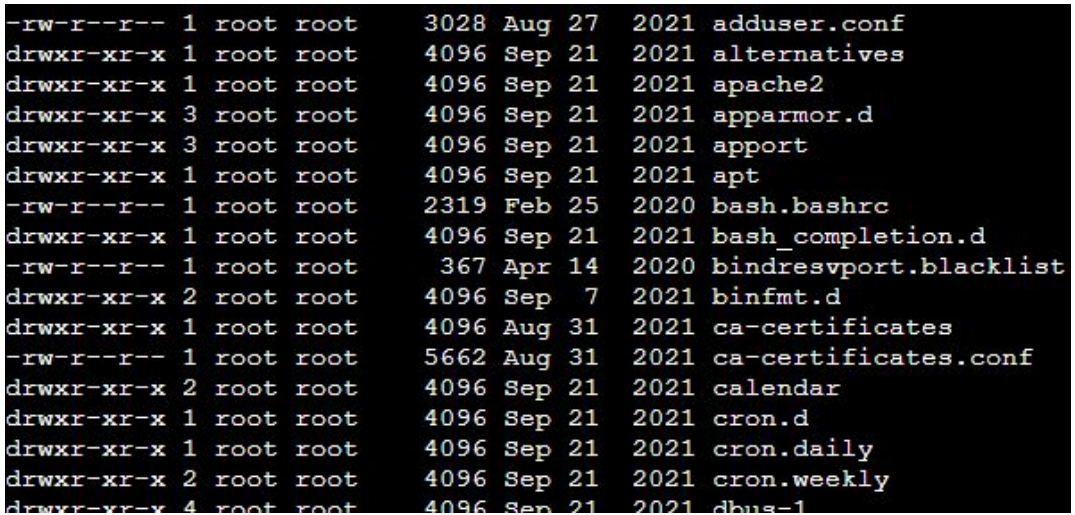
Sin embargo para los directorios el permiso de ejecución permite que se pueda acceder al directorio (ya que un directorio, por lógica, no se puede ejecutar).

Shell: Permisos

Teniendo esta regla en cuenta, lo habitual es que todos los directorios tengan el permiso de ejecución incluido en ellos para, al menos, el propietario del directorio. Los permisos de lectura, escritura y ejecución se muestran con las siglas r, w y x respectivamente.

Shell: Permisos

Los permisos de lectura, escritura y ejecución se muestran con las siglas r, w y x respectivamente. Estos permisos se pueden observar en el comando ls al inicio:



Shell: Permisos

La primera letra representa el tipo de archivo y determinará como Linux interpretará el archivo, como un dato, un directorio, etc. El tipo puede ser:

- d : Directorio: Contienen nombres de archivos que apuntan a inodos de disco.
- l : Enlace simbólico. Este archivo contiene el nombre de otro archivo o directorio.
- c : Dispositivo especial de caracteres. Un archivo que corresponde a un dispositivo de hardware el cual transfiere los datos en unidades de un bit.

Shell: Permisos

- p : Se le denomina canal, ya que permite tener disponibles dos programas de linux y que se puedan comunicar uno con el otro. Uno abre el canal para leer y otro para escribir, permitiendo que los datos sean transferidos de uno a otro.
- s : Socket. Es similar al llamado canal (p), pero éste permite redireccionar enlaces.
- - : Archivo convencional. Debe ser texto, un archivo ejecutable, gráficos, etc.

Shell: Permisos

Los restantes caracteres de la cadena se dividen en 3 grupos de 3 caracteres. El primer grupo, controla el acceso al fichero del propietario, el segundo grupo controla el acceso del grupo y el tercero controla los accesos de todos los demás usuarios.

En cada uno de los tres caracteres la cadena de permisos determina la presencia o ausencia de cada uno de los tres tipos de acceso: lectura, escritura y ejecución.

Shell: Permisos

Si se incluye el permiso de ejecución el fichero se podrá ejecutar como un programa y definir el bit de ejecución en un fichero que no es un programa no tiene ningún tipo de consecuencia. Si en la cadena de permisos aparece un guión (-) quiere decir que el permiso no tiene permisos. La presencia del permiso se indica con una r para lectura, un w para la escritura y una x para la ejecución.

Shell: Permisos

Por tanto, la cadena de permisos rwxr-xr-x significa que tanto el propietario del fichero como los miembros de su grupo y todos los demás usuarios pueden leer y ejecutar el fichero, pero sólo su propietario tiene permisos de escritura sobre él.

Linux codifica en formato binario la información de los permisos que se puede expresar como un único número de 9 bits. Este número se suele expresar en formato octal (base 8) debido a que un número en base 3 tiene 3 bits de longitud. Los permisos de lectura, escritura y ejecución se corresponden con cada uno de los bits.

Shell: Permisos

El permiso de ejecución tiene sentido para los ficheros normales pero no para el resto. Los directorios utilizan el bit de ejecución de otra manera: si está definido significa que se puede buscar en el contenido del directorio. Por eso es difícil ver un directorio en el que el bit de ejecución no se defina junto al de lectura. Si a un usuario se le da permiso para escribir en un directorio, este usuario podrá crear, borrar o renombrar los ficheros del directorio aunque no sea el propietario de los ficheros y no tenga permisos para escribir en ellos.

Shell: Permisos

Los enlaces simbólicos siempre tienen permisos 777, sin embargo, este acceso se aplica sólo al propio fichero del enlace y no al fichero enlazado.

La mayoría de las reglas de permisos no se aplican a root, por lo que puede leer o escribir en cualquier fichero del ordenador, incluso en aquellos que tienen permisos 000, ya que root puede cambiar los permisos de cualquier fichero. Hay algunos ficheros que pueden ser inaccesibles para root, pero sólo porque existe una restricción subyacente, como por ejemplo que un disco duro no esté instalado en el ordenador.

Shell: Permisos de un archivo

El superusuario (root) puede cambiar el propietario de un fichero mediante el comando chown.

```
# chown [opciones] [nuevo_propietario] [:nuevo_grupo] nombres_ficheros
```

Las opciones más usuales del comando chown son:

-R: Cambia el permiso de archivos que están en subdirectorios del directorio en el que estés en ese momento.

-C : Cambia el permiso para cada archivo.

-f: cuando es incapaz de cambiar la titularidad del archivo, previene al comando chown de mostrar mensajes de error.

Shell: Permisos de un archivo

En la sintaxis de este comando, los argumentos "nuevo propietario" y" nuevo grupo" se refieren al nuevo propietario y el nuevo grupo del fichero. Se pueden proporcionar ambos pero sólo se puede omitir uno de ellos. El comando chown admite muchas opciones, pero la más utilizada es -R o –recursive , que permite cambiar de propiedad un árbol de directorios completo. Sólo root puede utilizar el comando chown; si trata de utilizarlo un usuario normal devolverá un mensaje de error.

Shell: Permisos de un archivo

El comando chgrp cambia los permisos del grupo de un fichero de un archivo o directorio. Este comando lo pueden ejecutar tanto el usuario root como los usuarios normales. Los usuarios sólo pueden cambiar el grupo de un fichero por un grupo al que pertenezcan. Su sintaxis es la siguiente:

```
# chgrp [opciones] nuevo_grupo nombres_ficheros
```

Shell: Permisos de un archivo

Este comando acepta muchas opciones, entre las que se incluyen -R o –recursive. Esta herramienta es un subconjunto de funcionalidades de chown , pero la pueden utilizar los usuarios normales.

Las opciones más usuales del comando chgrp son las siguientes:

- R: Cambia el permiso de archivos que están en subdirectorios del directorio en el que estés en ese momento.
- C : Cambia el permiso para cada archivo.
- f : Esta opción lo que hace es forzar el cambio. No informa de mensajes de errores.

Shell: Permisos de un archivo

Con el comando chmod se pueden modificar los permisos de un fichero. Este comando se puede ejecutar de varios modos para obtener el mismo efecto. Su sintaxis es:

```
chmod [opciones] [modo[,modo...]] nombre_fichero…
```

La opción –recursive o -R modifica los permisos de un árbol de directorios. Se puede especificar el modo de dos formas básicas: con un número en octal o con un modo simbólico.

```
chmod 644 report.txt
```

Shell: Permisos de un archivo

Además de modificar los 3 dígitos de permisos del archivo, si al comando anterior se le añade un cuarto bit al principio se pueden definir permisos especiales:

- Si se añade un 4, se definirá el bit de la ID de usuario (SUID).
- Si se añade un 2 se definirá la ID de grupo (SGID).
- Si se añade un 1 se definirá el sticky bit: solo el usuario creador pueda eliminar o renombrar el archivo/elementos directorio.

Si se omite el primer dígito como en el ejemplo anterior Linux tomará los 3 bits como bits para permisos y no como bits para permisos especiales.

Shell: chmod

Código del conjunto de permisos	Significado	Código de tipo de cambio	Significado	Permisos para modificar el código	Significado
u	propietario	+	añadir	r	lectura
g	grupo	-	eliminar	w	escritura
o	global	=	definir igual que	x	ejecución
a	todos			X	ejecutar solo si el fichero es un directorio o tiene permisos de ejecución
				s	SUID o SGID
				t	sticky bit
				u	permisos del propietario existentes
				g	permisos del grupo existente
				o	permisos globales existentes

ACL

ACl es una lista de reglas que especifica los permisos de acceso a archivos y directorios. Las ACL extienden o complementan los permisos tradicionales de Unix (lectura, escritura y ejecución) permitiendo definir permisos más detallados y flexibles.

El sistema de archivos ext4 admite ACL para proporcionar un control de acceso más granular. No todos los sistemas de archivos lo pueden admitir.

ACL

Para trabajar con ACL en Ubuntu, se pueden utilizar comandos como **getfacl** para ver la lista de control de acceso de un archivo o directorio, y **setfacl** para establecer nuevas reglas de acceso.

El uso de ACL permite asignar permisos a usuarios individuales, grupos específicos o incluso a usuarios que no pertenecen al grupo propietario del archivo.

FIN