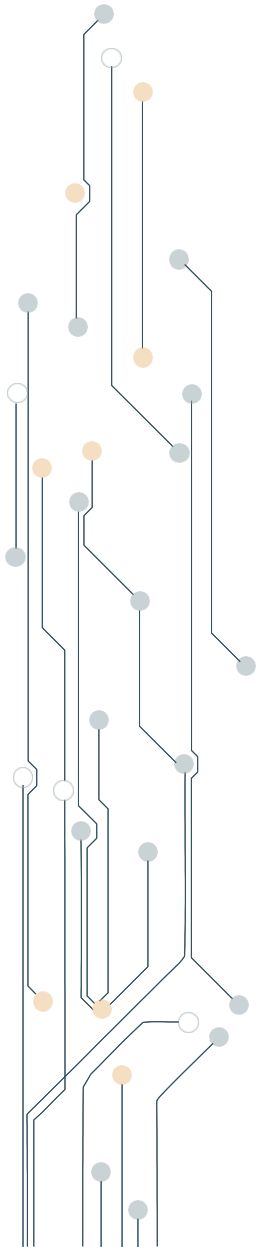
Administración de archivos y directorios

Índice



Unidad 4 – Administración de Archivos y Directorios

Qué es un sistema de archivos? Un sistema de archivos son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos de un disco o partición; es decir, es la manera en la que se organizan los archivos.

El sistema de archivos de linux es una organización jerárquica que se asemeja a un árbol donde cada hoja es un directorio o un archivo.

Originariamente, en los inicios de Linux, este árbol de directorios no seguía un estándar cien por cien, es decir, podíamos encontrar diferencias en él de una distribución a otra.

Todo esto hizo pensar a cierta gente que, posteriormente, desarrollarían el proyecto FHS (Filesystem Hierarchy Standard, o lo que es lo mismo: Estándar de Jerarquía de Sistema de archivos) en 1993.

FHS no es más que un documento guía, es decir, cualquier fabricante de software independiente o cualquier persona que decida crear una nueva distribución GNU/Linux, podrá aplicarlo o no a la estructura del sistema de archivos, con la ventaja de que si lo integra en el sistema, el entorno de éste será mucho más compatible con la mayoría de las distribuciones.

Es importante saber que el estándar FHS es en cierto modo flexible, es decir, existe cierta libertad en el momento de aplicar las normas. De ahí que existan en la actualidad leves diferencias entre distribuciones GNU/Linux.

Objetivos principales de FHS

- **Presentar un sistema de archivos coherente y estandarizado.**

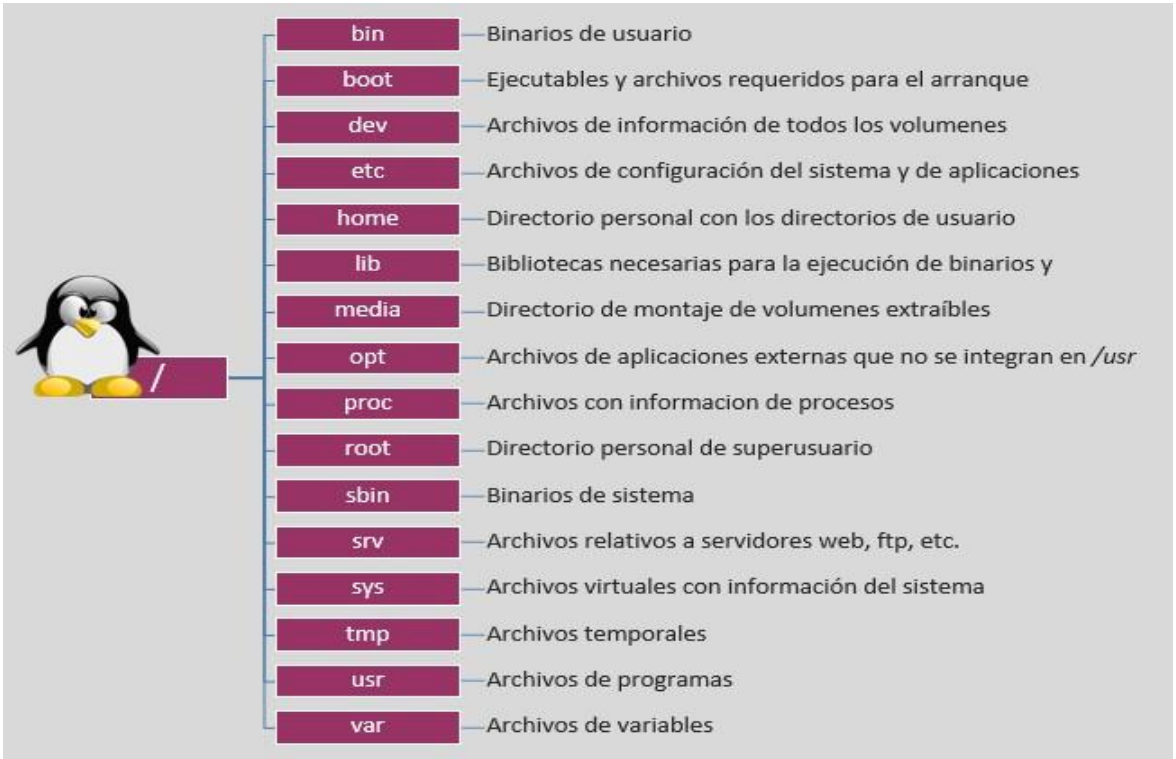
- **Facilidad para que el software prediga la localización de archivos y directorios instalados.**
- **Facilidad para que los usuarios prediga la localización de archivos y directorios instalados.**
- **Especificar los archivos y directorios mínimos requeridos.**

El estándar FHS está enfocado a:

Fabricantes de software independiente y creadores de sistemas operativos, para que establezcan una estructura de archivos lo más compatible posible.

Usuarios comunes, para que entiendan el significado y el contendido de cada uno de los elementos del sistema de archivos.

Principales Directorios y sus Funciones

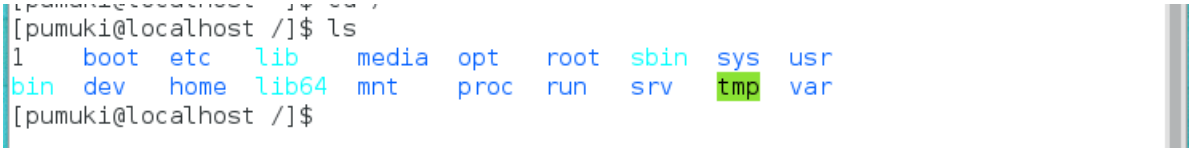


/

Toda la estructura de directorios en los sistemas basados en UNIX parte de un directorio raíz también llamado directorio root y que se simboliza por una barra inclinada o /. De este directorio, es desde donde nacen todo el resto de directorios, independientemente que estén almacenados físicamente en discos o unidades separadas.

Cualquier dirección de archivo o carpeta en Linux empieza por el directorio raíz o /, seguido de todos los directorios y subdirectorios que lo contienen, separados cada uno de ellos por /.

A continuación conocerás con más en detalle a todos los directorios principales que parten del directorio raíz, junto con sus subdirectorios más importantes y los archivos que suelen contener.



/bin

El directorio /bin es un directorio estático y es donde se almacenan todos los binarios necesarios para garantizar las funciones básicas a nivel de usuario. Solo almacena los ejecutables de usuario, ya que los binarios necesarios para tareas administrativas gestionadas por el usuario root o superusuario del sistema se encuentran en el directorio /sbin.

Incluye también los binarios que permiten la ejecución de varias utilidades estándar de la terminal de Linux, concretamente cat, cd, cp, echo, grep, gzip, kill, ls, mv, rm, ping, su, ps, tar y vi.

```
[pumuki@localhost bin]$ ls
[                                mformat
a2p                              minfo
abrt-action-analyze-backtrace    mixartloader
abrt-action-analyze-c            mkafmmap
abrt-action-analyze-ccpp-local   mkdir
abrt-action-analyze-core         mkfifo
abrt-action-analyze-oops         mkfontdir
abrt-action-analyze-python       mkfontscale
abrt-action-analyze-vmcore       mkhybrid
abrt-action-analyze-vulnerability mkinitrd
abrt-action-analyze-xorg         mkisofs
```

/boot

Es un directorio estático e incluye todos los ejecutables y archivos que son necesarios en el proceso de arranque del sistema, y que deberán ser utilizados antes que que el kernel empiece a dar las órdenes de ejecución de los diferentes módulos del sistema. Es también donde se encuentra el gestor de arranque GRUB.

En algunas distribuciones, es común que ese directorio se almacene en supropia partición separada del resto. Esto suele darse sobre todo en el caso de de que utilicen LVM (Logical Volume Manager - veremos más adelante en detalle de que se trata esto) por defecto, ya que tradicionalmente el gestor de arranque GRUB (en versiones anteriores a la 2) no podía arrancar desde LVM, por lo que se requería que estuviera en una partición separada.

De hecho, si en una instalación normal de Ubuntu o Debian optas por utilizar LVM, verás que el instalador ya te genera un esquema de particiones con el directorio boot en una partición aparte.

En estos casos, en el momento de instalar el sistema es importante prever bien el espacio que le vayas a dar a la partición, ya que a la larga, con la acumulación de diferentes actualizaciones del Kernel, es común que se quede sin espacio. Si esto sucede, puedes tener problemas a la hora de instalar futuras actualizaciones del núcleo, y debas hacer limpieza de versiones antiguas del kernel.

```
[pumuki@localhost bin]$ cd /boot
[pumuki@localhost boot]$ ls
config-3.10.0-514.26.2.el7.x86_64
config-3.10.0-514.el7.x86_64
grub
grub2
initramfs-0-rescue-9c3b430c6eea4096b1723307642a6b90.img
initramfs-3.10.0-514.26.2.el7.x86_64.img
initramfs-3.10.0-514.26.2.el7.x86_64kdump.img
initramfs-3.10.0-514.el7.x86_64.img
initramfs-3.10.0-514.el7.x86_64kdump.img
initrd-plymouth.img
symvers-3.10.0-514.26.2.el7.x86_64.gz
symvers-3.10.0-514.el7.x86_64.gz
System.map-3.10.0-514.26.2.el7.x86_64
System.map-3.10.0-514.el7.x86_64
vmlinuz-0-rescue-9c3b430c6eea4096b1723307642a6b90
vmlinuz-3.10.0-514.26.2.el7.x86_64
vmlinuz-3.10.0-514.el7.x86_64
[pumuki@localhost boot]$
```

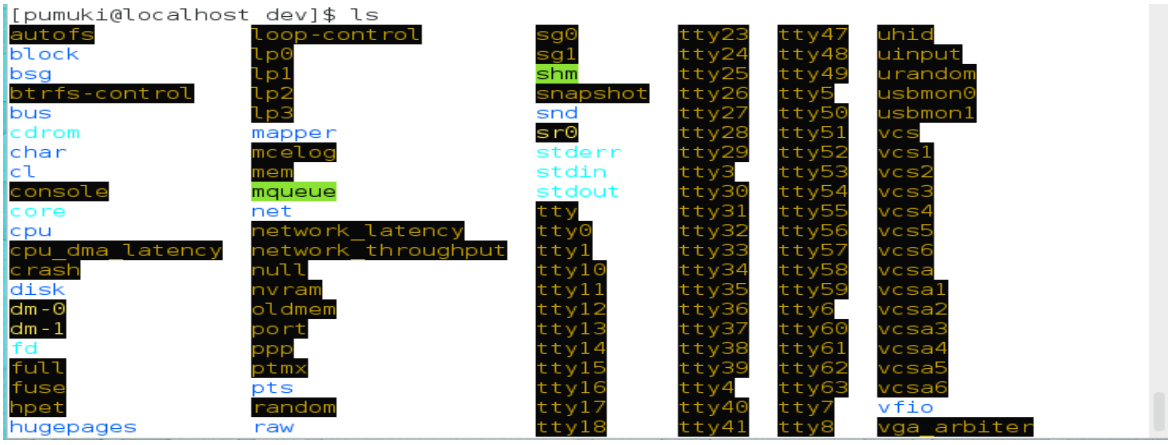
/dev

Este directorio incluye todos los dispositivos de almacenamiento, en forma de archivos, conectados al sistema, es decir, cualquier disco duro conectado, partición, memoria USB, o CDROM conectado al sistema y que el sistema pueda entender como un volumen lógico de almacenamiento.

Siendo esto así, verás que la ruta en la que se encuentra cualquier volumen (partición o dispositivo externo) conectado al sistema siempre empieza por /dev. Este es el directorio que contiene, por decirlo de algún modo, la información de cada uno de los volúmenes, a diferencia del directorio /media, que verás más adelante, que lo que contiene son solo los puntos de montaje, pero no la información real de estos volúmenes.

Para ver esto en la práctica, si abres una ventana de terminal y ejecutas el comando "fdisk –l" (como super usuario), verás la estructura de particiones de tu sistema.

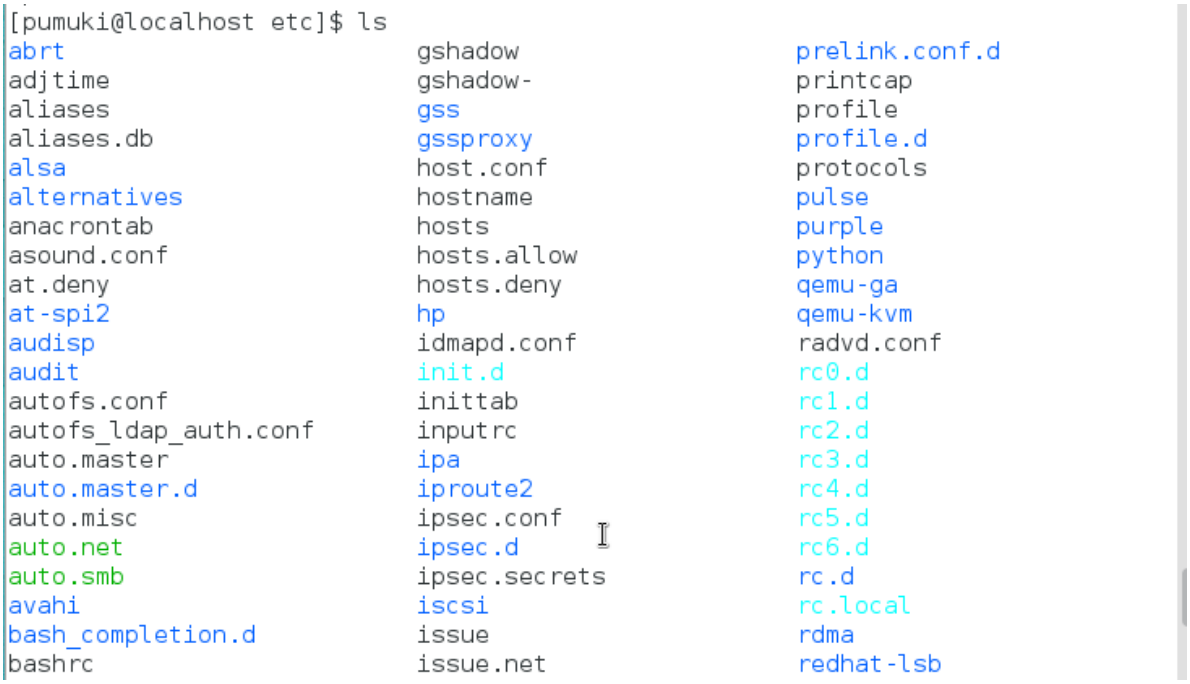
Eso en cuanto a particiones. Si se trata de un dispositivo externo, el volumen estará igualmente dentro de /dev, pero en este caso varía el nombre que el sistema le asigna a dicho volumen.



/etc

Es el encargado de almacenar los archivos de configuración tanto a nivel de componentes del sistema operativo en sí, como de los programas y aplicaciones instaladas a posteriori.

Es un directorio que debería contener únicamente archivos de configuración, y no debería contener binarios.



/home

Es el directorio de los usuarios estándar, y por lo tanto, el destinado a almacenar todos los archivos del usuario, como documentos, fotos, vídeos, música, plantillas, etc. También incluye archivos temporales de aplicaciones ejecutadas en modo usuario, que sirven para guardar las configuraciones de programas, etc.

Dentro /home están los directorios personales de todos los usuarios, nombrados según el nombre de usuario utilizado.

Cada directorio de usuario contiene asimismo diferentes carpetas para ayudarlo a clasificar la información. Estas generalmente son: /Documentos, /Imágenes, /Música, /Plantillas y /Vídeos /, así como otros archivos y carpetas ocultas, que son las encargados de guardar la información de configuraciones de las aplicaciones del usuario.

Por cierto, y muy importante, todas los archivos y carpetas ocultas en Linux empiezan por un punto, seguido del nombre de la carpeta.

En muchas distribuciones es una práctica recomendada el hecho de ubicar el directorio /home es una partición separada del resto, por tal de facilitar que, en caso de reinstalar el sistema operativo, puedas mantener intacta la partición de la /home, y de este modo mantener todos los archivos personales.

```
[pumuki@localhost etc]$ cd /home
[pumuki@localhost home]$ ls
pumuki
[pumuki@localhost home]$
```

/lib

Incluye las bibliotecas esenciales que son necesarios para que se puedan ejecutar correctamente todos los binarios que se encuentran en los directorios /bin y /sbin, así como los módulos del propio kernel.

En los sistemas operativos de 64 bits, además de /lib existe otro directorio denominado /lib64, referida a las bibliotecas para aplicaciones de 64 bits.

```
[pumuki@localhost home]$ cd /lib
[pumuki@localhost lib]$ ls
alsa        grub         jvm-exports   modules-load.d    systctl.d
binfmt.d    java         jvm-private   mozilla           systemd
crda        java-1.5.0   kbd           NetworkManager    tmpfiles.d
cups        java-1.6.0   kdump         polkit-1          tuned
debug       java-1.7.0   kernel        python2.7         udev
dracut      java-1.8.0   locale        rpm               udisks2
firewalld   java-ext     lsb           sendmail          yum-plugins
firmware    jvm          modprobe.d    sendmail.postfix
games       jvm-commmon  modules       sse2
[pumuki@localhost lib]$
```

/media

Representa el punto de montaje de todos los volúmenes lógicos que se montan temporalmente, ya sean unidades externas USB, otras particiones de disco, etc.

En la mayoría de distribuciones GNU/Linux, desde hace ya algún tiempo, cada vez que se monta una unidad externa, partición, etc.,esta se monta dentro del directorio /media y a su vez dentro de un directorio especifico dependiendo del usuario del sistema que monta el volumen.

/mnt

Es un directorio vacío que cumple funciones similares a /media, pero que actualmente no se suele utilizar, ya que la mayoría de distribuciones hacen uso de este último para los puntos de montaje temporales.

/opt

En cierto modo vendría a ser como una extensión del directorio /usr, pero en este caso van todos aquellos archivos de solo lectura que son parte de programas auto-contenidos y que, por lo tanto, no siguen los estándares de almacenar los diferentes archivos dentro de los diferentes subdirectorios de /usr (que sería lo recomendable) Haciendo una analogía con Windows, vendría a ser algo como el directorio de "Archivos y Programas", pero en este caso, como hemos dicho, para determinados programas que ya vienen auto-contenidos.

```
[pumuki@localhost lib]$ cd /opt
[pumuki@localhost opt]$ ls
google  rh
[pumuki@localhost opt]$
```

/proc

Este directorio contiene información de los procesos y aplicaciones que se están ejecutando en un momento determinado en el sistema, pero realmente no guarda nada como tal, ya que lo que almacena son archivos virtuales, por lo que el contenido de este directorio es nulo.

Básicamente son listas de eventos del sistema operativo que se generan en el momento de acceder a ellos, y que no existen dentro del directorio como tales.

```
[pumuki@localhost proc]$ ls
1     2366  2885  3091  3401  42   644  945          fb           pagetypeinfo
10    2387  2888  3096  3405  43   645  950          filesystems  partitions
104   24    2909  31    3406  44   649  951          fs           sched_debug
11    2447  2921  3107  3431  45   655  953          interrupts   schedstat
12    2475  2939  3147  3542  46   657  957          iomem        scsi
1251  25    2955  3149  3604  473  66   965          ioports      self
13    2516  296   3150  3643  496  660  967          irq          slabinfo
15    2619  297   3154  368   499  661  968          kallsyms     softirqs
1504  2667  2991  3197  369   501  662  99           kcore        stat
158   2680  2996  32    3706  586  663  acpi         keys         swaps
17    2682  3     3202  3710  587  668  asound       key-users    sys
18    2684  30    3224  378   588  669  buddyinfo    kmsg         sysrq-trigger
19    2691  3000  3230  379   589  67   bus          kpagecount   sysvipc
2     2692  3003  3244  394   590  670  cgroups      kpageflags   timer_list
20    2760  3012  3256  395   591  673  cmdline      loadavg      timer_stats
21    2765  3014  3280  396   592  680  consoles     locks        tty
2149  278   3019  3290  397   593  693  cpuinfo      mdstat       uptime
2153  280   302   3296  398   6    7    crypto       meminfo      version
2154  281   3030  33    399   618  700  devices      misc         vmallocinfo
22    283   3071  3343  400   629  705  diskstats    modules      vmstat
2257  2857  3076  3348  401   634  721  dma          mounts       zoneinfo
```

/root

Vendría a ser como el directorio /home del usuario root o superusuario del sistema. A diferencia de los otros usuarios, que se encuentran todos dentro de /home en sus respectivas subcarpetas, el directorio del usuario root está en su propia carpeta colgando directamente de la raíz del sistema.

Como es de esperarse solamente el usuario root puede ver su contenido y modificarlo.

```
[pumuki@localhost /]$ cd /root
bash: cd: /root: Permiso denegado
[pumuki@localhost /]$
```

/sbin

Si hemos dicho que en /bin se almacenaban los binarios relativos a las funciones normales de usuario, /sbin hace lo mismo pero para los binarios relativos tareas propias del sistema operativo, y que solamente pueden ser gestionadas por el usuario root, tales como el arranque, tareas de restauración, reparación, etc.

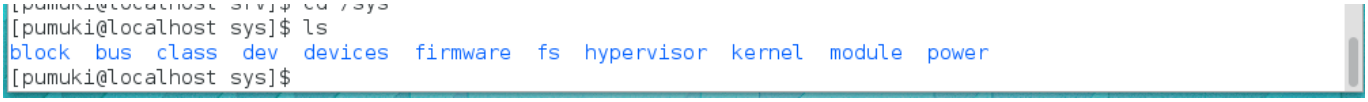
```
[pumuki@localhost sbin]$ ls
abrt-auto-reporting      fixparts                 mii-tool                 safe_finger
abrt-configuration       fsadm                    mkdict                   saned
abrtd                    fsck                     mkdosfs                  saslauthd
abrt-dbus                fsck.btrfs               mkdumprd                 sasldblistusers2
abrt-harvest-pstoreoops  fsck.cramfs              mke2fs                   saslpasswd2
abrt-harvest-vmcore      fsck.ext2                mkfs                     sedispatch
abrt-install-ccpp-hook   fsck.ext3                mkfs.btrfs               sefcontext_compile
abrt-server              fsck.ext4                mkfs.cramfs              selabel_digest
accept                   fsck.fat                 mkfs.ext2                selabel_lookup
accessdb                 fsck.minix               mkfs.ext3                selabel_lookup_best_match
accton                   fsck.msdos               mkfs.ext4                selabel_partial_match
adcli                    fsck.vfat                mkfs.fat                 selinuxconlist
addgnupghome             fsck.xfs                 mkfs.minix               selinuxdefcon
addpart                  fsfreeze                 mkfs.msdos               selinuxenabled
adduser                  fstrim                   mkfs.vfat                selinuxexeccon
agetty                   fuser                    mkfs.xfs                 selinux_restorecon
alsabat-test.sh          fxload                   mkhomedir_helper         semanage
alsactl                  gdisk                    mklost+found             semodule
alsa-info                gdm                      mkswap                   sendmail
alsa-info.sh             genhomedircon            ModemManager             sendmail.postfix
alternatives             genhostid                modinfo                  service
anaconda                 genl                     modprobe                 sestatus
```

/srv

Sirve para almacenar archivos y directorios relativos a servidores que puedas tener instalados dentro de tu sistema, ya sea un servidor web www, un servidor FTP, CVS, etc. Así, por ejemplo, en el caso de tener instalado un servidor web, sería buena idea tener el directorio web público dentro de /srv.

/sys

Al igual que /proc, contiene archivos virtuales que proveen información del kernel relativa a eventos del sistema operativo. Es en cierto modo una evolución de /proc, y a diferencia de este último, los archivos se distribuyen de forma jerárquica.



/tmp

Como ya da a entender su nombre, sirve para almacenar archivos temporales de todo tipo, ya sea de elementos del sistema, o también de diferentes aplicaciones a nivel de usuario.

Es un directorio dispuesto para almacenar contenido de corta duración, de hecho en la gran mayoría de los casos se suele vaciar de forma automática en cada reinicio del sistema. Aun así, no deben borrar su contenido de forma manual, puesto que puede contener archivos necesarios para ciertos programas o procesos que estén ejecutándose.

Las aplicaciones programadas para almacenar archivos en este directorio deben asumir que solo serán recuperables en la sesión actual. En este sentido, hay otro subdirectorio, /var/tmp, dispuesto igualmente para el almacenamiento de archivos temporales, pero cuyo contenido no se borra de forma automática tras el reinicio del sistema.

/usr

El directorio /usr viene de "User System Resources" y actualmente sirve para almacenar todos los archivos de solo lectura y relativos a las utilidades de usuario, incluyendo todo el software instalado a través de los gestores de paquetes de cada distribución.

Antiguamente /usr también contenía la carpeta particular de usuario, junto con todos sus documentos, vídeos, fotos, etc., pero más adelante se creó el directorio /home para este propósito, dejando /usr reservado para los archivos relativos a programas.

```
[pumuki@localhost usr]$ ls
bin  etc  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
[pumuki@localhost usr]$
```

/var

Contiene varios archivos con información del sistema, como archivos de logs, emails de los usuarios del sistema, bases de datos, información almacenada en la caché, información relativa a los paquetes de aplicaciones almacenados en /opt, etc. En cierto modo se podría decir que actúa a modo de registro del sistema.

```
[pumuki@localhost var]$ ls
account  cache  db    games   kerberos  local  log   nis  preserve  spool   tmp
adm      crash  empty gopher  lib       lock   mail  opt  run       target  yp
[pumuki@localhost var]$
```

Rutas relativas vs rutas absolutas

Cuando se trabaja con comandos es habitual tener que pasar como parámetros archivos o directorios. Para indicar un archivo o directorio se utiliza una ruta o path, que puede ser absoluta o relativa.

Antes de nada vamos a recalcar una cosa. Aunque todo el mundo habla de particiones cuando quiere hacer referencia al lugar donde guarda los datos, realmente los datos se guardan en el sistema de archivos que está creado dentro de la partición. Las particiones son una división del espacio de un dispositivo de almacenamiento como un disco duro o un pendrive y por si solas no pueden almacenar nada, por eso es necesario crea un sistema de archivos dentro de ellas. Este proceso de creación es lo que se llama dar formato o formatear la partición. En Windows los sistemas de archivos utilizados son FAT o NTFS. En Linux existen muchísimos sistemas, pero los más comunes son los EXT en sus versiones 3 y 4 (ext3 / ext4).

Como mencionamos anteriormente A diferencia de Windows, Linux no tiene unidades, así que se pueden olvidar de A:\, C:\, D:\ y todas las letras del alfabeto. En Linux las diferentes particiones se montan dentro de la jerarquía del file system o sistema de archivos y se referencian dentro del mismo.

Vamos con el tema de las rutas o paths. Supongamos que dentro de la partición de datos tenemos un archivo llamado pelicula.mpg. Para indicar la ruta a este archivo

se puede hacer de varias formas. Por ejemplo, la ruta absoluta a ese archivo es la siguiente:

/home/pepe/media/datos/pelicula.mpg

Las rutas absolutas se caracterizan por empezar SIEMPRE desde la raíz o root, es decir la /, y contener todos los directorios que hay desde la raíz hasta el archivo o directorio que queremos indicar, sólo es posible escribir de una forma una ruta absoluta. Todas las rutas siguientes son rutas absolutas por que empiezan desde la raíz.

/media/

/media/datos/

/etc/

/home/pepe/

/media/datos

/etc

/home/pepe/archivo.txt

Cuando tienen la / al final significa que la ruta es de un directorio, aunque no es necesaria la /. Cuando la ruta es de un archivo NUNCA se pone la / al final.

En cambio las rutas relativas NUNCA empiezan por la / y puede haber varias rutas relativas para el mismo archivo o directorio que queremos indicar. Esto es así porque la ruta relativa depende del directorio en el que se esté trabajando (donde estoy parado). Por ejemplo si estamos en el mismo directorio que el archivo pelicula.mpg, la ruta relativa es simplemente el nombre del archivo:

pelicula.mpg

En cambio, si estamos dentro del directorio media, pero fuera del directorio datos, la ruta relativa se escribiría:

datos/pelicula.mpg

Si estamos en la raíz del sistema de archivos: media/datos/pelicula.mpg

Y ¿Qué pasaría si estuviésemos dentro del directorio floppy? Aquí es necesario indicar que para llegar al archivo antes tenemos que ir al directorio padre de floppy

y después entrar en media y luego en datos. Para indicar el directorio padre se usan dos puntos y la /:

../media/datos/pelicula.mpg

Si tenemos que salir de más de un directorio se pueden poner más veces los dos puntos y la /. Por ejemplo:

../../otroarchivo.pdf

También existe el punto y la / para indicar el directorio actual de trabajo. Por eso, si estamos en un directorio donde existe un archivo llamado documento.pdf, podemos indicar la ruta relativa con sólo el nombre del archivo:

documento.pdf

O con el punto y la /:

./documento.pdf

Luego también pueden apareces cosas curiosas como por ejemplo:

./../../../../../../../../../../../../../documento.pdf

Que es lo mismo que las dos anteriores, pero evidentemente, nadie en su sano juicio hace esto.

Todo en Linux es un archivo

Como mencionamos anteriormente todo en un Linux es un archivo, tanto el Software como el Hardware, bueno el hardware es hardware, pero también es un archivo o por lo menos se puede acceder desde uno.

Desde el mouse, pasando por la impresora, el reproductor de DVD, el monitor, un directorio, un subdirectorio y un archivo de texto.

De ahí vienen los conceptos de montar y desmontar por ejemplo un CDROM.

El CDROM se monta como un subdirectorio en el sistema de archivos. En ese subdirectorio se ubicará el contenido del disco compacto cuando esté montado y, nada cuando esté desmontado.

Para ver que tenemos montado en nuestra distribución GNU/Linux, podemos ejecutar el comando “mount”.

Este concepto es muy importante para conocer cómo funciona Linux.

Como vimos anteriormente Linux el directorio /dev es donde podemos ver los dispositivos que tenemos en nuestro sistema operativo.

Algo interesante para hacer es ver información de nuestros dispositivos mediante los archivos que podemos encontrar en /proc , para ello utilizaremos el utilitario llamado "cat"

```
[pumuki@localhost var]$ cd /proc
[pumuki@localhost proc]$ ls
1     2154  2692  2991  3147  34    400   587  66   945        devices      kpageflags    stat
10    22    2760  2996  3149  3401  401   588  660  950        diskstats    loadavg       swaps
104   2257  2765  3     3150  3405  402   589  661  951        dma          locks         sys
11    2258  278   30    3154  3406  403   590  662  953        driver       mdstat        sysrq-trigger
12    23    280   3000  3197  3431  42    591  663  957        execdomains  meminfo       sysvipc
1251  2366  281   3003  32    3542  43    592  668  965        fb           misc          timer_list
13    2387  283   3012  3202  3604  44    593  669  967        filesystems  modules       timer_stats
15    24    2857  3014  3224  368   4489  6    67   968        fs           mounts        tty
1504  2447  286   3019  3230  369   4490  618  670  99         interrupts   mtrr          uptime
158   2475  2880  3030  3244  378   45    629  673  acpi       iomem        net           version
17    25    2885  3071  3256  379   4503  634  680  asound     ioports      pagetypeinfo  vmallocinfo
18    2516  2888  3076  3280  394   4548  642  693  buddyinfo  irq          partitions    vmstat
19    2619  2909  3081  3290  395   46    643  7    bus        kallsyms     sched_debug   zoneinfo
2     2667  2921  3086  3296  3951  473   644  700  cgroups    kcore        schedstat
20    2680  2939  3091  33    396   496   645  705  cmdline    keys         scsi
21    2682  2955  3096  3343  397   499   649  721  consoles   key-users    self
2149  2684  296   31    3348  398   501   655  8    cpuinfo    kmsg         slabinfo
2153  2691  297   3107  3363  399   586   657  9    crypto     kpagecount   softirqs
[pumuki@localhost proc]$
```

```
[pumuki@localhost proc]$ cat cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 58
model name      : Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz
stepping        : 8
microcode       : 0x19
cpu MHz         : 2893.446
cache size      : 4096 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clfl
sse2 ht syscall nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc pni pclmulqdq ss
se4_2 x2apic popcnt aes xsave avx rdrand hypervisor lahf_lm
bogomips        : 5786.89
clflush size    : 64
cache_alignment : 64
address sizes   : 36 bits physical, 48 bits virtual
```

Es muy importante mencionar que Linux es un sistema operativo “Case Sensitive”. Esto quiere decir que al contrario de Windows, en Linux las letras mayúsculas y minúsculas son interpretadas de forma diferente. Esto implica que un directorio o un archivo llamado “Pepe” es diferente a uno llamado “pepe”, “pePe”, “pEpe”, etc...

Permisos de Archivos en Linux

Una de las cosas más importantes en GNU/Linux son los permisos de los archivos. Comprender esto debería considerarse como algo esencial. Se suele creer que es algo complicado, pero veremos que no es así, y trataremos de verlo de manera sencilla y clara.

El sistema de permisos que utiliza GNU/Linux está directamente tomado de los permisos de archivos que utiliza UNIX. Un sistema potente que apenas ha sufrido variaciones durante todos estos años y que sigue siendo totalmente vigente. Por tanto esto hace pensar que es algo importante, y bien hecho.

Entender el sistema de archivos de GNU/Linux no es algo que sea para gurús, ni algo que te vaya a llevar varios días de estudio. Es un sistema sencillo que trataremos de aprender a usarlo.

Lo primero que deben comprender es que todos los archivos en Linux pertenecen a un grupo de usuarios y a un usuario en particular.

También que a su vez los usuarios en UNIX/Linux pertenecen a grupos y que los permisos se asignan tanto a grupos como a usuarios.

Los usuarios pueden pertenecer a uno o más grupos y por ende tendrás diferentes permisos sobre los archivos/directorios.

Existen tres tipos de permisos que se pueden aplicar, estos son:

Lectura: otorga al grupo el permiso de poder leer el archivo. Se indica con la letra r (inicial de Read, que significa leer en inglés)

Escritura: otorga al grupo el permiso de poder editar el archivo pudiendo escribir en él. Se indica con la letra w (inicial de Write, que significa escribir en inglés)

Ejecución: otorga al grupo el permiso de poder ejecutar el archivo. Se indica con la

tecla x (que viene de la palabra Execute, que significa ejecutar en inglés)

Para entender mejor cómo esto es aplicado a un grupo, podrían, por ejemplo, darle a un grupo de usuarios el permiso de leer y escribir en un archivo, pero no la capacidad de poder ejecutarlo.

O podrían darle los permisos de poder leer y ejecutar un archivo, pero no de poder modificarlo. Incluso pueden darle a un grupo todos los permisos de lectura, escritura y ejecución de un archivo, o ningún permiso quitándoselos todos.

Visto el tema de los tres permisos, veamos ahora el tema de los grupos, del que has estado leyendo.

Los grupos del archivo a los que nos podremos referir son cuatro:

- **Usuario: el propietario actual del archivo (nos referimos a este grupo con la letra u)**
- **Grupo: un grupo de usuarios de un archivo (nos referimos a este grupo con la letra g)**
- **Todos: todos los usuarios (nos referimos a este grupo con la letra a de all, que significa todos en inglés)**

En términos generales, sólo tendremos que trabajar con los tres primero grupos. El grupo de usuarios todos es sólo utilizado como un atajo (después veremos qué quiero decir con esto).

Si abren una terminal, y en la línea de comandos, escriben el comando "ls –l", verán un listado de todos los archivos y directorios que hay dentro del directorio de trabajo actual en el que nos encontremos.

```
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

Los permisos se agrupan en grupos de 3 letras, las primeras 3 posiciones corresponden a los permisos del owner (creador), las siguientes 3 al grupo y los últimos 3 al resto del mundo...

La primera columna es lo que nos brinda la información sobre los permisos, verán que hay muchos directorios (comienzan con la letra d y a continuación los permisos

d rwx r-x r-x

En el caso de Descargas por ejemplo, se interpreta de la siguiente manera:

Es un Directorio

Read, Write , Execute para el dueño.

Read , Execute para el grupo

Read , Execute para el resto del mundo.

También verán que hay un archivo llamado prueba (tiene un guión en vez de la d inicial) y los siguientes permisos:

- **rw- rw– rw– Se interpreta de la siguiente manera:**

NO es un Directorio

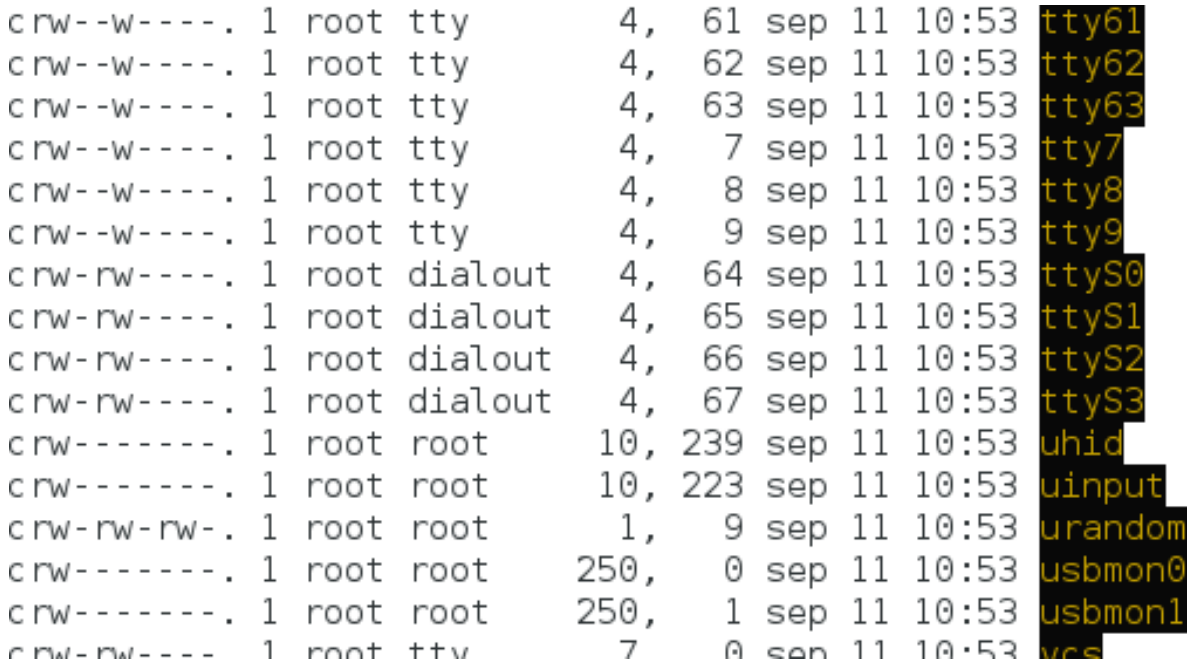
Read, Write para el dueño.

Read , Write para el grupo

Read , Write para el resto del mundo.

Luego verán que hay 2 columnas, que dicen pumuki y pumuki. Esto es el usuario y grupo, que en este caso se llama igual porque al crear un usuario siempre se crea un grupo con el mismo nombre.

Aquí un ejemplo donde no son lo mismo:



Con lo explicado hasta ahora ya sabemos que en el ejemplo del archivo que se muestra en la captura el usuario tiene otorgados los permisos de lectura y escritura, el grupo al que pertenece el usuario tiene otorgado el permiso de lectura, escritura y otros tienen también permiso lectura y escritura.

Si alguno de los grupos tuviera el permiso de ejecutar ese archivo (en el caso de que se pudiera porque fuera un script) entonces como hemos visto estaría representado con una x. ¿Sencillo, verdad?

Manipulando los permisos

Para añadir, o quitar permisos a un archivo se utiliza el comando "chmod". Para otorgar o quitar derechos podremos utilizar tanto las letras referidas a permisos y grupos que hemos visto como las equivalencias numéricas, lo que nos sea más fácil o lo que necesitemos en cada momento.

Y ahora veamos cómo poder manipular esos permisos. Lo primero aclarar que para poder manipular los permisos de un archivo debes ser el propietario de ese archivo o debes tener el permiso de poder editar el archivo, o tener acceso de superusuario con poderes totales (recuerden que un gran poder lleva una gran responsabilidad).

Sigamos con nuestro archivo del ejemplo, y supongamos que ahora le llamaremos

script.sh y que es un script (secuencia de comandos con una lógica dentro de un archivo) escrito en bash y necesita poder ejecutarse... pero que sólo quieres darte a ti mismo permiso de ejecución.

```
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 16:03 script.sh
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

Con todo lo que han leído quizás piensen "necesito añadir una x de ejecución al primer grupo que es el usuario, o sea, yo mismo." Correcto, ahora veamos cómo añadir esa x a nosotros mismos mediante la línea de comandos. Para ello tan simple como:

chmod u+x script.sh

```
[pumuki@localhost ~]$
[pumuki@localhost ~]$ chmod u+x script.sh
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
-rwxrw-r--. 1 pumuki pumuki  0 sep 11 16:03 script.sh
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

Así de simple. Indicándole con la u de user que queremos añadir el permiso de ejecución con la x al archivo. Pruébenlo con un archivo suyo, y verán que no es difícil de entender y de ejecutar.

Ahora hagámoslo un poco más interesante. Supongamos que por alguna razón un archivo tiene el permiso de ejecución para todos los grupos algo parecido a: -rwx-r-x-r-x. Si quieres quitar el permiso de ejecutarlo al grupo de otros tan sencillo como

chmod ugo-x script.sh

```
[pumuki@localhost ~]$ chmod ugo-x script.sh
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 16:03 script.sh
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

Efectivamente ugo=usuario/grupo/otros y con -x eliminas este permiso. Otra manera de hacerlo sería con el siguiente atajo (¿te acuerdas lo que comenté al inicio del artículo?):

chmod a-x script.sh

```
[pumuki@localhost ~]$ chmod a-x script.sh
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 16:03 script.sh
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

Tengan cuidado al quitar o dar permisos a un archivo si utilizan este método.

Permisos a directorios

También puedes otorgar o quitar permisos a directorios con el comando “chmod”. Cuando creas un nuevo directorio como usuario, lo normal es crearlo con los siguientes permisos

drwxrwxr-x

Cabe señalar que los archivos que creemos dentro del directorio no tienen por qué heredar estos permisos, pudiendo tener otros distintos.

Como ya hemos visto, la d inicial indica que es un directorio. Como puedes ver en este ejemplo tanto el usuario como el grupo tienen plenos poderes sobre el directorio. Pero supongamos que dentro de un directorio queremos darles a todos los archivos que contiene unos permisos iguales para eso añadiremos al comando

chmod el parámetro -R que indica que efectuará de manera recursiva la asignación de permisos que estipulemos.

Imaginemos que tenemos un directorio llamando Pruebas con un montón de scripts dentro. Todos esos archivos y el directorio en sí tiene los siguientes permisos -rwxrwxr-x. Si queremos quitar al grupo el permiso de escritura tendremos que utilizar el siguiente comando:

```
chmod -R g-w TEST
```

Es decir de manera recursiva (-R) al grupo (g) le quitaremos el permiso de escribir (-w) en los archivos que contiene el directorio (Pruebas).

Equivalente numérico

Hagámoslo ahora un poco más complejo (no se asusten, verán que no es para tanto). Cada permiso que hemos expresado con letras, también puede ser representado con números, lo que en algún caso nos puede resultar útil. Estas equivalencias son:

lectura = 4

escritura = 2

ejecución = 1

En el ejemplo del archivo que hemos visto anteriormente, la sustitución de las letras de los permisos por números sería la siguiente:

```
- 42- 4-- 4--
```

Ahora podemos sumar el número de los distintos grupos entre sí. Los permisos del usuario sería 4+2 lo que da 6 (para el owner). Los permisos de grupo y otros son simplemente 4, por tanto no hay nada que sumar. El equivalente numérico quedaría tal que así:644

Enlaces físicos

Un enlace físico no es más que una etiqueta o un nuevo nombre asociado a un archivo. Es una forma de identificar el mismo contenido con diferentes nombres. Éste enlace no es una copia separada del archivo anterior sino un nombre diferente para exactamente el mismo contenido.

Para crear un enlace físico en Linux del archivo "prueba" a "prueba_ln" ejecutamos:

```
$ ln prueba prueba_ln
```

El enlace aparecerá como otro archivo más en el directorio y apuntará al mismo contenido de "prueba". Cualquier cambio que se haga se reflejará de la misma manera tanto para "prueba" como para "prueba_ln".

Un enlace se puede borrar usando el comando "rm" de la misma manera en que se borra un archivo. Esto puede tener varias ventajas, pero también puede complicar la tarea de seguimiento de los archivos. Un enlace físico tampoco puede usarse para hacer referencia a directorios o a archivos en otros equipos.

Enlaces simbólicos

Un enlace simbólico también puede definirse como una etiqueta o un nuevo nombre asociado a un archivo pero a diferencia de los enlaces físicos, el enlace simbólico no contiene los datos del archivo, simplemente apunta al registro del sistema de archivos donde se encuentran los datos. Tiene mucha similitud a un acceso directo en Windows o un alias en OS X.

Para crear un enlace simbólico del archivo "prueba" a "prueba_ln", ejecutamos:

$ ln -s prueba prueba_ln

```
[pumuki@localhost ~]$ ln -s prueba prueba_ln
[pumuki@localhost ~]$ ls -l
total 0
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Descargas
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Documentos
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Escritorio
drwxr-xr-x. 2 pumuki pumuki 60 sep  2 17:12 Imágenes
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Música
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Plantillas
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 12:17 prueba
lrwxrwxrwx. 1 pumuki pumuki  6 sep 11 16:38 prueba_ln -> prueba
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Público
-rw-rw-r--. 1 pumuki pumuki  0 sep 11 16:03 script.sh
drwxr-xr-x. 2 pumuki pumuki  6 sep  2 17:06 Vídeos
[pumuki@localhost ~]$
```

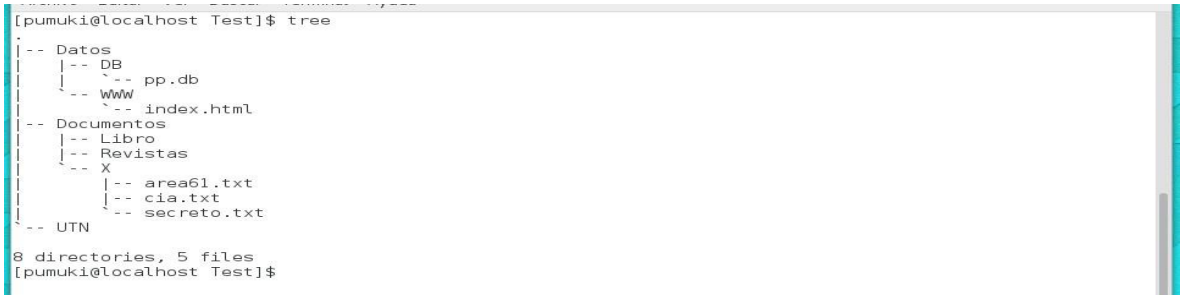
Este enlace también aparecerá como otro archivo más en el directorio y apuntará al mismo contenido de "prueba", reflejando todos los cambios que se hagan tanto para "prueba" como para "prueba_ln".

Sobre un enlace simbólico también se pueden usar todos los comandos básicos de archivos (rm, mv, cp, etc). Sin embargo cuando el archivo original es borrado o movido a una ubicación diferente el enlace dejará de funcionar y se dice que el enlace está roto.

Un enlace simbólico permite enlazar directorios y, también permite enlazar archivos fuera del equipo. En un principio puede parecer complicado, pero luego de leer detalladamente seguro que tendrás más claro cuándo usar un enlace simbólico y cuándo usar uno físico.

Ejercicios

Dada la siguiente estructura de directorios y archivos, escribe la ruta absoluta de todos los archivos visibles. Escribe después la ruta relativa a todos los directorios vacíos tomando como directorio de trabajo UTN.



Usando la estructura de directorios anterior crear un link simbólico del archivo area61.txt tomando como directorio de trabajo UTN.

Usando la estructura de directorios anterior crear un link físico del archivo cia.txt tomando como directorio de trabajo Test.

Estimados alumnos

Con motivo de un control de las unidades expuestas, les solicitamos que en caso de alguna incoherencia, error o inconvenientes de lectura, nos haga saber a través del foro de la unidad.

Bibliografía Recomendada

Aprendiendo a Aprender Linux

Vladimir Támara/Jaime Irving Dávila/Pablo Chamorro/Igor Támara Febrero de 2003

Consultas y/o dudas a través de la plataforma o por mail al Instructor.
Las mismas serán respondidas en un plazo no mayor de 72 horas.