Curso Básico de Linux

Sistema de archivos

Contenido

Archivos....1
Directorios....2
Nombres de archivo....2
Directorio propio, directorio actual....3
Rutas absolutas y relativas....3
Crear y borrar directorios....4
Tipos de archivo....4
Propiedad de los archivos....5
Permisos de archivos....5
Todo es un archivo....6
Los directorios de Linux....7
Referencias, lecturas complementarias....8

Los sistemas Unix/Linux funcionan enteramente en base a archivos. Los programas, los datos, los directorios y aún los controladores de dispositivos tales como discos, modems e impresoras, son o se visualizan y se manejan como archivos. Esta concepción uniformiza y simplifica la visión del operador, dejando a cargo del sistema el manejo de las distintas formas organizar los datos y la forma de soportar esos datos en dispositivos de hardware,

Un sistema de archivos es una forma de organización de datos. Esto requiere una estructura interna propia de cada sistema operativo. Un mismo sistema operativo puede soportar diversas formas de organización de estos datos. Las formas de organización de datos más habitualmente usadas en Linux son ext2, ext3 y ext4. En MS Windows es usual NTFS, y anteriormente VFAT y FAT32.

El sistema de archivos en Unix/Linux se organiza en una jerarquía de directorios cuyo punto de partida o directorio raíz se designa con /. Debajo de este directorio principal están todos los demás directorios, sean del sistema o de los usuarios, estén en tal o cual dispositivo externo, interno o aún remoto en otra máquina. Este capítulo describe el árbol de directorios en Linux.

Archivos

Un *archivo* o también *fichero* es un conjunto de datos almacenable en algún dispositivo informático como la memoria interna, un disco, un pendrive, un CD, una tarjeta SD o cualquier otro elemento capaz de retener información digital codificada como ceros y unos. Los dígitos 0 y 1 son llamados *dígitos binarios*.

Cada archivo tiene asociado un nombre, un contenido, un lugar de ubicación en la jerarquía de directorios, e información de tipo administrativo tal como fechas, tamaño, usuario o dueño del archivo. El contenido de bajo nivel de un archivo son siempre *bits*, o sea ceros y unos, pero esos bits pueden resultar de codificaciones diferentes, i.e. representar diferentes cosas. Un cierto conjunto de bits puede así corresponder a una caracter alfabético o numérico, a unas instrucciones en lenguaje de máquina, a números correspondientes a localizaciones de memoria, u otros. Por encima de este

nivel básico, estos conjuntos de datos pueden corresponder a textos, programas fuente escritos como texto, programas ejecutables en binario (bits interpretables como instrucciones de máquina), imágenes, sonido, u otros.

Estructuralmente un archivo es una secuencia de bytes. Un *byte* es una unidad de información consistente en una secuencia de bits, generalmente 8. En informática, cuando se habla de datos, se alude a conjuntos de bytes. Un archivo es entonces un conjunto de bytes almacenados en algún tipo de dispositivo físico capaz de conservar los 0s y 1s; a este conjunto se lo reconoce por un nombre, y se lo maneja como una entidad única (ese conjunto de bytes).

Directorios

Los archivos se agrupan en directorios. Un directorio es un contenedor de archivos. Más específicamente, es una entidad capaz de contener una lista de nombres de archivo e información acerca de los mismos. En definitiva, un directorio es también un conjunto de bytes, en este sentido, también un archivo. Un directorio puede concebirse como una localización capaz de contener otros directorios o archivos. Dos archivos que se encuentren en distinto directorio pueden tener el mismo nombre sin confundirse.

El comando `ls` permite listar el contenido de un directorio:

```
ls /var
```

el argumento es un directorio; la salida son los nombres de archivos y subdirectorios en ese directorio.

```
ls nota.txt
```

el argumento es un archivo, la salida es el nombre del archivo.

```
ls -l /var
```

muestra los archivos y subdirectorios contenidos en `/var` en formato largo.

```
ls -ld /var
```

muestra características del propio directorio `/var` en lugar de los archivos y subdirectorios contenidos en él.

Nombres de archivo

Para el nombre de archivo se puede usar prácticamente cualquier caracter, incluso caracteres no visibles. Los caracteres `$ ; \ & ! * |` son interpretados por el shell, y por lo tanto su uso en los nombres de archivos tiende a causar confusión, es mejor evitarlos. Para los nombres de archivo, se aconseja comenzar con una letra o número y usar solamente letras, números, punto, guión o signo de menos `-` y subraya o guión bajo _. Linux diferencia mayúsculas de minúsculas: el archivo `NOTA.TXT` es distinto del archivo `nota.txt` y también de `Nota.txt` o `Nota.Txt`. No se distingue entre nombre de archivo y extensión: `nota.nueva.txt`, `LCK..modem`, `nota.nueva`, `555`, `m`, son todos nombres de archivo válidos, con o sin extensión.

Un archivo que comienza por un punto no es visible:

```
touch .noseve
ls
ls -a
```

`ls` no lo muestra, pero sí `ls -a`.

```
rm .noseve
```

lo borra como a cualquier archivo.

Directorio propio, directorio actual

Al ingresar al sistema cada usuario entra en su directorio propio, un directorio privado que no es tocado por el sistema ni por los otros usuarios. El directorio en el cual se encuentra posicionado el usuario en un momento dado se denomina su *directorio actual* o *directorio de trabajo*.

```
cd /usr/bin
pwd
```

cambia al directorio `/usr/bin`; el directorio actual es ahora `/usr/bin`.

```
cd
pwd
```

devuelve al usuario a su directorio propio, que es ahora el directorio actual.

```
echo $HOME
```

nuestra el nombre del directorio propio. HOME es una variable de ambiente que contiene el nombre del directorio propio del usuario.

```
cd $HOME
cd ~
```

también llevan al usuario a su directorio propio. El caracter `~` es una forma abreviada de indicar el directorio propio del usuario.

Rutas absolutas y relativas

Un directorio puede contener otros directorios así como archivos ordinarios, lo que genera una jerarquía o árbol de directorios. El directorio superior o *directorio raíz* se denomina `/`.

```
cd /
pwd
```

lleva al directorio raíz.

```
cd
```

vuelve al directorio propio del usuario.

```
cd -
```

vuelve al directorio actual anterior, en este caso el raíz.

```
ls -l
```

muestra el contenido del directorio raíz.

El camino para llegar a un directorio o a un archivo se denomina *ruta* (en inglés *path*). Una *ruta* es una secuencia de directorios separados por `/`. Una *ruta absoluta* comienza en el directorio raíz e indica el camino completo para llegar a un archivo o directorio.

```
ls -l /var/lib/dpkg/available
```

indica cómo llegar al archivo `available` desde el directorio raíz; es una ruta absoluta.

Una *ruta relativa* comienza en el directorio de trabajo (directorio actual) e indica el camino para llegar a un archivo o directorio desde ese directorio actual.

```
cd /var/lib
ls -l dpkg/available
```

indica el camino para llegar al archivo `available` desde el directorio `/var/lib`; es una ruta relativa.

Caracteres indicativos de directorios:

- `.` un punto, el directorio actual
- `..` dos puntos, el directorio superior al actual
- `~` tilde, el directorio propio del usuario.

```
cd
ls -l .
```
muestra los archivos del directorio actual; en este caso el punto puede omitirse.
```
ls -l ..
```
muestra el contenido del directorio superior, entre los cuales se encuentra el directorio actual.

Crear y borrar directorios

Los siguientes comandos realizan cambios en la jerarquía de directorios.

```
cd /tmp
mkdir nuevo.dir
```
crea un nuevo directorio.

```
rmdir nuevo.dir
```
borra un directorio existente; actúa sólo sobre directorios vacíos.

```
mkdir dir1
mkdir dir1/dir2
touch dir1/dir2/arch2 dir1/arch1
ls -lR dir1
```
muestra recursivamente todos los archivos y directorios creados bajo `dir1`.

```
rmdir dir1
rmdir dir1/dir2
```
no borra ninguno de los dos directorios, no están vacíos.

```
rm -r dir1
```
borra recursivamente el directorio y todos los archivos y subdirectorios que pueda contener.

Observación: la opción para recursivo en los comandos `ls` y `rm` son diferentes, en un caso `-R` en otro `-r`. Sin embargo, ambos comandos soportan `--recursive`. Las opciones largas requieren más digitación, pero son menos ambiguas y más fáciles de recordar.

Tipos de archivo

Un archivo es una secuencia de bytes. Un byte equivalente a un caracter. Los sistemas operativos Unix/Linux no imponen ninguna estructura interna a los archivos, no asignan significado a su contenido, ni a su nombre. El significado de los bytes contenidos en un archivo depende totalmente de los programas que interpretan el archivo.

En los sistemas Unix/Linux nunca hay en un archivo ningún byte que no haya sido colocado por el usuario o un programa. No hay caracter de fin de archivo. El núcleo del sistema Unix se mantiene al tanto del tamaño de los archivos por contadores, sin introducir ningún caracter especial. Los cambios de línea se indican con un caracter especial, LF o nl, según la denominación, llamado *nueva línea*.

¿Qué hay dentro de un archivo? El núcleo no puede decirnos nada del tipo de archivo: no lo conoce. Todos los archivos tienen la misma estructura interna. El comando `file` hace una conjetura: lee algunos bytes y busca indicios sobre el tipo de contenido. En los sistemas Unix/Linux hay sólo una clase de archivo, una secuencia de bytes. Para accederlo basta con su nombre, no es preciso ninguna información adicional. Luego, el programa o el usuario deberán interpretarlo.

```
file /bin/nano
file /usr/share/doc/nano/README
file /usr/lib/python2.7/os.py
file /usr/sbin/adduser
file /etc/init.d/umountfs
```
detecta archivos de tipo ejecutable binario, texto, scripts en Python, Perl y Bash.

Propiedad de los archivos

Cada usuario es dueño de los archivos creados por él, hasta que los borre o los ceda a otro usuario. Un usuario pertenece a uno o varios grupos, y puede compartir archivos con otros usuarios que integren alguno de esos grupos. Cada archivo está asignado a un grupo de usuarios, al cual debe pertenecer su dueño. El comando `ls -l` muestra dueño y grupo de los archivos listados.

Si bien los usuarios y los grupos se presentan con nombres, el sistema maneja números como identificadores de usuarios y grupos. A cada usuario corresponde un identificador de usuario llamado UID (User ID); a cada grupo corresponde un identificador de grupo llamado GID (Group ID). Este esquema permite cambiar los nombres de usuarios y grupos sin afectar los archivos y directorios asignados a esos usuarios o grupos. Lo identificadores UID y GID se pueden ver con el comando `id`.

```
id          # UID del operador y GIDs de sus grupos
id nombre   # UID del usuario indicado y GIDs de sus grupos
```

Permisos de archivos

Cada archivo tienen un conjunto de permisos asociados; estos permisos determinan qué se puede hacer con ese archivo y quién o quiénes pueden hacerlo.

Los permisos de un archivo se indican con 10 caracteres:
- 1 caracter para tipo de archivo,
- 3 caracteres [rwx] para permisos del dueño,
- 3 caracteres [rwx] para permisos del grupo,
- 3 caracteres [rwx] para permisos de otros.

Caracter para tipo de archivo:
- `d` directorio
- `l` enlace simbólico
- `-` archivo normal
- `b` archivo controlador de dispositivo orientado a bloques
- `c` archivo control de dispositivo orientado a caracteres

Caracteres de permisos:
- `r` acceso de lectura (read)
- `w` acceso de escritura (write)
- `x` acceso de ejecución (execute)

El significado del permiso `x` varía según se trate de archivos o directorios.

Permiso	Archivos	Directorios
`r`	leer archivos	ver contenido de directorios
`w`	escribir en un archivo	crear y borrar archivos
`x`	ejecutar como programa	ingresar a un directorio

–	sin derechos	sin derechos

Permisos usuales al crear archivos y directorios:
archivo `rw-rx--r--` archivo: dueño `rwx`, grupo `r--`, otros `r--`.
directorio `rwxr-xr-x` directorio: dueño `rwx`, grupo `r-x`, otros `r-x`.

El ingreso a un directorio (permiso `x`) permite ver o ejecutar un archivo contenido en ese directorio, o trasladarse a ese directorio; para estas operaciones no es necesario poder ver los nombres de los archivos contenidos (permiso `r`). Para crear o borrar archivos en un directorio se requiere el permiso `w`.

```
cd /tmp
mkdir dir1
echo "Archivo en directorio" > dir1/untexto.txt
```
crea un directorio y un archivo dentro de él.
```
ls -ld dir1
cat dir1/untexto.txt
```
muestra los permisos del directorio, verifica acceso al archivo creado.
```
chmod a-x dir1
```
quita permisos de acceso al directorio a todos.
```
cat dir1/untexto.txt
```
el contenido del archivo se ha hecho inaccesible.
```
chmod a+x dir1
cat dir1/untexto.txt
```
repone permiso de acceso al directorio, recupera la visibilidad del contenido del archivo.

```
chmod a-w
ls -ld dir1
```
quita permiso de modificación al directorio.
```
rm dir1/untexto.txt
```
no puede borrar el archivo del directorio sin permisos de modificación.
```
chmod ug+w
rm dir1/untexto.txt
ls -l dir1
```
recupera permisos de modificación para el dueño y el grupo, borra el archivo.

Un archivo con comandos se declara ejecutable dándole permiso de ejecución `x`. Esto permite ejecutarlo por su nombre indicando su ruta, ya sea absoluta o relativa. Los comandos de Linux son archivos ejecutables. Un archivo sin permiso `r` no permite ver su contenido. Un archivo sin permiso `w` no se puede modificar.

```
echo "Archivo de contenido oculto" > opaco.txt
ls -l opaco.txt
cat opaco.txt
```
crea un archivo, muestra sus permisos, luego su contenido.
```
chmod a-r opaco.txt
ls -l opacto.txt
```
quita permisos de lectura, muestra los permisos.
```
 cat opaco.txt
```
da error de lectura, no se puede ver el contenido del archivo.
```
chmod ug+r opaco.txt
ls -l opaco.txt
cat opaco.txt
```

repone permisos de lectura para usuario (dueño) y grupo, muestra los permisos, muestra el contenido, ahora nuevamente visible.

Todo es un archivo

El árbol de directorios de Linux puede contener miles de archivos, directorios, y referencias a otros archivos y directorios (enlaces). En Unix/Linux existe el eslogan "todo es un archivo", porque muchas entidades se modelan como archivos sin realmente serlo, sin ser específicamente un conjuntos de datos en un soporte permanente tal como un disco o un pendrive. Los distintos dispositivos de hardware tanto de almacenamiento como de comunicaciones se modelan como si fueran archivos en el directorio `/dev`, así como diversos canales de información del kernel se modelan como archivos en el directorio `/proc`. Un disco o cualquier otro soporte de datos se integra al árbol de directorios como un directorio más, lo cual oculta los distintos dispositivos donde se almacena información, y el usuario solo ve el árbol de directorios, sin saber dónde residen físicamente los datos. En los sistemas grandes, el árbol de directorios abarca aún soportes de datos en otras máquinas de la red local, o en algunas ocasiones en máquinas más remotas.

Esta abstracción propuesta por el sistema de archivos tiene como grandes virtudes no solo ocultar al usuario una estructura de hardware potencialmente compleja, sino también uniformizar la forma de interactuar con los diferentes dispositivos: desde el punto de vista del usuario se leen datos de la misma forma ya se trate de un disco, un pendrive, una tarjeta SD, una tarjeta de red Ethernet, una antena de WiFi, una cámara digital, un micrófono o un sensor de temperatura. Esta facilidad exige la existencia de "*device drivers*", *manejadores de dispositivos* o *controladores*, programas de software específicos para cada dispositivo de hardware. cuyo propósito es exponer ese dispositivo ante el sistema operativo tal como se lo espera ver, o sea, como si fuera un archivo.

Los directorios de Linux

El árbol de directorios de Linux responde a un estándar denominado *Linux Filesystem Hierarchy Standard*, aunque puede haber diferencias menores entre las distintas distribuciones. El árbol de directorios puede explorarse libremente; el sistema de permisos de Linux impide a un usuario alterar los archivos del sistema o los de otros usuarios. El siguiente resumen reúne los directorios más usuales.

`/`	el directorio raíz
`/bin`	archivos binarios ejecutables, comandos de usuario
`/boot`	kernel de Linux, imagen de disco para el arranque, cargador de arranque
	Archivos importantes:
	`/boot/grub/grub.conf o menu.lst` configuración cargador arranque
	`/boot/vmlinuz-4.8.0-53-generic` o similar, el kernel de Linux
`/dev`	dispositivos, se presentan como un archivo
`/etc`	configuración del sistema, scripts en lenguaje del shell, textos con comandos
`./init.d`	scripts de arranque y detención de programas
`./rc?.d`	enlaces a scripts para el arranque del sistema en diferentes niveles
`./skel`	archivos de inicialización para nuevos usuarios
`/home`	directorios propios de los usuarios
`/lib`	bibliotecas compartidas usadas por los programas del núcleo del sistema
`/lost+found`	rescate de archivos ante fallas graves; cada partición tiene uno de estos
`/media`	puntos de montaje de dispositivos removibles, e.g. discos o pendrives USB
`/mnt`	para el montaje manual de dispositivos removibles
`/opt`	ubicación de software opcional o comercial no incluido en la distribución

`/proc`	sistema de archivos virtual del kernel, con información del sistema
`/root`	directorio propio de la cuenta de usuario del supervisor, llaamado `root`
`/run`	datos de procesos en ejecución
`/sbin`	ejecutables binarios para administración del sistema, para el supervisor
`/tmp`	archivos temporales generados por programas, se borra al apagar la máquina
`/usr`	todos los programas y archivos de soporte para usuarios del sistema
`/usr/bin`	ejecutables instalados por la distribución
`/usr/lib`	bibliotecas compartidas de programas en `/usr/bin`
`/usr/local`	programas fuera de la distribución pero de uso a nivel de sistema
`/usr/sbin`	más programas de administración del sistema
`/usr/share`	bibliotecas datos compartidos de los programas en `/usr/bin`
`/usr/share/doc`	documentación de todos los paquetes instalados
`/usr/share/man`	las páginas man, comprimidas
`/var`	datos que cambian, relativos a diversos programas
`/var/log`	registros de la actividad del sistema, muchos reservados al supervisor

Referencias, lecturas complementarias

- Comandos: `id ls cat cd chmod echo file mkdir pwd rm rmdir touch`
- Wikipedia, artículos principales: File system.
