Manejar los sistemas de archivos

1. Definición básica

a. Bloque

El bloque es la unidad básica, atómica, de almacenamiento del sistema de archivos. Un archivo ocupa siempre un número entero de bloques. Así, si el tamaño de un archivo es de un byte y el bloque en el que está tiene un tamaño de 4096 bytes, se desperdician 4095 bytes. De esta manera, es posible llenar un sistema de archivos con n archivos de 1 bytes, donde n representa el número de bloques, mientras que ¡el volumen total de los datos sólo es de n bytes!

Supongamos un disco que contiene 102.400 bloques de 4096 bytes. Su tamaño total es de 400 MB. Supongamos 102.400 archivos de 384 bytes. El tamaño total de los datos es de 37,5 MB. Ahora bien, el disco está lleno, ¡ya que todos los bloques están siendo utilizados! Hay 362,5 MB perdidos. Por lo tanto, es muy importante tener cuidado con el tamaño de bloques, sobre todo si los archivos que se van a almacenar son de pequeño tamaño.

Algunos comandos calculan el tamaño de los archivos en bloques, como du, df o find. Históricamente, el tamaño de un bloque era de 512 o 1024 bytes. Es una unidad cuyo valor puede cambiar según el comando, y a menudo sin relación con el tamaño de los bloques del sistema de archivos en el cual se trabaja. Se impone la prudencia.

b. Superbloque

Cada sistema de archivos dispone de al menos un superbloque (superblock en inglés). Un superbloque es una zona de metadatos que contiene varios datos sobre el sistema de archivos:

- su tipo;

- su tamaño;
- su estado;
- información (posición) en las demás zonas de metadatos (otros superbloques, tabla de inodos, etc.).

Linux intenta primero leer el superbloque primario, el primero del disco. Puede ocurrir que este último esté corrompido después de operaciones erróneas o una avería. En este caso, ya no se puede acceder a los datos del disco (es imposible saber, por ejemplo, dónde están los inodos). Un sistema de archivos de Linux dispone de copias (backups) de los superbloques en varios sitios del disco. Como las escrituras en los diversos superbloques son síncronas, son todos idénticos. Como último recurso, si se suprime uno de ellos, se puede volver a copiar desde otro.

A continuación verá cómo disponer de toda la información en un sistema de archivos de tipo ext (de 2 a 4).

c. Tabla de inodos

La palabra **inodo** es la contracción de "índice nodo", o sea nodo de índice. Es una estructura de datos que contiene la información que describe y representa un archivo. Son los **atributos** de un archivo. Cada archivo dispone de un número de inodo (i-number). Todos los inodos están presentes dentro de una tabla de inodos. Se suele repartir partes de esta tabla después de cada superbloque. Una tabla de inodo forma parte de los metadatos.

Un archivo sólo puede tener un único inodo. Un inodo es único dentro de un único sistema de archivos. Cada sistema de archivos dispone de una tabla de inodos independiente. Si el archivo titi lleva el número de inodo 12345 en un primer sistema de archivos, aunque pepito lleve el número de inodo 12345 en otro, estos archivos no tiene relación entre sí.

Sin embargo, veremos más adelante que se puede asignar el mismo número de inodo a dos nombres de archivos dentro de un mismo sistema de archivos. Estos dos nombres representan entonces un solo y mismo archivo.

Contenido

El contenido de un inodo cambia de un sistema de archivos a otro, pero la norma POSIX impone que cada uno de ellos disponga de, al menos, los atributos siguientes para cada

archivo:

- tamaño;
- identificador del periférico que lo contiene;
- propietario;
- grupo;
- número de inodo;
- modo (permisos) de acceso;
- fecha de última modificación de inodo (change time), no confundir con la fecha de creación;
- fecha de última modificación de contenido (modification time);
- fecha de último acceso (access time);
- contador de hard links (vínculos físicos o duros, ver más adelante).

Puede obtener datos sobre un inodo con el comando **stat**.

Un inodo no contiene el nombre del archivo ni información acerca de la creación de un archivo. Algunos sistemas de archivos disponen de atributos, como crtime, que añaden esta información, pero estos no son POSIX.

Aunque no sea POSIX, la fecha de creación **crtime** está almacenada en los sistemas de archivos XFS, BTRFS o ext4. Si dispone de un kernel 4.12 o superior, una biblioteca C glibc 2.28 o superior, y los coreutils 8.31 o superior, la información se encuentra directamente accesible con el comando **stat**, como es el caso es un sistema Fedora 31, por ejemplo:

```
# stat vmlinuz-5.4.12-200.fc31.x86_64
  Fichero : vmlinuz-5.4.12-200.fc31.x86_64
  Tamaño : 10285768 Bloques : 20096     Bloques d'E/S : 4096    fichero regular
Dispositivo : 801h/2049d Nodo-i : 133         Enlaces : 1
Acceso: (0755/-rwxr-xr-x) UID : (    0/    root)   GID : (    0/    root)
Contexto : system_u:object_r:boot_t:s0
  Acceso: 2020-01-14 21:20:55.000000000 +0100
Modificación: 2020-01-14 21:20:55.000000000 +0100
    Cambio: 2020-01-23 21:21:45.575244140 +0100
  Creación: 2020-01-23 21:21:44.856252696 +0100
```

Para obtener información precisa, use el parámetro `-c` seguido del valor deseado (encontrará en la página del manual la lista completa) por ejemplo, para esta famosa fecha de creación:

```
# stat -c "%w" vmlinuz-5.4.12-200.fc31.x86_64
2020-01-23 21:21:44.856252696 +0100
```

Sobre sistemas con núcleos o bibliotecas más antiguas, como es el caso de la distribución CentOS o RHEL 8, se puede acceder a los atributos usando los comandos de depuración del sistema de archivos, como **debugfs** para ext4, estudiado en este capítulo.

Direcciones

El inodo contiene también campos de direcciones de bloques repartidos en general entre dos tipos:

- direcciones que apuntan a los primeros bloques de datos del archivo;
- direcciones que apuntan a bloques que contienen otros campos de direcciones;
- y en este último caso de manera recursiva (direcciones de direcciones que apuntan ellas mismas a otras direcciones), formando un árbol, o bien cada una las hojas (terminaciones) apunta a un bloque de datos. Se habla de bloques **de indirección** (simple, doble, triple).

A continuación presentamos un ejemplo concreto de direcciones en un inodo dentro de un sistema de archivos ext2.

Un inodo ext2 contiene diez campos que apuntan cada uno a un bloque de datos, y tres campos de indirección (apuntan a direcciones).

- El primero de estos tres campos apunta, en indirección simple, a 256 direcciones de bloques de datos.
- El segundo apunta, en indirección doble, a 256 direcciones, las cuales apuntan cada una a otras 256 direcciones de bloques de datos, o sea, 256^2 bloques de datos.
- El tercer campo apunta, en triple indirección, a 256 direcciones que a su vez apuntan a 256 direcciones, que a su vez apuntan a 256 direcciones de bloques de datos, o sea 256^3 bloques de datos.

Si n es el tamaño de un bloque en bytes, el tamaño máximo de un archivo es, por lo tanto **$n*(10+256+256^2+256^3)$ bytes**.

Para un bloque de 4096 bytes, ¡eso representa unos 64 GB!

d. Tablas de catálogo

Si un inodo no contiene el nombre del archivo, éste se coloca en una tabla de catálogo. Esta tabla no es más que un directorio. Como un directorio contiene una lista de archivos y un inodo representa un archivo, entonces cada nombre de archivo se asocia a su inodo dentro del directorio.

Puede representarse esta tabla como un cuadro de dos columnas:

Tabla catálogo dir1 (directorio dir1)	
Inodo	Nombre
12345	Documento.txt
214579	Archivo.doc
47321	Música.mp3
98542	Copia.odt
...	...

e. Hard link

Un hard link permite añadir una referencia a un inodo. El hard link añade una asociación en una tabla de catálogo. No se modifican los permisos del archivo.

Un hard link no permite asignar varios nombres a un mismo directorio, ni tampoco permite efectuar vínculos desde o hacia otro sistema de archivos. Además, tenga cuidado con el contador de vínculos facilitado por el comando **ls -l**: un 1 indica que este archivo no posee otros vínculos o, dicho de otro modo, es el último. Si lo suprime, se pierde de manera definitiva. En cambio, mientras este contador sea superior a 1, si se suprime un vínculo, queda una copia del archivo en alguna parte.

```
$ touch fic1
$ ln fic1 fic2
$ ls -li
2394875 -rw-r--r-- 2 seb users    0 mar 21 22:40 fic1
2394875 -rw-r--r-- 2 seb users    0 mar 21 22:40 fic2
```

El ejemplo anterior muestra que los hard links representan el mismo archivo, ya que sólo se trata de los nombres asociados al mismo inodo. Cada uno tiene dos vínculos, lo que es lógico ya que los dos archivos apuntan al mismo inodo. Finalmente, puede ver que fic1 y fic2 tienen el mismo inodo, a saber 2394875.

2. Crear un sistema de archivos

a. mkfs, sintaxis general

Los comandos de “formateo” disponibles en Microsoft no son similares a los de Linux. Un formato de tipo Microsoft es en realidad la creación y comprobación de un sistema de archivos en una partición. La primera etapa consiste en rellenar diferentes sectores, bloques y clústeres de ceros (o de otro motivo binario) con una verificación del soporte, y la segunda consiste en la escritura de un sistema de archivos. Esta última operación única basta para la creación de un sistema de archivos virgen en el disco o la partición.

El comando para crear un sistema de archivos es **mkfs**. **mkfs** llama a otros programas en función del tipo de sistema de archivos seleccionado.

```
mkfs -t typefs opciones periférico
```

`typefs` determina el tipo de sistema de archivos y, por lo tanto, el programa al que se llama. Existe un programa por tipo de sistema de archivos:

- **ext2**: mkfs.ext2
- **ext3**: mkfs.ext3
- **ext4**: mkfs.ext4
- **btrfs**: mkfs.btrfs
- **vfat**: mkfs.vfat (para todos los formatos FAT, pero existe mkfs.msdos)
- **ntfs**: mkfs.ntfs

Observe que es muy importante que se indiquen las opciones de cada sistema de archivos DESPUÉS de haber especificado el sistema de archivos. Si las especifica antes, serán las opciones de mkfs.

En vez de utilizar mkfs, puede utilizar directamente los programas correspondientes al tipo de sistema de archivos que se va a escribir.

b. Un primer ejemplo en ext2

Va a crear un sistema de archivos de tipo ext2 en la primera partición creada anteriormente, a saber, sdb1. El uso de este antiguo sistema de archivos es voluntario, más adelante se convertirá en un sistema de archivos más reciente. El comando básico es el siguiente:

```
mkfs -t ext2 -v /dev/sdb1
mke2fs 1.45.6 (20-Mar-2020)
resolución de fs_types para mke2fs.conf: 'ext2'
Filesystem label=
Tipo de SO: Linux
Tamaño del bloque=4096 (log=2)
Tamaño del fragmento=4096 (log=2)
Stride=0 bloques, anchura de stripe=0 bloques
```

```
262144 nodos-i, 1048320 bloques
52416 bloques (5.00%) reservados para el superusuario
Primer bloque de datos=0
Número máximo de bloques del sistema de ficheros=1073741824
32 grupos de bloques
32768 bloques por grupo, 32768 fragmentos por grupo
8192 nodos-i por grupo
UUID del sistema de ficheros: 28272a5c-75d5-4507-b76c-e8fabf13734b
Respaldos del superbloque guardados en los bloques:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
```

Esta salida proporciona información interesante:

- Es posible poner una etiqueta (un nombre) al sistema de archivos.
- Cada bloque es de 4096 bytes.
- Hay 2569728 bloques.
- Los nodos i (inodos) representan el número máximo de archivos: 125440.
- Se reserva el 5 % del espacio en disco a root, lo que significa que un usuario lambda no podrá llenar el disco a más del 95 %.
- Se reparten las tablas de inodos por grupos.
- Hay un superbloque principal y ocho superbloques de emergencia (uno por grupo).
- Es posible modificar algunos parámetros del sistema de archivos con el comando **tune2fs**.

c. ext2, ext3 y ext4

Como los sistemas de los archivos ext2 y ext3 son compatibles, comparten los mismos parámetros. Entre los cuales los más corrientes deben de utilizarse después de `-t`:

Parámetro	Significado
`-b`	Tamaño de los bloques en bytes, múltiplo de 512. Si no se especifica el tamaño, se determinará por el tamaño de la partición. Cualquier archivo creado en el disco ocupa al menos un bloque y, por lo tanto, si se prevé un gran número de pequeños archivos, hay que poner un valor bajo (p. ej.: 1024).
`-c`	Verifique los bloques defectuosos antes de crear el sistema de archivos. También se puede utilizar el comando **badblocks**.
`-i`	Ratio bytes/inodo. Se calcula el tamaño de la tabla de los inodos en función del tamaño total del sistema de archivos. Un inodo ocupa 128 bytes. El hecho de meter menos limita el número de archivos posibles, pero permite ganar espacio. -i 4096: un inodo por cada 4 KB.
`-m`	Porcentaje reservado al superusuario, por defecto el 5 %. Ponerlo a cero permite ganar espacio, y root podrá trabajar en ello a pesar de todo.
`-L`	Label, etiqueta (nombre) del sistema de archivos, útil para el montaje.
`-j`	Crea un diario ext3 y por lo tanto un sistema de archivos ext3.
`-v`	Modo verbose (detallado).

El ejemplo siguiente crea un sistema de archivos transaccional ext3 (opción `-j`) con un tamaño de bloques de 2048 bytes, y un inodo por cada 16 KB. Los usuarios pueden utilizar la totalidad del sistema (no se reserva ningún espacio para root). La etiqueta es DATA.

```
# mkfs -t ext2 -v -j -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
mke2fs 1.45.6 (20-Mar-2020)
```

```
resolución de fs_types para mke2fs.conf: 'ext2'
Filesystem label=DATA
Tipo de SO: Linux
Tamaño del bloque=2048 (log=1)
Tamaño del fragmento=2048 (log=1)
Stride=0 bloques, anchura de stripe=0 bloques
262144 nodos-i, 2096640 bloques
0 bloques (0.00%) reservados para el superusuario
Primer bloque de datos=0
Número máximo de bloques del sistema de ficheros=538968064
128 grupos de bloques
16384 bloques por grupo, 16384 fragmentos por grupo
2048 nodos-i por grupo
UUID del sistema de ficheros: 7f44011e-e69b-41d2-b49f-9df63fd24524
Respaldos del superbloque guardados en los bloques:
    16384, 49152, 81920, 114688, 147456, 409600, 442368, 802816, 1327104,
    2048000

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (16384 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
```

Observe que la línea de comandos siguiente tiene el mismo efecto, ya que el sistema de archivos ext3 induce el parámetro `-j`:

```
# mkfs -t ext3 -v -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
```

De ext2 a ext3

Ext3 es un sistema de archivos ext2 al cual se ha añadido un soporte transaccional. Puede convertir un sistema de archivos ext2 en ext3 utilizando **tune2fs**.

```
# tune2fs -j /dev/sdb1
tune2fs 1.45.6 (20-Mar-2020)
Creando el nodo-i del fichero de transacciones: hecho.
```

De ext3 a ext2

Para volver a ext2, hay que suprimir otra vez el soporte transaccional con `tune2fs` y el parámetro `-O` (Option, con O mayúscula):

```
# tune2fs -O ^has_journal /dev/sdb1
```

Verifique la posible presencia de un archivo `.journal` y suprímalo. Finalmente, efectúe una verificación con fsck.

De ext3 a ext4

Primero, convierta su sistema de archivos a ext3 añadiendo un diario, como se ha visto anteriormente. Aunque a continuación ya se podría montar directamente este sistema ext3 como ext4, no podría beneficiarse de las ventajas de ext4 si no utilizara los extents. El comando tune2fs permite añadirlos, así como otros parámetros necesarios:

- **extents**: añadir extents.
- **unint_bg**: optimización de la tabla de inodos que permite reducir las comprobaciones del sistema de archivos.
- **dir_index**: modificación de la tabla de catálogos para acelerar el acceso, para directorios de gran tamaño.

```
# tune2fs -O extents,unint_bg,dir_index /dev/sdb1
```

Las optimizaciones no se activarán hasta después de un primer control del sistema de archivos, lo que verá en la sección Comprobar, ajustar y arreglar. Sin embargo, debe ejecutar el comando siguiente:

```
# fsck -pDf /dev/sdb1
```

Después de esta etapa, el sistema de archivos puede montarse y usarse directamente, como se indica en la sección Acceder a los sistemas de archivos. Sin embargo, la conversión todavía no es completa, ya que los archivos que están ya en el sistema de

archivos no se han optimizado para usarse con extents. El comando **chattr** permite modificar los atributos específicos de cada archivo, a los que se les añade los extents con el parámetro +e. Necesitará listar todos los archivos y añadirles este atributo; lo podrá hacer con el comando **find**. Para empezar, monte el sistema de archivos:

```
#mount /dev/sdb1 /mnt
```

A continuación aplique el atributo `+e` en cada archivo. El parámetro `xdev` indica a find que no tiene que descender en subdirectorios que no sean del mismo sistema de archivos que el inicial:

```
# find /mnt -xdev -type f -exec chattr +e {} \ ;
```

```
# find /mnt -xdev -type d -exec chattr +e {} \ ;
```

¡Atención! Si convierte el sistema de archivos raíz o el que contiene el núcleo de Linux compruebe para empezar si GRUB es compatible con ext4. En caso contrario, corre el riesgo de no poder arrancar su sistema.

De ext4 a ext3

Si el sistema de archivos ext4 no tiene aún la opción de extents, entonces basta con montarlo a ext3. En caso contrario, no se puede realizar la conversión, ya que el atributo, una vez asignado, no debe quitarse. En este caso, la única solución consiste en realizar una copia de seguridad, para poder volver a crear el sistema de archivos ext3 y restaurar la copia.

Label

Puede visualizar y cambiar la etiqueta del sistema de archivos tecleando e2label.

```
# e2label /dev/sdb1
DATA
# e2label /dev/sdb1 OLDDATA
# e2label /dev/sdb1
OLDDATA
```

Es importante acordarse de modificar las opciones de montaje en consecuencia.

Una etiqueta de sistema de archivos no debe superar los 16 caracteres o se truncará.

d. XFS

Cree un sistema de archivos XFS como éste:

```
# mkfs -t xfs -f /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=262080 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=1048320, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

e. BTRFS

Atención, aunque se ve cierta estabilización y es empleado a veces en sistemas de producción muchas de las funcionalidades de BTRFS siguen siendo inestables o tienen problemas de rendimiento. Se aconseja comprobar la documentación de la distribución y los consejos del editor.

BTRFS sigue en evolución constante, el sitio web https://btrfs.wiki.kernel.org/index.php/Status le dará el estado actual del sistema de archivos para el núcleo y las herramientas utilizadas.

El sistema de archivos BTRFS dispone de muchas funciones avanzadas: RAID, subvolúmenes, snapshots, etc. Algunas de estas nociones serán abordadas en el último capítulo sobre el particionamiento avanzado.

La creación de un sistema de archivos BTRFS sigue el mismo principio:

```
# mkfs -t btrfs -f /dev/sdb1
btrfs-progs v5.4
See http://btrfs.wiki.kernel.org for more information.

Label:              (null)
UUID:               2b1c7c56-b5fc-42a7-b254-805ab742873a
Node size:          16384
Sector size:        4096
Filesystem size:    9.80GiB
Block group profiles:
  Data:             single          8.00MiB
  Metadata:         DUP             256.00MiB
  System:           DUP             8.00MiB
SSD detected:       no
Incompat features:  extref, skinny-metadata
Checksum:           crc32c
Number of devices:  1
Devices:
   ID        SIZE  PATH
    1     9.80GiB  /dev/sdb1
```

De ext3 o ext4 a BTRFS

Los comandos BTRFS permiten convertir fácilmente un sistema de archivos ext3 o ext4, con la posibilidad de volver al estado anterior si surgen problemas. En este caso, evidentemente, las modificaciones se borrarán. Para empezar, realice una comprobación del sistema de archivos:

```
# fsck -f /dev/sdb1
```

A continuación ejecute el comando **btrfs-convert**; puede tardar un poco según el tamaño de su sistema de archivos y la cantidad de datos que contenga:

```
# btrfs-convert /dev/sdb1

create btrfs filesystem:
        blocksize: 4096
        nodesize: 16384
        features: extref, skinny-metadata (default)
        checksum: crc32c
creating ext2 image file
creating btrfs metadata
copy inodes [o] [       2/       11]
conversion complete
```

Monte el sistema de archivos:

```
# mount -t btrfs /dev/sdb1 /mnt
```

En este punto de montaje encontrará una imagen del antiguo sistema de archivos en el directorio **ext2_saved** que se podrá montar si lo desea. Encontrará toda la información necesaria en la wiki de btrfs: https://btrfs.wiki.kernel.org/index.php/Main_Page#Documentation

Para liberar el espacio y pasar definitivamente a BTRFS sin vuelta atrás, puede borrar el directorio **ext2_saved**.

Vuelta atrás

Si BTRFS no le convence, y si no ha borrado el directorio **ext2_saved**, puede anular la conversión, después de haber desmontado el sistema de archivos:

```
# btrfs-convert -r /dev/sdb1
rollback succeeded
```

f. VFAT

La creación de un sistema de archivos VFAT se hace también de la misma manera. Esta vez, va a crearlo en la partición sdb5 prevista a tal efecto. El comando va a seleccionar automáticamente, en función del tamaño de la partición, el tipo de FAT que se debe crear (12, 16 o 32). Se ha añadido el parámetro `-v` para ver las trazas de su creación.

```
mkfs -t vfat -v /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
Auto-selecting FAT32 for large filesystem
/dev/sdb1 has 133 heads and 62 sectors per track,
hidden sectors 0x0800;
logical sector size is 512,
using 0xf8 media descriptor, with 8386560 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 8 sectors per cluster.
FAT size is 8176 sectors, and provides 1046272 clusters.
There are 32 reserved sectors.
Volume ID is 74177e99, no volume label.
```

Los comandos **mkfs.vfat** y **mkfs.msdos** son vínculos simbólicos hacia el programa mkdosfs.

Puede especificar varios parámetros, en particular si desea forzar la creación de un tipo de FAT dado:

Parámetro	Significado
`-c`	Verifica el periférico antes de la creación.
`-F`	Tamaño de la FAT (12, 16, 32).
`-I`	Permite utilizar un disco completo, y no una partición (práctico para algunos lectores de MP3).
`-n`	Nombre del volumen (etiqueta, label).
`-v`	Visualización de detalles durante la creación.

Las mtools

Las mtools son herramientas que permiten trabajar en sistemas de archivos FAT y VFAT como si estuviera bajo MSDOS o la consola de Windows. Retoman la sintaxis de los comandos de origen, pero añaden una m delante: mdir, mformat, mlabel, mdeltree, etc.

Se representan los discos y las particiones con letras de unidad c :, d :, e :. Dichas letras pueden representar un disco, una partición o un directorio. Sin embargo, debe modificar un archivo de configuración /etc/mtools.conf. Por ejemplo, para declarar /dev/sdb1 como d: añada o modifique la línea siguiente:

```
drive d: file="/dev/sdb1"
```

Es útil para modificar después ciertos datos, como el nombre del volumen del sistema de archivos vfat:

```
# mlabel -s d:
Volume has no label
```

```
# mlabel d:
Volume has no label
Enter the new volume label : DATAFAT
# mlabel -s d:
Volume label is DATAFAT
```