Controlar el sistema de archivos

1. Estadísticas de ocupación

a. Por sistema de archivos

El comando **df** permite obtener estadísticas de ocupación de cada sistema de archivos montado. Sin argumento, **df** facilita información sobre todos los sistemas de archivos. Puede pasar como argumento un periférico montado o un punto de montaje. Si pasa un directorio cualquiera, df da la información del sistema de archivos que contenga este directorio.

```
# df
Sis. de fich.          1K-bloques   Ocupado Disponible Capacidad Montado en
/dev/mapper/fedora-root   15718400   6504544  9213856   42%     /
/dev/sda1              1038336    229732   808604    23%     /boot
/dev/sdb1              10051792   36888    9484576   1%      /mnt/DATA
```

El resultado es explícito. La unidad por defecto es el KB (idéntico al parámetro `-k`) aunque la norma POSIX define una unidad de bloque en 512 bytes. Puede modificar los parámetros para pedir el resultado en MB (`-m`).

```
# df -m /mnt/DATA
Sis. de fich.  1M-bloques  Ocupado  Disponible Capacidad  Montado en
/dev/sdb1      9817        37       9263       1%        /mnt/DATA
```

Para que sea más legible, añada el parámetro `-h` (*Human readable*).

```
# df -h /mnt/DATA
Sis. de fich.     Tam.   Oc.  Disp.  %Oc.  Montado en
/dev/sdb1          9,6G   37M  9,1G   1%    /mnt/DATA
```

No confunda este último parámetro con `-H`, que visualiza el resultado en unidades **SI** (*Sistema Internacional*).

```
# df -H /home
Sis. de fich.    Tam.  Oc.  Disp.  %Oc.  Montado en
/dev/sdb1        158G   51G  100G   34%   /home
```

Las unidades del SI que definen las unidades de peso y medida se basan en potencias de 10. Es sencillo memorizar que 1 Kg es igual a 10^3 gramos, o sea, 1000 gramos. Por lo tanto, 1 KB equivale a 10^3 bytes, o sea 1000 bytes... ¿No está de acuerdo? Un ordenador no trabaja con potencias de 10, sino de 2. A nivel físico, 1 KB equivale 2^{10} bytes, o sea 1024 bytes; 1 MB vale 2^{20} bytes, y así sucesivamente. Este método se llama método tradicional. El sistema internacional prefiere emplear los términos kibibyte (kilo Binario, Kib), mebibytes (Meb) y gibibytes (Gib) para las representaciones binarias, y KB, MB y GB para las potencias de 10, como en el caso de los metros y los gramos. Ahora entiende por qué un disco de 160 GB se corresponde en realidad a 152,5 Gib. En cierto modo, nos dejamos engañar de manera legal y oficial.

La `-T` añade la visualización del tipo de sistema de archivos.

```
# df -T /mnt/DATA
Sis. de fich. Tipo  1K-bloques  Ocupado Disponible  Capacidad  Montado en
/dev/sdb1        10051792   36888   9484576    1%       /mnt/DATA
```

El comando **df** permite también facilitar estadísticas para el uso de los inodos. Puede combinar los parámetros `-i` y `-h`.

```
# df -ih /mnt/DATA/
```

```
Sis. de fich.      Inodos   IUtil.   ILib.   %IUti. Montado en
/dev/sdb1           629K     11      629K    1%     /mnt/DATA
```

b. Por estructura

El comando **du** (*disk usage*) facilita la información relativa al espacio ocupado por una estructura (un directorio y todo su contenido). Si no se especifica nada, se utiliza el directorio corriente. Los parámetros `-k` (Kb) y `-m` (MB) determinan la unidad. Se facilita el tamaño para cada elemento (incluso redondeado). El tamaño total de la estructura está en la última línea.

```
# du -m LIBRO_LPI
1       LIBRO_LPI/Nuevos/ce00-introducción/imágenes
2       LIBRO_LPI/Nuevos/ce00-introducción/
1       LIBRO_LPI/Nuevos/ce00-a-descriptivo
1       LIBRO_LPI/Nuevos/ce01
...
42      LIBRO_LPI
```

Para obtener el total, y no todos los detalles, utilice `-s`.

```
# du -ks LIBRO_LPI
42696 LIBRO_LPI/
```

Observe que no se limita **du** a un único sistema de archivos y sigue calculando si encuentra un punto de montaje en la estructura que analiza. Si quiere limitar el cálculo al sistema de archivos corriente sin entrar en los puntos de montaje presentes en la estructura, especifique `-x`.

```
# du -msx /
1064 /
```

2. Comprobar, ajustar y arreglar

a. fsck

El comando **fsck** permite comprobar y arreglar un sistema de archivos.

```
fsck -t typefs periférico
```

El sistema de archivos que se quiere comprobar o arreglar no debería estar montado, o, como mucho, montado en modo de sólo lectura.

De la misma forma que **mkfs**, **fsck** invoca a otro comando teniendo en cuenta el tipo del sistema de archivos para comprobar: esos otros comandos más especializados son fsck.ext2, fsck.ext3, fsck.ext4, fsck.btrfs, fsck.xfs, etc. Cada uno puede presentar opciones particulares. Si **fsck** no reconoce la opción que se le proporciona, la transmite al programa correspondiente. Si no indica un tipo, **fsck** intenta determinarlo por sí mismo.

La verificación se hace con el sistema de archivos desmontado. Para este ejemplo, se pasa el parámetro `-f` a fsck para forzar la comprobación (no ha sido posible producir una corrupción), así como el parámetro `-V` para facilitar todos los detalles.

```
# fsck -fV /dev/sdb1
fsck de util-linux 2.33.1
e2fsck 1.44.5 (15-Dec-2018)
Paso 1: verificación de los i-nodos, de los bloques y de los tamaños
Paso 2: verificación de la estructura de los directorios
Paso 3: verificación de la conectividad de los directorios
Paso 4: verificación de los contadores de referencia
Paso 5: verificación de la información del sumario del grupo

   42 inodes used (0.06%)
    1 non-contiguous inode (2.4%)
      # of inodes with ind/dind/tind blocks: 10/1/0
 8864 blocks used (6.69%)
    0 bad blocks
    1 large file
```

```
    27 regular files
     3 directories
     0 character device files
     0 block device files
     0 fifos
     0 links
     3 symbolic links (3 fast symbolic links)
     0 sockets
--------
    33 files
```

Cuando el sistema de archivos está dañado, **fsck** inicia una batería de preguntas por cada acción necesaria. Puede pasar el parámetro `-p` para intentar una reparación automática, o también `-y` para forzar las respuestas a sí. Se aconseja comprobar los parámetros del manual del comando **fsck** asociado:

```
# man fsck.ext4
```

Durante el inicio del sistema, éste comprueba desde hace cuánto tiempo, o después de cuántos montajes, no se ha comprobado el sistema de archivos. Si el intervalo de tiempo es demasiado grande, ejecutará un **fsck** en el sistema de archivos correspondientes. Se pueden modificar los intervalos mediante el comando **tune2fs**.

Una partición de tipo BTRFS no necesita fsck, emplea **btrfsck** o **btrfs check**. El comando fsck.btrfs sólo existe para indicar el uso de los dos primeros.

b. badblocks

El comando **badblocks** intenta comprobar los bloques defectuosos en el periférico de almacenamiento proporcionado como argumento. **mkfs** o **fsck** pueden llamar a este comando si se les proporciona el parámetro `-c` (check).

Por defecto, **badblocks** lee la totalidad de los bloques del soporte y devuelve un error si uno o varios de ellos son ilegibles. Se puede ejecutar el comando incluso aunque el sistema de archivos esté montado, excepto si usted una prueba en lectura y escritura, incluso no destructiva.

```
# badblocks -v /dev/sdb1
Comprobación de los bloques 0 a 10278911
Comprobación de los bloques defectuosos (prueba en modo de sólo lectura): done
Paso completado, 0 bloques defectuosos localizados.
```

Los parámetros `-n` (no destructivo) y `-w` (write, con motivos, destructivo) intentan escribir en los bloques. El primero lee y vuelve a escribir lo leído en el bloque, el segundo escribe otro tipo de información (0xaa, 0x55, 0xff, 0x00) y por lo tanto sobreescribe lo anterior.

La ejecución de badblocks puede ser larga, varias horas en algunos centenares de GB.

c. dumpe2fs

El comando **dumpe2fs** acepta como argumento un periférico que contiene un sistema de archivos ext2, ext3 o ext4. Devuelve un gran número de información sobre el sistema de archivos.

```
dumpe2fs /dev/sdb1
dumpe2fs 1.45.6 (20-Mar-2020)
Filesystem volume name:   DATA
Last mounted on:          <not available>
Filesystem UUID:          713064f6-0e6c-429a-8868-a0ec31675aa8
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index
filetype extent 64bit flex_bg sparse_super large_file huge_file dir_nlink
extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              2097152
Block count:              8388352
Reserved block count:     419417
Free blocks:              8211646
```

```
Free inodes:              2097141
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      1024
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Wed Apr 14 19:36:38 2021
Last mount time:          n/a
Last write time:          Wed Apr 14 19:36:38 2021
Mount count:              0
Maximum mount count:      -1
Last checked:             Wed Apr 14 19:36:38 2021
Check interval:           0 (<none>)
Lifetime writes:          4182 kB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      149b068c-be80-43a3-87eb-f576e17dfbf9
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0x29883976
Journal features:         (none)
Journal size:             128M
Journal length:           32768
Journal sequence:         0x00000001
Journal start:            0


Grupo 0: (Bloques 0-32767) csum 0x09c9
  Superbloque primario en 0, descriptores de grupo en 1-4
```

```
  Se reservaron los bloques GDT en 5-1028
  Mapa de bits de bloques en 1029 (+1029), csum 0x2d4563fb
  Mapa de bits de nodos-i en 1045 (+1045), csum 0xa8b8430a
  Tabla de nodos-i en 1061-1572 (+1061)
  23509 bloques libres, 8181 nodos-i libres, 2 directorios, 8181 nodos-i sin usar
  Bloques libres: 9259-32767
  Nodos-i libres: 12-8192
Grupo 1: (Bloques 32768-65535) csum 0xdc95 [INODE_UNINIT, BLOCK_UNINIT]
  Superbloque de respaldo en 32768, descriptores de grupo en 32769-32772
  Se reservaron los bloques GDT en 32773-33796
  Mapa de bits de bloques en 1030 (bg #0 + 1030), csum 0x0000
  Mapa de bits de nodos-i en 1046 (bg #0 + 1046), csum 0x0000
  Tabla de nodos-i en 1573-2084 (bg #0 + 1573)
  31739 bloques libres, 8192 nodos-i libres, 0 directorios, 8192 nodos-i sin usar
  Bloques libres: 33797-65535
  Nodos-i libres: 8193-16384
...
```

Esta salida es muy larga (y ello a pesar de que ha sido truncada), pero le da todos los detalles posibles sobre el sistema de archivos. Puede aislar únicamente el encabezamiento (hasta el tamaño del diario) con el parámetro `-h`. Si busca un dato en concreto, lo mejor es utilizar el comando **grep**, ya que la redirección de la salida de error evita que se muestre la versión.

```
# dumpe2fs -h /dev/sdb1|grep -i "block size"
Block size:               4096
```

d. tune2fs

El comando **tune2fs** permite modificar algunos parámetros de un sistema de archivos ext2 o ext3. Hemos visto anteriormente este comando cuando se explicó cómo convertir ext2 a ext3, y viceversa. Le presentamos algunos parámetros soportados por el comando:

Parámetro	Significado

`-c n`	Número de veces que se debe montar el sistema de archivos antes de ser comprobado automáticamente. Por ejemplo, si n vale 10, se iniciará de manera automática fsck la décima vez que se monte. Si n vale 0 o -1, se desactivará la verificación.
`-i n`	Intervalo de tiempo entre dos comprobaciones. La unidad por defecto es el día. Se pueden juntar los sufijos d (días), w (semanas) o m (mes) al número. `-i` 180d significa que se controlará el sistema de archivos pasados 180 días.
`-j`	Añade un diario sobre un sistema de archivos ext2. Es preferible hacerlo con el sistema desmontado. En caso contrario, se añade un archivo oculto `.journal` a la raíz del sistema, inmutable (salvo si destruye el diario), que se integrará dentro del sistema durante la próxima ejecución de fsck.
`-L`	Modifica la etiqueta (label, nombre de volumen). La etiqueta no debe superar 16 caracteres o se truncará.
`-e err`	Indica cómo debe reaccionar el núcleo si se detecta un error en el sistema de archivos durante el boot. El valor por defecto es “continuo”. Los demás valores posibles son “panic” (bloqueo del núcleo en modo kernel panic) y “remount-ro”: remontaje en modo de sólo lectura.
`-m n`	El porcentaje n representa el tamaño reservado a los procesos iniciados por root (y el propio root) sobre el tamaño total de la partición. En un sistema de archivos reservados a los usuarios, poner 0 permite llenar el sistema de archivos hasta el 100 %. Pero es importante reservar en la raíz, o /var, una zona para que algunas bitácoras como syslogd puedan seguir escribiendo los registros. Por defecto se reserva el 5 %.

`-o [^]option`	Añade o suprime (con ^) la opción indicada por defecto al montar. Las opciones pueden ser, por ejemplo, acl o user_xattr.
`-O [^]function`	Añade o suprime (con ^) la función indicada. La función más famosa es "has_journal". `-O has_journal` equivale a -j. `-O ^has_journal` convierte ext3 en ext2.
`-U UUID`	Modifica el valor del UUID a su conveniencia (formato hexadecimal). Es posible suprimirlo (clear), generar uno de manera aleatoria (random) o generar uno en función de la fecha (time).
`-s 0/1`	Activa o desactiva la "sparse super feature". En discos de gran tamaño, la activación reduce el número de bloques de emergencia para ganar espacio. Después debe ejecutar fsck.

```
# tune2fs -m 0 -s 1 -U random -e remount-ro -c 60 -i 180 /dev/sdb1
```

Observe que `-c` e `-i` van de la mano. A vencimiento del plazo se efectúa la comprobación con fsck. Luego el sistema pone de nuevo a cero la fechas y los contadores. Como consecuencia de un fsck, el sistema pone de nuevo a cero las fechas y los contadores. Se verifican los contadores durante el montaje en el momento del inicio (boot) del sistema. Si hace 300 días que el sistema no se ha reiniciado y no se ha verificado el sistema de archivos durante este intervalo, no se verificará automáticamente durante estos 300 días, sino en el próximo reinicio. Como algunos servidores reinician pocas veces (por ejemplo, cada dos años), no se comprueba el sistema de archivos de manera automática durante todo este tiempo...

e. debugfs

El comando **debugfs** permite depurar un sistema de archivos de tipo ext2 a ext4. Se emplea bien de forma interactiva, en forma de Shell, o como línea de comando, en cuyo caso usará el parámetro **-R**.

Es un comando a la vez útil y peligroso que permite efectuar todo tipo de operaciones en el sistema de archivos, que no permite usar tune2fs o los comandos Linux clásicos. La mayoría de los comandos se parecen, pero trabajan a muy bajo nivel. El comando **ls**, por ejemplo, proporciona el tipo de archivo en forma de valor numérico: 2 para un archivo ordinario, 4 para un directorio, 12 para un enlace simbólico, etc.

```
# debugfs -R 'ls -l' /dev/sda1 | tee
debugfs 1.43.3 (04-Sep-2016)
     2   40755 (2)      0     0    409617-Jan-2017 21:50 .
     2   40755 (2)      0     0    409617-Jan-2017 21:50 ..
    11   40700 (2)      0     0   1638414-Jan-2017 17:48 lost+found
129281   40755 (2)      0     0    409614-Jan-2017 17:48 home
258561   40755 (2)      0     0   1228818-Jan-2017 21:36 etc
387841   40755 (2)      0     0    409614-Jan-2017 17:48 media
 18132  120777 (7)      0     0      3217-Jan-2017 21:20 initrd.img
```

Los inodos y el comando **stat** se presentan además en este capítulo. He aquí como obtener información detallada acerca de un archivo específico y también cómo obtener la fecha de la creación del archivo en las distribuciones un poco antiguas (observe la línea **crtime** que ya se vio en la presentación de los inodos):

```
# debugfs -R 'stat /root/skype.deb' /dev/sda1 | tee
debugfs 1.43.3 (04-Sep-2016)
Inode: 288908   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 3396615331    Version: 0x00000000:00000001
User:     0   Group:     0   Project:     0   Size: 20118938
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 39296
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x587fc98a:a8d61870 -- Wed Jan 18 21:01:14 2017
 atime: 0x587fc98e:e7c24c54 -- Wed Jan 18 21:01:18 2017
 mtime: 0x57df9be1:00000000 -- Mon Sep 19 10:03:45 2016
```

```
crtime: 0x587fc986:1e9328a8 -- Wed Jan 18 21:01:10 2017
Size of extra inode fields: 32
EXTENTS:
(0-4095):1914880-1918975, (4096-4911):1975540-1976355
```

Usted puede volcar un inodo, lo que copiará de hecho el archivo vinculado a este inodo:

```
# stat -c %i skype.deb
288908
# debugfs -R 'dump <288908> file1-debugfs' /dev/sda1
```

3. XFS

El uso de XFS se ha vuelto frecuente en Linux, principalmente gracias a su uso por defecto en las distribuciones profesionales Red Hat Enterprise Linux y CentOS a partir de la versión 7, al igual que la distribución Fedora a partir de la versión 22. Conviene conocer los comandos básicos específicos de este sistema de archivos.

a. xfs_info

El comando **xfs_info** muestra las características del sistema de archivos, de forma semejante a las proporcionadas durante su creación.

```
$ xfs_info /
meta-data=/dev/sda3         isize=256    agcount=4, agsize=1744064 blks
         =                  sectsz=512   attr=2, projid32bit=1
         =                  crc=0        finobt=0 spinodes=0
data     =                  bsize=4096   blocks=6976256, imaxpct=25
         =                  sunit=0      swidth=0 blks
naming   =version 2         bsize=4096   ascii-ci=0 ftype=0
log      =internal          bsize=4096   blocks=3406, version=2
         =                  sectsz=512   sunit=0 blks, lazy-count=1
realtime =none              extsz=4096   blocks=0, rtextents=0
```

b. xfs_growfs

Un sistema de archivos XFS puede expandirse, pero no puede ser reducido, tampoco en frío (desmontado).

xfs_growfs se asemeja a resize2fs (véase el capítulo Particionamiento avanzado: RAID, LVM y BTRFS) y se emplea de idéntica forma. Si no se especifica nada, el sistema de archivos se extenderá a todo el espacio nuevo disponible.

```
# xfs_growfs /
```

Al contrario que un sistema de archivos de tipo ext, el número de inodos no está limitado en un sistema de archivos XFS.

c. xfs_repair

El comando **xfs_repair** es el equivalente de fsck. A diferencia de otros sistemas de archivos, xfs_repair no se ejecuta nunca al arrancar ni al montar de forma automática durante el arranque del sistema. En caso de incidente en el sistema de archivos, usted deberá ejecutar este comando por su cuenta. El comando se ejecuta con el sistema de archivos desmontado

```
# xfs_repair /dev/sdb1
# xfs_repair /dev/sdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
```

```
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify and correct link counts...
done
```

El registro de eventos de XFS se llama log. Podemos en principio obtener una copia empleando el comando **xfs_logprint**. Si este está corrompido (desmontaje incorrecto, fallo súbito), xfs_repair, basado en la lectura de este registro, no podrá funcionar. Habrá que recrear este registro. Observe que los datos pueden haber desaparecido durante la operación. El sistema de archivos no tendrá coherencia. Por lo que esto solo se debe usar como último recurso.

Monte y desmonte el sistema de archivos y limpiará los registros, simplemente porque estos se volverán a ejecutar, como con todo sistema de archivos transaccional. Sin embargo, en caso de problema, xfs_repair los puede borrar:

```
# xfs_repair -L
```

Vuelve a montar el sistema de archivos y verifica posibles errores.

d. xfs_db y xfs_admin

Los comandos **xfs_db** y **xfs_admin** equivalen a debugfs y tune2fs (ext). Estos permiten el mismo tipo de operaciones; sin embargo, deben emplearse con el sistema de archivos desmontado.

Por ejemplo, para obtener los detalles de un inodo:

```
# xfs_db /dev/sdb1
xfs_db> inode 67
xfs_db> print
core.magic = 0x494e
core.mode = 0100644
core.version = 3
core.format = 2 (extents)
core.nlinkv2 = 1
...
core.uid = 0
core.gid = 0
core.flushiter = 0
core.atime.sec = Sun Jan 22 16:42:44 2017
core.atime.nsec = 723113480
core.mtime.sec = Sun Jan 22 16:42:44 2017
core.mtime.nsec = 723113480
core.ctime.sec = Sun Jan 22 16:42:44 2017
core.ctime.nsec = 723113480
...
```

El comando **xfs_metadump** es un alias del comando **metadump** de **xfs_db**. Permite hacer un dump (vacío/copia exacta) de los metadatos de un sistema de archivos para permitir analizar el archivo resultante más tarde. El dump se efectúa también con el sistema de archivos desmontado.

e. xfs_fsr

El comando **xfs_fsr** permite reorganizar el sistema de archivos XFS: es decir, que lo desfragmenta. XFS usa la noción de extents. Cuando es posible, se intentará usar extents consecutivos para almacenar la información en un mismo archivo. Pero más adelante, si el archivo crece, el extent siguiente no estará disponible y otra será usado en su lugar. El comando **xfs_bmap** permite obtener esta información:

```
# xfs_bmap -v /boot/vmlinuz-5.4.12-200.fc31.x86_64
/boot/vmlinuz-5.4.12-200.fc31.x86_64:
```

```
EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET       TOTAL
  0: [0..20095]:      192..20287       0 (192..20287)     20096
```

Aquí solo hay una línea EXT, es decir, un solo extent usado. Si aparecieran algunas líneas, el archivo se repartiría sobre algunos extents. En ese caso, el uso de **xfs_fsr** puede ser relevante. El comando funciona en un sistema de archivos completo, montado, pero también directamente en los archivos:

```
# xfs_fsr /boot
/boot start inode=0
...
```