Engineer

Sistema de archivos

estructura de archivos y directorios con nombre en disco

Qué es

Un sistema de archivos es la parte de un ordenador que convierte un disco, que por sí solo no ofrece más que bloques numerados de bytes, en archivos con nombre organizados en directorios. En inglés también se escribe filesystem, en una palabra, la forma más usada por la documentación del kernel Linux. POSIX, el estándar que siguen los sistemas de tipo Unix, lo define en su edición de 2024 como «una colección de archivos y algunos de sus atributos». Añade que los números de serie de los archivos pertenecen a un espacio de nombres proporcionado por el sistema de archivos: identifican de forma única cada archivo dentro del sistema que lo contiene.

El resto se deduce de unas pocas definiciones más del estándar. Un archivo es un objeto en el que se puede escribir, del que se puede leer o ambas cosas, con atributos como permisos de acceso y tipo: archivo normal, directorio, enlace simbólico, FIFO, socket o archivo especial de caracteres o bloques. Un directorio es un archivo formado por entradas de directorio; en un mismo directorio no puede haber dos con el mismo nombre. Una entrada de directorio, que POSIX también llama enlace duro, vincula un nombre con un archivo; varias pueden nombrar el mismo archivo. Por tanto, se separan el nombre y el propio archivo. En Linux, el registro del archivo es el inodo, que, según el manual de GNU df, guarda propietario, permisos, marcas de tiempo y ubicación en el sistema de archivos. Los nombres viven en los directorios que apuntan a él.

Un sistema de tipo Unix presenta todos sus sistemas de archivos como un único árbol de nombres. El primero aporta el directorio raíz y los demás se acoplan a un directorio del árbol, su punto de montaje. Desde entonces, las rutas que hay debajo pertenecen al sistema acoplado. Dentro del kernel Linux, el sistema de archivos virtual, VFS, se sitúa entre los programas y los muchos sistemas de archivos compatibles con el kernel. Su documentación lo describe, también bajo el nombre Virtual Filesystem Switch, como la capa que da a los programas su

interfaz de sistema de archivos y permite coexistir a distintas implementaciones. Un programa llama a `open(2)`, `stat(2)`, `read(2)` o `write(2)`; VFS resuelve la ruta y entrega el trabajo al sistema que contiene el archivo: por ejemplo, ext4 en una partición o un pseudosistema de archivos cuyos inodos solo viven en memoria. Los tipos registrados por el kernel en ejecución figuran en `/proc/filesystems`.

Utilice el concepto en una decisión

Aplique «¿Qué es un sistema de archivos?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. El cuidado diario de servidores, correo, cuentas y actualizaciones para que los servicios de los que depende todo el mundo sigan funcionando sin sorpresas.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 10 servidores PostgreSQL y MS SQL en Ubuntu — 10 servidores de bases de datos desplegados y mantenidos

10 servidores PostgreSQL y MS SQL en Ubuntu Póngase en contacto →

Por qué importa

Casi todo lo que un servidor conserva más allá del siguiente reinicio se guarda en un sistema de archivos: archivos de datos de sus bases, registros, documentos subidos y configuración. Que sobrevivan a un corte eléctrico, un disco lleno o una unidad averiada depende en gran parte de las decisiones tomadas al crear y montar el sistema de archivos. Algunas no se pueden deshacer después.

La creación fija la capacidad en dos «monedas» distintas. Los bloques guardan el contenido de los archivos y los inodos los propios archivos. mke2fs, la herramienta que crea sistemas ext2, ext3 y ext4, crea un inodo por cierta cantidad de bytes de espacio; su manual advierte de que esa relación de bytes por inodo no puede cambiarse una vez creado el sistema. Un volumen puede, por tanto, quedarse sin sitio para nuevos archivos aunque aún tenga espacio para su contenido. Sorprende a quien solo vigila el porcentaje de espacio ocupado.

El montaje decide el coste de una caída. ext4 conserva un diario y su documentación enumera tres modos de datos. El predeterminado, `data=ordered`, fuerza la escritura de los datos de archivo en el sistema principal antes de confirmar en el diario los metadatos que los referencian. `data=writeback` elimina ese orden; la documentación advierte de que, tras un apagado incorrecto, archivos escritos poco antes pueden exponer datos antiguos, lo que en ciertos casos supone un riesgo de seguridad. `data=journal` también pasa los datos por el

diario y su uso está desaconsejado, con la retirada del soporte prevista para enero de 2028. La opción `errors=` decide qué hace el sistema al detectar un error propio: volver a montarse en solo lectura, continuar o provocar un pánico del kernel y detener la máquina.

La recuperación también depende de ello. Cuando se daña un sistema de archivos, el contenido puede seguir en disco mientras está rota la estructura que lo nombra. Ahí trabaja la recuperación de datos. La entrada sobre recuperar datos de tarjetas SD, discos duros, SSD y RAID dice que cada medio se avería de manera distinta y requiere una forma distinta de obtener los datos, y que los sistemas de archivos de Mac tienen sus peculiaridades. Lo importante era elegir el enfoque adecuado al medio en el banco de trabajo, más que una técnica concreta.

Cómo se aplica en la práctica

Suponga una partición nueva, `/dev/sdb1`, que guardará los datos de un servicio en `/srv/data`. Crear ext4 y montarlo a mano requiere tres órdenes:

```
mke2fs -t ext4 -L data /dev/sdb1
mkdir -p /srv/data
mount -t ext4 /dev/sdb1 /srv/data
```

`-t ext4` indica el tipo, como en la documentación del kernel sobre ext4, y `-L` da una etiqueta al sistema de archivos, limitada por mke2fs a 16 bytes. Lo demás procede de valores predeterminados: en la configuración distribuida con e2fsprogs, por ejemplo, cada inodo ocupa 256 bytes. Dos órdenes `df` muestran después los dos tipos de capacidad:

```
df -hT /srv/data
df -i /srv/data
```

La primera imprime el tipo de sistema de archivos y sus bloques usados y disponibles, en unidades basadas en potencias de 1.024; `-H` usaría potencias de 1.000. La segunda enumera inodos en lugar de bloques: cuántos hay, cuántos están usados y disponibles y el porcentaje utilizado.

Un montaje manual dura hasta el siguiente reinicio. Para hacerlo permanente, se añade una línea a `/etc/fstab`, el archivo cuyas entradas leen por orden mount, umount y fsck:

```
LABEL=data  /srv/data  ext4  defaults,nofail  0  2
```

El primer campo identifica el sistema por su etiqueta y no como `/dev/sdb1`. El manual de fstab recomienda usar `LABEL=` o `UUID=`, porque los nombres de dispositivo suelen seguir el orden de detección del hardware y pueden cambiar al añadir o quitar discos. `defaults` suele significar lectura y escritura junto con las demás opciones estándar del kernel; `nofail` dice a mount que no notifique un error si no existe el dispositivo. El quinto campo lo lee dump(8), y 0 excluye al sistema de archivos. El sexto indica a fsck el orden de las comprobaciones al arrancar: 1 para la raíz, 2 para los demás y 0 para ninguna. Con systemd, el manual recomienda `systemctl daemon-reload` después de editar. `mount -a` monta luego las entradas no marcadas como `noauto`, para que un error en la línea nueva aparezca mientras aún hay alguien al teclado.

Vigilar el espacio forma parte de administrarlo. La entrada sobre monitorización del host con interruptor de hombre muerto nombra el disco lleno como uno de los dos modos realistas de que deje de funcionar una máquina de 464 MB con una forge y un servidor web, junto al agotamiento de memoria; ninguno se anuncia por sí mismo. Su respuesta es un script pequeño que un temporizador inicia cuatro veces por hora, con un momento ligeramente desplazado cada vez, para medir disco libre y memoria disponible. Solo envía señales a un servicio externo mientras ambos superan sus umbrales, por lo que la alarma procede de la interrupción de esas señales. Una comprobación así también puede leer la columna de inodos de `df -i` además de la de espacio: cualquiera puede alcanzar primero su límite.

Dónde surgen los problemas

La mayoría de los problemas aparece como un disco lleno, una máquina que no monta lo que montaba ayer o datos perdidos tras una caída. Muchos se remontan a un ajuste que nadie miró al crear el sistema de archivos.

El disco está lleno aunque sobra espacio. Cada archivo necesita un inodo y mke2fs crea una cantidad fija al crear el sistema. Un volumen con millones de archivos pequeños puede agotar los inodos mientras `df -h` aún muestra

espacio libre. Desde entonces rechaza todo archivo nuevo. Vigile `df -i` junto a `df -h`. En un volumen para muchos archivos pequeños, fije la relación con `-i` o la cantidad con `-N` al ejecutar mke2fs, porque la relación no se puede cambiar después. Redimensionar el sistema mantiene la relación y añade o quita inodos en proporción.

Se comprueba un sistema de archivos montado. El manual de e2fsck dice que, en general, no es seguro ejecutarlo en un sistema montado. La única excepción es `-n`, que lo abre en solo lectura y responde no a todo; aun así, los resultados no son válidos mientras siga montado. Cuando e2fsck pregunta si debe continuar sobre un sistema montado, el manual responde que no. Desmonte primero o, para la raíz, deje que la comprobación se haga al arrancar, como dispone el sexto campo de fstab.

«Solo lectura» escribe en disco. Montar ext4 con `ro` no impide escribir: la documentación señala que reproduce el diario y por tanto escribe en la partición, aunque esté montada en solo lectura. Importa sobre todo al rescatar un disco, donde cualquier escritura puede caer sobre algo aún necesario. Las opciones `ro,noload` impiden las escrituras, con el coste indicado en la misma página: tras un desmontaje incorrecto, omitir la reproducción deja inconsistencias. Para un disco cuyo contenido importa más que su disponibilidad, móntelo con `ro,noload`, copie lo necesario fuera de ese sistema de archivos y trate lo leído como posiblemente inconsistente.

Un disco cambia de nombre. Una línea que monta `/dev/sdb1` monta lo que el kernel llama hoy `/dev/sdb1`; el manual de fstab advierte de que puede cambiar al añadir o quitar discos. Un disco nuevo puede montarse donde iba el antiguo o este puede faltar al arrancar. Use `LABEL=` o `UUID=` en el primer campo y establezca las etiquetas con `mke2fs -L` o después con una herramienta como `e2label`.

Falla la corriente entre la escritura y el disco. De forma predeterminada, ext4 confirma su diario cada cinco segundos. Un corte puede hacer que se pierdan los últimos cinco segundos de cambios de metadatos, aunque el diario mantiene intacto el propio sistema de archivos. La pérdida de datos puede abarcar un periodo aún mayor: con asignación diferida de bloques, su escritura en disco solo empieza después del intervalo de `/proc/sys/vm/dirty_expire_centisecs`. La documentación considera defectuosas las aplicaciones que sustituyen archivos sin `fsync()`. La opción predeterminada `auto_da_alloc` de ext4 detecta los dos patrones habituales, escribir un archivo nuevo y renombrarlo sobre el antiguo, o

truncar y reescribir sobre él, para evitar archivos vacíos tras una caída. La solución está en el programa que escribe: llame a `fsync()` antes de decir que los datos están guardados, en lugar de apoyarse en esa red de seguridad.

Las marcas de tiempo se detienen en 2038. El manual de mke2fs dice que los sistemas con inodos de 128 bytes no admiten marcas posteriores al 19 de enero de 2038, mientras que los inodos de 256 bytes o más admiten marcas ampliadas. Los sistemas de archivos nuevos usan de forma predeterminada inodos de 256 bytes, pero uno antiguo o creado con `-I 128` puede conservar los inodos pequeños. Detecte esos sistemas de archivos antes de que los alcance la fecha: la documentación de ext4 muestra `tune2fs -I 256` convirtiendo un sistema creado con inodos de 128 bytes.

Lecturas adicionales

- Definiciones, POSIX.1-2024 (The Open Group Base Specifications Issue 8) — Las definiciones de sistema de archivos, archivo, directorio, enlace duro y punto de montaje.
- Visión general del sistema de archivos virtual de Linux — Cómo resuelve VFS una ruta y entrega el trabajo a un sistema de archivos.
- Información general de ext4 — Creación y montaje de ext4, modos de datos, `commit=` y `ro` con el diario.
- mke2fs(8) — Relación de bytes por inodo, tamaño del inodo, límite de 2038 y etiquetas de volumen.
- e2fsck(8) — Comprobación de ext2, ext3 y ext4, y por qué nunca en un sistema montado.
- fstab(5) — Los seis campos, `LABEL=` y `UUID=` , `nofail` y el orden de comprobaciones al arrancar.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un sistema de archivos?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/file-system/