Engineer

Dateisystem

Struktur für benannte Dateien und Ordner auf einem Medium

Was es ist

Ein Dateisystem ist der Teil eines Computers, der aus einem Datenträger mit bloß nummerierten Byteblöcken benannte Dateien in Verzeichnissen macht. Im Englischen wird es auch zusammengeschrieben, filesystem; diese Schreibweise verwendet die Linux-Kernel-Dokumentation überwiegend. POSIX, der Standard, dem Unix-ähnliche Systeme folgen, definiert es in der Ausgabe von 2024 als „eine Sammlung von Dateien und bestimmten ihrer Attribute“. Dieselbe Definition ergänzt, dass Dateiseriennummern zu einem vom Dateisystem bereitgestellten Namensraum gehören. Diese Nummern identifizieren jede Datei eindeutig innerhalb des Dateisystems, das sie enthält.

Der Rest folgt aus einigen weiteren Definitionen des Standards. Eine Datei ist ein Objekt, das gelesen oder beschrieben werden kann, gegebenenfalls auch beides. Zu seinen Attributen gehören Zugriffsrechte und der Typ: eine reguläre Datei, ein Verzeichnis, ein symbolischer Link, eine FIFO-Datei, ein Socket oder eine zeichen- oder blockorientierte Gerätedatei. Ein Verzeichnis ist eine Datei aus Verzeichniseinträgen, von denen keine zwei im selben Verzeichnis denselben Namen tragen. Ein Verzeichniseintrag, von POSIX auch Hardlink genannt, verknüpft einen Dateinamen mit einer Datei. Mehrere Einträge können dieselbe Datei benennen. Dateiname und Datei selbst werden daher getrennt gehalten. Unter Linux ist der Datensatz der Datei selbst der Inode. Das GNU-df-Handbuch beschreibt ihn als Speicher für Eigentümer, Berechtigungen, Zeitstempel und Lage der Datei im Dateisystem. Die Namen liegen dagegen in den darauf verweisenden Verzeichnissen.

Ein Unix-ähnliches System zeigt alle seine Dateisysteme als einen einzigen Namensbaum. Das erste stellt das Wurzelverzeichnis bereit, jedes weitere wird an einem Verzeichnis des Baums eingehängt, seinem Einhängepunkt. Die Pfade unter diesem Verzeichnis gehören danach zum eingehängten Dateisystem. Im Linux-Kernel sitzt das Virtual File System zwischen Programmen und den vielen

unterstützten Dateisystemen. Die Kernel-Dokumentation beschreibt diese auch Virtual Filesystem Switch genannte Schicht als Dateisystemschnittstelle für Programme, die verschiedene Implementierungen nebeneinander ermöglicht. Ein Programm ruft `open(2)`, `stat(2)`, `read(2)` oder `write(2)` auf. Das VFS löst den Pfad auf und übergibt die Arbeit an das Dateisystem, das die Datei enthält: etwa ext4 auf einer Plattenpartition oder ein Pseudodateisystem, dessen Inodes nur im Speicher existieren. Die vom laufenden Kernel registrierten Typen stehen in `/proc/filesystems`.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Dateisystem?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Server, E-Mail, Konten und Updates werden im Alltag betreut, damit die Dinge, auf die alle angewiesen sind, unauffällig funktionieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 10 PostgreSQL-/MS-SQL-Server auf Ubuntu — 10 bereitgestellte und funktionsfähig gehaltene Datenbankserver

10 PostgreSQL-/MS-SQL-Server auf Ubuntu Kontakt aufnehmen →

Warum es wichtig ist

Fast alles, was ein Server über den nächsten Neustart hinaus behält, liegt in einem Dateisystem: die Datendateien seiner Datenbanken, Logs, hochgeladene Dokumente und Konfigurationen. Ob sie einen Stromausfall, eine volle Platte oder ein defektes Laufwerk überstehen, hängt wesentlich von Entscheidungen beim Anlegen und Einhängen des Dateisystems ab. Manche lassen sich später nicht zurücknehmen.

Beim Anlegen wird Kapazität in zwei Währungen festgelegt. Blöcke enthalten Dateiinhalte, Inodes die Dateien selbst. mke2fs, das Werkzeug zum Anlegen von Dateisystemen der Typen ext2, ext3 und ext4, erzeugt für jeweils eine festgelegte Zahl von Speicherbytes einen Inode. Sein Handbuch warnt, dass dieses Verhältnis von Bytes zu Inodes nach dem Anlegen nicht mehr geändert werden kann. Ein Datenträger kann daher keinen Platz für neue Dateien mehr haben, obwohl noch Platz für deren Inhalte frei ist. Das überrascht jeden, der nur den prozentual belegten Speicher beobachtet.

Das Einhängen entscheidet, was ein Absturz kostet. ext4 führt ein Journal, und seine Dokumentation nennt drei Datenmodi. Der Standard `data=ordered` erzwingt, dass Dateidaten in das eigentliche Dateisystem geschrieben werden,

bevor die auf sie verweisenden Metadaten im Journal festgeschrieben werden. `data=writeback` hebt diese Reihenfolge auf. Die Dokumentation warnt, dass kurz vor einem unsauberen Herunterfahren geschriebene Dateien danach veraltete Daten offenlegen können, was in manchen Fällen ein Sicherheitsrisiko darstellt. `data=journal` leitet auch die Daten durch das Journal. Dieser Modus ist veraltet; seine Unterstützung soll im Januar 2028 entfernt werden. Die Option `errors=` entscheidet, was das Dateisystem beim Erkennen eines eigenen Fehlers tut: schreibgeschützt neu einhängen, weiterlaufen oder mit einer Kernel-Panic den Rechner anhalten.

Auch die Datenrettung hängt davon ab. Ist ein Dateisystem beschädigt, können die Dateiinhalte noch auf der Platte liegen, während die namensgebende Struktur defekt ist. Das ist der Ansatzpunkt der Datenrettung. Der Portfolioeintrag zur Datenrettung von SD-Karten, Festplatten, SSDs und RAID beschreibt, dass jede Medienart anders ausfällt und anders zur Herausgabe ihrer Daten gebracht werden muss. Mac-Dateisysteme bringen eigene Besonderheiten mit. Entscheidend war dort eher das zum jeweils vorliegenden Medium passende Vorgehen als irgendeine einzelne Technik.

Wie es in der Praxis aussieht

Eine neue Partition `/dev/sdb1` soll die Daten eines Dienstes unter `/srv/data` aufnehmen. Drei Befehle reichen, um darauf ein ext4-Dateisystem anzulegen und es von Hand einzuhängen:

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

`-t ext4` nennt wie in der ext4-Dokumentation des Kernels den Typ, und `-L` gibt dem Dateisystem eine Bezeichnung, die mke2fs auf 16 Bytes begrenzt. Alles andere folgt den Standardwerten. In der mit e2fsprogs ausgelieferten Konfiguration belegt beispielsweise jeder Inode 256 Bytes. Zwei `df` -Befehle zeigen danach die beiden Kapazitätsarten:

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

Der erste gibt den Dateisystemtyp sowie belegte und verfügbare Blöcke in Einheiten auf Basis von Potenzen von 1.024 aus. `-H` würde Potenzen von 1.000 verwenden. Der zweite listet statt Blöcken Inodes auf: Gesamtzahl, belegte und verfügbare Anzahl sowie den belegten Prozentsatz.

Eine von Hand vorgenommene Einhängung hält bis zum nächsten Neustart. Dauerhaft wird sie durch eine Zeile in `/etc/fstab`, deren Einträge mount, umount und fsck der Reihe nach lesen:

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

Das erste Feld bezeichnet das Dateisystem über seine Bezeichnung statt als `/dev/sdb1`. Das fstab-Handbuch empfiehlt dort `LABEL=` oder `UUID=`, weil Gerätenamen häufig der zufälligen Erkennungsreihenfolge der Hardware folgen und sich beim Hinzufügen oder Entfernen von Platten ändern können. `defaults` bedeutet gewöhnlich Lesen und Schreiben sowie die übrigen Standardoptionen des Kernels. `nofail` weist mount an, bei einem nicht vorhandenen Gerät keinen Fehler zu melden. Das fünfte Feld liest dump(8); 0 lässt das Dateisystem aus. Das sechste legt die Prüfungsreihenfolge von fsck beim Start fest: 1 für das Wurzeldateisystem, 2 für die übrigen und 0 für keine Prüfung. Auf einem von systemd verwalteten System empfiehlt das Handbuch nach einer Änderung `systemctl daemon-reload`. `mount -a` hängt anschließend jeden nicht mit `noauto` markierten Eintrag ein. Ein Fehler in der neuen Zeile zeigt sich damit, während noch jemand an der Tastatur sitzt.

Zum Betrieb gehört, den freien Platz zu beobachten. Der Portfolioeintrag zum Host-Monitoring mit Totmannschalter nennt neben Arbeitsspeichermangel eine volle Festplatte als einen von zwei realistischen Ausfallgründen eines Rechners mit 464 MB, auf dem eine Forge und ein Webserver laufen. Er weist darauf hin, dass sich keiner der beiden Ausfälle selbst ankündigt. Die Antwort ist ein kleines Skript, das ein Timer viermal pro Stunde jeweils leicht zeitversetzt startet. Es misst freien Plattenplatz und verfügbaren Arbeitsspeicher und pingt einen externen Dienst nur an, solange beide über ihren Schwellenwerten bleiben. Der Alarm entsteht dadurch, dass die Pings ausbleiben. Eine solche Prüfung kann neben der Speicherplatzspalte auch die Inode-Spalte von `df -i` lesen, weil jede zuerst ihre Grenze erreichen kann.

Wo es schiefgeht

Die meisten Dateisystemprobleme zeigen sich als volle Platte, als Rechner, der nicht mehr einhängt, was er gestern einhängte, oder als fehlende Daten nach einem Absturz. Vieles geht auf eine Einstellung zurück, die beim Anlegen des Dateisystems niemand angesehen hat.

Die Platte ist voll, obwohl noch Platz frei ist. Jede Datei braucht einen Inode. mke2fs legt beim Erzeugen des Dateisystems eine feste Anzahl davon an. Ein Datenträger mit Millionen kleiner Dateien kann seine Inodes aufbrauchen, während `df -h` noch freien Platz zeigt. Danach verweigert er jede neu anzulegende Datei. Die Lösung ist, `df -i` neben `df -h` zu beobachten und für einen Datenträger mit vielen kleinen Dateien beim Aufruf von mke2fs das Verhältnis mit `-i` oder die Anzahl mit `-N` festzulegen. Das Verhältnis lässt sich später nicht ändern. Eine Größenänderung des Dateisystems behält es bei und ergänzt oder entfernt entsprechend Inodes.

Eine Prüfung läuft auf einem eingehängten Dateisystem. Laut e2fsck-Handbuch ist das im Allgemeinen nicht sicher. Die einzige Ausnahme ist `-n` : Damit wird das Dateisystem schreibgeschützt geöffnet und jede Frage mit Nein beantwortet. Doch selbst dann sind die ausgegebenen Ergebnisse ungültig, solange das Dateisystem eingehängt ist. Fragt e2fsck, ob auf einem eingehängten Dateisystem fortgefahren werden soll, lautet die Antwort des Handbuchs Nein. Die Lösung ist, es zuerst auszuhängen oder beim Wurzeldateisystem die Prüfung beim Start laufen zu lassen. Dafür sorgt das sechste Feld von fstab.

„Schreibgeschützt“ schreibt auf die Platte. ext4 mit `ro` einzuhängen verhindert Schreibvorgänge nicht: Laut Dokumentation spielt es das Journal erneut ab und schreibt daher trotz schreibgeschützter Einhängung auf die Partition. Besonders wichtig ist das bei einer zu rettenden Platte, auf der jeder Schreibvorgang noch benötigte Daten treffen kann. Die Optionen `ro,noload` verhindern das Schreiben, zu dem auf derselben Seite genannten Preis: Nach einem unsauberen Aushängen hinterlässt das Überspringen der Journal-Wiedergabe ein inkonsistentes Dateisystem. Bei einer Platte, deren Inhalte wichtiger sind als ihre Betriebszeit, lautet die Lösung: mit `ro,noload` einhängen, Benötigtes herauskopieren und das Gelesene als möglicherweise inkonsistent behandeln.

Eine Platte ändert ihren Namen. Eine Zeile zum Einhängen von `/dev/sdb1` hängt ein, was der Kernel heute `/dev/sdb1` nennt. Das fstab-Handbuch warnt, dass sich das beim Hinzufügen oder Entfernen von Platten ändern kann. Eine neue Platte wird dann womöglich am Platz der alten eingehängt, oder die alte fehlt beim Start. Die Lösung sind `LABEL=` oder `UUID=` im ersten Feld. Bezeichnungen werden mit `mke2fs -L` oder später mit einem Werkzeug wie `e2label` gesetzt.

Der Strom fällt zwischen Schreiben und Platte aus. Standardmäßig schreibt ext4 sein Journal alle fünf Sekunden fest. Bei einem Stromausfall können daher die Metadatenänderungen der letzten fünf Sekunden verloren gehen, während das Journal das Dateisystem selbst unbeschädigt hält. Dateidaten können weiter zurückliegen: Bei verzögerter Blockzuweisung beginnt das Schreiben auf den Datenträger erst nach dem in `/proc/sys/vm/dirty_expire_centisecs` gesetzten Intervall. Die Dokumentation bezeichnet Anwendungen, die Dateien ohne `fsync()` ersetzen, als fehlerhaft. ext4 erkennt mit der Standardoption `auto_da_alloc` die zwei üblichen Muster: eine neue Datei schreiben und durch Umbenennen die alte ersetzen oder die vorhandene Datei kürzen und neu schreiben. So werden nach einem Absturz leere Dateien vermieden. Die Lösung liegt im schreibenden Programm: Rufen Sie `fsync()` auf, bevor Sie Daten als gespeichert melden, statt sich auf dieses Sicherheitsnetz zu verlassen.

Zeitstempel enden 2038. Laut mke2fs-Handbuch können Dateisysteme mit 128-Byte-Inodes keine Zeitstempel nach dem 19. Januar 2038 speichern. Inodes mit mindestens 256 Bytes unterstützen dagegen erweiterte Zeitstempel. Neue Dateisysteme erhalten standardmäßig 256 Bytes. Ein altes oder mit `-I 128` erzeugtes kann jedoch noch die kleine Größe haben. Die Lösung ist, diese Dateisysteme zu finden, bevor das Datum sie einholt. Die ext4-Dokumentation zeigt, wie `tune2fs -I 256` ein mit 128-Byte-Inodes erzeugtes Dateisystem umwandelt.

Weiterführende Lektüre

- Definitionen, POSIX.1-2024 (The Open Group Base Specifications Issue 8) — die Definitionen von Dateisystem, Datei, Verzeichnis, Hardlink und Einhängepunkt.
- Überblick über das Linux Virtual File System — wie das VFS des Kernels einen Pfad auflöst und die Arbeit an ein Dateisystem übergibt.
- Allgemeine Informationen zu ext4 — ext4 anlegen und einhängen, seine Datenmodi, `commit=` und `ro` mit dem Journal.

- mke2fs(8) — das Verhältnis von Bytes zu Inodes, Inode-Größe, die Grenze von 2038 und Datenträgerbezeichnungen.
- e2fsck(8) — ext2, ext3 und ext4 prüfen und warum nie im eingehängten Zustand.
- fstab(5) — die sechs Felder, `LABEL=` und `UUID=`, `nofail` und die Reihenfolge der Prüfungen beim Start.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Dateisystem?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/file-system/