Engineer

filsystem

strukturen som gör en disk till namngivna filer och mappar

Vad det är

Ett filsystem är den del av en dator som förvandlar en disk, som på egen hand bara lagrar numrerade block av byte, till namngivna filer ordnade i kataloger. På engelska skrivs det både file system i två ord och filesystem i ett, och det är den senare formen som Linuxkärnans dokumentation oftast använder. POSIX, standarden som Unixliknande system följer, definierar i 2024 års utgåva ett filsystem som en samling filer tillsammans med vissa av deras attribut. Samma definition tillägger att filsystemet ger ett namnutrymme för filernas serienummer, de nummer som entydigt identifierar varje fil inom det filsystem den ligger i.

Resten följer av ytterligare några definitioner i standarden. En fil är ett objekt som går att skriva till, läsa från eller både och, med attribut som åtkomsträttigheter och typ: vanlig fil, katalog, symbolisk länk, FIFO, socket eller tecken- eller blockspecialfil. En katalog är en fil som består av katalogposter, och två poster i samma katalog kan inte ha samma namn. En katalogpost, som POSIX också kallar en hård länk, knyter ett filnamn till en fil, och flera poster kan peka på samma fil. Namnet och själva filen är alltså två skilda saker. På Linux kallas posten för själva filen en inod, som GNU-manualen för df beskriver som platsen där filens ägare, åtkomsträttigheter, tidsstämplar och placering i filsystemet står, medan namnen finns i katalogerna som pekar på den.

Ett Unixliknande system visar alla sina filsystem som ett enda träd av namn. Det första står för rotkatalogen, och vart och ett av de övriga hängs på en katalog i trädet, dess monteringspunkt (mount point), varefter sökvägarna under den katalogen hör till filsystemet som monterats där. Inne i Linuxkärnan sitter Virtual File System, VFS, mellan programmen och de många filsystem kärnan stöder. Kärnans dokumentation beskriver det, som också kallas Virtual Filesystem Switch, som lagret som ger programmen deras gränssnitt mot filsystemen och låter olika implementationer finnas sida vid sida. Ett program anropar `open(2)`, `stat(2)`,

`read(2)` eller `write(2)`, och VFS slår upp sökvägen och lämnar över arbetet till det filsystem som har filen: ext4 på en diskpartition, till exempel, eller ett pseudofilsystem vars inoder bara finns i minnet. Vilka typer den körande kärnan har registrerat står i `/proc/filesystems`.

Använd begreppet i ett beslut

Använd ”Vad är ett filsystem?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Daglig skötsel av servrar, e-post, konton och uppdateringar, så att det som alla förlitar sig på förblir odramatiskt.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu — 10 databasservrar som driftsattes och hölls i gott skick

Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu Hör av er →

Varför det spelar roll

Nästan allt som en server ska minnas efter nästa omstart ligger i ett filsystem: databasernas datafiler, loggarna, uppladdade dokument, konfigurationen. Om de klarar ett strömavbrott, en full disk eller en disk som håller på att ge upp avgörs till stor del av val som gjordes när filsystemet skapades och monterades, och en del av de valen går inte att ändra i efterhand.

Skapandet låser kapaciteten i två valutor. Block rymmer innehållet i filerna, och inoder rymmer själva filerna. mke2fs, verktyget som skapar filsystem av typen ext2, ext3 och ext4, skapar en inod för varje visst antal byte utrymme, och manualen varnar för att det här förhållandet mellan byte och inoder inte går att ändra när filsystemet väl finns. En volym kan därför få slut på plats för nya filer medan det fortfarande finns plats för deras innehåll, och det överraskar alla som bara håller ögonen på hur många procent av utrymmet som används.

Monteringen avgör vad en krasch kostar. ext4 för en journal, och dokumentationen beskriver tre datalägen. Standardläget, `data=ordered`, tvingar ut fildata till själva filsystemet innan metadata som hänvisar till dem skrivs till journalen. `data=writeback` släpper den ordningen, och dokumentationen varnar för att läget efter en oren avstängning kan lämna gamla data synliga i filer som nyligen skrivits. `data=journal` skickar även data genom journalen; det läget är utfasat (deprecated), och stödet ska enligt planen tas bort i januari 2028. Flaggan `errors=` bestämmer vad filsystemet gör när det hittar ett fel i sig självt: monterar om sig skrivskyddat, fortsätter, eller får kärnan att stanna maskinen med panik.

Räddning hänger också på det. När ett filsystem är skadat kan filernas innehåll fortfarande ligga kvar på disken medan strukturen som ger dem namn är trasig, och det är där dataräddning arbetar. Portföljsidan om dataräddning ur minneskort, hårddiskar och diskarrayer berättar att varje medium går sönder på sitt eget vis och kräver sin egen metod för att ge ifrån sig det som finns kvar, och att filsystemen på Mac har egna särdrag. Det som räknades var därför att välja rätt angreppssätt för just det medium som låg på bordet, snarare än att lita på en enda teknik.

Så ser det ut i praktiken

Ta en ny partition, `/dev/sdb1`, som ska rymma en tjänsts data under `/srv/data`. Att skapa ett ext4-filsystem på den och montera det för hand kräver tre kommandon:

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

`-t ext4` anger typen, som i kärnans egen ext4-dokumentation, och `-L` ger filsystemet en etikett (label), som mke2fs begränsar till 16 byte. Allt annat kommer från standardvärdena; i konfigurationen som följer med e2fsprogs tar till exempel varje inod 256 byte. Två `df`-kommandon visar sedan de två sorternas kapacitet:

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

Det första visar filsystemets typ och använda och lediga block i enheter byggda på potenser av 1 024 (`-H` skulle använda potenser av 1 000). Det andra räknar inoder i stället för block: hur många som finns, hur många som är använda och lediga, och hur många procent som används.

En montering för hand håller bara till nästa omstart. För att göra den beständig får filsystemet en rad i `/etc/fstab`, filen som mount, umount och fsck läser i tur och ordning:

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

Det första fältet anger filsystemet med dess etikett i stället för som `/dev/sdb1`. Manualen för fstab rekommenderar `LABEL=` eller `UUID=` där, eftersom enhetsnamn ofta följer den ordning som hårdvaran råkade upptäckas i och kan ändras när diskar läggs till eller tas bort. `defaults` betyder vanligen läsning och skrivning plus kärnans övriga standardval, och `nofail` får mount att avstå från att rapportera ett fel om enheten inte finns. Det femte fältet läses av dump(8), och 0 håller filsystemet utanför. Det sjätte ger fsck ordningen för kontroller vid uppstart: 1 för rotfilsystemet, 2 för de övriga och 0 för ingen. På ett system som körs av systemd rekommenderar manualen `systemctl daemon-reload` efter en ändring. `mount -a` monterar därefter varje rad som inte är märkt `noauto`, så att ett fel i den nya raden syns medan någon fortfarande sitter vid tangentbordet.

Att hålla ett filsystem igång betyder också att hålla koll på utrymmet. Portföljsidan om värdövervakning med dödmansgrepp nämner en disk som blir full som det ena av två realistiska sätt som en maskin på 464 MB med en forge och en webbserver kan gå under på. Det andra är att minnet tar slut, och inget av dem säger ifrån självmant. Svaret där är ett litet skript på en timer, var kvart med slumpmässig förskjutning, som mäter ledigt diskutrymme och tillgängligt minne och bara hör av sig utåt så länge båda håller sig över sina gränser, så att tystnaden blir larmet. En kontroll av det slaget kan lika gärna läsa inodkolumnen från `df -i` som utrymmeskolumnen, för vilken som helst av dem kan nå taket först.

Var det går fel

De flesta problem med filsystem visar sig som en full disk, en maskin som inte vill montera det den monterade i går, eller data som saknas efter en krasch, och mycket av det går att spåra tillbaka till en inställning som ingen tittade på när filsystemet skapades.

Disken är full fast det finns plats. Varje fil behöver en inod, och mke2fs skapar ett fast antal av dem när filsystemet skapas. En volym med miljontals små filer kan göra slut på alla inoder medan `df -h` fortfarande visar ledigt utrymme, och från den stunden går det inte att skapa en enda ny fil. Lösningen är att hålla koll på `df -i` vid sidan av `df -h` och, för en volym som är tänkt för många små filer,

att ange förhållandet med `-i` eller antalet med `-N` när mke2fs körs, eftersom förhållandet inte går att ändra efteråt. Ändras filsystemets storlek behålls förhållandet, och antalet inoder följer med.

En kontroll körs på ett monterat filsystem. Manualen för e2fsck säger att det i allmänhet inte är säkert att köra den på ett monterat filsystem. Det enda undantaget är `-n`, som öppnar filsystemet skrivskyddat och svarar nej på alla frågor, och även då är resultaten den skriver ut inte giltiga medan filsystemet är monterat. Frågar e2fsck om den ska fortsätta på ett monterat filsystem är manualens svar nej. Lösningen är att avmontera först, eller, för rotfilsystemet, att låta kontrollen köras vid uppstart, vilket är vad det sjätte fältet i fstab ordnar.

”Skrivskyddat” skriver till disken. Att montera ext4 med `ro` stoppar inte skrivningen: dokumentationen påpekar att journalen spelas upp igen, och att partitionen därmed skrivs till även när den är monterad skrivskyddat. Det spelar störst roll på en disk som ska räddas, där varje skrivning kan träffa något som fortfarande behövs. Flaggorna `ro,noload` hindrar skrivningen, men samma sida gör priset tydligt: efter en oren avmontering blir filsystemet kvar med inkonsekvenser när uppspelningen hoppas över. Lösningen för en disk vars innehåll betyder mer än dess drifttid är att montera den med `ro,noload`, kopiera ut det som behövs och behandla det som läses som möjligen inkonsekvent.

En disk byter namn. En rad som monterar `/dev/sdb1` monterar det som kärnan kallar `/dev/sdb1` i dag, och fstab-manualen varnar för att det kan ändras när diskar läggs till eller tas bort. Då kan en ny disk monteras där den gamla hör hemma, eller den gamla saknas vid uppstart. Lösningen är `LABEL=` eller `UUID=` i första fältet, med etiketter satta av `mke2fs -L` eller senare av ett verktyg som `e2label`.

Strömmen går mellan skrivning och disk. Som standard skriver ext4 sin journal var femte sekund, så ett strömavbrott kan kosta de senaste fem sekundernas ändringar i metadata, medan journalen håller själva filsystemet oskadat. Fildata kan gå längre tillbaka: med fördröjd allokering börjar utskrivningen först efter intervallet som är satt i `/proc/sys/vm/dirty_expire_centisecs`. Dokumentationen kallar program som ersätter filer utan `fsync()` för trasiga, och ext4:s standardval `auto_da_alloc` känner igen de två vanliga mönstren, att skriva en ny fil och byta namn på den över den gamla eller att korta en fil och skriva om den, för att undvika filer som kommer tillbaka tomma efter en krasch. Lösningen är att rätta programmet som skriver filen: det ska anropa `fsync()` innan någon får veta att data är sparade, i stället för att luta sig mot det skyddsnätet.

Tidsstämplarna tar slut 2038. Manualen för mke2fs säger att filsystem med inoder på 128 byte inte stöder tidsstämplar efter den 19 januari 2038, medan inoder på 256 byte eller mer stöder utökade tidsstämplar. Nya filsystem får 256 byte som standard, men ett gammalt filsystem, eller ett som skapats med `-I 128`, kan fortfarande ha den lilla storleken. Lösningen är att hitta de filsystemen innan datumet gör det; ext4-dokumentationen visar `tune2fs -I 256` konvertera ett filsystem som skapats med inoder på 128 byte.

Läs mer

- Definitions, POSIX.1-2024 (The Open Group Base Specifications Issue 8) — definitionerna av filsystem, fil, katalog, hård länk och monteringspunkt.
- Overview of the Linux Virtual File System — hur kärnans VFS slår upp en sökväg och lämnar över arbetet till ett filsystem.
- ext4 General Information — att skapa och montera ext4, datalägena, `commit=` och `ro` med journalen.
- mke2fs(8) — förhållandet mellan byte och inoder, inodstorleken, gränsen 2038 och etiketter.
- e2fsck(8) — kontroll av ext2, ext3 och ext4, och varför aldrig på ett monterat filsystem.
- fstab(5) — de sex fälten, `LABEL=` och `UUID=`, `nofail` och ordningen för kontroller vid uppstart.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ett filsystem?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/file-system/