**Engineer**

# filsystem

strukturen som gjør en disk om til navngitte filer og mapper

## Hva det er

Et filsystem er den delen av en datamaskin som gjør en disk, som i seg selv bare lagrer nummererte blokker med byte, om til navngitte filer ordnet i kataloger. På engelsk skrives det både file system i to ord og filesystem i ett, og det er den siste formen Linux-kjernens dokumentasjon stort sett bruker. POSIX, standarden Unix-lignende systemer følger, definerer i 2024-utgaven et filsystem som en samling filer og visse av attributtene deres. Den samme definisjonen legger til at filsystemet gir et navnerom for filenes serienumre, numrene som entydig identifiserer hver fil innenfor filsystemet den ligger i.

Resten følger av noen flere definisjoner i standarden. En fil er et objekt som kan skrives til, leses fra eller begge deler, med attributter som tilgangsrettigheter og type: vanlig fil, katalog, symbolsk lenke, FIFO, socket eller tegn- eller blokkspesialfil. En katalog er en fil som består av katalogoppføringer, og to oppføringer i samme katalog kan ikke ha samme navn. En katalogoppføring, som POSIX også kaller en hard lenke, knytter et filnavn til en fil, og flere oppføringer kan peke på den samme filen. Navnet og selve filen er altså to forskjellige ting. På Linux heter posten for selve filen en inode, som GNU-manualen for df beskriver som stedet der filens eier, tilgangsrettigheter, tidsstempler og plassering i filsystemet står, mens navnene bor i katalogene som peker på den.

Et Unix-lignende system viser alle filsystemene sine som ett samlet tre av navn. Det første leverer rotkatalogen, og hvert av de andre henges på en katalog i treet, monteringspunktet (mount point), og deretter hører stiene under den katalogen til filsystemet som er montert der. Inne i Linux-kjernen ligger Virtual File System, VFS, mellom programmene og de mange filsystemene kjernen støtter. Kjernens dokumentasjon beskriver det, også kjent som Virtual Filesystem Switch, som laget som gir programmene grensesnittet deres mot filsystemene og lar ulike implementasjoner eksistere side om side. Et program kaller `open(2)`, `stat(2)`, `read(2)` eller `write(2)`, og VFS slår opp stien og sender arbeidet

videre til filsystemet som har filen: ext4 på en diskpartisjon, for eksempel, eller et pseudofilsystem der inodene bare finnes i minnet. Typene den kjørende kjernen har registrert, står i `/proc/filesystems` .

Bruk begrepet i en beslutning

Bruk «Hva er et filsystem?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Daglig ettersyn av servere, e-post, kontoer og oppdateringer, slik at det alle er avhengige av, forblir kjedelig.

**Se tjenestene våre** Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu — 10 databaseservere produksjonssatt og holdt i god stand

**Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu** Ta kontakt →

## Hvorfor det er viktig

Nesten alt en server skal huske etter neste omstart, ligger i et filsystem: datafilene til databasene, loggfilene, opplastede dokumenter, konfigurasjonen. Om de overlever et strømbrudd, en full disk eller en disk som holder på å gi opp, avgjøres i stor grad av valg som ble tatt da filsystemet ble opprettet og montert, og noen av dem kan ikke gjøres om senere.

Opprettelsen fastsetter kapasiteten i to valutaer. Blokker rommer innholdet i filene, og inoder rommer selve filene. mke2fs, verktøyet som oppretter ext2-, ext3- og ext4-filsystemer, lager én inode for hver så og så mange byte med plass, og manualen advarer om at dette forholdet mellom byte og inoder ikke kan endres når filsystemet først finnes. Et volum kan derfor gå tom for plass til nye filer mens det fortsatt er plass til innholdet deres, og det overrasker alle som bare følger med på hvor mange prosent av plassen som er brukt.

Monteringen avgjør hva et krasj koster. ext4 fører en journal, og dokumentasjonen beskriver tre datamoduser. Standarden, `data=ordered` , tvinger fildata ut i selve filsystemet før metadataene som viser til dem, skrives til journalen. `data=writeback` dropper den rekkefølgen, og dokumentasjonen advarer om at den etter en uren nedstengning kan la gamle data bli synlige i filer som nylig er skrevet. `data=journal` sender også dataene gjennom journalen; den modusen frarådes (deprecated), og støtten skal etter planen fjernes i januar 2028. Valget `errors=` bestemmer hva filsystemet gjør når det finner en feil i seg selv: monterer seg på nytt skrivebeskyttet, fortsetter, eller går i panikk og stanser maskinen.

Gjenoppretting henger også på det. Når et filsystem er skadet, kan innholdet i filene fortsatt ligge på disken mens strukturen som gir dem navn, er ødelagt, og det er der datagjenoppretting arbeider. Porteføljesiden om datagjenoppretting fra SD-kort, HDD, SSD og RAID forteller at hvert medium svikter på sitt eget vis og også må få dataene lokket ut på sitt eget vis, og at et Mac-filsystem har egne særheter. Det viktigste var derfor å velge riktig fremgangsmåte for nettopp det mediet som lå på bordet, mer enn å stole på én bestemt teknikk.

## Slik ser det ut i praksis

Ta en ny partisjon, `/dev/sdb1`, som skal romme dataene til en tjeneste under `/srv/data`. Å opprette et ext4-filsystem på den og montere det for hånd krever tre kommandoer:

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

`-t ext4` angir typen, som i kjernens egen ext4-dokumentasjon, og `-L` gir filsystemet en etikett (label), som mke2fs begrenser til 16 byte. Resten kommer fra standardverdiene; i konfigurasjonen som følger med e2fsprogs, tar for eksempel hver inode 256 byte. To `df` -kommandoer viser deretter de to typene kapasitet:

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

Den første viser filsystemets type og brukte og ledige blokker i enheter bygd på potenser av 1 024 ( `-H` ville brukt potenser av 1 000). Den andre teller inoder i stedet for blokker: hvor mange det er, hvor mange som er brukt og ledige, og hvor mange prosent som er i bruk.

En montering for hånd varer bare til neste omstart. For å gjøre den permanent får filsystemet en linje i `/etc/fstab`, filen som mount, umount og fsck leser i rekkefølge:

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

Det første feltet angir filsystemet ved etiketten i stedet for som `/dev/sdb1`. Manualen for fstab anbefaler `LABEL=` eller `UUID=` der, fordi enhetsnavn ofte følger rekkefølgen maskinvaren tilfeldigvis ble oppdaget i, og kan endre seg når disker legges til eller fjernes. `defaults` betyr vanligvis lese og skrive pluss kjernens øvrige standardvalg, og `nofail` får mount til å la være å melde feil hvis enheten ikke finnes. Det femte feltet leses av dump(8), og 0 holder filsystemet utenfor. Det sjette gir fsck rekkefølgen for sjekker ved oppstart: 1 for rotfilsystemet, 2 for de andre og 0 for ingen. På et system som kjører systemd, anbefaler manualen `systemctl daemon-reload` etter en endring. `mount -a` monterer deretter alle linjer som ikke er merket `noauto`, slik at en feil i den nye linjen viser seg mens noen fortsatt sitter ved tastaturet.

Å holde et filsystem i gang betyr også å følge med på plassen. Porteføljesiden om vertsovervåking med dødmannsknapp nevner en disk som blir full, som den ene av to realistiske måter en maskin på 464 MB med en forge og en webserver kan dø på. Den andre er mangel på minne, og ingen av dem sier fra av seg selv. Svaret der er et lite skript på en timer, hvert kvarter med jitter, som måler ledig diskplass og tilgjengelig minne og bare gir lyd fra seg utad så lenge begge holder seg over grensene sine, slik at stillheten er alarmen. En sjekk av den typen kan like gjerne lese inode-kolonnen fra `df -i` som plasskolonnen, for begge kan nå taket først.

## Hvor det går galt

De fleste problemer med filsystemer viser seg som en full disk, en maskin som ikke vil montere det den monterte i går, eller data som mangler etter et krasj, og mye av det kan spores tilbake til en innstilling ingen så på da filsystemet ble laget.

**Disken er full selv om det er plass.** Hver fil trenger en inode, og mke2fs lager et fast antall av dem når filsystemet opprettes. Et volum med millioner av små filer kan bruke opp alle inodene mens `df -h` fortsatt viser ledig plass, og fra da av kan det ikke opprettes en eneste ny fil. Løsningen er å følge med på `df -i` ved siden av `df -h` og, for et volum som er ment for mange små filer, å sette forholdet med `-i` eller antallet med `-N` når mke2fs kjøres, for forholdet kan ikke endres etterpå. Endres størrelsen på filsystemet, beholdes forholdet, og antallet inoder følger med.

**En sjekk kjører på et montert filsystem.** Manualen for e2fsck sier at det generelt ikke er trygt å kjøre den på et montert filsystem. Det eneste unntaket er `-n`, som åpner filsystemet skrivebeskyttet og svarer nei på alle spørsmål, og

selv da er resultatene den skriver ut, ikke gyldige mens filsystemet er montert. Spør e2fsck om den skal fortsette på et montert filsystem, er manualens svar nei. Løsningen er å avmontere først, eller, for rotfilsystemet, å la sjekken kjøre ved oppstart, slik det sjette feltet i fstab sørger for.

**«Skrivebeskyttet» skriver til disken.** Å montere ext4 med `ro` stopper ikke skrivingen: dokumentasjonen påpeker at journalen spilles av på nytt, og at det dermed skrives til partisjonen også når den er montert skrivebeskyttet. Det betyr mest på en disk som skal reddes, der hver skriving kan treffe noe som fortsatt trengs. Valgene `ro,noload` hindrer skrivingen, men samme side gjør prisen tydelig: etter en uren avmontering blir filsystemet stående med inkonsistenser når avspillingen hoppes over. Løsningen for en disk der innholdet betyr mer enn oppetiden, er å montere den med `ro,noload`, kopiere ut det som trengs, og behandle det som leses, som mulig inkonsistent.

**En disk bytter navn.** En linje som monterer `/dev/sdb1`, monterer det kjernen kaller `/dev/sdb1` i dag, og fstab-manualen advarer om at det kan endre seg når disker legges til eller fjernes. Da kan en ny disk bli montert der den gamle hører hjemme, eller den gamle kan mangle ved oppstart. Løsningen er `LABEL=` eller `UUID=` i første felt, med etiketter satt av `mke2fs -L` eller senere av et verktøy som `e2label`.

**Strømmen går mellom skriving og disk.** Som standard skriver ext4 journalen sin hvert femte sekund, så et strømbrudd kan koste de siste fem sekundene med endringer i metadata, mens journalen holder selve filsystemet uskadd. Fildata kan gå lenger tilbake: med forsinket allokering begynner utskrivingen først etter intervallet som er satt i `/proc/sys/vm/dirty_expire_centisecs`. Dokumentasjonen kaller programmer som erstatter filer uten `fsync()`, for ødelagte, og ext4s standardvalg `auto_da_alloc` kjenner igjen de to vanlige mønstrene, å skrive en ny fil og gi den nytt navn over den gamle eller å kutte en fil og skrive den på nytt, for å unngå filer som kommer tomme tilbake etter et krasj. Løsningen er å rette programmet som skriver filen: det skal kalle `fsync()` før noen får vite at dataene er lagret, i stedet for å lene seg på det sikkerhetsnettet.

**Tidsstemplene stopper i 2038.** Manualen for mke2fs sier at filsystemer med inoder på 128 byte ikke støtter tidsstempler etter 19. januar 2038, mens inoder på 256 byte eller mer støtter utvidede tidsstempler. Nye filsystemer får 256 byte som standard, men et gammelt filsystem, eller et som er laget med `-I 128`, kan

fortsatt ha den lille størrelsen. Løsningen er å finne de filsystemene før datoen gjør det; ext4-dokumentasjonen viser `tune2fs -I 256` konvertere et filsystem som er opprettet med inoder på 128 byte.

## Les mer

- Definitions, POSIX.1-2024 (The Open Group Base Specifications Issue 8) — definisjonene av filsystem, fil, katalog, hard lenke og monteringspunkt.
- Overview of the Linux Virtual File System — hvordan kjernens VFS slår opp en sti og overlater arbeidet til et filsystem.
- ext4 General Information — opprettelse og montering av ext4, datamodusene, `commit=` og `ro` med journalen.
- mke2fs(8) — forholdet mellom byte og inoder, inodestørrelsen, grensen i 2038 og etiketter.
- e2fsck(8) — sjekk av ext2, ext3 og ext4, og hvorfor aldri på et montert filsystem.
- fstab(5) — de seks feltene, `LABEL=` og `UUID=`, `nofail` og rekkefølgen for sjekker ved oppstart.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

**Se hva det koster** Ta kontakt →

# Gjør begrepene praktiske

Spør hvordan «Hva er et filsystem?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/file-system/