Engineer

Système de fichiers

organisation d'un disque en fichiers et dossiers nommés

Ce que c’est

Un système de fichiers est la partie d’un ordinateur qui transforme un disque, lequel ne conserve à lui seul que des blocs d’octets numérotés, en fichiers nommés organisés dans des répertoires. En anglais, on l’écrit aussi en un seul mot, filesystem, l’orthographe la plus courante dans la documentation du noyau Linux. POSIX, la norme suivie par les systèmes de type Unix, le définit dans son édition 2024 comme « un ensemble de fichiers et de certains de leurs attributs ». La même définition ajoute que les numéros de série des fichiers appartiennent à un espace de noms fourni par le système de fichiers : ces numéros identifient chaque fichier de façon unique à l’intérieur du système de fichiers qui le contient.

La suite découle de quelques autres définitions de la norme. Un fichier est un objet permettant l’écriture, la lecture ou les deux, avec des attributs tels que ses droits d’accès et son type : fichier ordinaire, répertoire, lien symbolique, FIFO, socket, fichier spécial de caractères ou fichier spécial de blocs. Un répertoire est un fichier composé d’entrées de répertoire dont les noms sont tous distincts au sein de ce répertoire. Une entrée de répertoire, que POSIX appelle aussi lien physique, associe un nom à un fichier, et plusieurs entrées peuvent nommer le même fichier. Le nom d’un fichier et le fichier lui-même restent donc séparés. Sous Linux, l’enregistrement du fichier lui-même est l’inode : selon le manuel GNU de df, il contient le propriétaire, les droits, les horodatages et l’emplacement du fichier dans le système de fichiers. Les noms restent dans les répertoires qui pointent vers lui.

Un système de type Unix présente tous ses systèmes de fichiers comme une seule arborescence de noms. Le premier fournit le répertoire racine. Chaque autre système est attaché à un répertoire de l’arborescence, son point de montage, après quoi les chemins sous ce répertoire appartiennent au système de fichiers attaché. Dans le noyau Linux, le Virtual File System se place entre les programmes et les nombreux systèmes de fichiers pris en charge. La

documentation du noyau le décrit, sous ce nom ou celui de Virtual Filesystem Switch, comme la couche qui fournit aux programmes leur interface de système de fichiers et permet à différentes implémentations de coexister. Un programme appelle `open(2)`, `stat(2)`, `read(2)` ou `write(2)`, et le VFS résout le chemin puis confie le travail au système de fichiers qui contient le fichier : ext4 sur une partition de disque, par exemple, ou un pseudo-système de fichiers dont les inodes n'existent qu'en mémoire. Les types enregistrés par le noyau en cours d'exécution sont énumérés dans `/proc/filesystems`.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un système de fichiers ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. L'entretien quotidien des serveurs, de la messagerie, des comptes et des mises à jour, pour que les outils indispensables restent sans histoire.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 10 serveurs PostgreSQL et MS SQL sur Ubuntu — 10 serveurs de bases de données déployés et maintenus

10 serveurs PostgreSQL et MS SQL sur Ubuntu Contactez-nous →

Pourquoi c'est important

Presque tout ce qu'un serveur conserve au-delà de son prochain redémarrage est dans un système de fichiers : les fichiers de données de ses bases, ses journaux, les documents téléversés, sa configuration. Leur survie à une coupure de courant, à un disque plein ou à un support défaillant dépend largement des choix faits à la création et au montage du système de fichiers. Certains de ces choix ne peuvent pas être annulés ensuite.

À la création, la capacité se fixe dans deux « monnaies » différentes. Les blocs contiennent le contenu des fichiers et les inodes représentent les fichiers eux-mêmes. mke2fs, l'outil qui crée les systèmes de fichiers ext2, ext3 et ext4, crée un inode pour un certain nombre d'octets d'espace. Son manuel avertit que ce rapport d'octets par inode ne peut plus être changé une fois le système de fichiers créé. Un volume peut donc manquer de place pour de nouveaux fichiers tout en ayant encore de la place pour leur contenu, ce qui surprend quiconque ne surveille que le pourcentage d'espace occupé.

Le montage détermine ce qu'un plantage peut faire perdre. ext4 conserve un journal, et sa documentation distingue trois modes de traitement des données. Le mode préconfiguré, `data=ordered`, impose l'écriture des données dans le

système de fichiers principal avant la validation dans le journal des métadonnées qui les référencent. `data=writeback` abandonne cet ordre. La documentation avertit qu'après un arrêt incorrect, les fichiers écrits peu avant peuvent exposer des données anciennes, avec un risque de sécurité dans certains cas. `data=journal` fait aussi passer les données par le journal ; ce mode est déprécié et sa prise en charge doit être retirée en janvier 2028. L'option `errors=` décide de la réaction lorsqu'une erreur est détectée dans le système de fichiers lui-même : le remonter en lecture seule, continuer ou déclencher une panique du noyau et arrêter la machine.

La récupération en dépend aussi. Lorsqu'un système de fichiers est endommagé, le contenu de ses fichiers peut encore être sur le disque alors que la structure qui leur donne un nom est cassée. C'est le point de départ de la récupération de données. L'entrée du portfolio sur la récupération de données de cartes SD, disques durs, SSD et RAID explique que chaque type de support tombe en panne différemment et exige une manière différente d'en extraire les données, et que les systèmes de fichiers Mac ont leurs particularités. Le choix de l'approche adaptée au support posé sur l'établi comptait davantage que le recours à une technique en particulier.

À quoi cela ressemble en pratique

Prenez une nouvelle partition, `/dev/sdb1`, destinée à contenir les données d'un service sous `/srv/data`. Trois commandes suffisent pour y créer un système de fichiers ext4 et le monter à la main :

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

`-t ext4` indique le type, comme dans la documentation ext4 du noyau, et `-L` attribue au système de fichiers une étiquette, dont mke2fs limite la longueur à 16 octets. Tout le reste vient des réglages prédéfinis ; dans la configuration fournie avec e2fsprogs, par exemple, chaque inode occupe 256 octets. Deux commandes `df` montrent ensuite les deux types de capacité :

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

La première affiche le type du système de fichiers, ses blocs utilisés et disponibles, avec des unités fondées sur des puissances de 1 024. `-H` utiliserait des puissances de 1 000. La seconde affiche les inodes à la place des blocs : leur nombre total, ceux utilisés et disponibles et le pourcentage utilisé.

Un montage fait à la main dure jusqu'au prochain redémarrage. Pour le rendre permanent, ajoutez une ligne pour ce système de fichiers dans `/etc/fstab`, le fichier dont mount, umount et fsck lisent les entrées dans l'ordre :

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

Le premier champ désigne le système de fichiers par son étiquette plutôt que par `/dev/sdb1`. Le manuel de fstab y recommande `LABEL=` ou `UUID=`, car les noms des périphériques suivent souvent l'ordre dans lequel le matériel a été détecté et peuvent changer lorsqu'on ajoute ou retire des disques. `defaults` signifie généralement lecture-écriture et les autres options standard du noyau, tandis que `nofail` demande à mount de ne pas signaler d'erreur si le périphérique n'existe pas. Le cinquième champ est lu par dump(8), et 0 exclut le système de fichiers de ses sauvegardes. Le sixième donne à fsck l'ordre des vérifications au démarrage : 1 pour le système de fichiers racine, 2 pour les autres et 0 pour aucune vérification. Sur un système géré par systemd, le manuel recommande `systemctl daemon-reload` après une modification. `mount -a` monte ensuite toutes les entrées qui ne portent pas `noauto`, ce qui révèle une erreur dans la nouvelle ligne tant que quelqu'un est encore au clavier.

Surveiller l'espace fait partie de l'exploitation. L'entrée du portfolio sur la surveillance des hôtes par un mécanisme d'alerte en cas de silence cite le manque d'espace disque comme l'une des deux causes réalistes de panne d'une machine de 464 Mo hébergeant une forge et un serveur web, avec le manque de mémoire. Aucune de ces pannes ne s'annonce elle-même. La réponse est un petit script lancé par un minuteur quatre fois par heure, chaque fois à un moment légèrement décalé, qui mesure l'espace disque libre et la mémoire disponible. Il n'envoie un signal à un service externe que si les deux restent au-dessus de leurs seuils : c'est donc l'arrêt des signaux qui déclenche l'alarme. Une vérification de cette forme peut lire la colonne des inodes de `df -i` en plus de celle de l'espace, car l'une comme l'autre peut atteindre sa limite en premier.

Où les problèmes surviennent

La plupart des problèmes de système de fichiers se présentent comme un disque plein, une machine incapable de monter ce qu'elle montait la veille ou des données manquantes après un plantage. Beaucoup remontent à un réglage que personne n'a examiné lors de la création du système de fichiers.

Le disque est plein alors qu'il reste de la place. Chaque fichier a besoin d'un inode, et mke2fs en crée un nombre fixe lors de la création du système de fichiers. Un volume contenant des millions de petits fichiers peut épuiser ses inodes alors que `df -h` affiche encore de l'espace libre. Il refuse alors tout nouveau fichier qu'on lui demande de créer. Surveillez `df -i` à côté de `df -h` et, pour un volume destiné à de nombreux petits fichiers, fixez le rapport avec `-i` ou le nombre avec `-N` lors de l'exécution de mke2fs, puisque le rapport ne peut plus changer ensuite. Redimensionner le système de fichiers conserve ce rapport et ajoute ou retire des inodes en conséquence.

Une vérification s'exécute sur un système de fichiers monté. Le manuel d'e2fsck indique que l'exécuter sur un système de fichiers monté n'est généralement pas sûr. La seule exception est `-n`, qui l'ouvre en lecture seule et répond non à chaque question ; même alors, les résultats affichés ne sont pas valides tant que le système de fichiers est monté. Quand e2fsck demande s'il faut poursuivre sur un système de fichiers monté, la réponse du manuel est non. Démontez-le d'abord ou, pour le système de fichiers racine, laissez la vérification se faire au démarrage, ce qu'organise le sixième champ de fstab.

La lecture seule écrit sur le disque. Monter ext4 avec `ro` ne l'empêche pas d'écrire : la documentation indique qu'il rejoue le journal et écrit donc sur la partition, malgré le montage en lecture seule. C'est particulièrement important pour un disque en cours de récupération, où toute écriture peut tomber sur quelque chose d'encore nécessaire. Les options `ro,noload` empêchent ces écritures, au prix indiqué sur la même page : après un démontage incorrect, ne pas rejouer le journal laisse le système de fichiers avec des incohérences. Pour un disque dont le contenu compte plus que la disponibilité, montez-le avec `ro,noload`, copiez ce dont vous avez besoin vers un autre support et considérez que les données lues peuvent être incohérentes.

Un disque change de nom. Une ligne qui monte `/dev/sdb1` monte ce que le noyau appelle `/dev/sdb1` aujourd'hui. Le manuel de fstab avertit que cela peut changer lorsqu'on ajoute ou retire des disques. Un nouveau disque peut alors

être monté à la place de l'ancien, ou l'ancien manquer au démarrage. Utilisez `LABEL=` ou `UUID=` dans le premier champ, avec des étiquettes définies par `mke2fs -L` ou, ensuite, par un outil tel qu' `e2label` .

Le courant se coupe entre l'écriture et le disque. Sans réglage contraire, ext4 valide son journal toutes les cinq secondes. Une coupure peut donc faire perdre les cinq dernières secondes de modifications des métadonnées, tandis que le journal préserve l'intégrité du système de fichiers lui-même. Pour les données des fichiers, la perte peut remonter plus loin : avec l'allocation différée, leur écriture sur disque ne commence qu'après l'intervalle fixé dans `/proc/sys/vm/dirty_expire_centisecs` . La documentation qualifie de défectueuses les applications qui remplacent des fichiers sans `fsync()` . L'option `auto_da_alloc` d'ext4, activée sans réglage contraire, détecte les deux schémas habituels, écrire un nouveau fichier puis le renommer à la place de l'ancien, ou tronquer puis réécrire sur place, pour éviter de retrouver des fichiers vides après un plantage. La correction relève du programme qui écrit : appelez `fsync()` avant de dire que les données sont enregistrées, au lieu de vous reposer sur ce filet de sécurité.

Les horodatages s'arrêtent en 2038. Le manuel de mke2fs indique que les systèmes de fichiers à inodes de 128 octets ne peuvent pas conserver d'horodatages après le 19 janvier 2038, tandis que des inodes de 256 octets ou plus prennent en charge les horodatages étendus. Les nouveaux systèmes utilisent sans option contraire des inodes de 256 octets, mais un ancien système, ou un système créé avec `-I 128` , peut encore utiliser des inodes de petite taille. Repérez ces systèmes de fichiers avant que l'échéance ne vous rattrape ; la documentation d'ext4 montre comment `tune2fs -I 256` convertit un système de fichiers créé avec des inodes de 128 octets.

Pour aller plus loin

- Définitions, POSIX.1-2024, spécifications de base de l'Open Group, édition 8 — les définitions de système de fichiers, fichier, répertoire, lien physique et point de montage.
- Vue d'ensemble du Virtual File System de Linux — la manière dont le VFS du noyau résout un chemin et confie le travail à un système de fichiers.
- Informations générales sur ext4 — la création et le montage d'ext4, ses modes de données, `commit=` et `ro` avec le journal.
- mke2fs(8) — le rapport d'octets par inode, la taille des inodes, la limite de 2038 et les étiquettes de volume.

- e2fsck(8) — la vérification d'ext2, ext3 et ext4, et pourquoi ne jamais la faire sur un système de fichiers monté.
- fstab(5) — les six champs, `LABEL=` et `UUID=`, `nofail` et l'ordre des vérifications au démarrage.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un système de fichiers ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

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

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/file-system/