Informazioni aggiuntive su file system e Linux

Introduzione al file system V7 di Linux

Prime versioni di UNIX avevano un file system multiutente derivato da MULTICS chiamato V7, implementato nel PDP-11.

La struttura era quella di un albero che nasce dalla directory principale, con aggiunta di link, a formare un grafp aciclico orientato.

> **Nomi dei file**
>
> Massimo 14 caratteri e non potevano usare il carattere '/' (Separatore dei nomi nel percorso) e NUL (Usato per riempire i nomi con meno di 14 caratteri).

Una voce di directory aveva soli due campi:

- **Nome del file (14 byte)
- **Numero degli I-node del file (2 byte)

Questi parametri limitano il numero di file per file system a 64K.

Gli I-node di UNIX contengono alcuni attributi tipo:

- Dimensione.
- Tre orari (Creazione, Ultimo accesso, Ultima modifica).
- Proprietario.
- Gruppo.
- **Numero di voci di directory che puntano a quel file.**
- Ecc....

Se viene creato un link ad un I-node, il suo contatore viene incrementato, al contrario se viene rimosso decrementa.

Se raggiunge 0, l'I-node viene riciclato e i suoi blocchi del disco tornano nella lista di blocchi liberi.

Per tenere traccia dei blocchi su disco, i primi 10 indirizzi del disco erano memorizzati nell'I-node del file, in questo modo:

- File piccoli --> Tutte le informazioni erano negli I-node, venivano presi e messi in memoria.
- File grandi --> Uno degli indirizzi nell'I-node era l'indirizzo di un blocco chiamato **blocco indiretto singolo**, contenente altri indirizzi del disco.

Si possono usare anche blocchi indiretti doppi e tripli.

Esempio di ricerca del percorso usr/ast/mbox

Il file system individua la directory radice tramite un I-node che sta sul disco.
Legge la directory radice è cerca il primo componente del percorso, per individuare il numero di I-node del file. Da questo I-node troviamo la directory usr in cui poi cerca la componente successiva nello stesso modo.

Gestioni directory speciali

Usiamo . e .. per indicare la directory corrente e precedente.

Minimizzazione d'acceso tramite caching

Usiamo una **block cache** o **buffer cache** per ridurre i tempi di accesso al disco.
Una tecnica di allocazione inteliggente è quella di allocare blocchi vicini nello stesso cilindro per minimizzare il movimento della testina, altrimenti si può usare una bitmap.

Concetti importanti del caching:

- **Buffer cache** --> Memorizza i blocchi del disco sulla cache.
- **Page cache** --> Memorizza le pagine del file system virtuale in RAM prima di passare al driver del dispositivo.

Ottimiziamo tutto combinando le due cache in un'unica.

Comando free in Linux

Uno strumento essenziale per monitorare l'utilizzo della memoria in Linux è il comando *free*.
Che fornisce una panoramica della RAM includendo cache e spazio libero.

Funzionamento:

```
free -h
```

Output leggibile che include dimensione e utilizzo della cache.

La colonna **"buff/cache"** rivela quanto spazio è utilizzato per buffer, inclusi i dati letti dal disco.

La colonna **"free"** mostra la memoria fisica non utilizzata attualmente.

La colonna **"Available"** stima la memoria disponibile per nuovi processi, considerando anche la memoria facilemente liberabile come cache, offre una visione realistica della memoria effettivamente disponibile per applicazioni senza influire sulle prestazioni.

> **Ricorda**
>
> La cache aiuta a velocizzare l'accesso ai file frequentemente usati, riducendo la necessità di leggere ripetutamente dal disco più lento.

rsync

Comando usato nei sistemi UNIX per la **sincronizzazione di file e cartelle tra due location diverse**, sia una stessa macchina che su macchine diverse.

Ottimizza il traferimento dei dati trasmettendo solo le parti di file che sono state modificate.
Usate principalemente per backup, ripristino e sincronizzazione di dati in ambienti di rete.

Quindi:

- Trasferisce solo le differenze tra le sorgetni e le destinazioni.
- Support la copia di link, dispositivi, attributi, permessi, dati utente e gruppo.
- Può utilizzare SSH per trasferimenti cripati.

Funzionamento:

```
rsync -av /sorgente/cartella
/destionazione/cartella
```

- -a: Modalità archivio, mantiene i peremssi e la struttura del file.
- -v: Modalità verbosa, mostra i dettagli del trasferimento.

```
rsync -av /sorgente/cartella
user@remoto: /destionazione/cartella
```

Sincronizza la cartella dalla macchina locale a quella remota tramite l'account utente.

RAID

Nel corso degli il gap fra CPU e dischi magnetici ha continuato ad allargarsi.
L'idea di migliorare le prestazioni tramite il **parallelismo** era perfetta sia per CPU che per dispositivi di I/O.

1988 Patterson --> Introduce sei organizzazioni di dischi che avrebbero migliorato prestioni e affidabilità di questi.

Le idee vennero adottate rapidamente facendo nascere una nuova classe di dispositivi I/O: i **RAID (Redundant Array of Inexpensive/Independent disks)** che combattevano i **SLED(Single Large Expnsive Disks)**.

Idea del RAID --> Installare contenitore pieno di dischi accanto al computer, sostituire il controller dei dischi con uno RAID e copiare i dati sul RAID. Il RAID verrà visto come un SLED dal SO. Oggi i RAID supportano anche unità SATA e SSD.

Esistono 7 livelli di RAID.

- **RAID livello 0** -> Distribuzione dei dati in strip, di k settori ciasuna, su dischi multipli , migliora le prestazioni con richieste grandi, nessuna ridonanda quindi meno affidabile.
- **RAID livello 1** -> Duplicazione dei dischi per tolleranza agli errori, prestazioni di lettura migliorate, ma non quelle di scrittura perché dobbiamo scrivere su più dischi.
- **RAID livello 2** -> Opera sulle parole o byte con **codice di Hamming**.
- **RAID livello 3** -> Usa un singolo bit di parità (Inserito in un'unità di parità) per la parola richiede la sincronizzazione delle unità.
- **RAID livello 4** -> Utilizzo di strip con unità extra per la parità, se un'unità va in crash si possono ricalcolare i byte persi tramite l'unità di parità.
- **RAID livello 5** -> Distribuisce i bit di parità in modo uniforme su tutte le unità.
- **RAID livello 6** -> Simile al 5, ma con un blocco di parità in più per una maggiore tolleranza agli errori.

Questi schemi si possono combinare tra di loro.

> **RAID 0+1**
>
> Duplicazione ridondante come nel RAID 1 di dischi associati allo schema 0.

Nella situazione in cui un disco (RAID 5) si guasta possiamo ricostruire i suoi dati tramite l'operazione di xor.

Esempio

Disco A: 1011 (Dati orignali)
Disco B: 1100 (Dati orignali)
Disco C: 0111 (Ottenuti dallo XOR, quindi dati di parità)

Se il disco B si guasta possiamo fare XOR tra i dati del disco A e disco C per ritrovare quelli del disco B.

Evoluzione da ext2 a ext3 e ext4

- **Ext (1992): Il primo file system di Linux** -> Creato specificatamente per il kernel da Remy Card, superava i limiti di MINIX, primo ad usare il VFS nel kernel Linux.
- **Transizione a Ext2 (1993)** -> Introdotto per risolvere i problemi di Ext, come immutabilità degli I-node e frammentazione. Gli I-node non potevano cambiare per tutta la durata di vita del file. Prevale su Xiafs.
- **Ext3: Introduzione del jornaling** -> Maggiore integrità dei dati.
- **Ext4 (2006-2008)**
- **Ext4 viene adottato da Google**

File system EXT4 di Linux

Introduce il journaling per la perdita di adti in caso di crash, (Journaling Block Device) per le operazioni di log.

Miglioramenti:

- Aumento della dimensione massima dei file del file system.
- Introduzione degli extend

Extend

Strutture che indicano un intervallo contiguo di blocchi su disco specificando l'indirizzo di inizio e la quantità di blocchi consecutivi.

- Compatibilità con le versioni precedenti, ma con prestazioni migliori.

Inoltre è possibili configurare il journaling solo per metadati o per tutto il disco, con un support di file fino a 16TB e file system fino a 1EB.

BTRFS

File system progettato per l'era dei moderni dispositivi di archiviazione, supporta snapshot e rollbacks, ottimale per backup e ripristini. Ci sta un miglioramento nell'integrità dei dati con **checksum** e miglioramento nella compressione dei dati e deduplicazione.

> **Caratteristiche:**
>
> 1. Supporto per file enormi (Fino a 16 exbibyte).
> 2. Archiviazione efficiente tramite riduzione dei metadati nei file.
> 3. Supporto RAID.
> 4. Facili operazioni per deframmentazione.
> 5. Allocazione dinamica degli I-node.
> 6. Snapshot per il ripristino del file system.
> 7. Supporto del checksum.
> 8. ottimizzazione per SSD.

Siccome è un file system molto nuovo non è ancora stato completamente testato.
Invece Ext3 e Ext4 sono molto stabili, ma non hanno tutte le caratteristiche moderne di BTRFS.

Struttura cartelle in Linux

Di seguito abbiamo una lista delle cartelle più importanti in Linux:

- **/bin**: contiene i binari dei principali comandi eseguibili dal sistema: cat, ls, pwd, ecc…
- **/boot**: contiene tutti i file necessari al Boot Loader per il processo di avvio del sistema. Inoltre, contiene il Kernel.
- **/etc**: contiene tutti i file di configurazione del sistema e di controllo.
- **/usr**: contiene i pacchetti del sistema e le directory e le applicazioni dell'utente.
- **/var**: contiene file temporanei, log, ecc…
- **/sbin**: ci sono i binari del sistema, quindi anche comandi come ifconfig, mkfs, fdisk , ecc…
- **/dev**: i file descrittori dei device, ovvero dei dispositivi/periferiche come gli hard disk interni. In linux anche i dispositivi vengono visti come file!
- **/home**: cartella principale dell'utente dove vengono salvati i file locali, contenente le cartelle: Documenti, Immagini, ecc…
- **/lib**: contiene le librerie essenziali, incluso il compilatore C e le relative librerie.
- **/media**: contiene cartelle che servono per gestire media rimovibili «montati» automaticamente dal sistema, come cd, floppy, ecc…

- **/mnt**: usata per eseguire il mount manuale dei filesystem temporanei dei dispositivi rimovibili come USB e cd.
- **/opt**: abbreviazione di “Optional”, contiene sia gli add-ons di alcuni software, sia programmi che non sono necessari al sistema.
- **/root**: Gome dell'utente root.

Visualizzazione delle partizioni e file system con mount e altri comandi

E' fondamentale conoscere le partizioni del disco per gestire correttamente lo spazio e l'organizzazione dei dati, inoltre è cruciale per le operazioni di backup, ripristiono e manutenzione del sistema.

Scoprire partizioni e file system disponibili:

```
lsblk //Mostra un elenco dei dispositivi di blocco, inclusi dischi e partizioni
```

```
fdisk -l //Elenca tutte le partizioni su disco, comprese quelle non montate
         //richiede privilegi di sudo
```

Visualizzazione dei file system montati:

```
mount -l
```

Mostra l'elenco dei file system attualmente montati con le opzioni di montaggio, utile per vedere dove e come sono montate le partizioni e file system.

Montaggio di una partizione/file system:

```
mount [opzioni] <dispositivo> <directory>
```

Esempio:

```
sudo mount /dev/sdal /mnt/mydisk
```

La directory di destinazione deve esistere prima del montaggio. In questo caso stiamo montando /dev/sdal su /mnt/mydisk.

Possiamo specificare anche il tipo di file system che vogliamo montare: `-f <tipo>`
Oppure aggiungere delle opzioni aggiuntive con: `-o <opzioni>` (es -o ro, sola lettura).

Smontaggio di un file sytem:

```
umount <dispositivo>
```

Usato per smontare in modo sicuro un file system o partizione.

Note finali:

- Queste operazioni richiedono privilegi di root.
- Lo smontaggio corretto è essenziale per prevenire la perdita di dati.