FILE SYSTEM

Un file-system è una struttura dati che sta su un **dispositivo a blocchi,** e quando si formatta un volume si inizializza la struttura dati di quel volume. Un volume è una **sequenza di blocchi indirizzabili.** Se voglio formattare una USB uso un'applicazione/programma per formattare in un formato. In base al formato abbiamo una struttura dati diversa.

L'interfaccia con cui il sistema operativo parla è sempre quella di **dispositivi a blocchi:** quando interagisco (leggo o scrivo) con un blocco, non posso solo leggere o scrivere un singolo byte ma **devo leggere o scrivere l'intero blocco**. Un esempio di dispositivo a blocco è USB.
Tipicamente sono divisi in 512 byte l'uno.

Ogni indirizzo di memoria **RAM** contiene dei byte. Leggere o scrivere un blocco è veloce: però se voglio cercare un blocco specifico "dall'altro lato" del disco devo aspettare che arrivi nella posizione giusta. Questo fenomeno si chiama **ritardo rotazionale**.
La differenza tra un blocco e un disco è la **persistenza:** Se stacco la corrente la RAM perde tutto.

Quando voglio scrivere un file e non lo salvo rimane nella **RAM.** Quando poi viene salvato si salva direttamente nel blocco.
Quando leggo un file viene salvata in una cache chiamata **buffer cache**, che memorizza i file letti o scritti. Se un processo deve rileggere quel file è già salvato nella **cache(nella RAM)** quindi non devo cercare ogni volta.

Se vogliamo avere due sistemi operativi sullo stesso disco lo partizioniamo ossia creiamo dei dischi logici nel disco originale. Abbiamo due standard per partizionare:

- **MBR:** con 4 partizioni primarie + partizioni estese
- **GPT:** 128 partizioni identificate da un UUID

Su Unix è tutto un file, i file speciali fanno riferimento ai dispositivi che possono essere a caratteri o a blocchi. Un file corrispondente ad un sistema a blocchi può fare riferimento all'intero disco oppure alle partizioni. Questi speciali vengono creati con uan system call chiamata mknod , chiamata da root e sono identificati da due numeri:

- Il **major number** identifica il tipo di dispositivo;
- Il **minor number** identifica uno dei vari dispositivi di quel tipo (istanze della classe).

Il file system è un albero che parte da una radice che è **/(root).**

In unix se attacco una USB mi si monta il drive quando l'attacco al pc, e quando la toglo mi si smonta. In Windows invece ogni disco ha una lettera (C:, D: ecc)

Struttura generale del file system: Quando formattiamo un volume andiamo ad inizializzare una struttura dati ovvero:

- **Super blocco:** contiene tutte le informazioni generali del file system come ad esempio:
  - dimensione di ogni blocco
  - numero totale blocchi

- **I-node:** struttura dati fondamentale che mantiene quasi tutti i metadati di un file, tranne il nome. Ogni i-node è contenuto in una tabella chiamata **tabella degli i-node**

- **Bit-mat:** è un settore di booleani, **viene usata per gli i-node**, abbiamo 1 bit per ogni i-node che mi dice se l'inode è occupato (1) o libero (0).
  Abbiamo anche la **bit-mat dei blocchi di dati** che svolge la stessa identica cosa ma facendo riferimento ai blocchi

All'interno di un file system alla Unix, possiamo avere più tipi di **file**; il più **semplice** è quello regolare. Dal punto di vista del sistema operativo, **un file regolare non è altro che una sequenza di byte**. Il **formato** del file **non** è un tipo di file, tutti i file tipo jpg, png, pdf, ecc. dal punto di vista del sistema operativo sono file regolari.
Un altro tipo di file molto importante sono le **directory.** Per l'utente una directory è una cartella che contiene file mentre per il sistema operativo **una directory è un file speciale** che contiene le associazioni fra nome e numero di i-node.

Quando **crei un file in un file system Unix-like**, il sistema deve:

1. **Trovare un i-node libero**, cioè una struttura che contiene i metadati del file (tipo, permessi, proprietario, dimensione, date, link count, e i puntatori ai blocchi di dati).
2. **Aggiornare la bitmap degli i-node** per segnare che quell'i-node è in uso.
3. **Scrivere nella directory** un'associazione: *nome del file → numero dell'i-node.*

Un file può avere più nomi (hard link), quindi più voci nella directory che puntano allo stesso i-node. Esistono diversi **tipi di file**:

- **File normali**
- **Link simbolici**: file speciali che contengono un *percorso*, non i dati.
- **FIFO (named pipe)**, **socket**, **dispositivi a carattere** (es. tastiera), **dispositivi a blocchi** (es. hard disk).

Le **directory sono file speciali** che contengono nomi di file e relativi numeri di i-node. Hanno due voci speciali:
- . che rappresenta la directory stessa.
- .. che rappresenta la directory padre.

Link:
- Il comando ln crea link:
  - **Hard link**: un altro nome per lo stesso file (stesso i-node, stesso file system).
  - **Link simbolico**: un file che contiene un percorso (può puntare a file anche inesistenti o su altri file system).

Eliminazione:
- Il comando rm elimina un nome (una voce di directory).
- Il file viene eliminato solo quando **non ha più nomi e nessun processo lo tiene aperto**.

Percorsi:
- **Assoluti**: iniziano con /, partono dalla root.
- **Relativi**: partono dalla directory corrente del processo.
- Ogni processo ha una directory corrente e una root (modificabile con chroot).

Durante la risoluzione di un percorso:

1. Si analizza ogni componente (es. qui/quo/qua).
2. Se si incontra un **link simbolico**, si cerca di risolverlo.
3. Se si va in loop (A punta a B, B punta ad A), il sistema si ferma dopo un po'

Problemi di coerenza:
Se mentre modifichi il file system **manca la corrente**, possono succedere vari errori:

- Un i-node risulta occupato ma non c'è il nome in directory.
- Un nome in directory punta a un i-node che è ancora libero.
- Due nomi diversi puntano allo stesso blocco (senza volerlo).

Non c'è un ordine di scrittura che eviti sempre questi problemi.
Soluzioni:

1. **fsck (file system check)**: programma che controlla la coerenza del file system dopo uno spegnimento anomalo.
    - Controlla superblocco, bitmap, i-node, puntatori, ecc.
    - Costa in termini di tempo, soprattutto su dischi grandi.
    - Viene eseguito se il flag "montato" non è stato correttamente rimosso prima dello spegnimento.
2. **Journaling** (soluzione moderna):
    - Prima si scrivono in un'area speciale (journal) tutte le operazioni che si vogliono fare.
    - Poi si eseguono.
    - Se il sistema crasha a metà, al riavvio legge il journal e **ripete le operazioni incomplete**.
    - Questo garantisce che le modifiche siano **atomiche**: o tutte fatte o nessuna.