Sistemi Operativi

Ionut Zbirciog

5 December 2023

Implementazione del File System

Layout del File System

Il file system è il metodo utilizzato per organizzare e memorizzare dati sui dispositivi di memoria non volatile. Fornisce un modo strutturato per gestire informazioni come file e directory su dispositivi di memoria. Un disco può essere suddiviso in più partizioni, ciascuna con un proprio file system indipendente.

MBR

Nel vecchio stile, il settore 0 del disco è chiamato MBR (Master Boot Record), essenziale per l'avvio del computer. Contiene la tabella delle partizioni (contiene gli indirizzi di inizio e fine di ciascuna partizione) e identifica la partizione attiva da cui avviare il sistema.

Quando si avvia il computer, il BIOS legge ed esegue l'MBR, che localizza la partizione attiva, ne legge il primo blocco, chiamato blocco di boot, e lo esegue. Ogni partizione inizia con un boot block, anche se non contiene un sistema operativo avviabile.

Il layout del file system cambia molto a seconda del file system, ma in generale, è strutturato nel seguente modo:

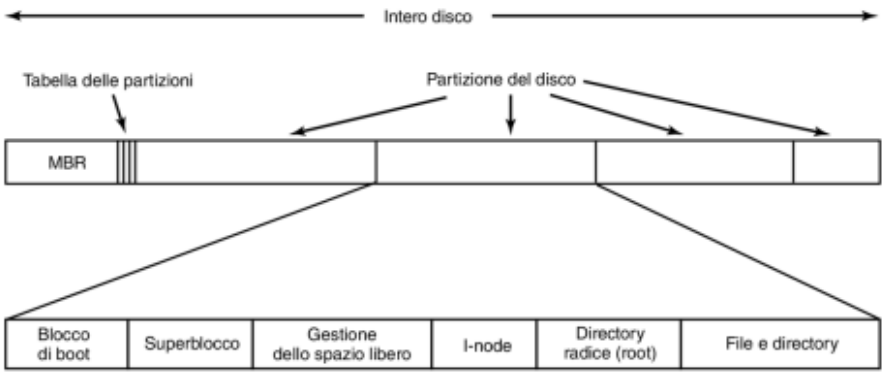

Figure 1: Layout del file system.

- **Superblocco:** Contiene tutti i parametri chiave riguardanti il file system e viene letto in memoria all'avvio del computer. Tipicamente contiene un numero magico per identificare il file system e altre informazioni chiave.
- **Bitmap o Linked List:** Usati per la gestione dello spazio libero.
- **I-node:** Un array di strutture dati, una per file, che contiene tutte le informazioni sui file.
- **Directory radice:** Contiene la cima dell'albero del file system.

UEFI

Avviare il computer con la MBR è lento e limitato a dischi di dimensione fino a 2 TB. Pertanto è stato introdotto UEFI (Unified Extensible Firmware Interface), attualmente il modo più diffuso per avviare il computer. UEFI è veloce, ammette infinite partizioni, supporta diverse architetture e dischi di dimensione fino a 8 ZiB.

UEFI non si basa sul MBR ma cerca la tabella delle partizioni nel secondo blocco, riservando il primo blocco per il software che si aspetta di trovare un MBR.

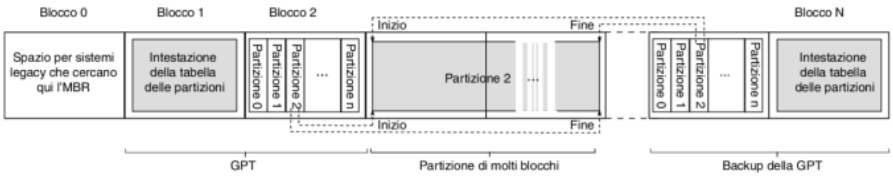

Figure 2: UEFI con GPT.

UEFI utilizza la GPT (GUID Partition Table), una struttura flessibile e dinamica che contiene informazioni sulla posizione delle partizioni sul disco. Nell'ultimo blocco conserva un backup della GPT. Una GPT contiene l'inizio e la fine di ogni partizione, e una volta trovata la GPT, il firmware ha funzionalità per leggere file system di tipi specifici.

Implementazione dei File

È importante avere una buona implementazione dei file per assicurare l'integrità, l'accesso efficiente e la gestione dello spazio sul disco.

Allocazione Continua

Lo schema di allocazione più semplice è memorizzare ciascun file come una sequenza contigua di blocchi sul disco.

Vantaggi:

- Semplice da implementare.
- Prestazioni di lettura eccellenti.

Svantaggi:

- Frammentazione del disco.
- Difficoltà nell'aggiungere nuovi file in un disco frammentato.
- Necessità di compattare il disco, operazione costosa.

Allocazione a Liste Concatenate

Il secondo metodo per memorizzare i file è configurare ciascuno come una lista concatenata di blocchi sul disco. La prima parte di ciascun blocco è usata come puntatore al successivo, il resto del blocco è per i dati.

Vantaggi:

- Utilizza in modo efficiente tutti i blocchi del disco.
- Minimizza la frammentazione esterna.

Svantaggi:

- Accesso casuale estremamente lento.
- Dimensioni dei blocchi non più una potenza di due, riducendo l'efficienza.

Allocazione a Liste Concatenate con FAT

Gli svantaggi dell'allocazione a liste concatenate possono essere eliminati spostando i puntatori in una tabella di memoria FAT (File Allocation Table). Ogni blocco del disco è rappresentato come una voce nella FAT, in memoria RAM.

Vantaggi:

- Accesso casuale semplificato, poiché i puntatori sono in RAM.

Svantaggi:

- La tabella deve rimanere sempre in memoria.
- Occupa considerevole quantità di RAM, specialmente su dischi grandi.

Figure 3: Allocazione a Liste Concatenate con FAT.

Domande

Quanti indirizzi si possono mettere su un blocco da 4 KB, se ogni numero/indirizzo è rappresentato a 32 bit (4 byte)?

$$\frac{2^{12}}{2^2} = 2^{10} = 1024 \text{ indirizzi}$$

Qual è il file più grande che si può indicizzare nei seguenti sistemi di file:

- **FAT12:** 32 MB
- **FAT16:** 2 GB
- **FAT32:** 4 GB

I-node

Gli i-node o index-node sono una struttura dati che elenca gli attributi (esclusi nome e contenuto) come permessi, proprietario, timestamp e gli indirizzi dei blocchi dei file. Ogni file e directory è rappresentato da un I-node univoco, indicizzato in una tabella di I-node.

Dato un i-node, è possibile trovare tutti i blocchi di quel file. Un grande vantaggio è che solo gli i-node dei file aperti sono mantenuti in memoria, riducendo significativamente l'utilizzo della memoria. L'array degli i-node in memoria è proporzionale al numero di file aperti, non alla dimensione del disco.

Gli i-node hanno uno spazio limitato per gli indirizzi, per file che superano il limite, uno degli indirizzi nell'i-node punta a un blocco contenente ulteriori indirizzi di blocchi dati.

Gli i-node sono un concetto fondamentale in UNIX e nei suoi file system derivati. NTFS, il file system di Windows, utilizza una struttura simile con i-node più grandi che possono contenere file di piccole dimensioni all'interno dell'i-node stesso.

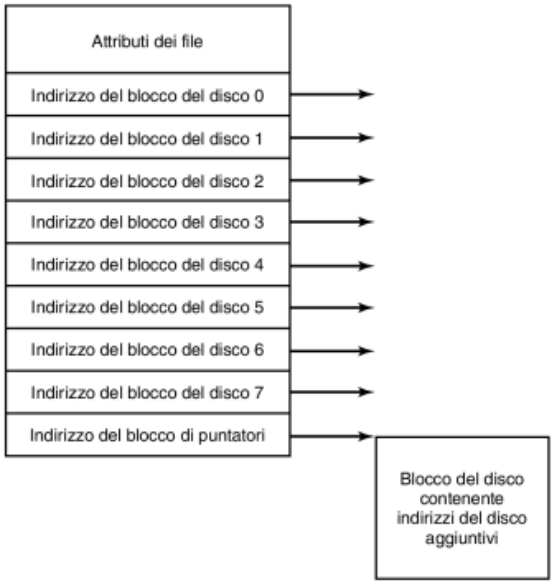

Figure 4: Struttura degli I-node.

Implementazione delle Directory

Implementazione Directory

Il ruolo principale delle directory è di mappare il nome ASCII del file sulle informazioni necessarie per localizzare i dati sul disco. A seconda del sistema, questa informazione può essere l'indirizzo del disco dell'intero file (allocazione contigua), il numero del primo blocco (linked list), o il numero dell'i-node.

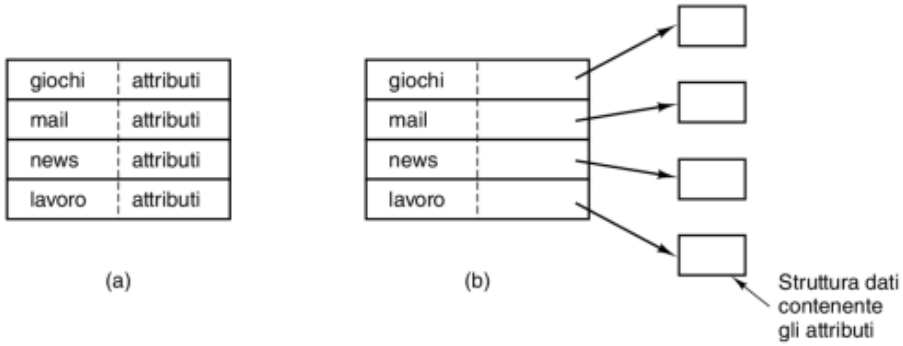

Figure 5: Struttura delle directory.

I moderni file-system supportano nomi di file variabili da 1 a 255 caratteri. Per fare ciò, le directory si possono strutturare in 2 modi:

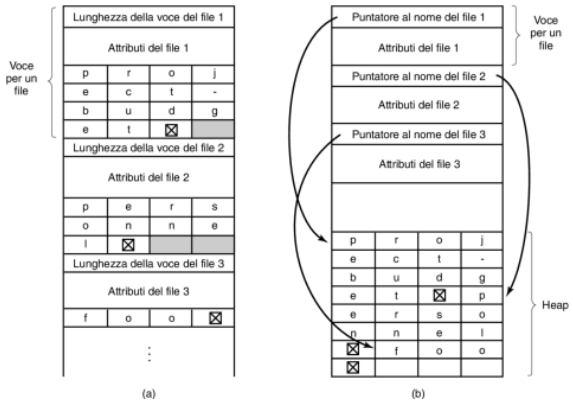

Figure 6: (a) Directory con header di lunghezza fissa. (b) Uso di heap per i nomi dei file.

(a) Ciascuna directory contiene l'header di lunghezza fissa seguito dal nome del file. Ogni nome di file termina con un carattere speciale. Per consentire a ciascuna voce di directory di iniziare alla fine della parola, ogni nome di file è riempito fino ad arrivare a un numero intero di parole. Lo svantaggio è che quando il file viene cancellato, nella directory è introdotto un vuoto di ampiezza variabile, che potrebbe non bastare a contenere il file da inserire successivamente.

(b) Uso di heap per tenere i nomi dei file. Questo metodo ha il vantaggio che quando viene rimossa una voce, il successivo file da inserire ci starà sempre. Una piccolissima conquista in questo caso è che non è più necessario che i nomi dei file inizino ai confini delle parole, quindi non sono necessari caratteri di riempimento.

Inizialmente, i file in una directory venivano cercati linearmente dall'inizio alla fine. Questo metodo può diventare lento in directory con un gran numero di file. Si è passato dunque all'utilizzo delle Hash Table, accelerando il processo di ricerca. Il nome di un file è sottoposto a hashing per generare un indice nell'intervallo da 0 a $n-1$. La voce corrispondente nella tabella di hash indica il punto di partenza per la ricerca del file. Per i file che condividono lo stesso hash, viene creata una lista concatenata. Un modo diverso per velocizzare la ricerca in grandi directory è salvare nella cache il risultato delle ricerche. Prima di avviare la ricerca, si verifica se quel nome di file si trova nella cache. Se si trova, può essere immediatamente individuato; altrimenti, si segue la ricerca all'interno della directory.

File Condivisi e Link nel File System

I file condivisi sono essenziali in ambienti collaborativi per permettere a più utenti di lavorare sugli stessi file.

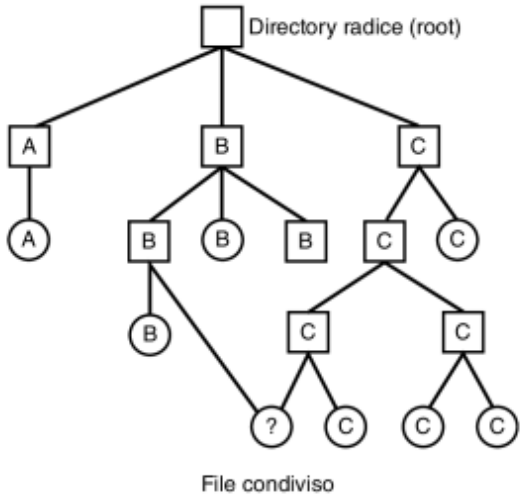

Figure 7: (a) Hard Link. (b) Soft Link.

- **Hard Link**: Puntano direttamente all'i-node di un file condiviso. Un file con hard link viene rimosso solo quando non ci sono più riferimenti ad esso. Sono efficienti in termini di spazio, usano un solo i-node indipendentemente dal numero di link. Il file permane fino all'eliminazione dell'ultimo link, potenzialmente causando confusione sulla proprietà del file.
- **Soft Link**: Puntano al nome di un file piuttosto che all'i-node. Sono più flessibili, possono riferirsi a nomi di file oltre i confini del file system e su macchine remote, ma sono meno

efficienti in termini di spazio in quanto richiedono un i-node per ogni link. I soft link diventano invalidi alla rimozione del file originale. Il sistema operativo impiega più tempo nella risoluzione del percorso rispetto agli hard link.

Entrambe le modalità, però, hanno un problema comune, ovvero che i file con più percorsi possono essere processati più volte da programmi di backup o di ricerca. Per esempio, c'è il rischio di duplicazione dei file su un'unità di backup.

Gestione Dello Spazio sul Disco

Dimensione dei Blocchi

Generalmente i file sono memorizzati su disco, tramite l'allocazione contigua, portando a maggiori spostamenti sul disco se le loro dimensioni aumentano, oppure suddividendo il file in blocchi più piccoli consentendo maggiore flessibilità e un migliore utilizzo dello spazio su disco. La dimensione comune di 4KB per blocco è un compromesso tra lo spazio su disco e le prestazioni di trasferimento dei dati.

Perché 4KB?

- **Prestazioni di Trasferimento Dati**: I dischi magnetici con blocchi più grandi consentono il trasferimento di più dati per operazione di lettura/scrittura. Ma blocchi grandi portano a spreco di memoria.
- **Efficienza dello Spazio**: Blocchi piccoli minimizzano lo spreco di spazio con file piccoli. Ma significa distribuire la maggior parte dei file su più blocchi e incorrere in più ricerche e ritardi per leggerli. L'efficienza dello spazio diminuisce con l'aumento della dimensione dei blocchi.

Gestione dei Blocchi Liberi

1. Linked List

   Si usa una lista concatenata, in cui si vanno a mettere solo i blocchi liberi. Per ospitare la lista si usano gli stessi blocchi del disco. Ogni blocco contiene numeri di blocchi del disco liberi. Richiede meno spazio solo se il disco è quasi pieno. Per ottimizzare la lista, si possono tracciare i blocchi liberi consecutivi. A ciascun blocco può essere associato un conteggio a 8, 16, 32 bit che rappresenta il numero di blocchi liberi consecutivi. Nell'ipotesi migliore, un disco fondamentalmente vuoto è rappresentato da due numeri, l'indirizzo del primo blocco libero, seguito dal conteggio dei blocchi liberi. Questo metodo è migliore per dischi quasi vuoti, ma meno efficiente per dischi molto frammentati.

   La gestione dei blocchi liberi può utilizzare una lista concatenata di puntatori, nota come "free list". Solo un blocco di puntatori è mantenuto in memoria contemporaneamente, ottimizzando così l'utilizzo della memoria. Quando si crea un file, i blocchi necessari vengono allocati dai puntatori disponibili nel blocco in memoria. Questo metodo evita I/O su disco inutili mantenendo una lista di blocchi liberi direttamente accessibili in memoria. Al riempimento del blocco di puntatori in memoria, un nuovo blocco viene letto da disco per proseguire con le operazioni. La presenza di file temporanei può portare a frequenti operazioni di I/O su disco se il blocco di puntatori in memoria è quasi pieno. Una strategia alternativa prevede di dividere il blocco pieno di puntatori per gestire meglio i blocchi liberi senza I/O su disco.

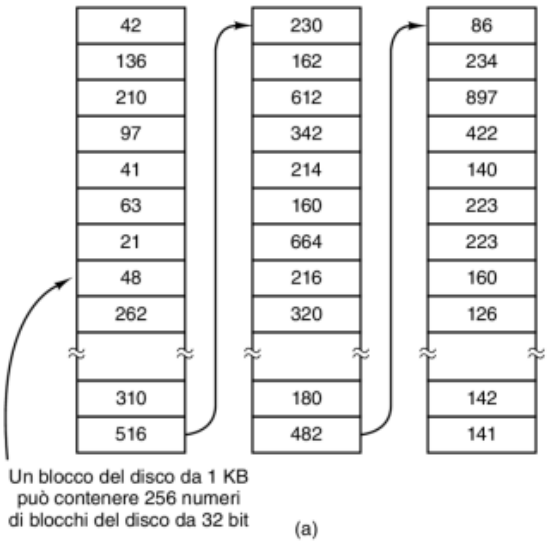

Figure 8: Linked List.

2. Bitmap

   Un disco con n blocchi richiede una bitmap con n bit. I blocchi liberi sono indicati dal valore 1 nella mappa, quelli allocati dallo 0. Per esempio, per un disco da 1TB serve una mappa da 1 miliardo di bit, il che richiede 130,000 blocchi da 1 KB per eseguire la memorizzazione. È più efficiente rispetto alla linked list, tranne in dischi quasi pieni.

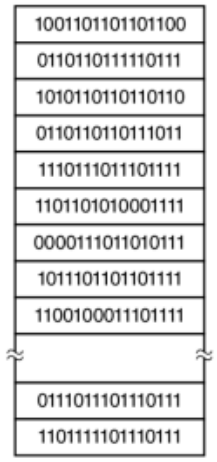

Figure 9: Struttura di una bitmap.

Quote del Disco

Per impedire che gli utenti occupino troppo spazio su disco, i sistemi operativi multiutente forniscono spesso un meccanismo per imporre le quote del disco. L'idea è che l'amministratore di sistema assegni a ciascun utente un numero massimo di file e blocchi, e che il sistema operativo si accerti che gli utenti non superino la loro quota. Ogni apertura di file coinvolge il controllo degli attributi e degli indirizzi sul disco. Gli attributi includono l'identificatore del proprietario del file. In una tabella dei file aperti, viene contabilizzata la quota di ciascun utente. Ci sono 2 limiti, soft e hard. Il limite soft può essere temporaneamente superato, mentre il limite hard, no.