

IL FILE SYSTEM: GESTIONE DIRECTORY, SPAZIO E PERFORMANCE

Danilo Croce

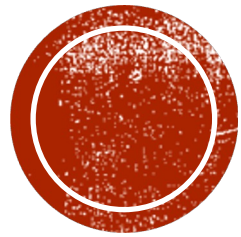
Dicembre 2024



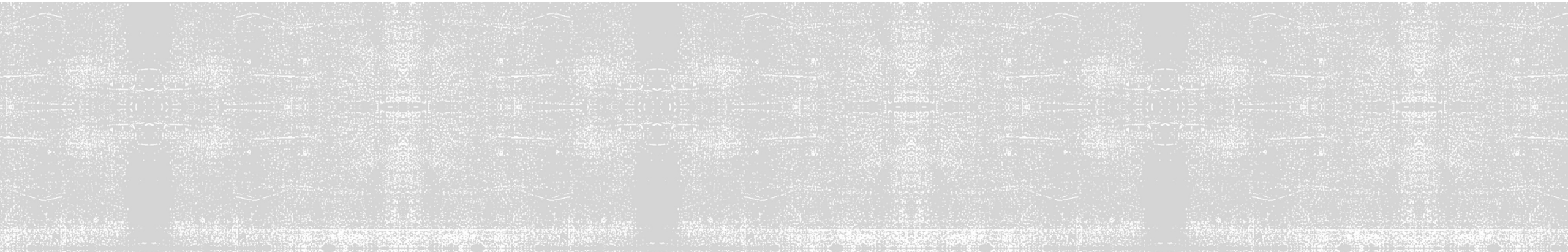
IMPLEMENTAZIONE DEL FILE SYSTEM

- In questa lezione cercheremo di rispondere alle seguenti domande relative i File System:
 - Come memorizzare i file?
 - Come implementare le directory?
 - Come gestire lo spazio su disco?
 - Come garantire le prestazioni del file system?
 - Come garantire l'affidabilità del file system?





IMPLEMENTAZIONE DEI FILE



IMPLEMENTAZIONE DEL FILE SYSTEM

- **Come memorizzare i file?**
- Come implementare le directory?
- Come gestire lo spazio su disco?
- Come garantire le prestazioni del file system?
- Come garantire l'affidabilità del file system?



INTRODUZIONE AL LAYOUT DEL FILE SYSTEM

- **Definizione:** Il file system è il **metodo utilizzato per organizzare e memorizzare dati** sui dispositivi di memoria non volatile (dischi/SSD).
- **Importanza:** Fornisce un **modo strutturato per gestire informazioni** come file e directory su dispositivi di memoria.
- **Partizioni del Disco:** Un disco può essere **suddiviso in più partizioni**, ciascuna con un proprio file system indipendente.
- **Evoluzione:** I metodi di strutturazione del file system **variano a seconda dell'epoca del computer**, influenzando come i dati vengono gestiti e acceduti.



VECCHIO STILE - BIOS CON MBR (MASTER BOOT RECORD)

- **MBR nel BIOS:**

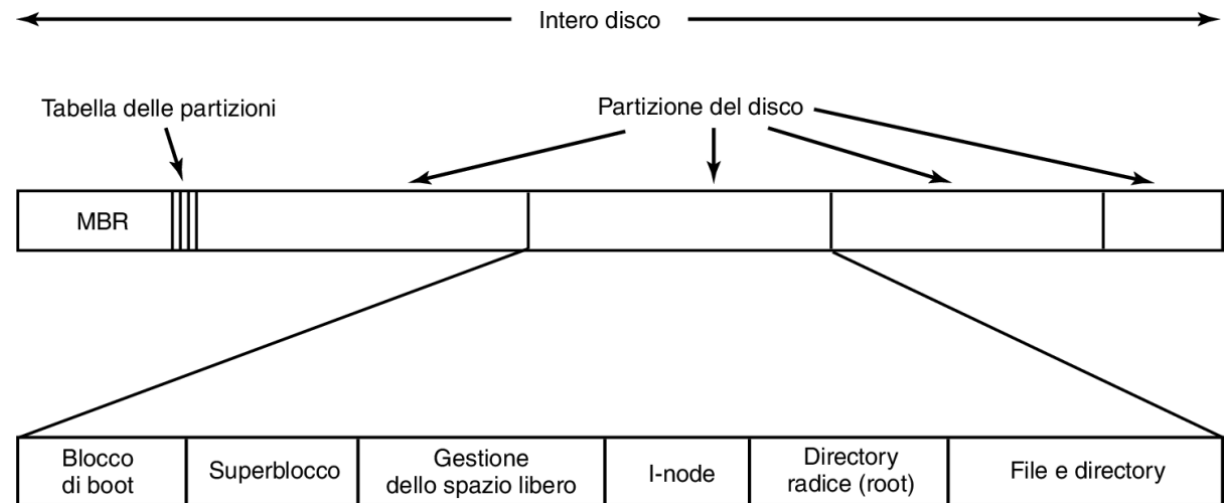
- Situato nel settore 0 del disco, l'MBR è essenziale per l'avvio del computer.
- Contiene la tabella delle partizioni con dettagli su inizio e fine di ciascuna partizione.
- Identifica la partizione attiva da cui avviare il sistema.

- **Processo di Avvio:**

- Il BIOS legge l'MBR per trovare la partizione attiva.
- Carica il **boot block** della partizione attiva per avviare il sistema operativo.

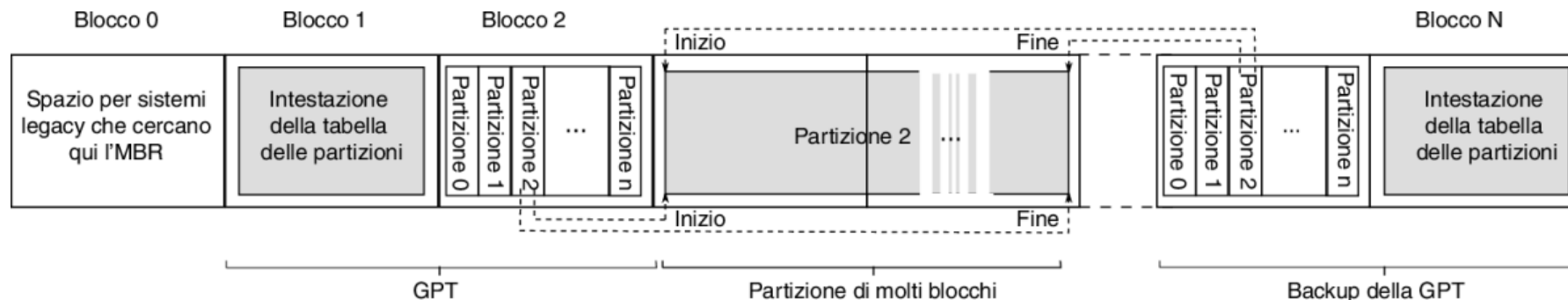
- **Layout del File System:**

- Ogni **partizione inizia con un boot block**, seguito da vari elementi di sistema.
- A destra un esempio di layout, includendo **superblocco**, **bitmap**, **I-node** e **directory radice** (vedi slide successive)



NUOVA SCUOLA - UEFI (UNIFIED EXTENSIBLE FIRMWARE INTERFACE)

- **UEFI (Unified Extensible Firmware Interface):**
 - **Cos'è:** Un sistema moderno che sostituisce il vecchio BIOS tradizionale.
- **Vantaggi:**
 - **Avvio più veloce:** Ottimizza il processo di inizializzazione dell'hardware.
 - **Compatibilità migliorata:** Supporta architetture hardware a 32 e 64 bit.
 - **Interfaccia utente avanzata:** Può includere grafica e supporto per mouse.
 - **Sicurezza:** Aggiunge funzionalità come il Secure Boot.
- **Supporto per dischi moderni:** UEFI funziona con **GPT**, consentendo di superare i limiti di 2,2 TB dei dischi gestiti con il vecchio schema MBR.



GPT E EFI SYSTEM PARTITION (ESP)

GUID Partition Table (GPT):

- **Cos'è:** Un avanzato sistema di gestione delle partizioni.
- **Caratteristiche:**
 - Supporta dischi fino a **8 ZiB**.
 - Consente un numero illimitato di partizioni (limite imposto solo dal sistema operativo).
 - Include **backup della tabella delle partizioni** per maggiore sicurezza.
 - Utilizza un controllo di integrità (**CRC**) per prevenire corruzione dei dati.

EFI System Partition (ESP):

- **Cos'è:** Una partizione speciale sui dischi GPT.
- **Funzione:**
 - Archivia i file di avvio come bootloader, driver e utility di diagnostica.
 - È essenziale per avviare il sistema operativo.
 - Usa il **file system FAT32**, garantendo compatibilità con il firmware UEFI.



SECURE BOOT - SICUREZZA DURANTE L'AVVIO

- **Cos'è:** Una funzionalità UEFI progettata per impedire l'avvio di software non autorizzato.
- **Funzionamento:**
 - Controlla le **firme digitali** di bootloader, driver e sistema operativo.
 - Avvia solo software autorizzato e firmato.
 - Blocca malware e rootkit durante l'avvio.
- **Vantaggi:**
 - Protegge contro attacchi all'avvio (es. rootkit, malware).
 - Mantiene l'integrità del sistema operativo.
 - Aumenta la sicurezza per utenti domestici e aziende.
- **Considerazioni:**
 - Alcuni sistemi operativi (es. alcune distribuzioni Linux) potrebbero richiedere la disattivazione del Secure Boot per funzionare.
 - Disattivabile nelle impostazioni UEFI, ma ciò può esporre il sistema a rischi.



IMPLEMENTAZIONE DEI FILE NEI FILE SYSTEM

- **Obiettivo Principale:** Gestire l'associazione tra i file e i blocchi del disco su cui sono memorizzati.
- **Importanza:** Fondamentale per assicurare l'integrità, l'accesso efficiente e la gestione dello spazio su disco.
- **Varietà di Metodi:** Diversi sistemi operativi adottano approcci differenti per questa associazione.
 - **Basato su**
 - Metodi basati su indici
 - Liste concatenate
 - Bitmap
 - Strutture ad albero
- **Focus:** Analisi dei vari metodi e delle loro caratteristiche specifiche nel contesto dei diversi file system.



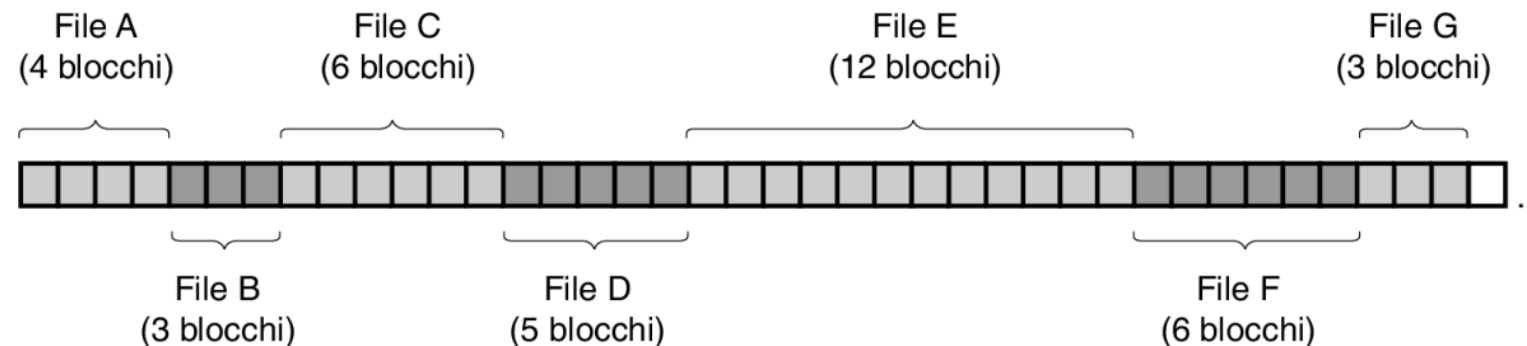
ALLOCAZIONE CONTIGUA NEL FILE SYSTEM

- **Concetto di Allocazione Contigua:**

- I file sono memorizzati come **sequenze contigue di blocchi sul disco**.
- **Esempio:** un file di 50 KB su un disco con blocchi da 1 KB occupa 50 blocchi consecutivi.
- In basso: l'allocazione contigua di sette file su disco.

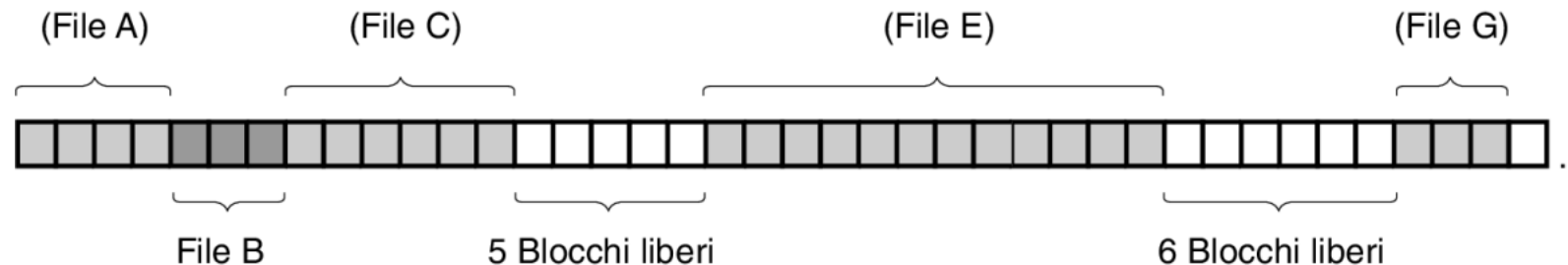
- **Implementazione e Vantaggi:**

- **Semplice da implementare:** richiede solo l'indirizzo del primo blocco e il numero totale di blocchi.
- **Alta efficienza di lettura:** l'intero file può essere letto in una sola operazione, senza ritardi.



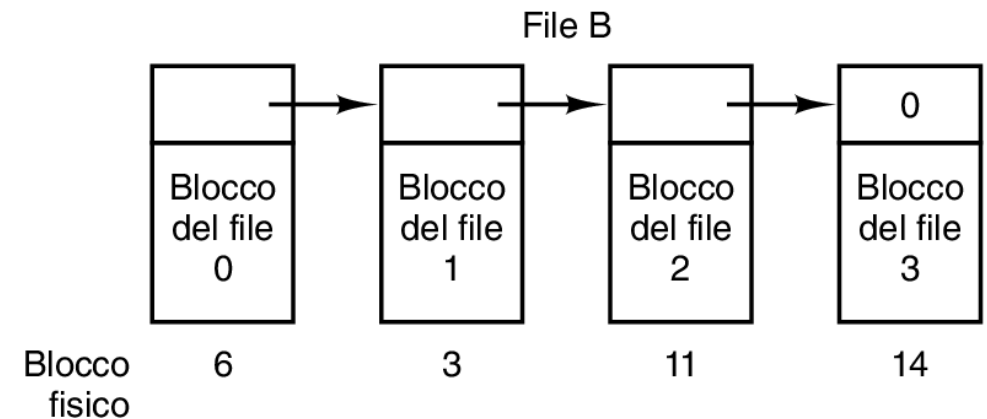
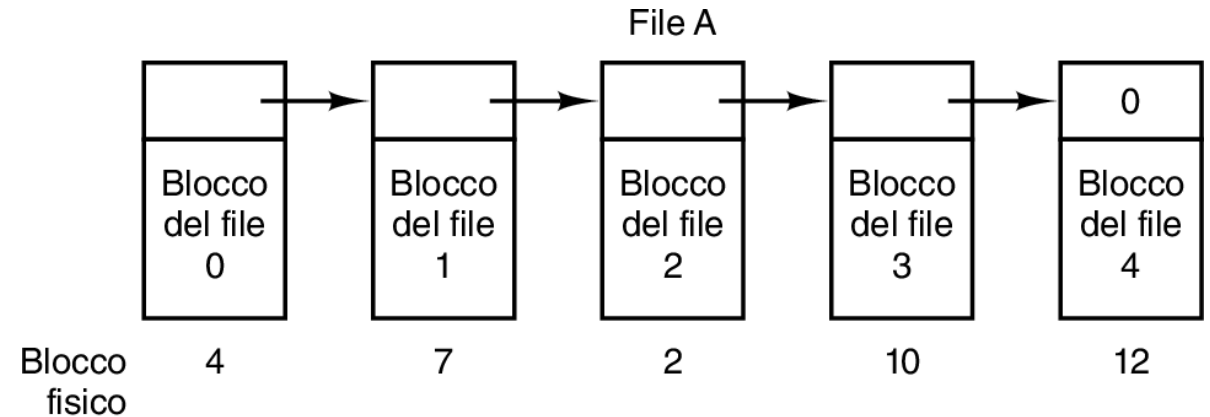
PROBLEMI E LIMITAZIONI DELL'ALLOCAZIONE CONTIGUA

- **Frammentazione del Disco:**
 - Col passare del tempo, **i dischi si frammentano** a causa della rimozione di file.
 - **In basso:** La frammentazione lascia intervalli di blocchi liberi
 - **Problema di allocazione di nuovi file in spazi liberi** frammentati.
- **Gestione dello Spazio Libero:**
 - Richiede una lista di spazi liberi e la conoscenza della dimensione finale dei nuovi file.
 - Problemi nella previsione della dimensione del file e nella ricerca di spazi adeguati.
- **Implicazioni Pratiche:**
 - Difficoltà nell'aggiungere nuovi file in un disco frammentato.
 - Necessità di compattazione del disco o di gestione intelligente dello spazio libero.



ALLOCAZIONE A LISTE CONCATENATE - CONCETTI BASE

- **Principio di Allocazione:**
 - I file sono organizzati come liste concatenate di blocchi su disco.
 - Ogni blocco contiene una parte di dati e un puntatore al blocco successivo.
- **Gestione dello Spazio:**
 - Efficiente utilizzo di tutti i blocchi disponibili sul disco.
 - Minima frammentazione esterna: riduce lo spreco di spazio non utilizzato.
- **Struttura delle Voci di Directory:**
 - Ogni voce di directory traccia solo l'indirizzo del primo blocco di un file.
 - Il percorso completo di un file è costruito seguendo i puntatori da un blocco all'altro.



ALLOCAZIONE A LISTE CONCATENATE - PRESTAZIONI E LIMITAZIONI

- **Accesso ai Dati:**
 - **Accesso Sequenziale:** Leggere un file è efficiente, procedendo blocco per blocco.
 - **Accesso Casuale (seek):** Estremamente lento, richiede la lettura sequenziale di ogni blocco precedente.
- **Dimensione dei Blocchi e Efficienza:**
 - Ogni blocco ha una dimensione effettiva ridotta a causa dello spazio occupato dai puntatori.
 - Letture e scritture di dimensioni standard (potenze di due) possono essere meno efficienti.
- **Implicazioni Pratiche:**
 - Il metodo offre un'elevata **efficienza** nello **sfruttamento dello spazio su disco** ...
 - **MA** introduce **complessità** e rallentamenti nelle operazioni di **accesso casuale**.
 - **Adatto per file** a cui si **accede** principalmente **in modo sequenziale**.



ALLOCAZIONE A LISTE CONCATENATE CON FAT

- **Ottimizzazione dell'Allocazione a Liste Concatenate:**
 - Eliminazione degli svantaggi dell'allocazione a liste concatenate 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**
- **Struttura della FAT:**
 - Contiene la sequenza dei blocchi di ciascun file.
 - Esempio:
 - File A utilizza i blocchi 4 => 7 => 2 => 10 => 12 (dove termina);
 - File B i blocchi 6 => 3 => 11 => 14 (dove termina).
 - Sequenze terminate da un indicatore speciale (es. -1) per marcare la fine.
 - In memoria principale
- **Vantaggi della FAT:**
 - L'intero blocco è disponibile per i dati, ottimizzando lo spazio.
 - Accesso casuale semplificato: la sequenza dei blocchi è interamente in memoria.

Blocco fisico		
0		
1		
2	10	
3	11	
4	7	← Il file A inizia qui
5		
6	3	← Il file B inizia qui
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Blocco non utilizzato



LIMITAZIONI E APPLICAZIONI DELLA FAT

- **Gestione in Memoria:**

- La FAT deve essere **mantenuta interamente in memoria principale**.
- Richiede una quantità significativa di memoria: per un disco da 1 TB con blocchi da 1 KB, la FAT richiederebbe fino a 3 GB di RAM.

- **Implicazioni di Efficienza:**

- Lo **spazio e la velocità influenzano la dimensione** della voce della FAT (da 3 a 4 byte per voce).
- L'approccio non è ottimale per dischi di grandi dimensioni a causa dell'elevato consumo di memoria.

- **Utilizzo Pratico:**

- Originariamente implementato in MS-DOS, ancora supportato da Windows e UEFI.
- **Comunemente usato** in dispositivi portatili come **schede SD** in fotocamere, lettori musicali e altri dispositivi elettronici.



I-NODE

- **I-node (Index Node) nei File System UNIX-like**
 - **Definizione:** Struttura dati fondamentale nei file system come ext2/ext3/ext4 in Linux.
 - **Contenuto:** Contiene tutte le informazioni su un file, esclusi il nome e il contenuto. Include metadati come permessi, proprietario, timestamp e indirizzi dei blocchi di dati.
 - **Funzione:** Ogni file e directory è rappresentato da un I-node univoco, indicizzato in una tabella di I-node.
- **Confronto con FAT (File Allocation Table)**
 - **Gestione dei File:**
 - FAT si basa su una tabella di allocazione per tracciare i file, mentre i sistemi basati su I-node utilizzano una tabella di I-node.
 - I sistemi I-node separano le informazioni sul file dalla sua posizione fisica sul disco.
 - **Informazioni sui File:**
 - FAT fornisce meno dettagli sui file, concentrandosi principalmente sull'allocazione dello spazio.
 - I sistemi I-node offrono una gestione più dettagliata dei metadati, inclusi permessi e proprietà.
 - **Efficienza e Performance:**
 - I file system basati su I-node tendono a essere più efficienti e performanti, specialmente su dischi di grandi dimensioni, grazie alla loro struttura avanzata.



FUNZIONAMENTO E VANTAGGI DEGLI I-NODE

- **I-node (Index-Node) nei File System**

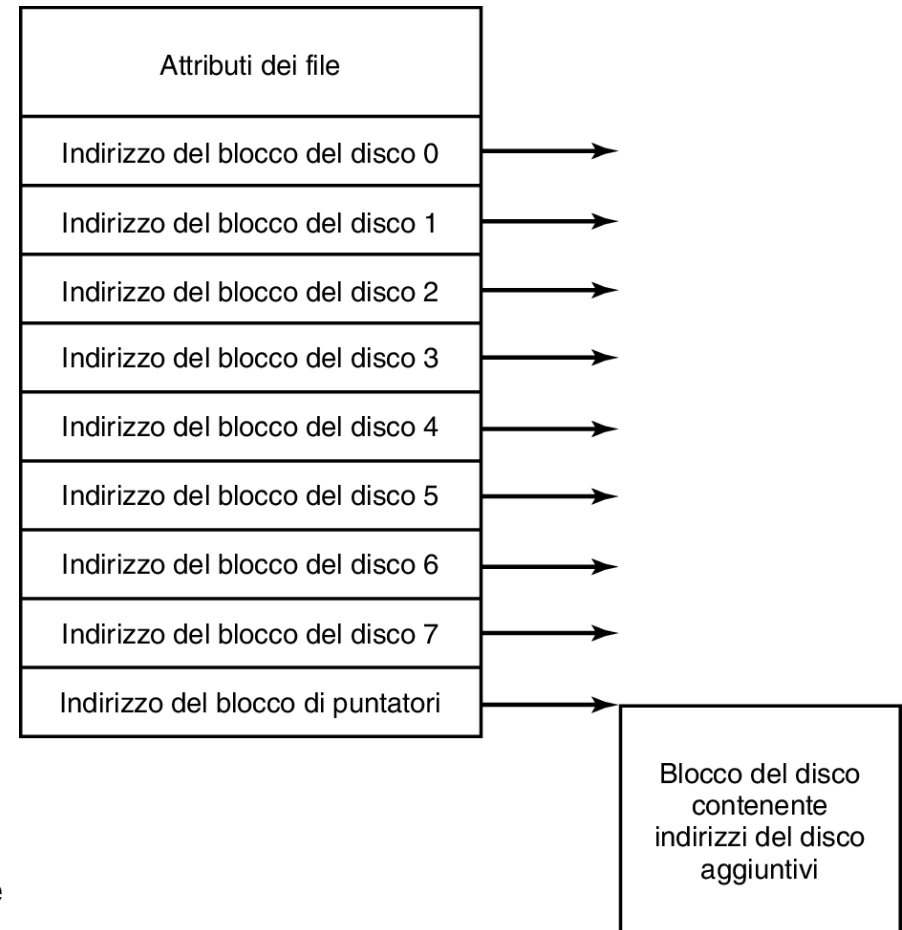
- Gli I-node sono strutture dati che elencano gli attributi e gli indirizzi dei blocchi dei file.
- Ogni I-node rappresenta un file, fornendo un metodo efficiente per trovare tutti i suoi blocchi di dati.

- **Efficienza della Memoria con I-node**

- **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.

- **Esempio e Struttura**

- Ogni I-node ha una dimensione fissa e contiene informazioni quali dimensione del file, permessi, proprietario, e indirizzi dei blocchi di dati.



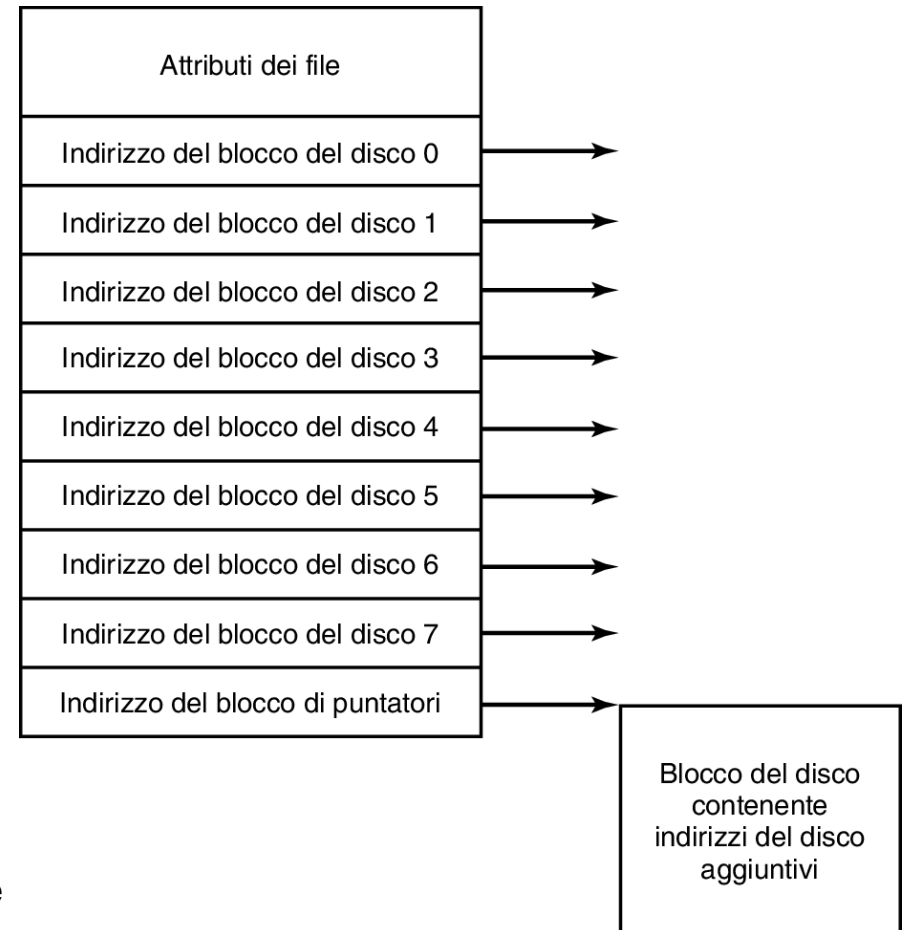
GESTIONE DEI FILE DI GRANDI DIMENSIONI E CONFRONTO CON NTFS

- **Gestione File di Grandi Dimensioni**

- Gli I-node hanno uno spazio limitato per gli indirizzi del disco.
- Per file che superano il limite, uno degli indirizzi nell'I-node punta a un blocco contenente ulteriori indirizzi di blocchi di dati.
- Questo sistema permette di gestire file di dimensioni molto grandi con efficacia.

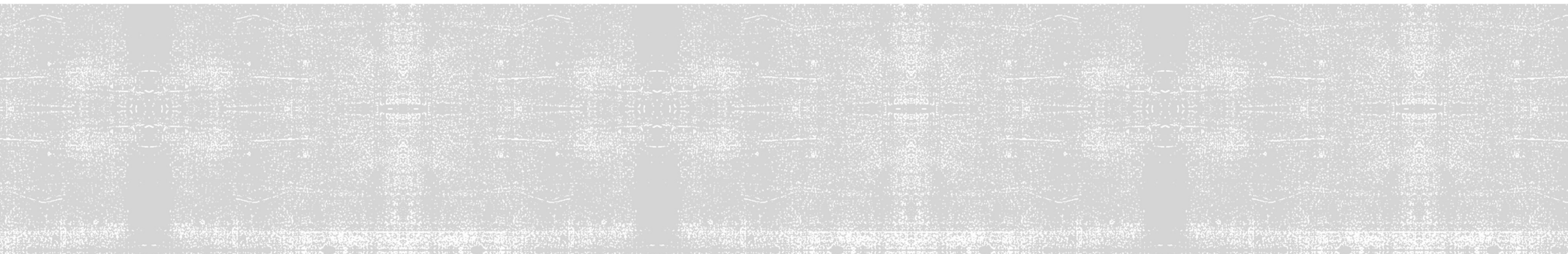
- **I-node nei File System UNIX e Windows NTFS**

- 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.





IMPLEMENTAZIONE DELLE DIRECTORY



IMPLEMENTAZIONE DEL FILE SYSTEM

- Come memorizzare i file?
- **Come implementare le directory?**
- Come gestire lo spazio su disco?
- Come garantire le prestazioni del file system?
- Come garantire l'affidabilità del file system?



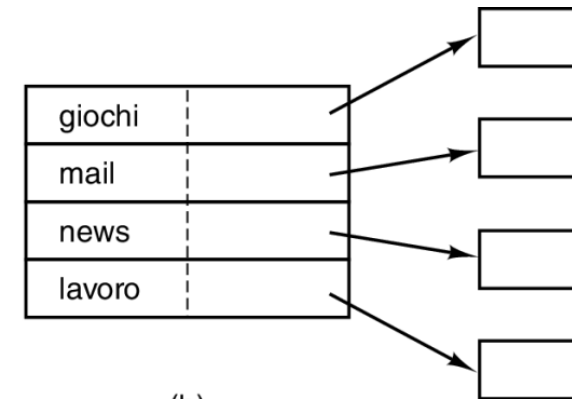
IMPLEMENTAZIONE DELLE DIRECTORY - CONCETTI DI BASE

- **Funzione Principale:** Le directory mappano i nomi ASCII dei file sulle informazioni necessarie per localizzare i dati su disco.
- **Metodi di Allocazione:** Variano a seconda del sistema operativo, includendo indirizzi di blocchi contigui, il primo blocco nelle liste concatenate, o i numeri degli I-node.

- a) Una semplice directory contenente voci a dimensione fissa con gli indirizzi del disco e gli attributi nella voce della directory.
- b) Una directory in cui ogni voce fa soltanto riferimento a un I-node.

giochi	attributi
mail	attributi
news	attributi
lavoro	attributi

(a)



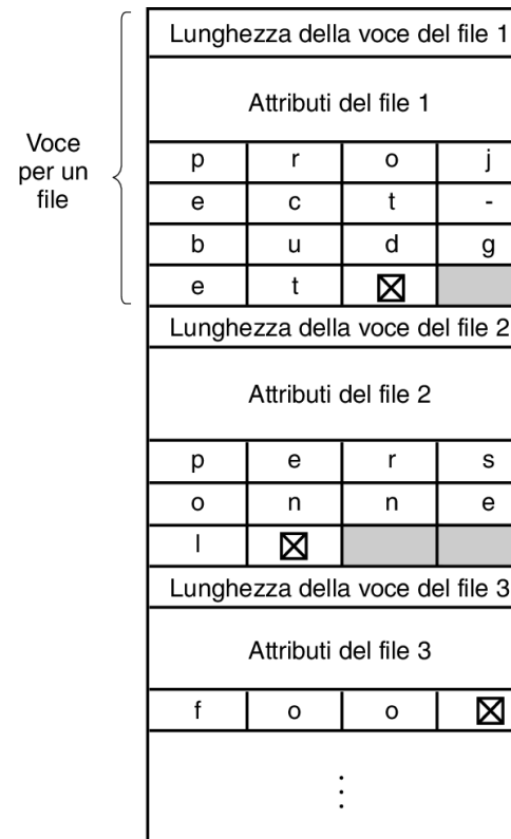
(b)

Struttura dati
contenente
gli attributi

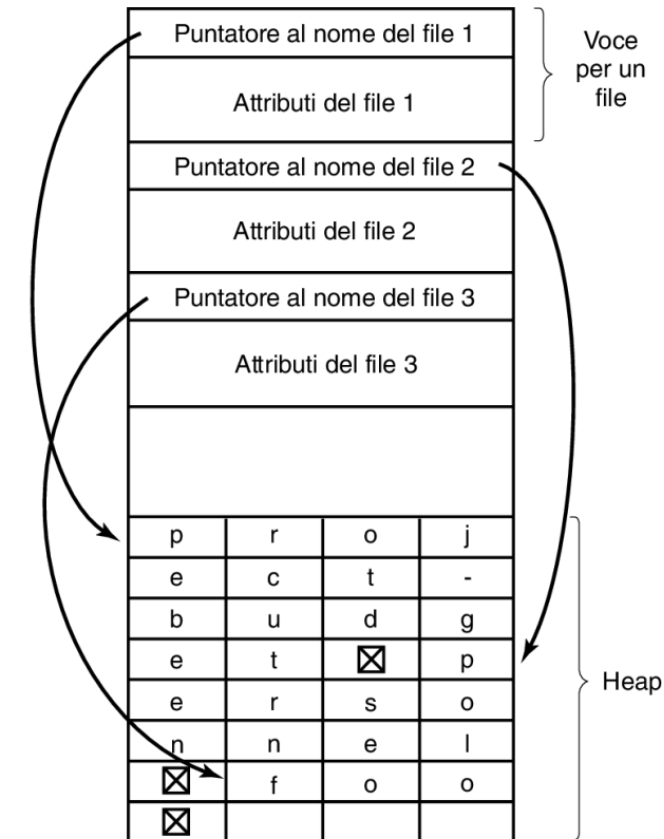


IMPLEMENTAZIONE DELLE DIRECTORY - GESTIONE DEI NOMI DEI FILE

- **Nomi di File Variabili:** Supporto per nomi di file di lunghezza variabile, con un limite tipico di 255 caratteri.
- **Strutture di Directory:**
 - Voci di Directory di Lunghezza Variabile:** l'header di lunghezza fissa seguito dal nome del file.
 - Gestione degli Heap:** le voci di directory di lunghezza fissa con nomi dei file gestiti in uno heap separato.
- **Efficienza e Limitazioni:** gestiscono i nomi di lunghezza variabile ma presentano sfide
 - nella gestione degli spazi vuoti (un file viene cancellato)



(a)



(b)



OTTIMIZZAZIONE DELLA RICERCA NELLE DIRECTORY CON TABELLE DI HASH

- **Ricerca Lineare Tradizionale:**

- 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.

- **Uso delle Tabelle di Hash:**

- Introduzione di tabelle di hash in ogni directory per accelerare 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.

- **Gestione delle Collisioni:**

- Creazione di liste concatenate per gestire più file che condividono lo stesso valore hash.
- La ricerca verifica tutte le voci nella catena per trovare il file desiderato.



GESTIONE DELLA CACHE PER RICERCHE EFFICIENTI

- **Caching delle Ricerche:**

- Salvataggio dei risultati di ricerche comuni nella cache per accesso rapido.
- Prima di avviare una ricerca, si verifica se il file si trova nella cache.

- **Vantaggi e Limitazioni:**

- La cache aumenta l'efficienza delle ricerche, specialmente per file frequentemente richiesti.
- Efficace quando la maggior parte delle ricerche riguarda un numero limitato di file.

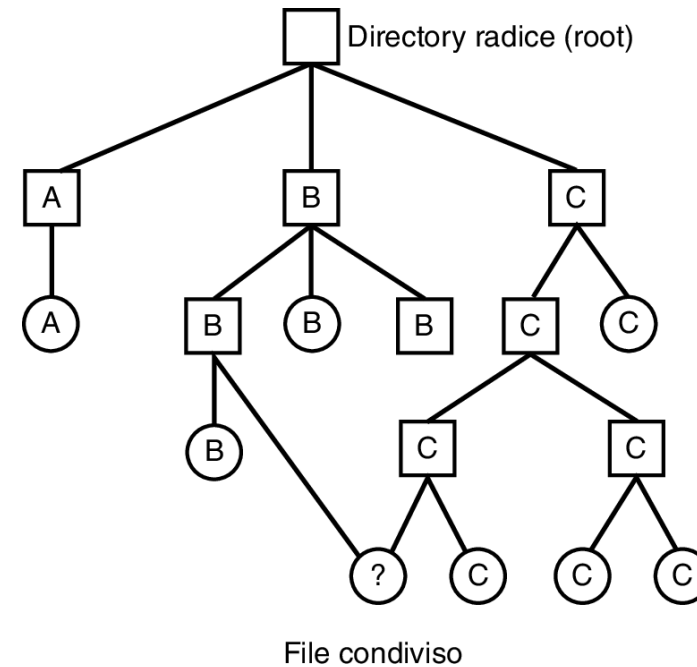
- **Complessità Amministrativa:**

- L'uso di tabelle di hash e cache introduce una maggiore complessità nella gestione delle directory.
- Più adatto a sistemi con directory molto estese, dove si prevede un elevato numero di file.



FILE CONDIVISI E LINK NEI FILE SYSTEM

- **File Condivisi:** Essenziali in ambienti collaborativi per permettere a più utenti di lavorare sugli stessi file.
- **Tipi di Link:**
 - **Hard Link:** Puntano direttamente all'I-node di un file condiviso.
 - **Link Simbolico (Soft Link):** Puntano al nome di un file piuttosto che all'I-node.
- **Gestione degli Hard Link:**
 - Un file con hard link **viene rimosso solo quando non ci sono più riferimenti ad esso.**
 - **Efficienza di spazio:** una sola voce di directory per ciascun hard link.
 - **Ideali per la gestione di file condivisi** tra più proprietari.



- Un file system con un file condiviso tra due utenti.
- Esempio: Un file di un utente può essere presente anche nella directory di un altro.



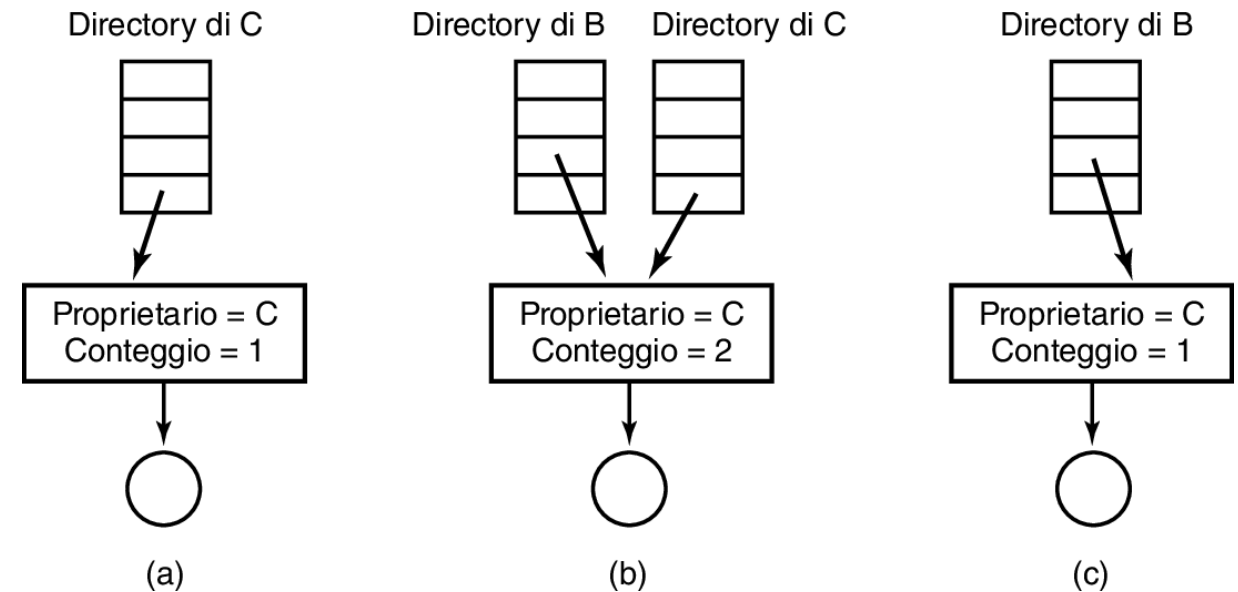
GESTIONE E PROBLEMI DEI FILE CONDIVISI: HARD LINKS

- **Vantaggi degli Hard Link:**

- **Spazio-efficienti:** usano un solo I-node indipendentemente dal numero di link.
- Gestione trasversale degli utenti: il file rimane accessibile finché almeno un hard link è presente.

- **Problemi e Limitazioni:**

- Il file permane fino all'eliminazione di tutti gli hard link, **potenzialmente causando confusione sulla proprietà del file**.
- **A destra:** Illustrano come i file condivisi sono gestiti nel file system e le implicazioni dell'eliminazione di hard link.



- a) Situazione prima del link.
- b) Dopo la creazione del link.
- c) Dopo che il proprietario originale elimina il file.



GESTIONE E PROBLEMI DEI FILE CONDIVISI: LINK SIMBOLICI

- **Vantaggi dei Link Simbolici:**

- **Maggiore flessibilità:** possono riferirsi a nomi di file oltre i confini del file system e su macchine remote.
- **Meno efficienti in termini di spazio:** richiedono un I-node per ogni link simbolico.

- **Problemi e Limitazioni:**

- Link simbolici **diventano invalidi** alla rimozione del file originale.
- **Overhead maggiore** nella risoluzione del percorso rispetto agli hard link.
- **Gestione più complessa**, ma con benefici in termini di flessibilità e organizzazione.

- **Problemi Comuni:**

- **I file** con più percorsi possono essere **processati più volte da programmi di backup** o di ricerca.
 - Rischio di **duplicazione dei file** su unità di backup.
- Necessità di software avanzato per gestire correttamente i file condivisi e i loro link.

