Sistemi Operativi

C.d.L. in Informatica (laurea triennale)
Anno Accademico 2011-2012

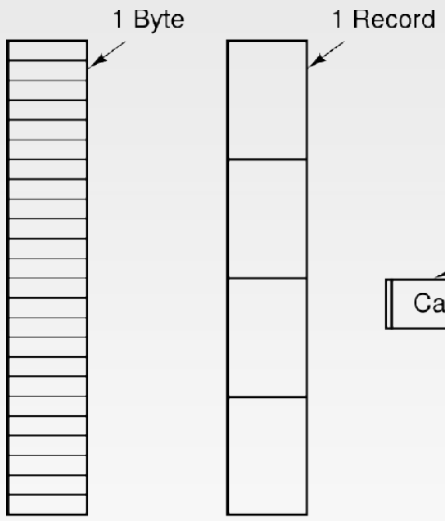
Dipartimento di Matematica e Informatica – Catania

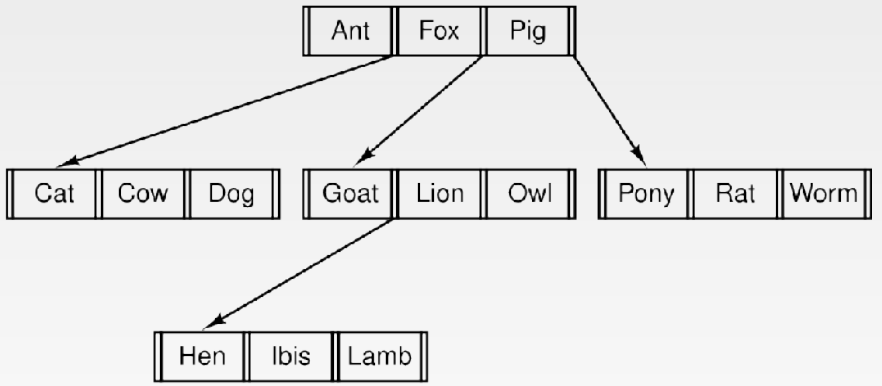
File System e Dischi

Prof. Mario Di Raimondo

I file system

- Problema di base: gestire **grandi quantità** di informazioni, in modo **persistente** e condiviso **tra più processi**;
- astrazione: **file**;
- i dettagli di gestione ed implementazione di questa astrazione costituiscono il **file system**;
- esempi di **dettagli**:
  - nomenclatura;
  - struttura interna;
  - tipi di file;
  - tipi di accesso;
  - metadati (o attributi);
  - operazioni supportate sui file;

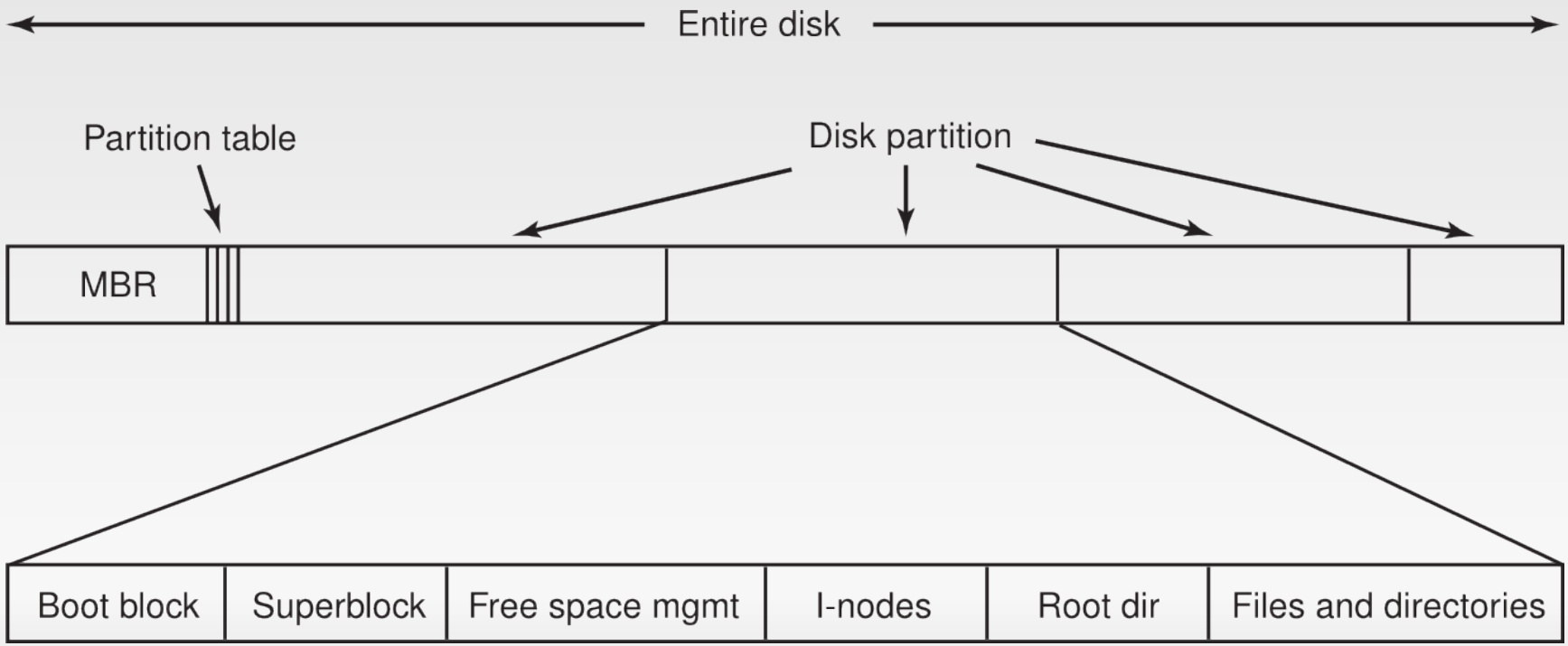




- altra astrazione per migliorare l'organizzazione: **directory**.

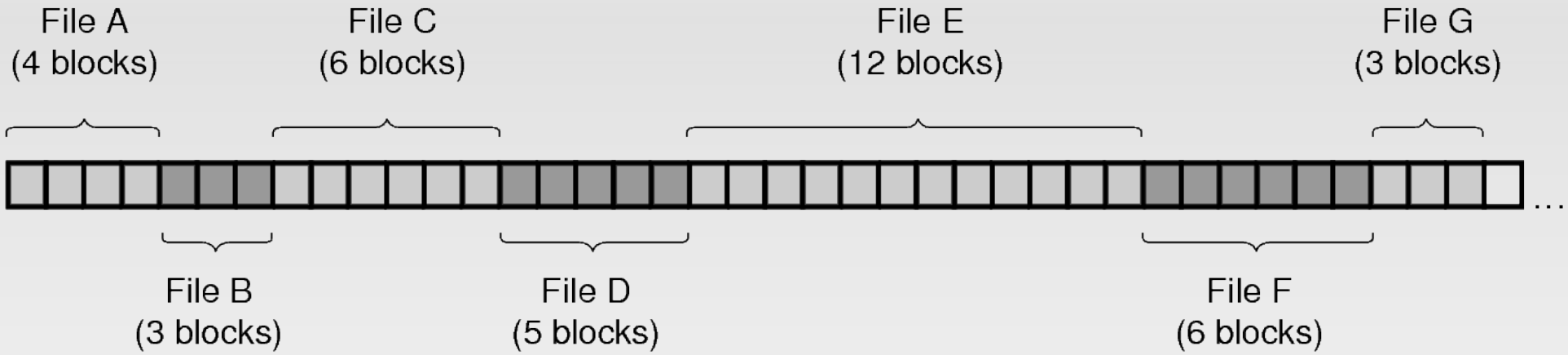
Struttura di un file system

- **Master Boot Record** (MBR);
- partizioni e **boot record** (o boot block);
- **superblocco**;

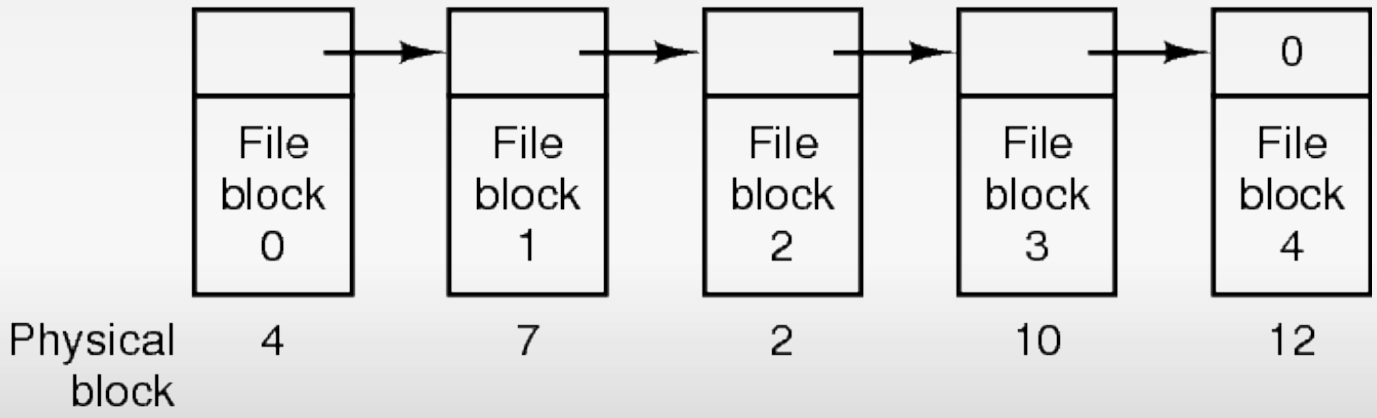


Implementazione dei file

- **Allocazione contigua**.

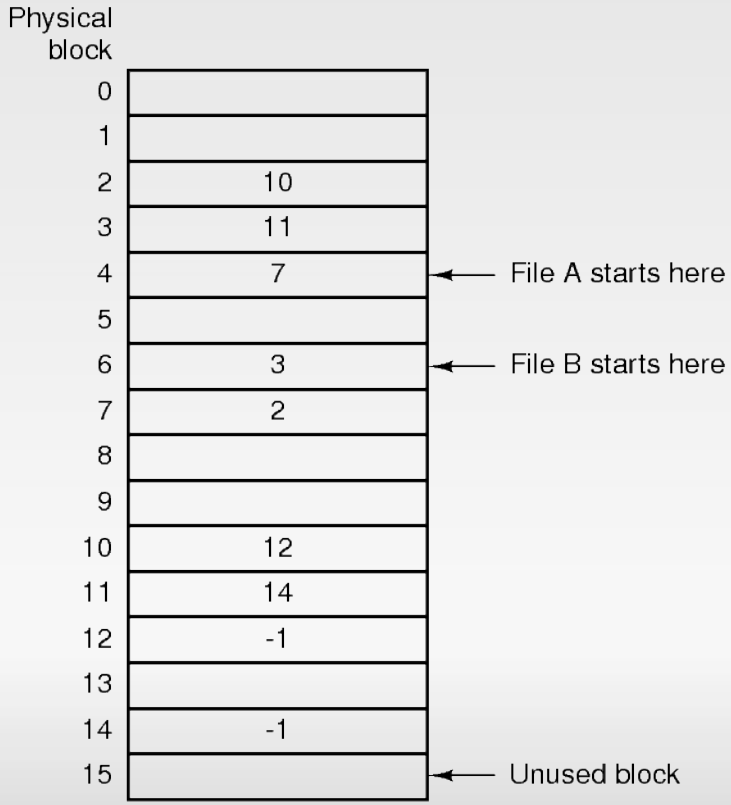


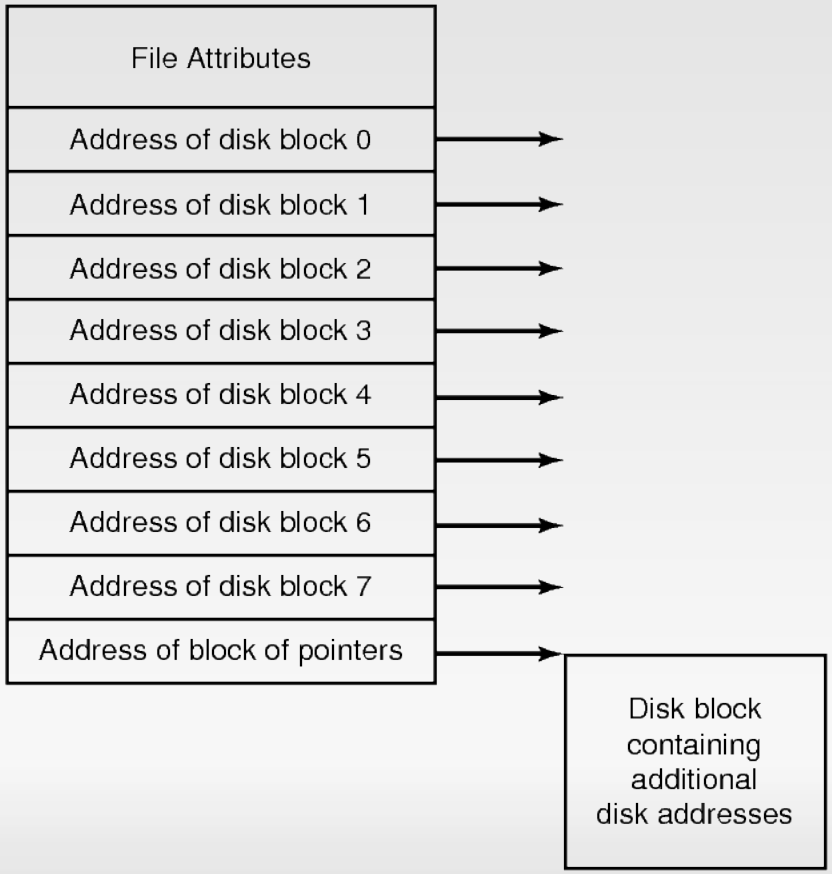
- **Allocazione con liste collegate** (o allocazione concatenata).



Implementazione dei file

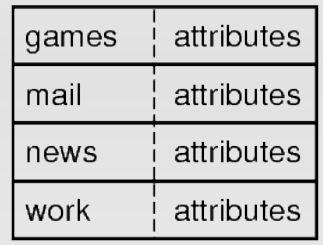
- **Allocazione con liste collegate su una tabella di allocazione dei file** (file allocation table – **FAT**) (o allocazione tabellare).
- **Allocazione con nodi indice** (index node – **i-node**) (o allocazione indicizzata):
  - varianti: **concatenata**, **multilivello**.

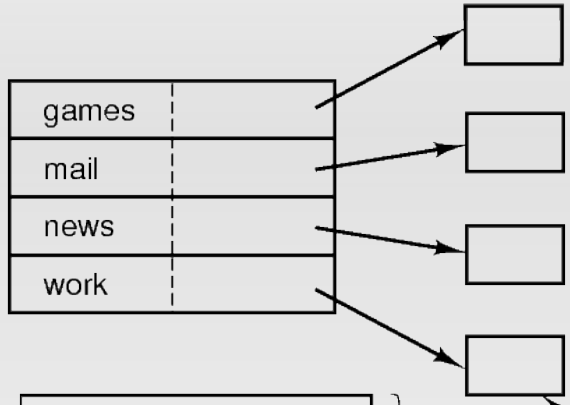




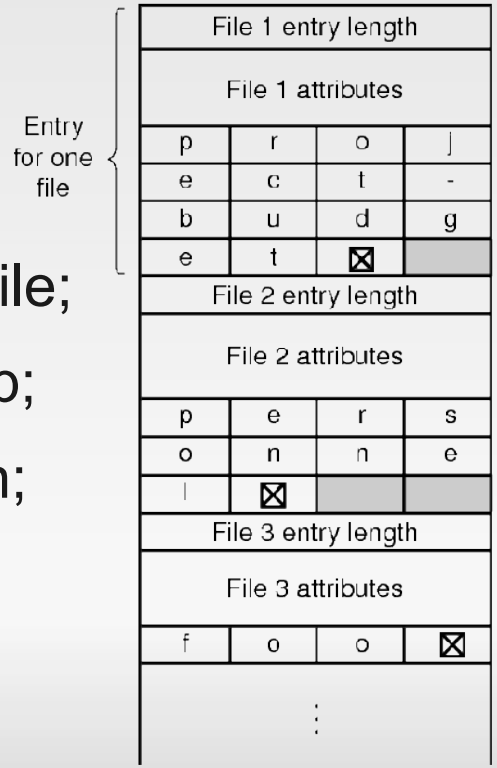
Implementazione delle directory

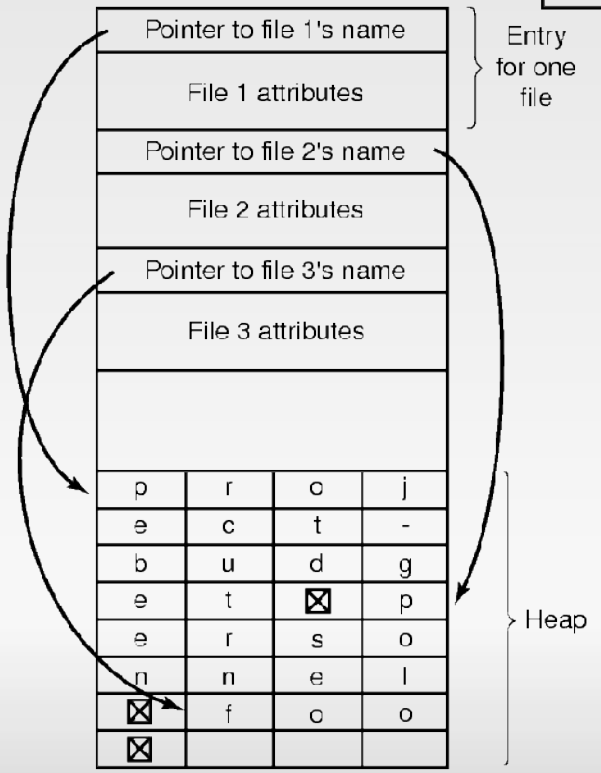
- Dove memorizzare i **metadati/attributi**?





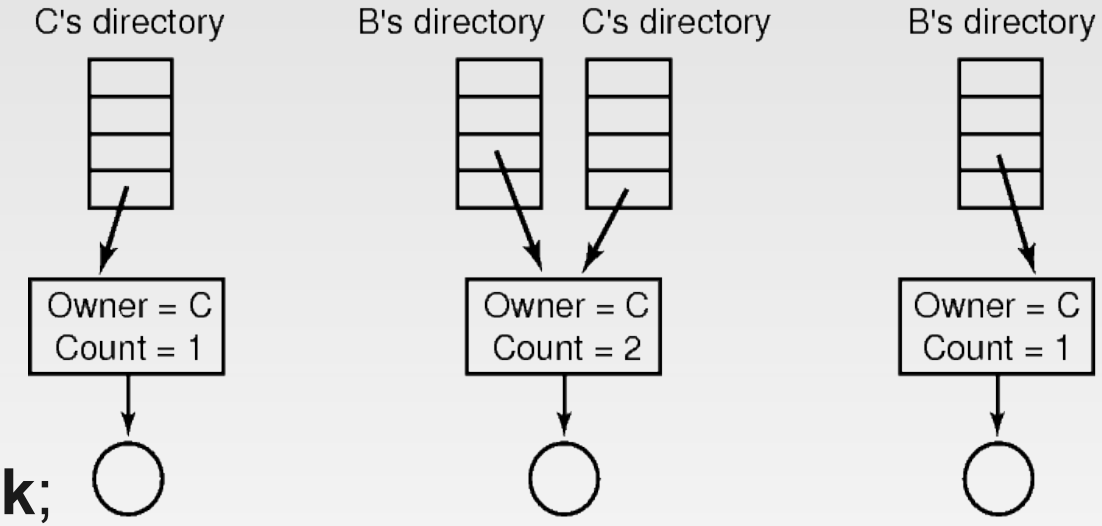
- **Nomi lunghi:**
  - lung. variabile;
  - tramite heap;
  - tabella hash;
  - cache.





Condivisione di file su un file system

- **Scenario**: due o più utenti vogliono condividere un file;
- usando una FAT: duplicare la lista con i riferimenti ai blocchi;
  - problemi in caso di append;
- usando i-node: **hard-link**;
  - contatore dei link;
  - anomalia con accounting;



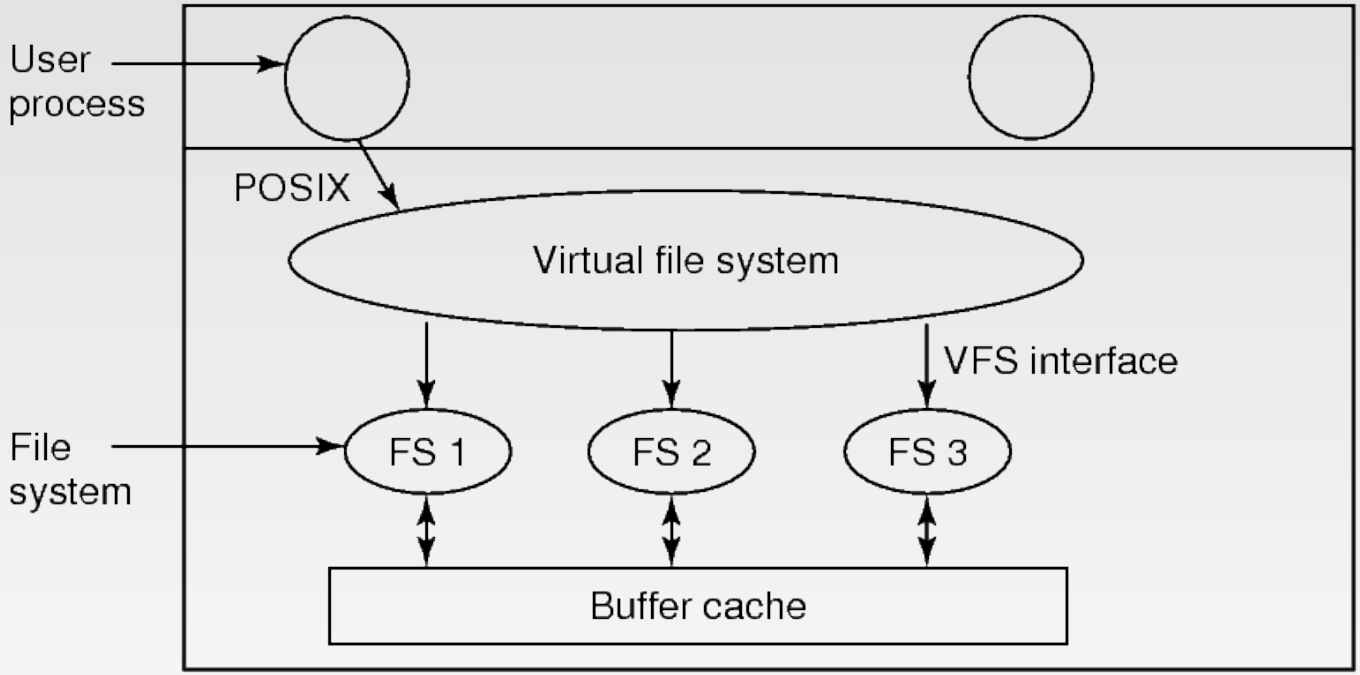
- ulteriore approccio: **soft-link**;
  - universale e permettere di fare riferimenti al di fuori del file system;
  - appesantimento nella gestione;
- eventuali **problemi** in fase di attraversamenti e backup.

File system con journaling

- **Problemi di integrità** dei file system in seguito a **crash** del sistema;
  - esempio: **cancellazione di un file**:
    (1) rimozione voce relativa al file dalla directory;
    (2) reinserimento dell'i-node nella lista degli i-node liberi;
    (3) reinserimento dei blocchi nella lista dei blocchi liberi;
- **strategia**:
  - le operazioni da fare sono preliminarmente appuntate in un **log** che viene poi ripulito a posteriori (subito dopo);
  - in caso di ripristino da crash: ripetere le operazioni in log;
- affinché tutto funzioni bisogna operare con **operazioni idempotenti**;
- possibilità di definire **transazioni atomiche**.

File system virtuale

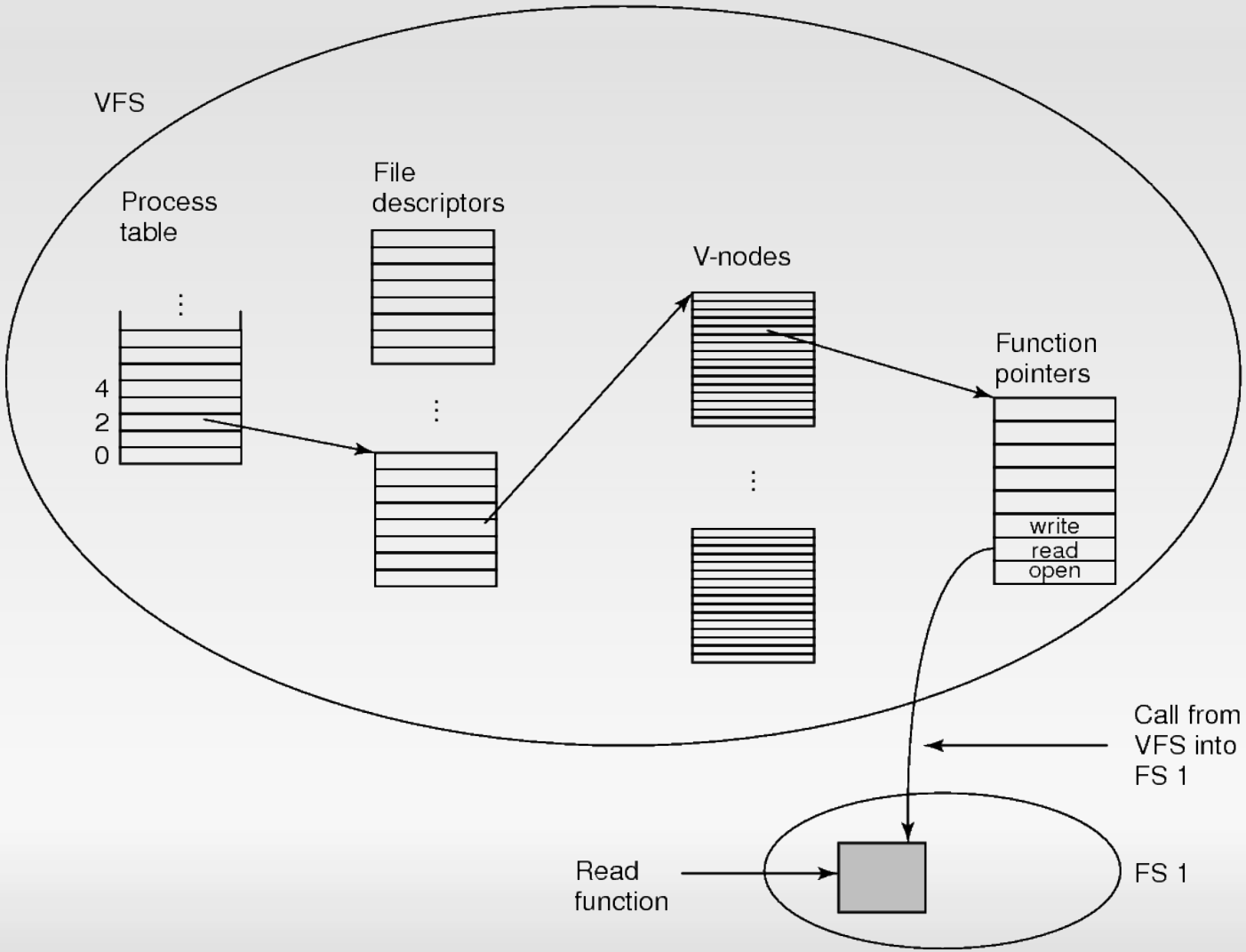
- **Virtual file system** (**VFS**): strategia dei sistemi UNIX per fornire un ulteriore livello di **astrazione** nella gestione dei file system;



- due interfacce: **interfaccia POSIX** e **interfaccia VFS**;
- permette l'utilizzo trasparente di vari FS anche di rete (NFS).

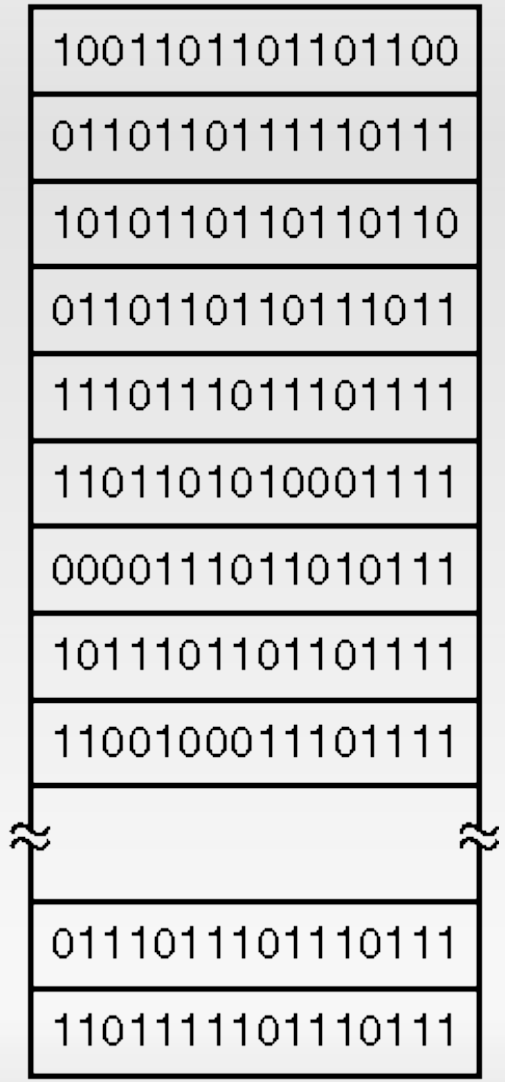
File system virtuale

- il file system specifico deve **implementare** le voci dell'interfaccia VFS;



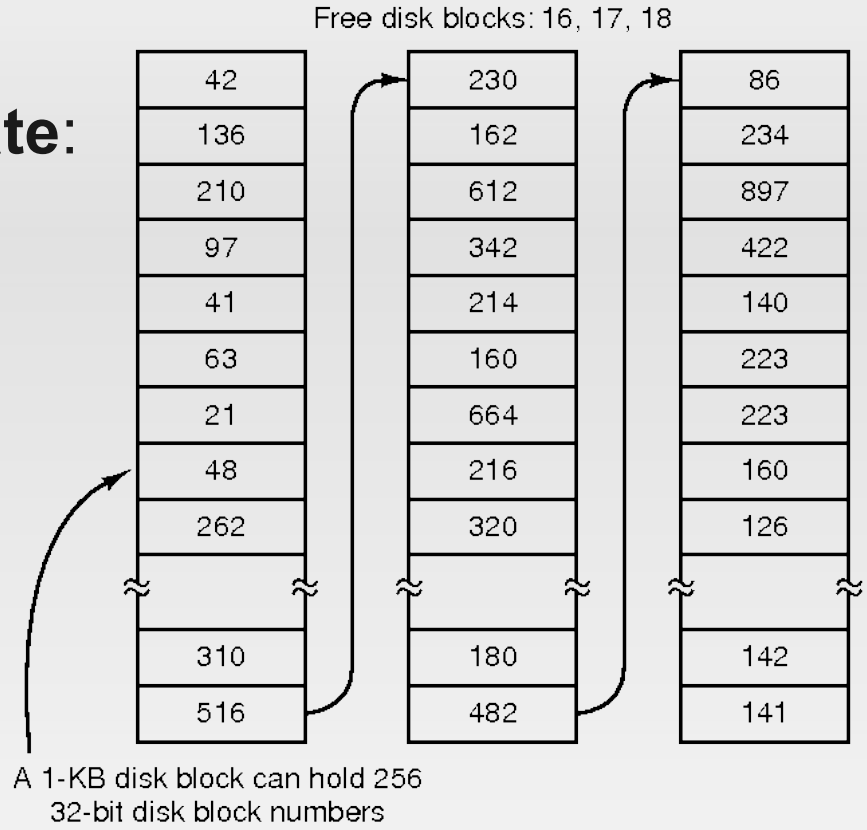
Gestione blocchi liberi

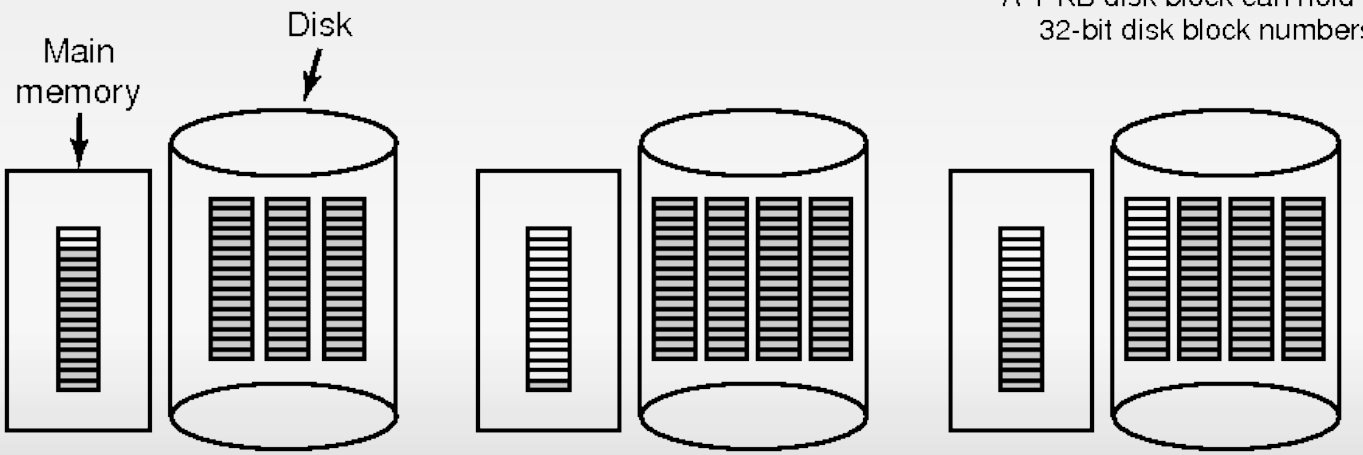
- Attraverso l'uso di una **bitmap**:
  - relativamente **piccola**;
  - strategie di **allocazione in memoria**:
    - tutta in memoria, o;
    - un blocco alla volta;
      - paginata con VM;



Gestione blocchi liberi

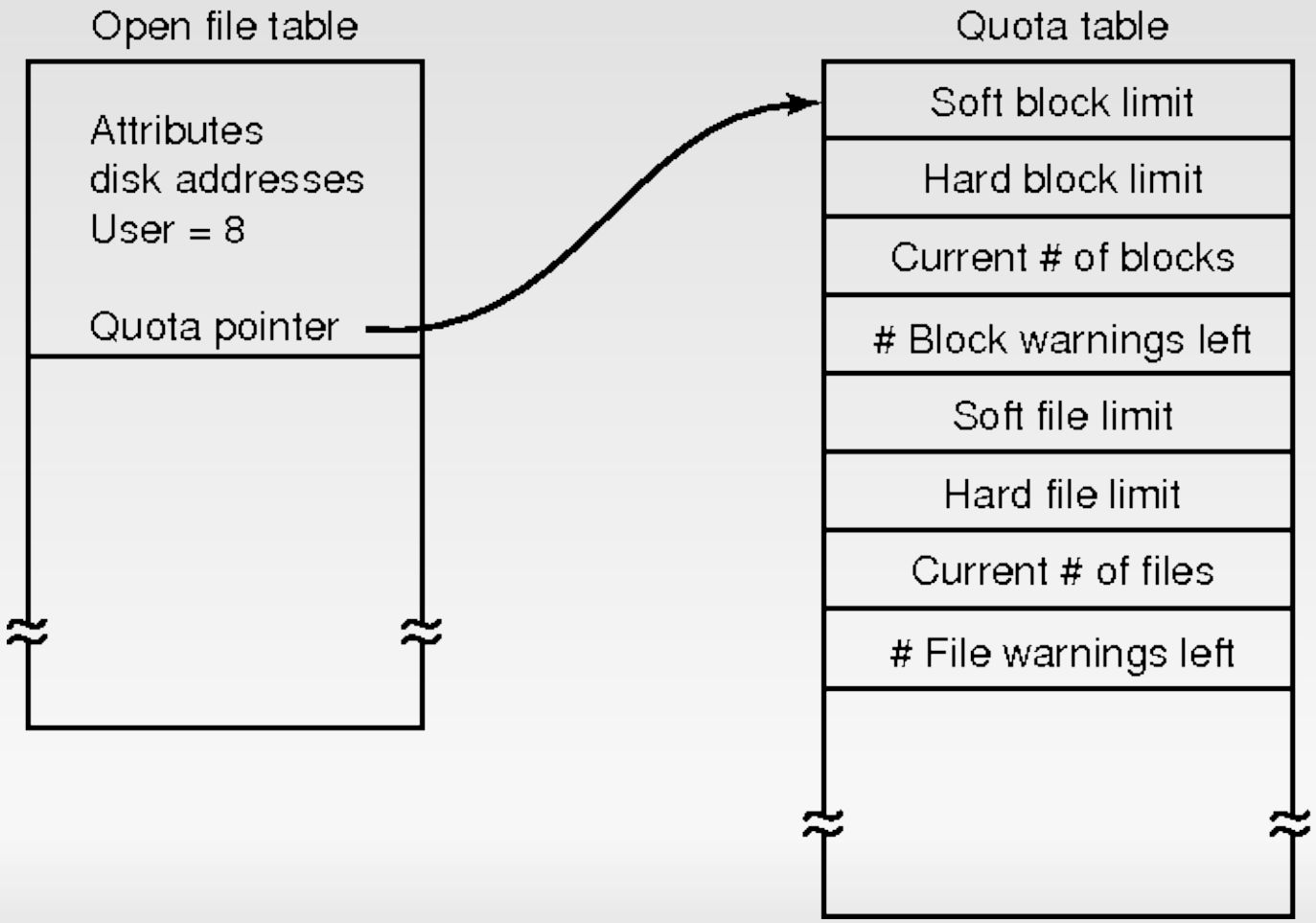
- attraverso l'uso di **liste concatenate**:
  - richiede più spazio;
    - ma si sfruttano i blocchi stessi liberi;
  - possibilità di inserire **contatori** per blocchi contigui;
- **anomalie** con spreco di I/O;
  - soluzione.





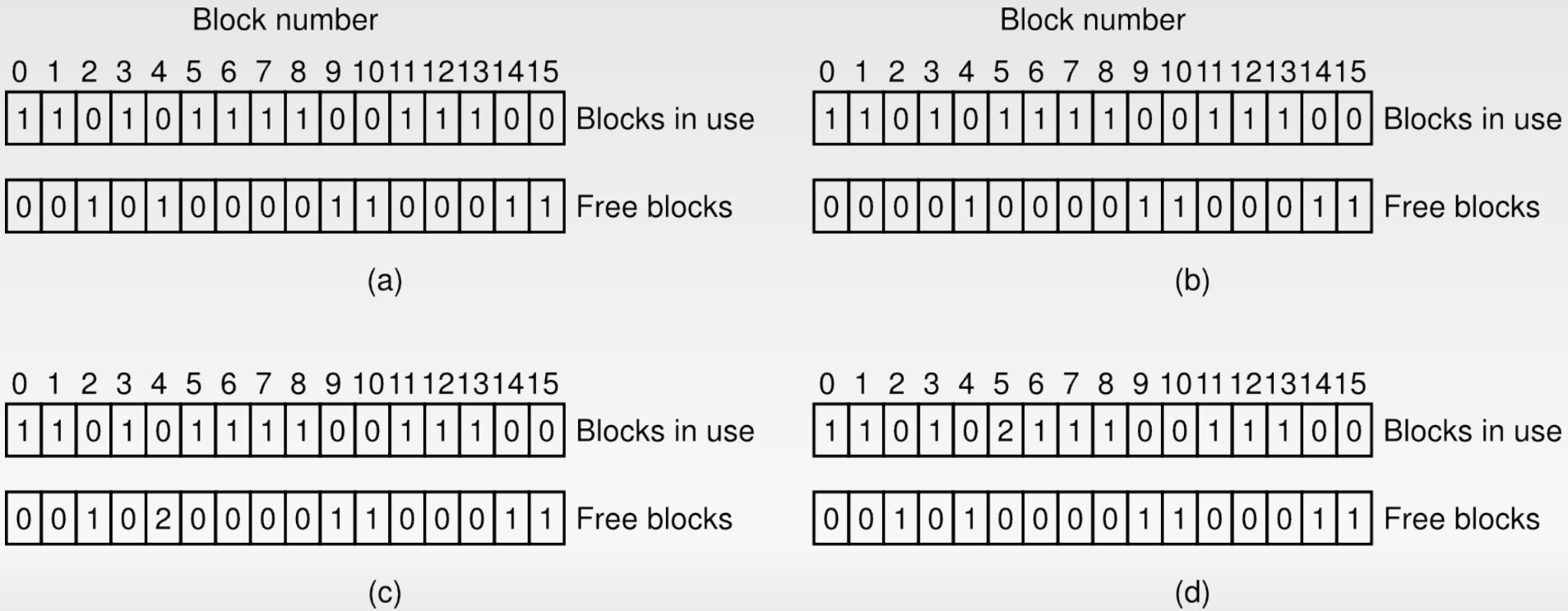
Quote su disco

- Quote per **file** e per **blocchi**;
- due tipi di limiti:
  - **limite ”soft”**;
  - **limite ”hard”**.



Controlli di consistenza

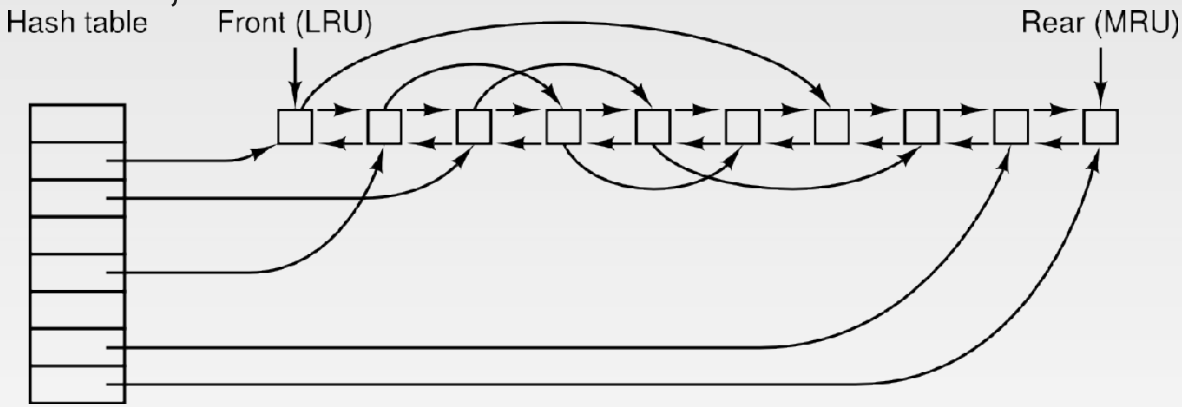
- A seguito di **crash** del sistema i file-system possono diventare inconsistenti;
- apposite **utility** possono effettuare dei **controlli di consistenza**:
  - sui **blocchi**;



  - sui **riferimenti agli i-node**.

Cache del disco

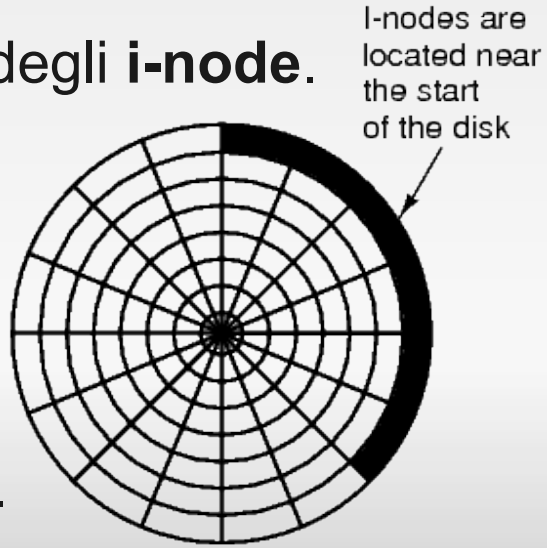
- Gli alti **tempi di accesso al disco** e la sua **relativa lentezza** nel trasferimento dei dati, implica la necessità di utilizzare metodologie per migliorarne le prestazioni.
- Uso di una **cache del disco** (o **block cache**, o **buffer cache**):
  - controllo preventivo degli accessi (lettura/scrittura);
  - struttura basata su **tabelle hash**;
  - **politiche di rimpiazzo**;
    - **LRU** (con problemi);

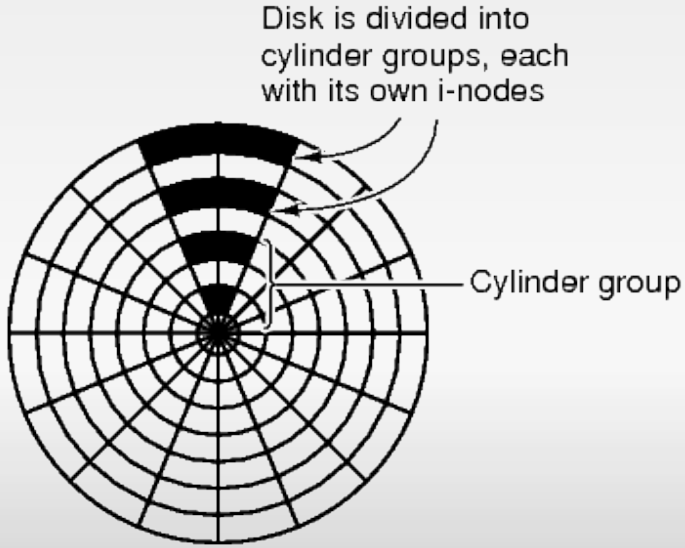


  - gestione delle scritture:
    - **cache non write-through**;
      - scrittura immediata dei **blocchi critici**;
      - **sync** periodico;
    - **cache write-through**;
      - più sicura, meno performante, pensata per i floppy.

Altre tecniche per migliorare le prestazioni

- **Lettura del blocco successivo** (block read ahead):
  - accesso **sequenziale/casuale**;
  - il file-system può cercare di tenere traccia del **pattern di utilizzo** di ogni file.
- Tecniche di allocazione che permettono la **riduzione dei movimenti del braccio del disco**:
  - scelta di "**blocchi vicini**" nell'allocazione di un file;
  - tenere conto del **posizionamento rotazionale**;
  - posizionamento degli **i-node**.

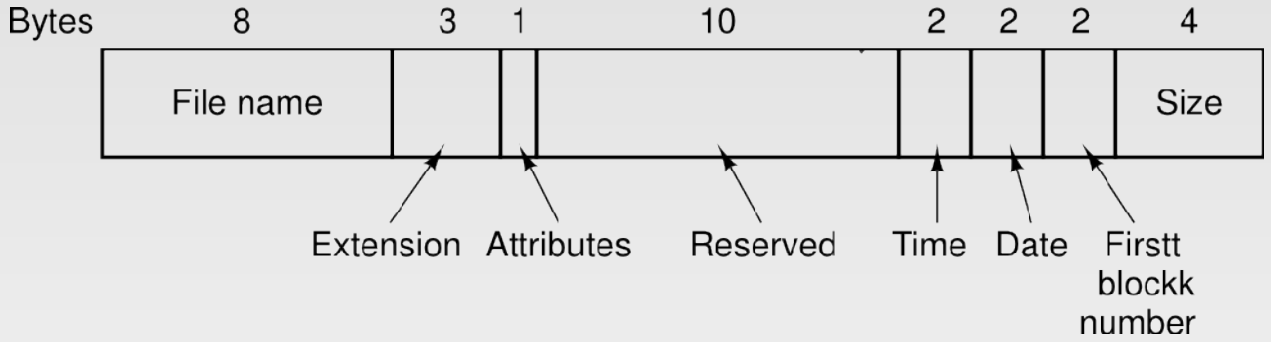




- **Deframmentazione**.

File System MS-DOS (FAT)

- Nato con **MS-DOS** si è poi evoluto in FAT-32;
- **nomi**: 8+3, con estensione per i nomi lunghi; case-insensitive;



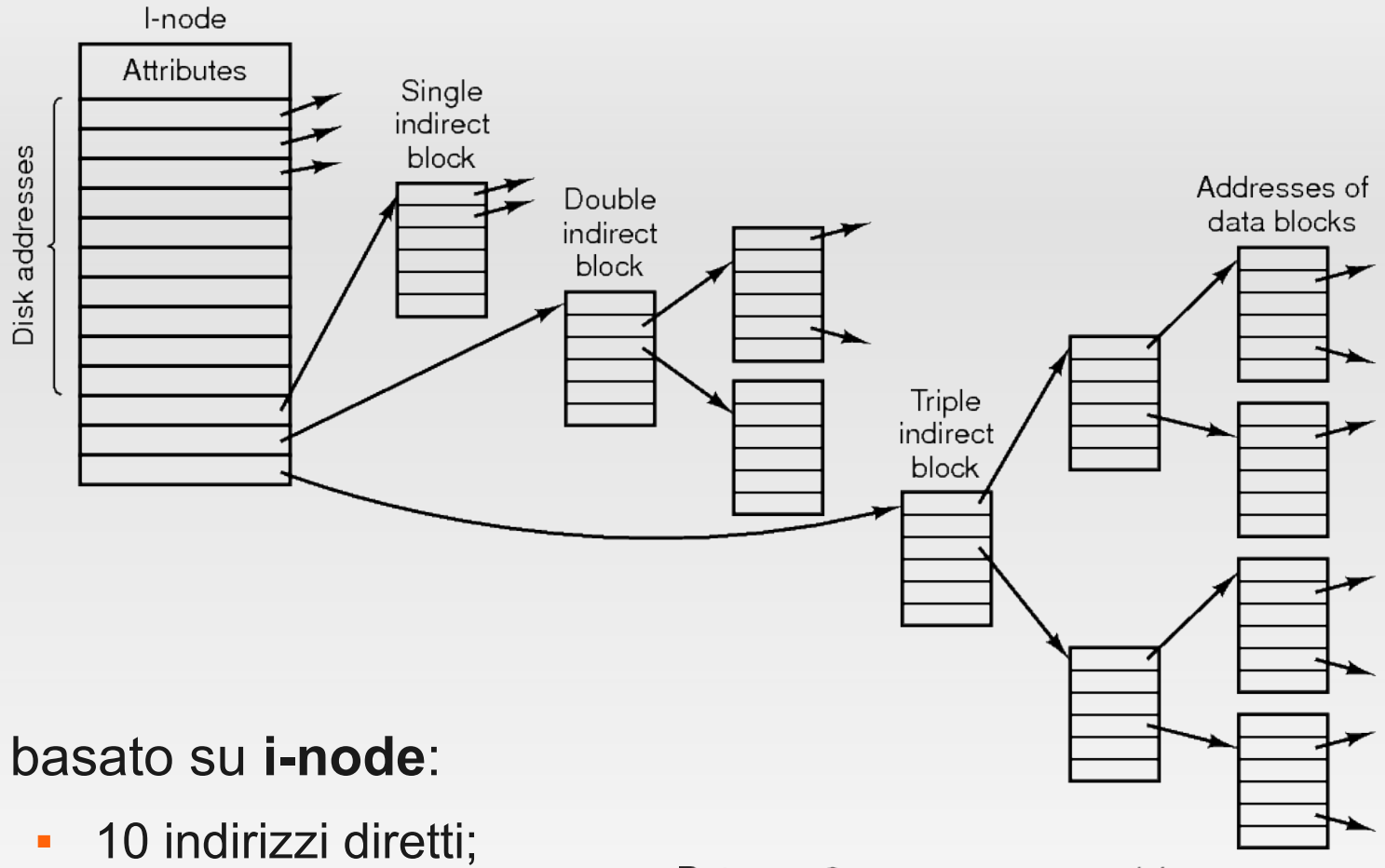
- impiega una **File Allocation Table** per la gestione dei blocchi;

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

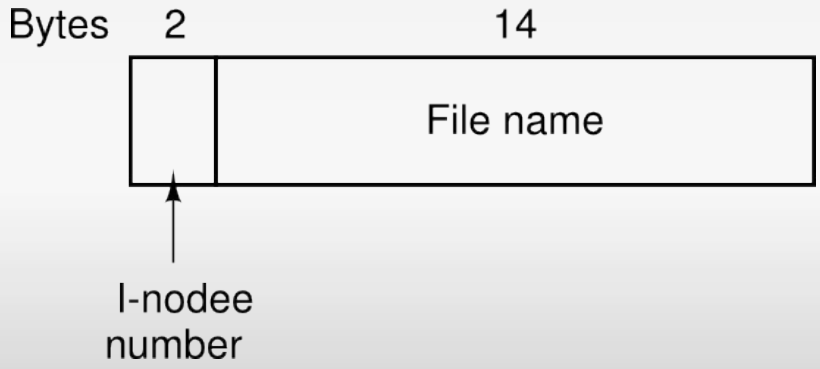
File System NTFS

- Nato con **Windows NT**: molto evoluto e complesso;
- ogni file consiste di molteplici **attributi** e può contenere molteplici **flussi**;
- basato su una **Master File Table** (**MFT**) composta da record;
- ogni **record** è associato ad un file/directory;
- sia il record che la MFT si possono **estendere**;
- alcuni attributi possono stare dentro il record, oppure esterni (**attributi non residenti**);
- nel record ci sono anche le info sull'**allocazione dei blocchi** al file;
- gestione **blocchi liberi** basata su **bitmap**;
- ultime versioni supportano: **hard-link**, **soft-link** e montaggio di altri FS;
- **compressione** e **cifratura** selettiva dei file; di recente anche la cifratura a livello di volume;
- **journaling**.

File System UNIX (V7)

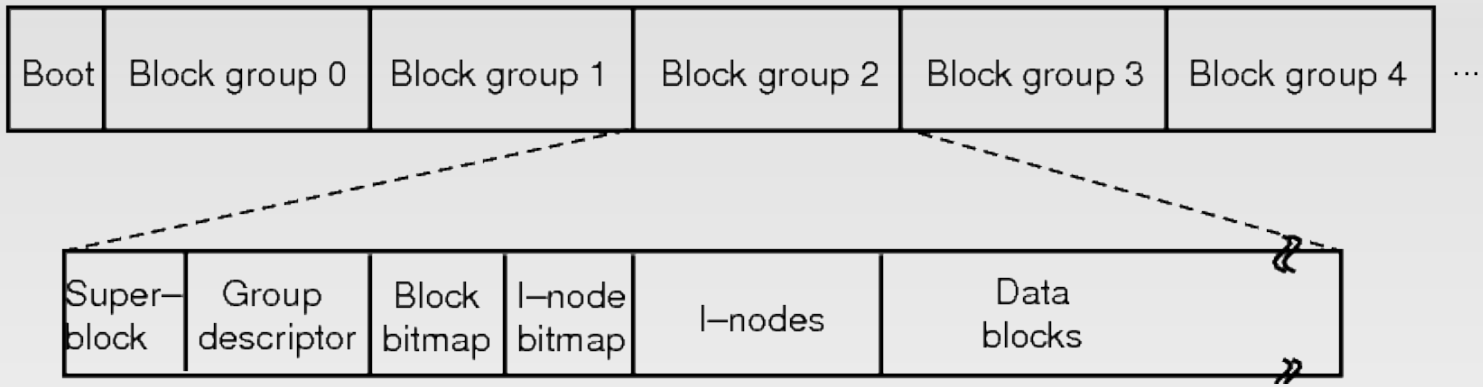


- basato su **i-node**:
  - 10 indirizzi diretti;
  - 1 blocco indiretto singolo;
  - 1 blocco indiretto doppio;
  - 1 blocco indiretto triplo;

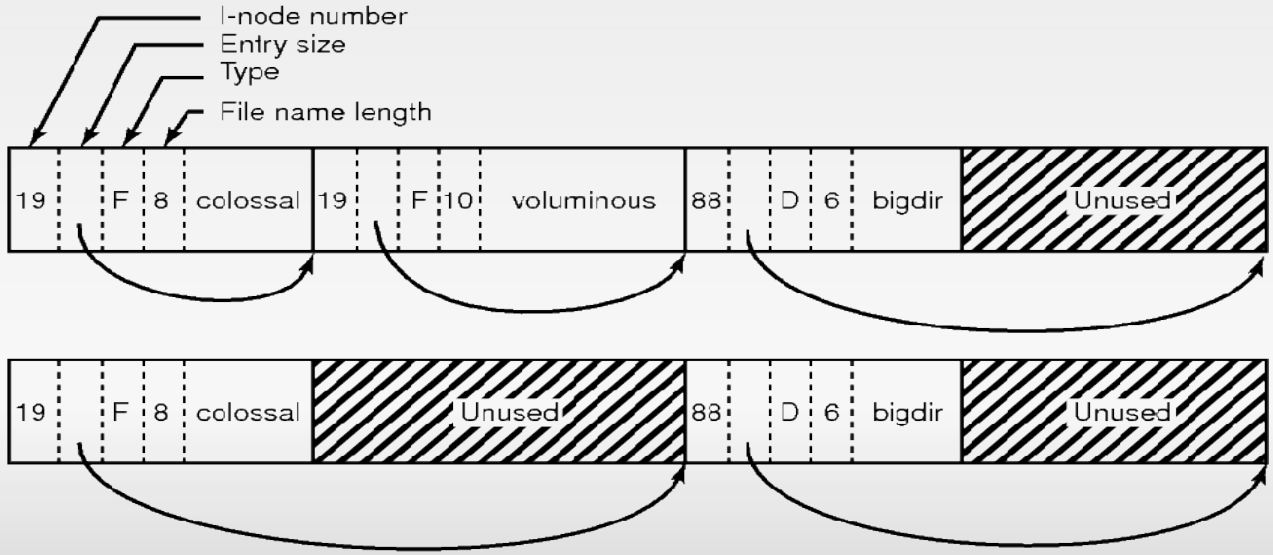


File System di Linux

- V7 → ext → ext2 → ext3 → ext4 → …
- strutture distribuite usando i **gruppi di blocchi**; **preallocazione**;



- nomi lunghezza variabile (max 255);
- **jornaling** (con ext3);
- **ext4**:
  - extent;
  - allocazione multi-blocco;
  - defram. on-line;
  - ...

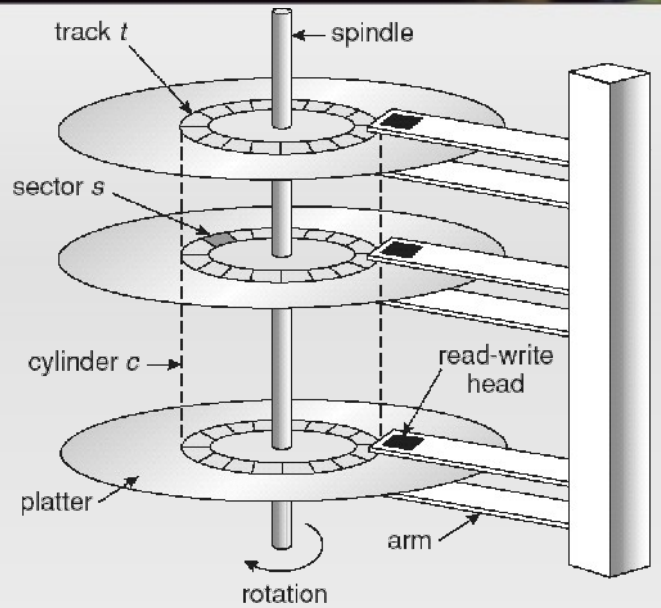


File System di Linux

- **BTRFS** (”*B-Tree FS*” o ”*Butter FS*”);
- caratteristiche addizionali rispetto ai precedenti FS Linux:
  - **clonazione** di singoli file con copy-on-write;
  - **sottovolumi**;
  - **snapshot** dei sottovolumi con copy-on-write;
  - meccanismo di **seeding** con copy-on-write di un volume rispetto ad un altro volume read-only;
  - directory basate su strutture dati **B-Tree**;
  - deframmentazione, aumento/riduzione on-line di volumi;
  - **checksum** sui blocchi di dati e dei metadati;
  - **compressione** trasparente dei file;
  - supporto a **meccanismi tipo-RAID** (mirror, striping – vedi dopo) all'interno stesso del volume;
  - ...

Scheduling del disco

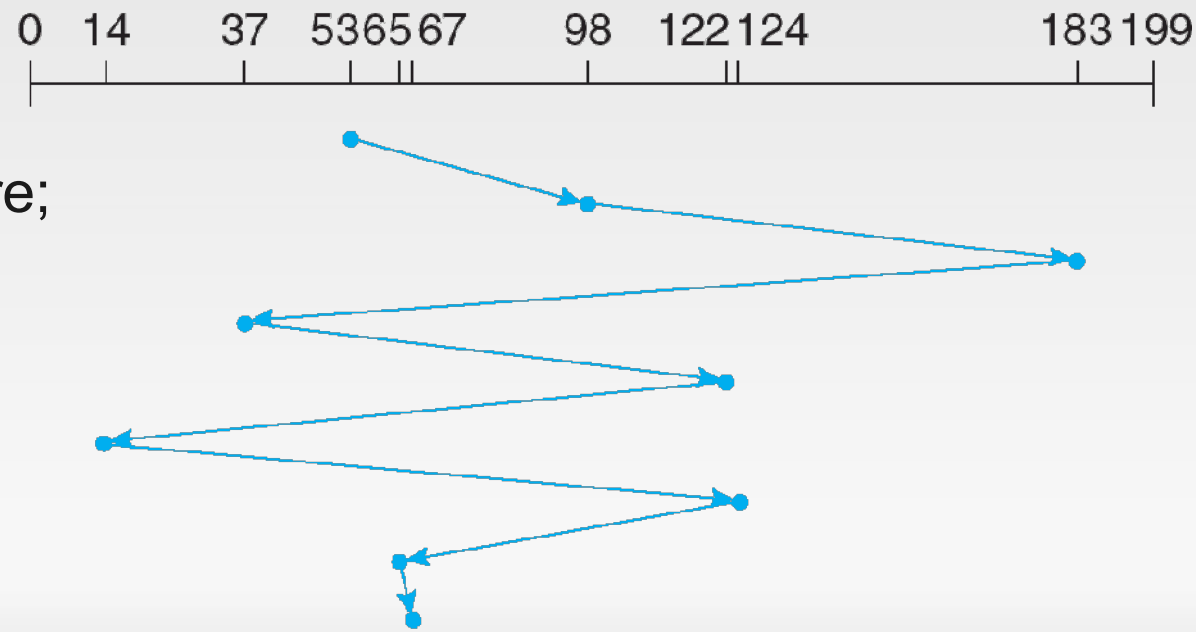
- **Obiettivi**:
  - massimizzare il numero di richieste soddisfatte in una unità di tempo (**throughput**);
  - minizzare il **tempo medio di accesso**;



- in un sistema, soprattutto se multiprogrammato, si vengono a creare varie richieste di I/O su disco che però, tipicamente, possono essere inviate al controller del disco solo una alla volta. Si crea quindi una **coda** di **richieste pendenti**;
- Il S.O. può adottare varie **politiche di selezione** della prossima richiesta da mandare;
- si può ottimizzare per:
  - **tempo di posizionamento (seek-time)**;
  - **latenza di rotazione**.

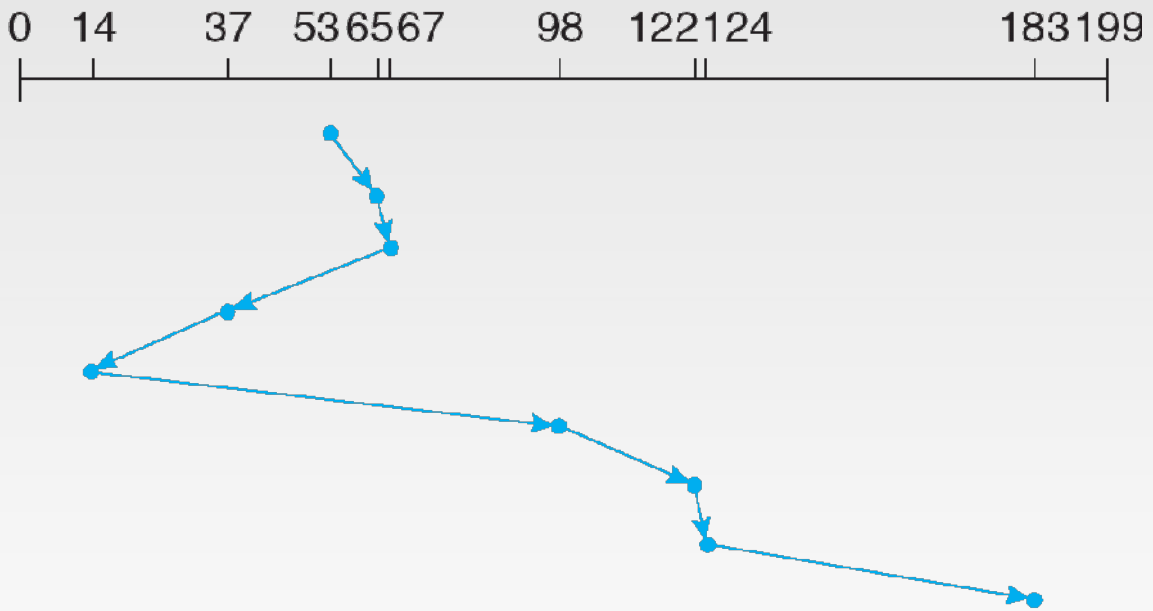
Ottimizzare il seek-time

- Vedremo varie politiche di scheduling su uno specifico **esempio**:
  - lista delle richieste in ordine di arrivo e per # di cilindro:
    98, 183, 37, 122, 14, 124, 65, 67
  - posizione iniziale della testina: cilindro ***53***
- **First Come First Served** (**FCFS**):
  - distanza totale percorsa: 640 tracce;
  - **semplice** da realizzare;
  - **equo**;
  - **inefficiente**;



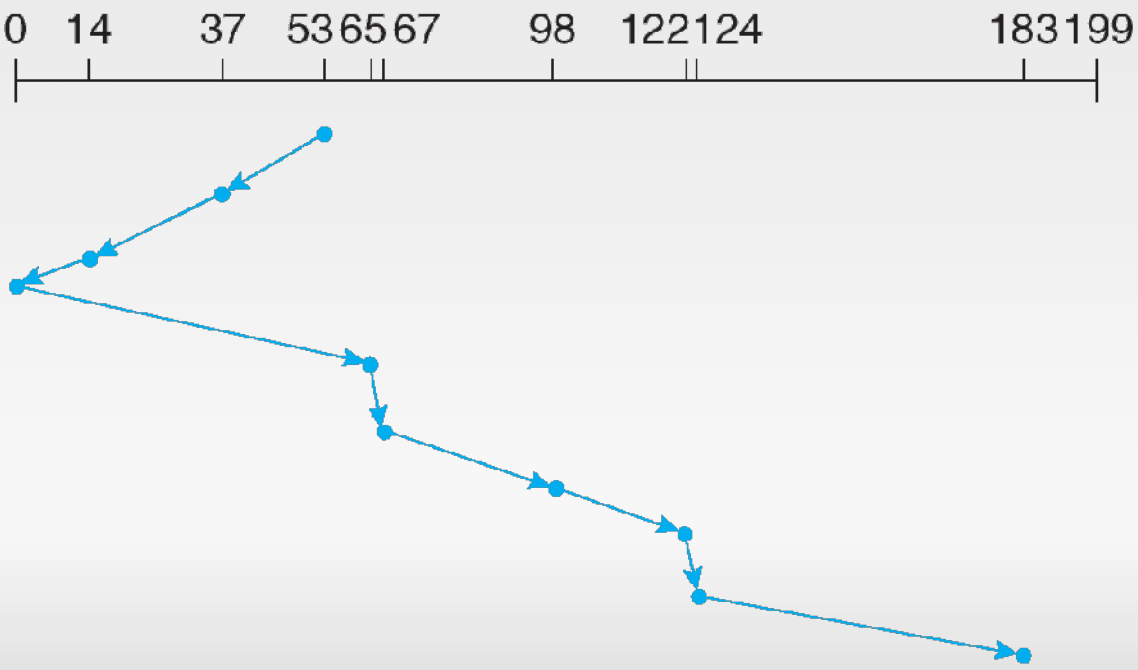
Ottimizzare il seek-time

- **esempio**: coda ***98, 183, 37, 122, 14, 124, 65, 67***; cilindro iniziale ***53***;
- **Shortest Seek Time First** (**SSTF**):
  - ordine usato: 65, 67, 37, 14, 98, 122, 124, 183;
  - distanza totale percorsa: 236 tracce;
  - **buone prestazioni** (ma non ottimale);
  - **non equo** (*starvation*);



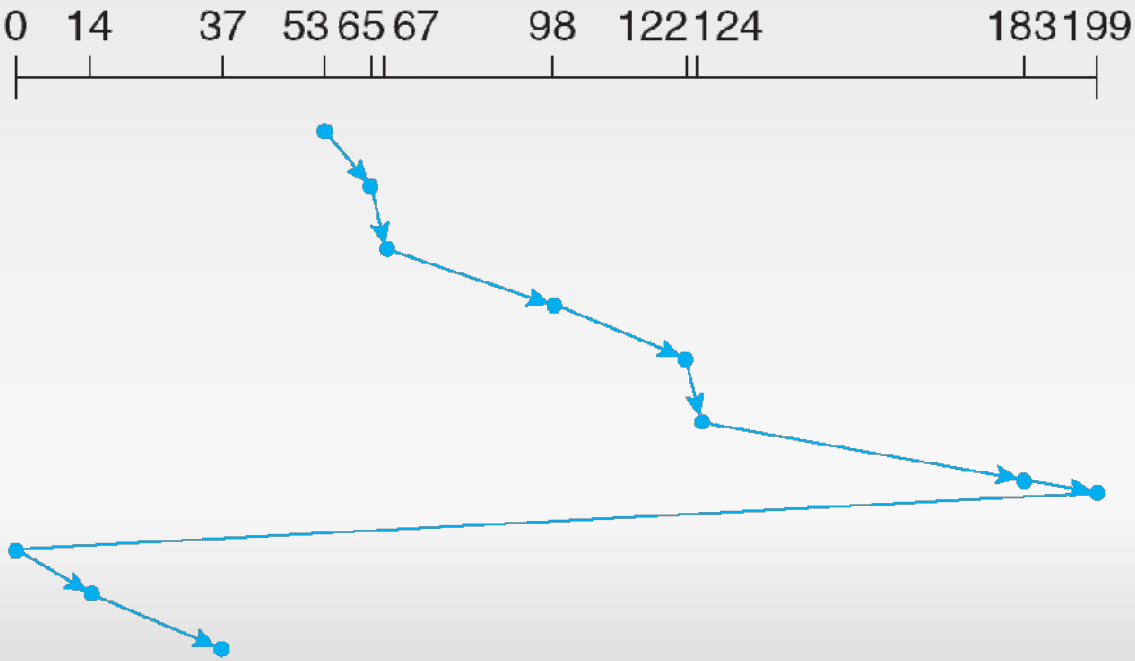
Ottimizzare il seek-time

- **esempio**: coda ***98, 183, 37, 122, 14, 124, 65, 67***; cilindro iniziale ***53***;
- **Scheduling per scansione** (**SCAN**):
  - cambia direzione solo dopo aver raggiunto un estremo;
  - detto quindi **algoritmo dell'ascensore**;
  - ordine usato: 37, 14, 0, 65, 67, 98, 122, 124, 183
  - scansione uniforme;
  - **garantisce** comunque una attesa massima per ogni richiesta, ma si può fare di meglio...



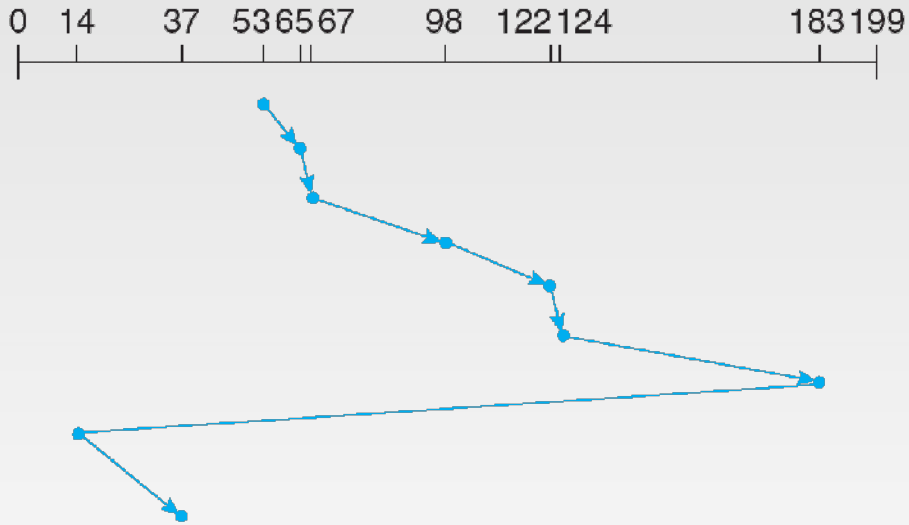
Ottimizzare il seek-time

- **esempio**: coda ***98, 183, 37, 122, 14, 124, 65, 67***; cilindro iniziale ***53***;
- **Scheduling per scansione circolare** (**C-SCAN**):
  - considera le posizioni come collegate in **modo circolare**: arrivato alla fine del disco torna sul primo cilindro senza servire alcuna richiesta;
  - ordine usato: 65, 67, 98, 122, 124, 183, 199, 0, 14, 37
  - garantisce un **tempo medio di attesa** più **uniforme**;



Ottimizzare il seek-time

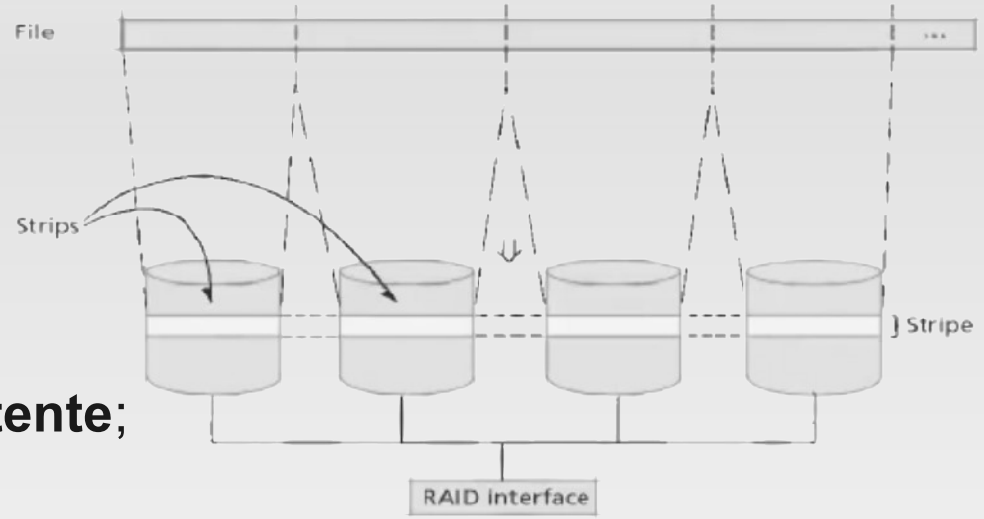
- **esempio**: coda ***98, 183, 37, 122, 14, 124, 65, 67***; cilindro iniziale ***53***;
- **Scheduling LOOK**:
  - piccola **ottimizzazione** per SCAN e C-SCAN: evita di arrivare agli estremi del disco se non necessario:
    - SCAN → LOOK;
    - C-SCAN → C-LOOK;



  - esempio di C-LOOK;
    - ordine usato: 65, 67, 98, 122, 124, 183, 14, 37.
- **Come scegliere?**
  - varianti circolari per alto carico;
  - varianti non circolari per basso carico.

Sistemi RAID

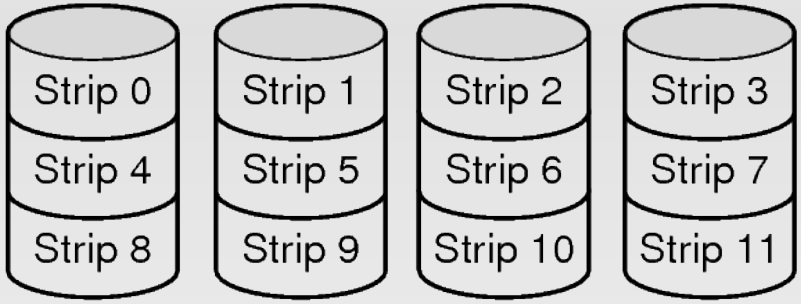
- Un altro modo per aumentare le **prestazioni** è sfruttare il **parallelismo** anche per l'I/O su disco:
  - servono più **dischi indipendenti**;
  - si suddividono i dati relativi ad una unità logica (un file, in generale un volume) su più dischi: **striping**;
    - suddivisione **trasparente all'utente**;



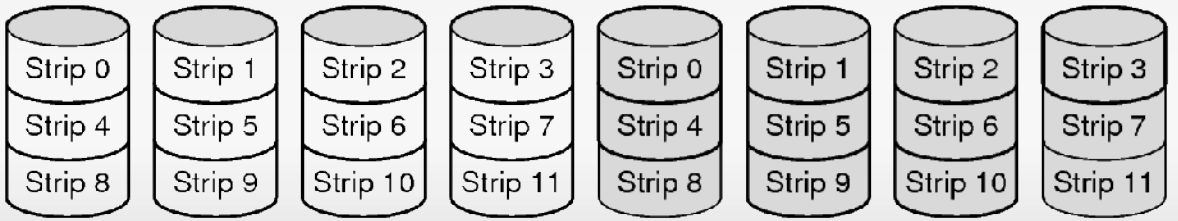
- problema: aumenta la probabilità che si **verifichi un guasto** sul volume logico RAID;
  - soluzione: aggiungiamo ridondanza per ottenere migliore **affidabilità**;
- **Redundant Array of Inexpensive Disks** (**RAID**);
  - aka **Redundant Array of Independent Disks**;
- vedremo vari schemi di gestione che bilanciano questi due aspetti;
  - **livelli RAID**;
- via **hardware** (trasparente al S.O.) o via **software** (con carico sulla CPU).

Sistemi RAID

- **RAID 0** (*striping*):
  - fa ***striping*** in modalità round-robin;
  - semplice con **prestazioni ottimali** con letture di grandi volumi;
  - niente ridodanza: **maggiore vulnerabilità**;

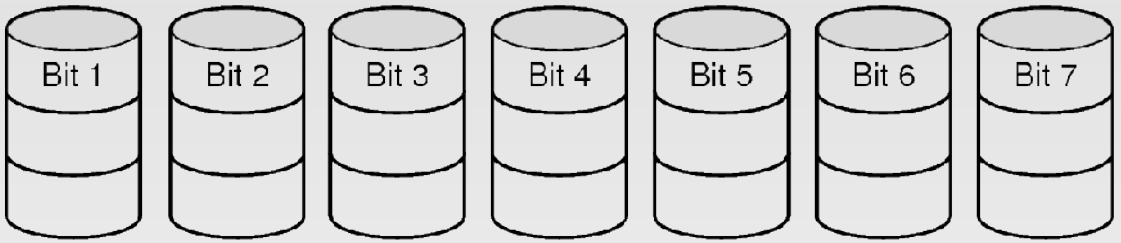


- **RAID 1** (*mirroring*):
  - gli eventuali stripe vengono anche duplicati (***mirroring***);
  - può anche essere usato senza striping;
  - raddoppio prestazioni in lettura;
  - migliore ***fault tollerance***;
  - alto **overhead** di storage;

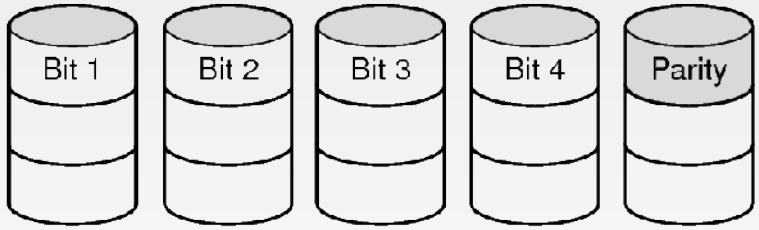


Sistemi RAID

- **RAID 2** (*striping a livello di bit con ECC*):
  - lavora sulle parole o byte applicando un codice di correzione degli errori – ECC (tipo **codice di Hamming** per singoli bit di errore);
  - esempio: 4 bit dati + 3 bit ridondanza;
  - buone prestazioni;
  - ottima **fault tollerance**;
  - serve **sincronizzare** le rotazioni dei dischi;



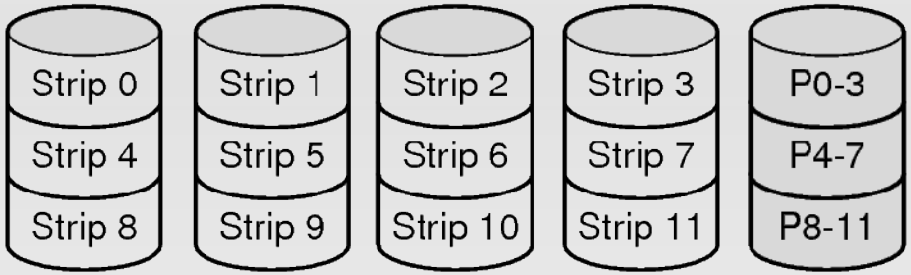
- **RAID 3** (*striping a livello di bit con bit di parità*):
  - usa un solo disco con raccolti i singoli **bit di parità**;
  - in realtà permette anche di **recuperare** i dati e offre la stessa capacità di fault tollerance del RAID 2;
  - serve ancora sincronizzazione;



- fare striping a livello di bit è comunque pesante se non gestito a livello hardware;

Sistemi RAID

- **RAID 4** (*striping a livello di blocchi con XOR sull'ultimo disco*):
  - basato su **stripe a blocchi**;
  - disco extra = **XOR** degli stripe;
  - non necessità sincronizzazione;
  - ottima fault tollerance;
  - **aggiornamento** lento in caso di modifica di un blocco?
    - **ottimizzazione**: il nuovo blocco di parità si può calcolare dal blocco sovrascritto e dal vecchio blocco di parità;



- **RAID 5** (*striping a livello di blocchi ma con informazioni di parità distribuite*):
  - il blocchi di parità del RAID 4 vengono distribuiti su tutti i dischi;
  - di fatti sostituisce il RAID 4.