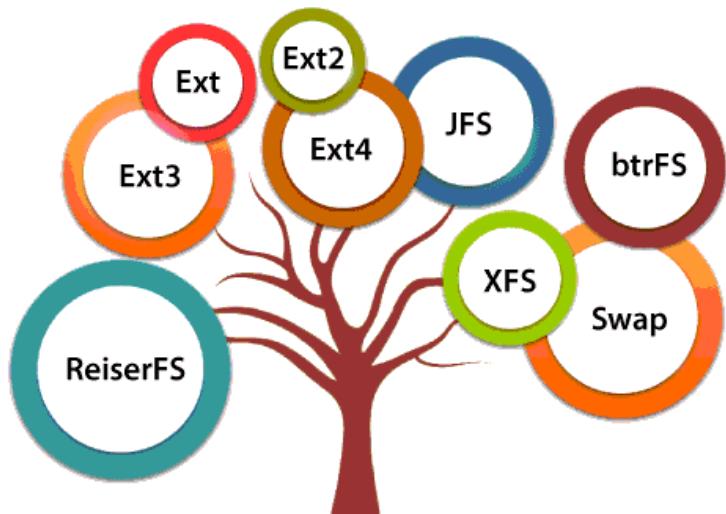


SISTEMI OPERATIVI (M-Z)

Types of Linux File System



C.d.L. in Informatica
(laurea triennale)

Dip. di Matematica e Informatica
Università degli Studi di Catania

Anno Accademico
2021-2022

Prof. Mario F. Pavone
mario.pavone@unict.it
mpavone@dmi.unict.it

File System e Dischi



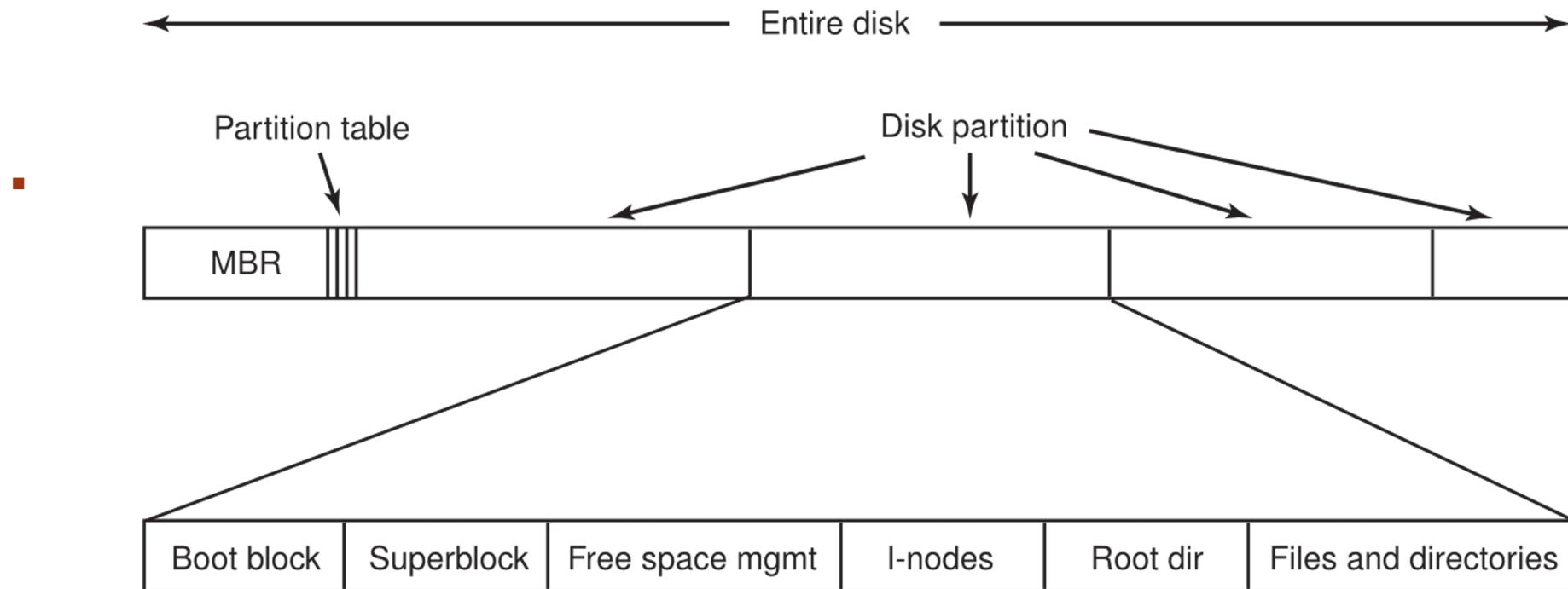
I FILE SYSTEM

- Problema di base: gestire **grandi quantità** di informazioni, in modo **persistente** e condiviso **tra più processi**;
- astrazione: **file** e **directory**;
- i dettagli di gestione ed implementazione costituiscono il **file system**;
- esempi di **dettagli**:
 - nomenclatura;
 - tipi di file;
 - tipi di accesso;
 - metadati (o attributi);
 - operazioni supportate sui file;
 - accesso condiviso ai file: i **lock**:
 - **shared vs. exclusive**;
 - **mandatory vs. advisory**;
- **strutture dati** per la gestione dei file: globale e per processo.



STRUTTURA DI UN FILE SYSTEM

- **Master Boot Record (MBR)** => settore 0 del disco;
- partizioni e **boot record** (o boot block);
- **superblocco**;

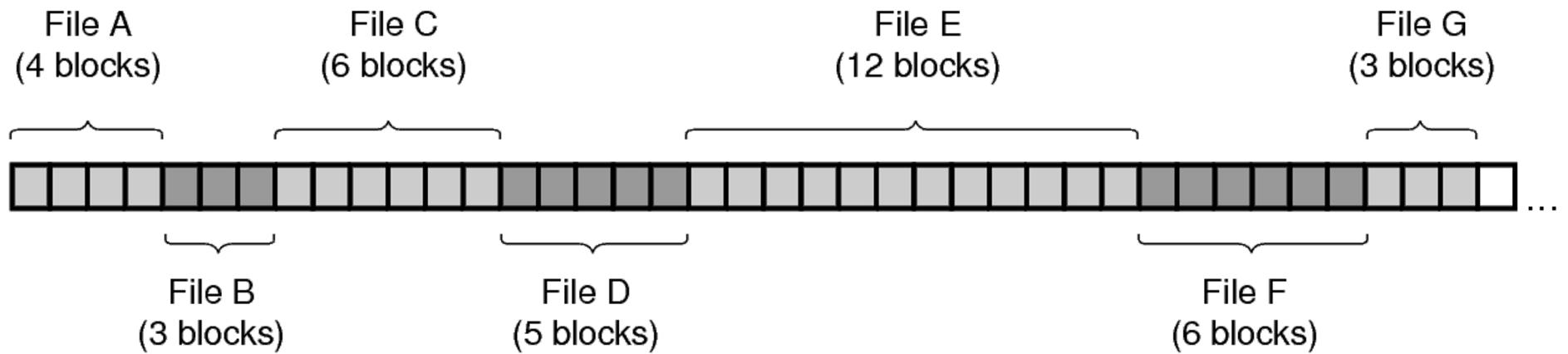


- layout moderno alternativo: **GPT** (GUID Partition Table) definito dallo standard **EFI**.

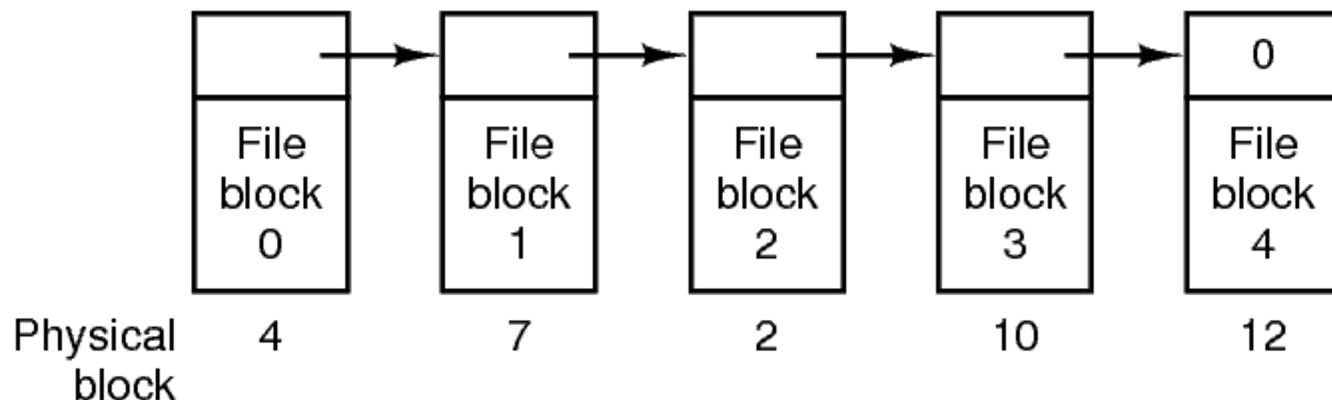


IMPLEMENTAZIONE DEI FILE

▪ Allocazione contigua.

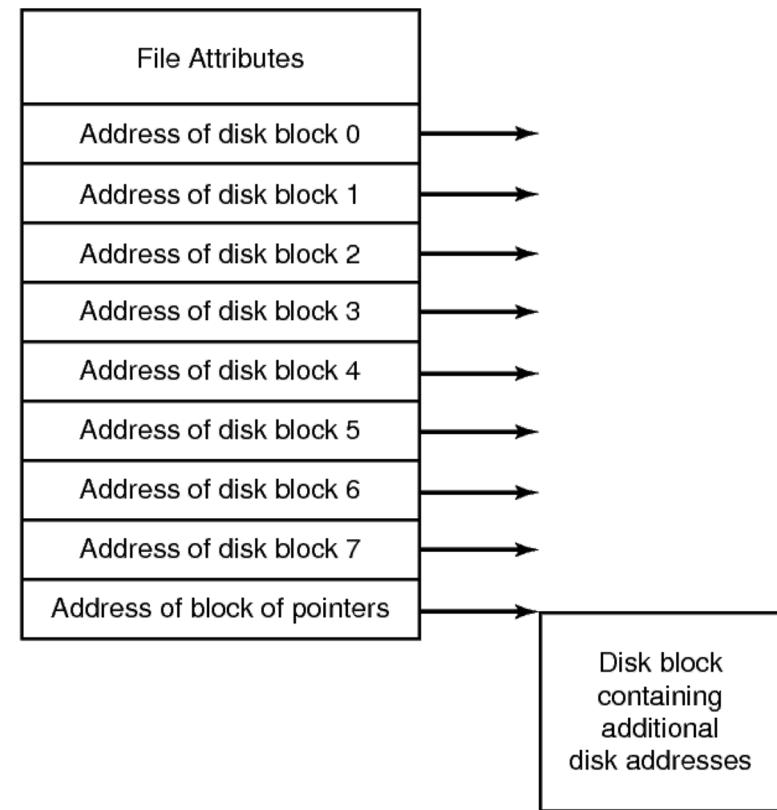
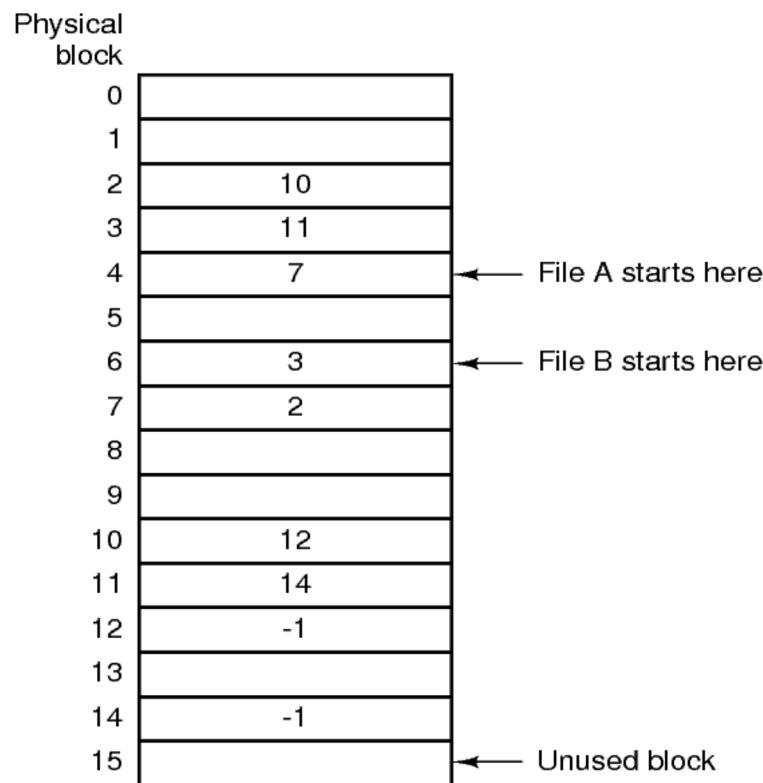


▪ Allocazione con liste collegate (o allocazione concatenata).



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: **collegata, multilivello, ibrida.**



IMPLEMENTAZIONE DELLE DIRECTORY

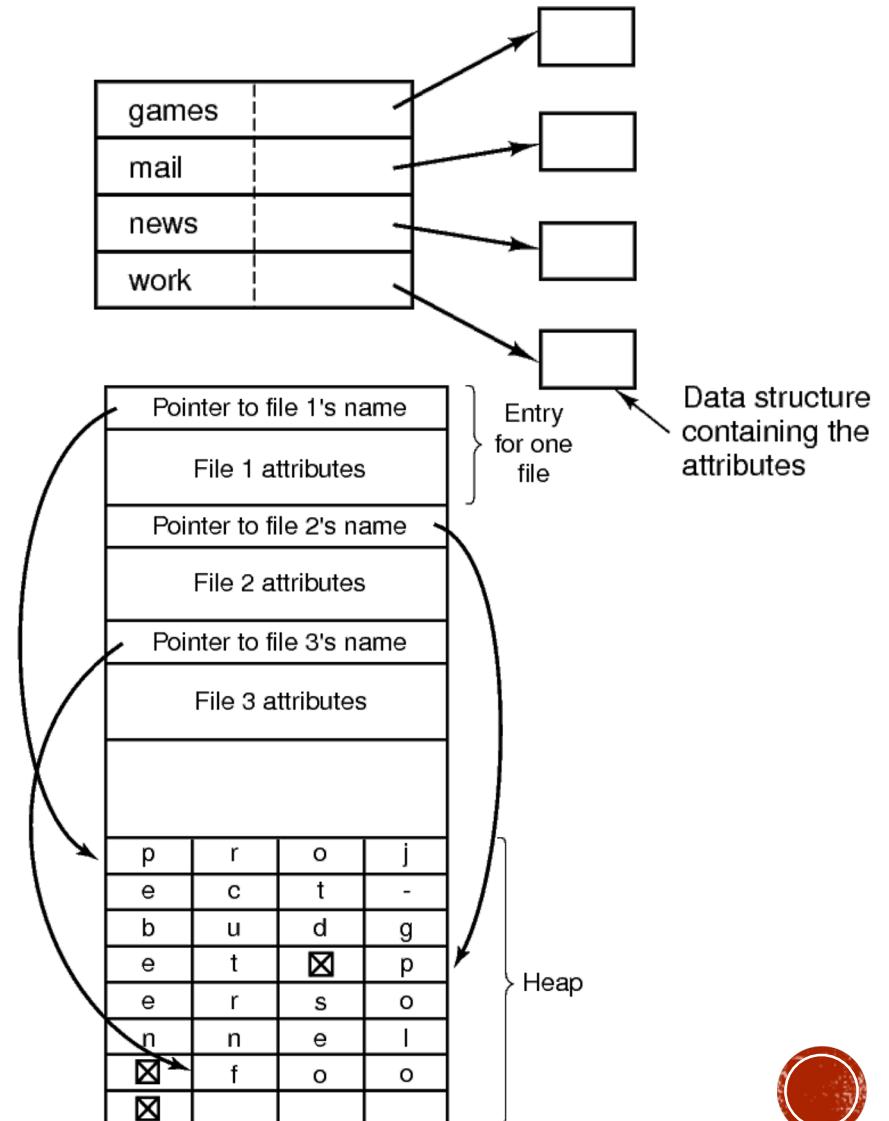
- Dove memorizzare i **metadati/attributi**?

games	attributes
mail	attributes
news	attributes
work	attributes

games	
mail	
news	
work	

Entry for one file

File 1 entry length			
File 1 attributes			
p	r	o	j
e	c	t	-
b	u	d	g
e	t	<input checked="" type="checkbox"/>	
File 2 entry length			
File 2 attributes			
p	e	r	s
o	n	n	e
i	<input checked="" type="checkbox"/>		
File 3 entry length			
File 3 attributes			
f	o	o	<input checked="" type="checkbox"/>
:			



- nomi lunghi:**

- lung. variabile;
- tramite heap;

- prestazioni:**

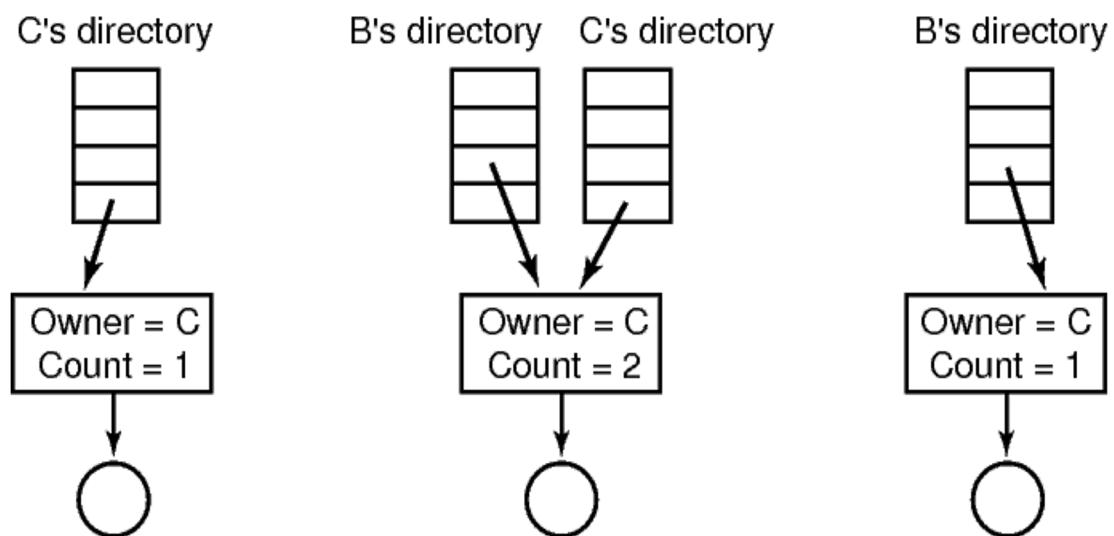
- tabella hash;
- cache.



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.

append=Consente ai programmi di aprire i file di dati nelle directory specificate come se fossero nella directory corrente. Se usato senza parametri, append visualizza l'elenco di directory accodato.



In informatica, con la parola inglese accounting ci si riferisce a tutte le azioni che tracciano ovvero registrano, misurano e documentano le risorse concesse ad un utente durante un accesso ad un server o più in generale ad un sistema informatico.



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;

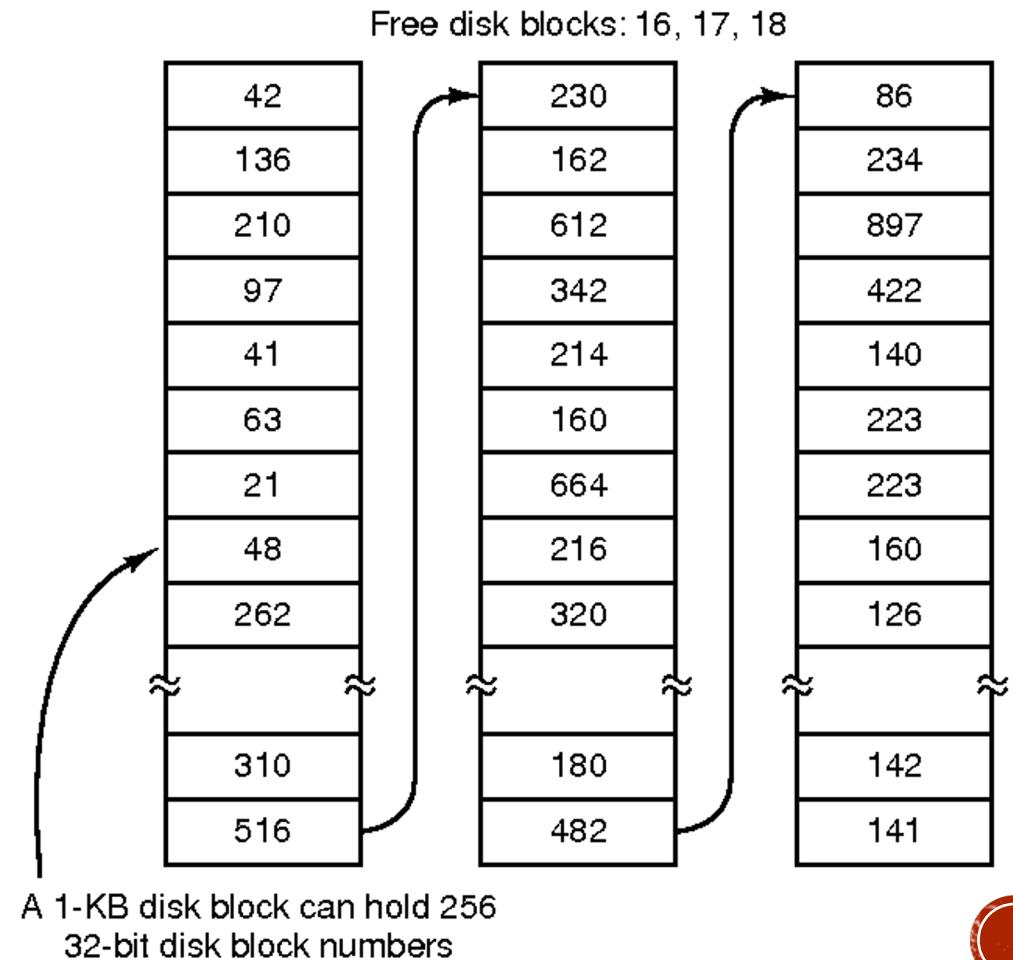
1001101101101100
011011011110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
~ ~
0111011101110111
1101111101110111



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.



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**;

Block number																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0	
Blocks in use																
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1	
Free blocks																

(a)

Block number																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	0	1	1	1	1	0	0	1	1	1	1	0	0
Blocks in use																
0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1	
Free blocks																

(b)

Block number																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	0	1	1	1	1	0	0	1	1	1	1	0	0
Blocks in use																
0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1	
Free blocks																

(c)

Block number																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	0	2	1	1	1	0	0	1	1	1	1	0	0
Blocks in use																
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1	
Free blocks																

(d)

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



JOURNALING

- **strategia:**

- le operazioni sui meta-dati 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;

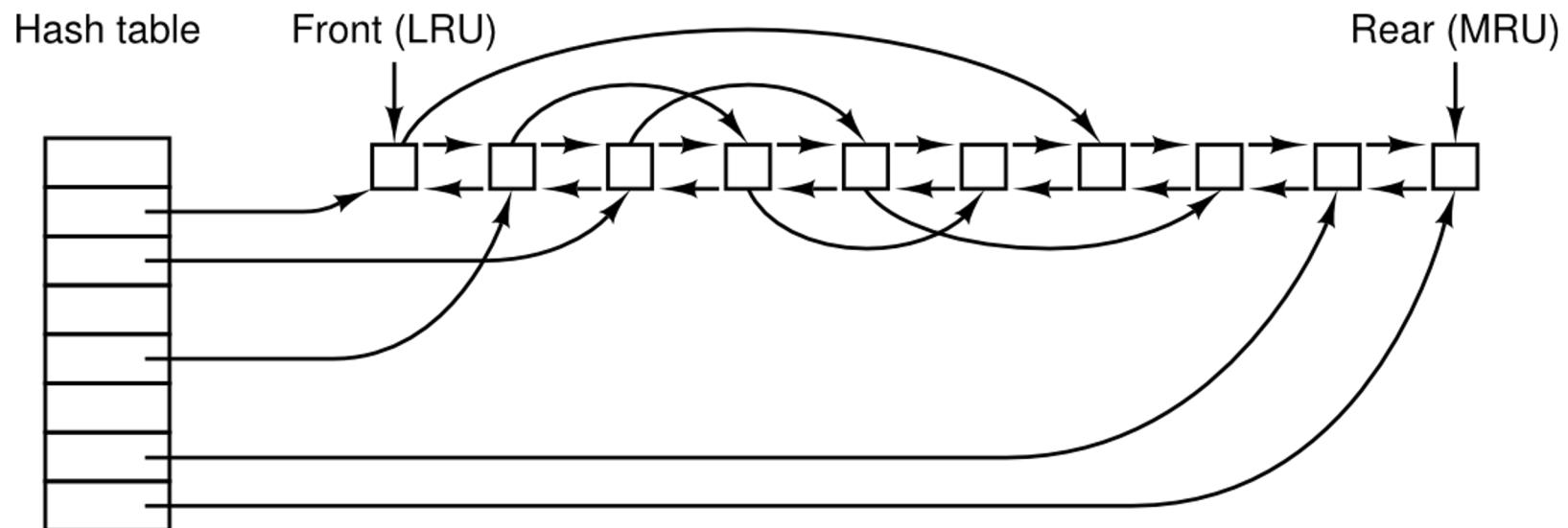
- **benefici:**

- maggiore robustezza dei metadati;
- veloce ripristino da crash/reboot;
- affinché tutto funzioni bisogna operare con operazioni **idempotenti**.



CACHE DEL DISCO

- Per migliorare le prestazioni dei dischi si fa spesso uso di una **cache del disco** (o **buffer cache**):
 - struttura basata su **tabelle hash**;

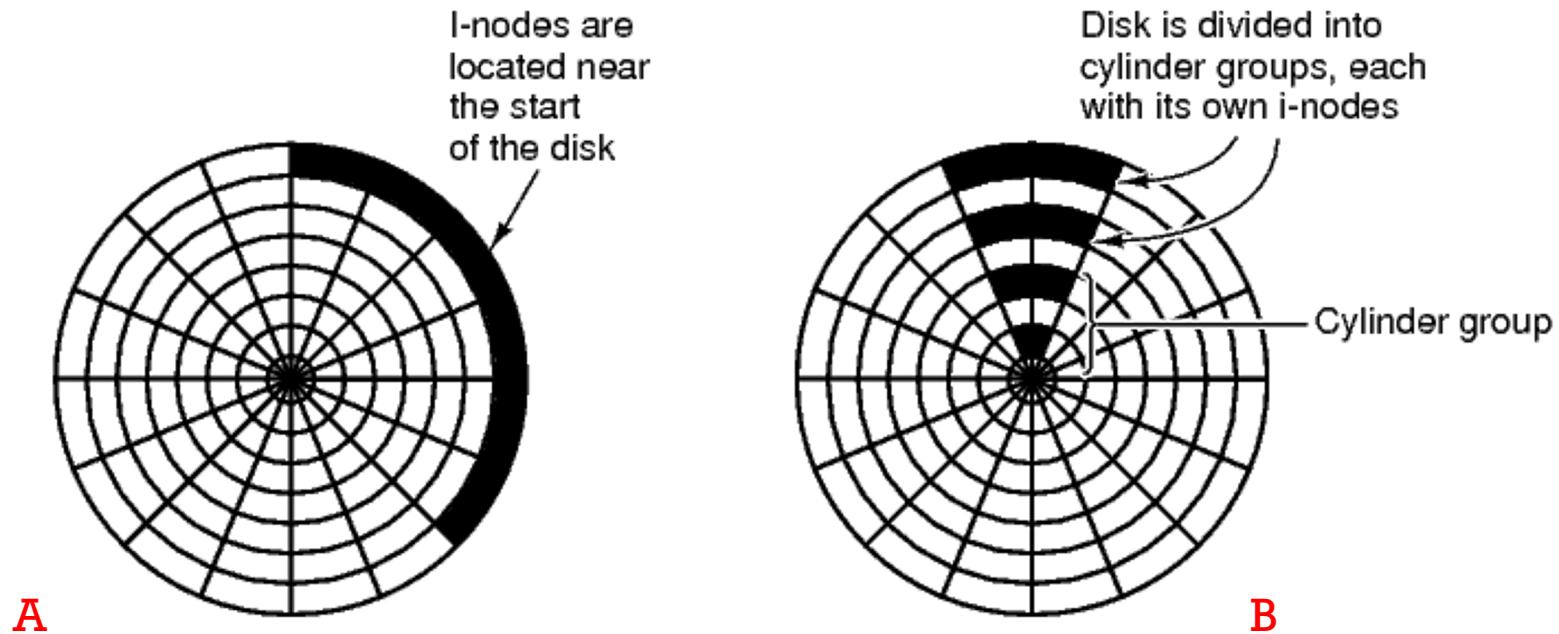


- strategie di gestione:
 - **LRU modificata**;
 - **free-behind & read-ahead**;
- scrittura: **sincrona vs. asincrona**.



ALTRÉ TECNICHE PER MIGLIORARE LE PRESTAZIONI

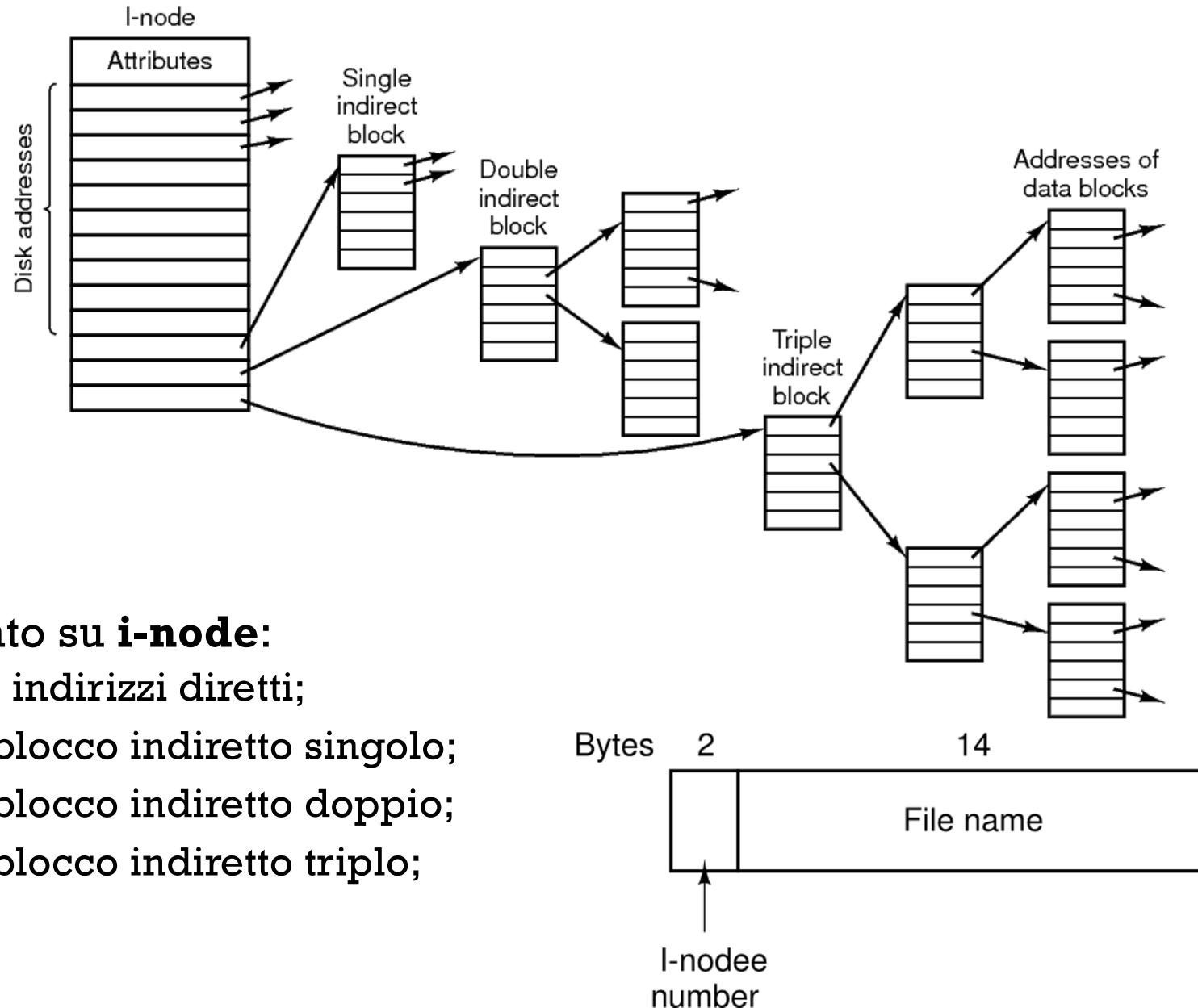
- Tecniche di allocazione che permettono la **riduzione dei movimenti del braccio del disco**:
 - scelta di "**blocchi vicini**" nell'allocazione di un file;
 - posizionamento degli **i-node**.



- **Deframmentazione.**

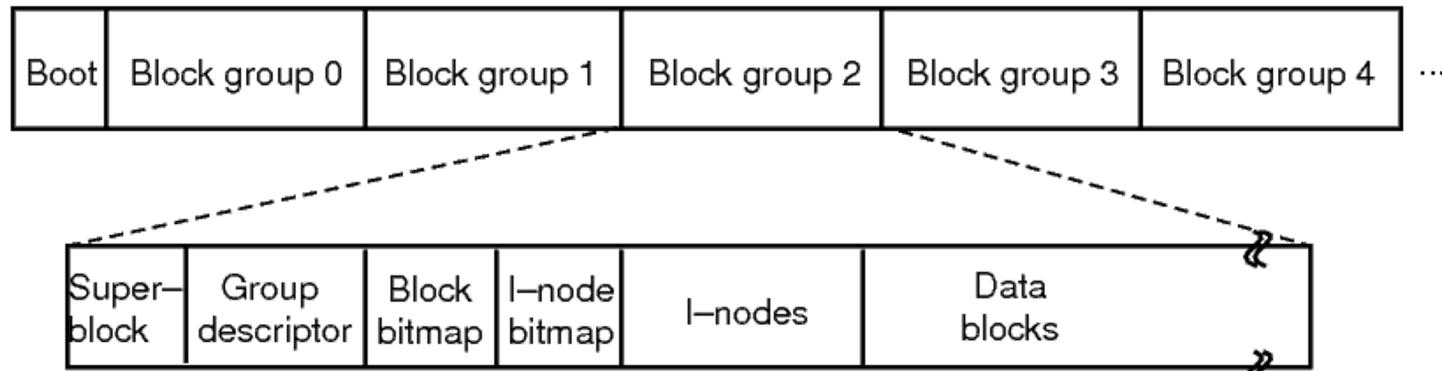


FILE SYSTEM UNIX (V7)



FILE SYSTEM DI LINUX

- V7 → ext → ext2 → ext3 → ext4 → ...
- strutture distribuite basata sui **gruppi di blocchi**; strategia di **preallocazione**;

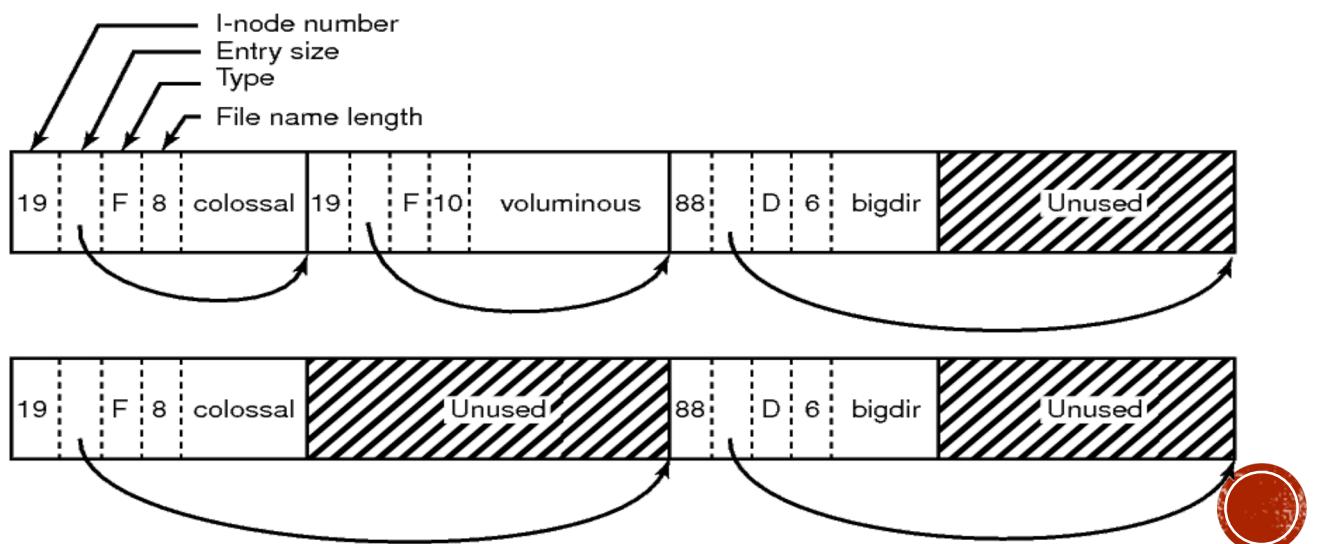


- nomi lunghezza variabile (max 255);

- **journaling** (con ext3);

- **ext4:**

- extent e i-node a dim. variabile;
- allocazione multi-blocco;
- allocazione ritardata;
- deframmentazione on-line;



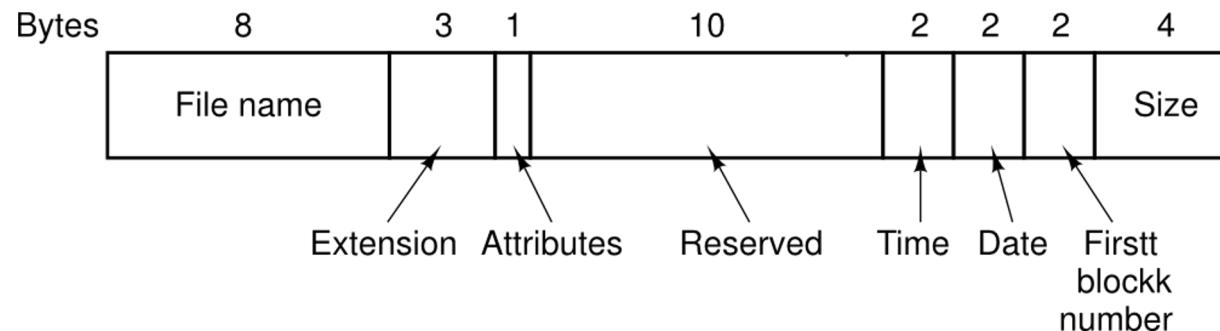
FILE SYSTEM DI LINUX

- **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;
 - primitive di **send/receive** su differenze tra snapshot;
 - 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;
 - ...



FILE SYSTEM MS-DOS (FAT)

- Nato con **MS-DOS** si è poi evoluto nella "famiglia FAT";
- **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

- versione attuale: **exFAT** (extended FAT o FAT-64).



FILE SYSTEM NTFS

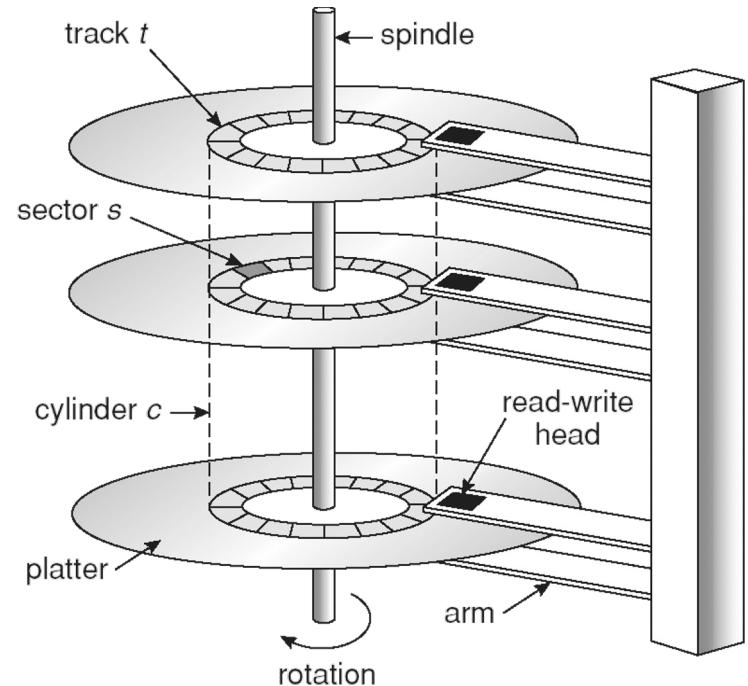
- Nato con **Windows NT**: molto evoluto e complesso;
- **volume** (una o più partizioni o dischi) e **cluster** (unità elementare di alloc.);
- **master file table (MFT)** composta da **record** (di dimensione fissa);
- **file**: collezione di più **attributi** di varia natura
 - un attributo può corrispondere ad un metadato del file o ad un flusso;
 - **residenti** e **non residenti**;
 - un file può richiedere più record
- directory basata sulla struttura dati **B+ Tree** per ricerche efficienti;
- gestione **blocchi liberi** basata su **bitmap**;
- **hard-link**, **soft-link** e montaggio di altri FS (nelle ultime versioni);
- **journaling**;
- **compressione** e **cifratura** selettiva dei file; di recente anche la cifratura a livello di volume;
- **copie shadow** di volumi con tecnica copy-on-write.



SCHEDULING DEL DISCO

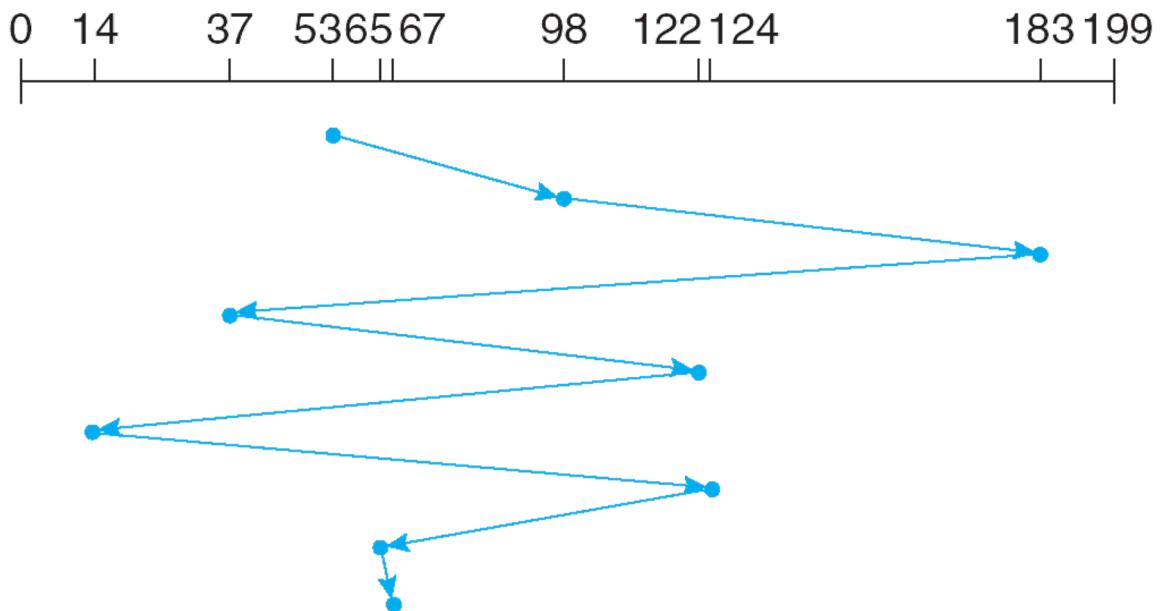
- **Obiettivi:**

- 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**.



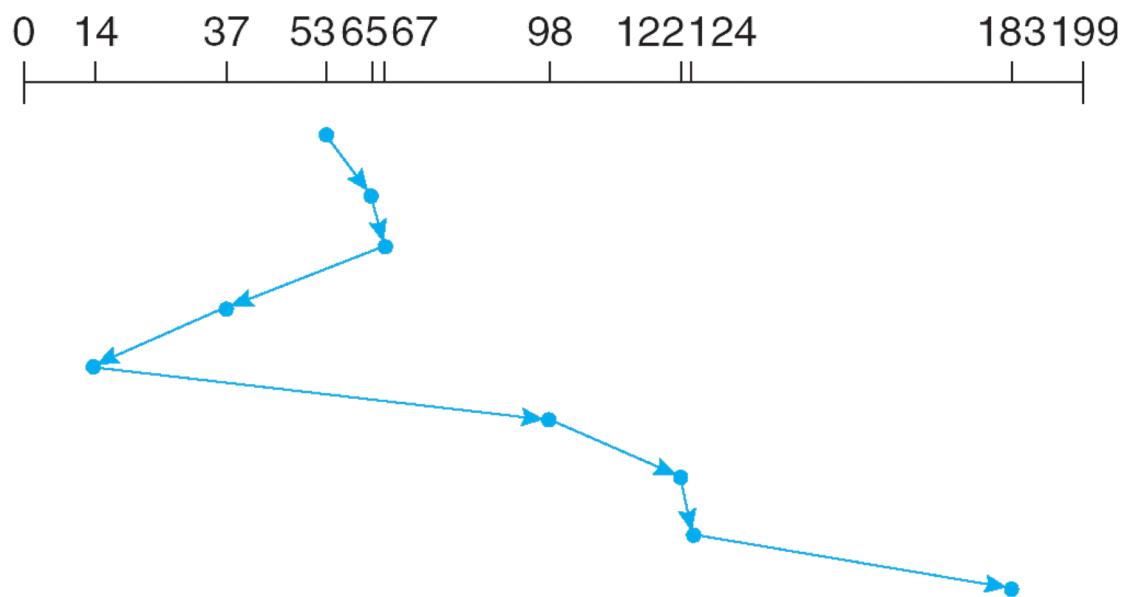
OTTIMIZZARE IL SEEK-TIME

- 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)**;



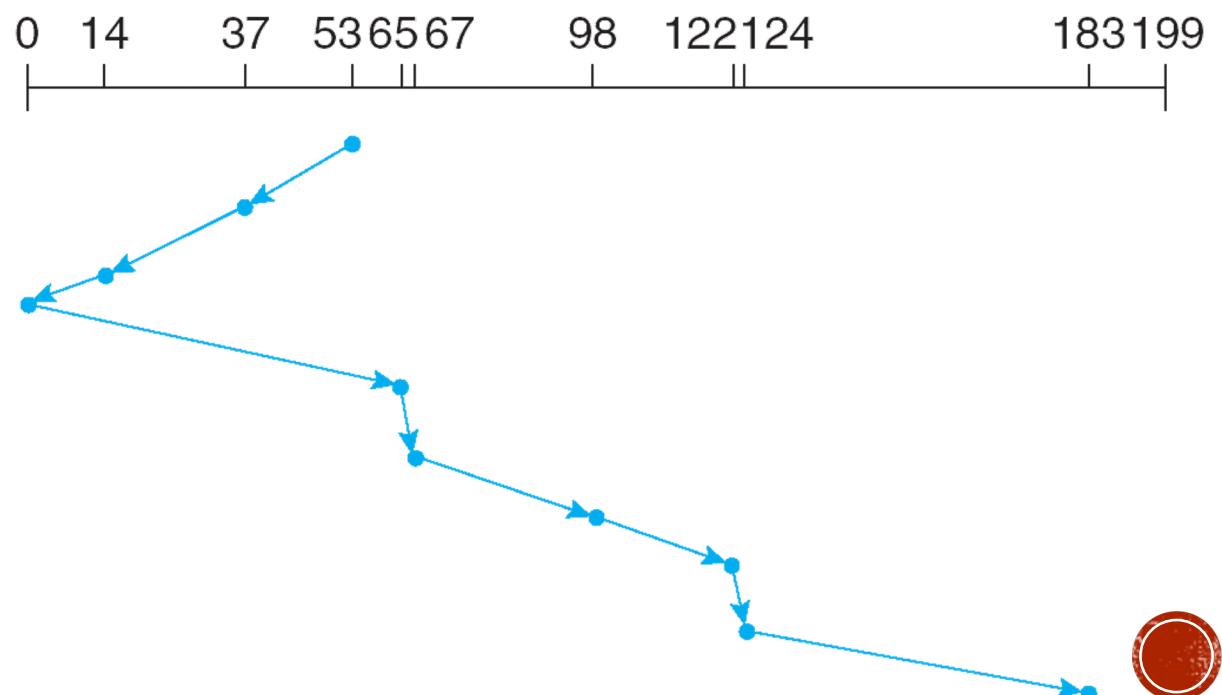
OTTIMIZZARE IL SEEK-TIME

- **esempio:** coda **98, 183, 37, 122, 14, 124, 65, 67**; cilindro iniziale **53**;

- **Scheduling per scansione (SCAN):**

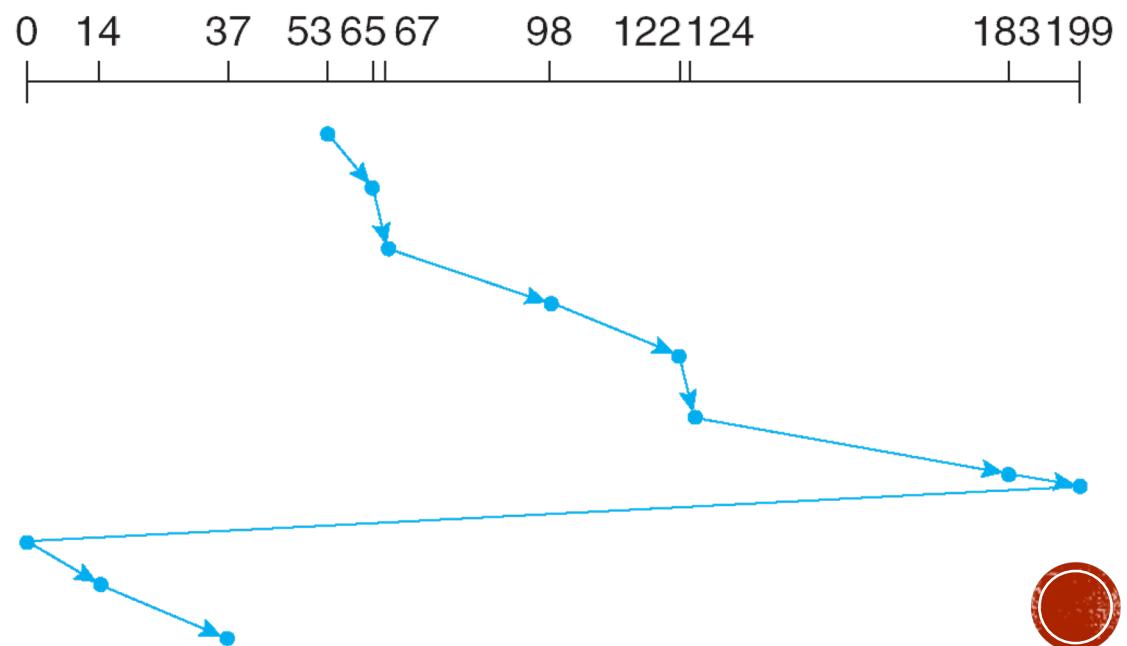
- cambia direzione solo dopo aver raggiunto un estremo;
- detto quindi **algoritmo dell'ascensore**;
- ordine usato: **37, 14, 0, 65, 67, 98, 122, 124, 183**

- 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**;



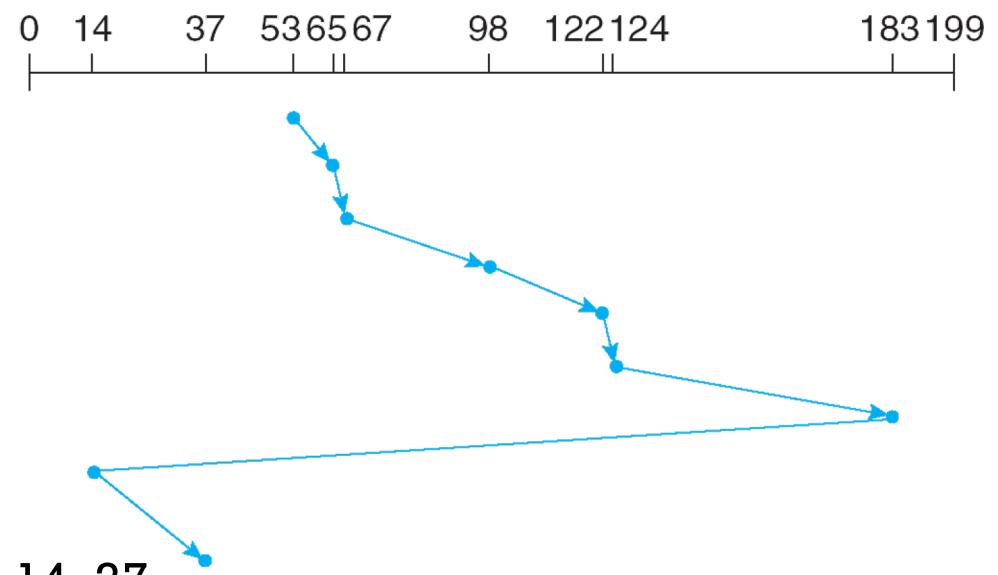
OTTIMIZZARE IL SEEK-TIME

- **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.**

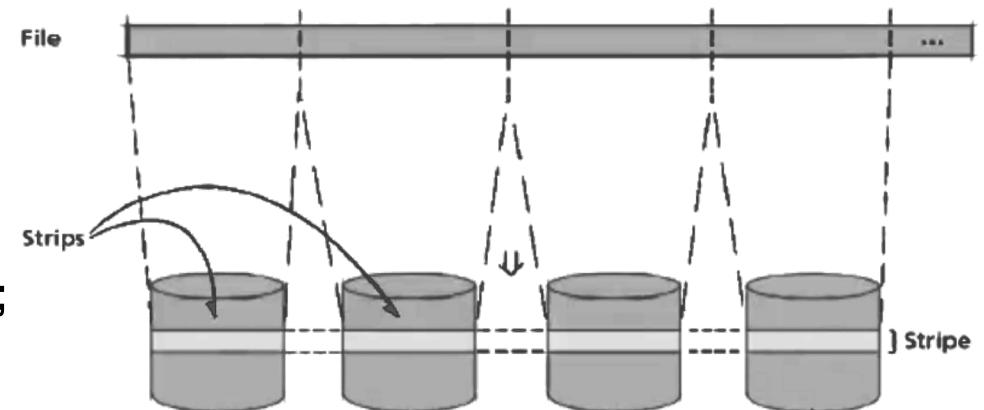
- **Come scegliere?**

- C-LOOK per alto carico;
 - LOOK o SSTF 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**;
 - sostituzione automatica: dischi **spare**;
- **Redundant Array of Inexpensive Disks (RAID)**;
 - noti anche come **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).

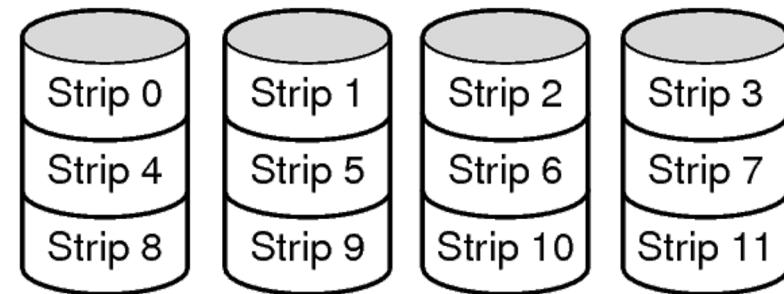


SISTEMI RAID

Overhead=Esso rappresenta il tempo medio di CPU necessario per eseguire i moduli del kernel.

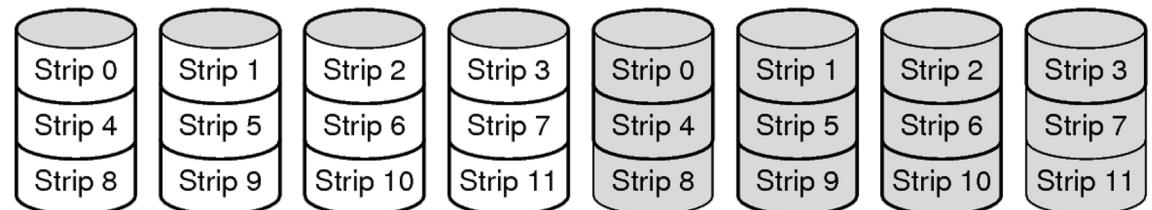
■ RAID 0 (*striping*): striscie

- fa ***striping*** in modalità round-robin;
- semplice con **prestazioni ottimali** con letture di grandi volumi;
- niente ridondanza:
maggiori vulnerabilità;



■ RAID 1 (*mirroring*): rispecchiamento

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



Nell'ingegneria dell'affidabilità la tolleranza ai guasti (o fault-tolerance, dall'inglese) è la capacità di un sistema di non subire avarie (cioè interruzioni di servizio) anche in presenza di guasti

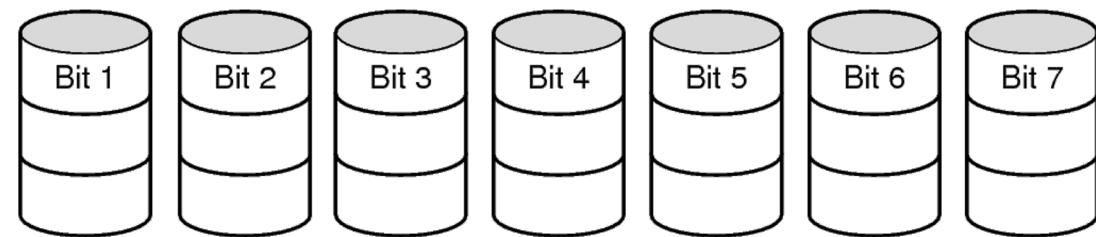


SISTEMI RAID

Le RAM con ECC (acronimo di Error Correction Code) hanno dei sistemi utili a rintracciare eventuali errori contenuti nell'informazione memorizzata e dei meccanismi capaci di correggere l'errore riscontrato.

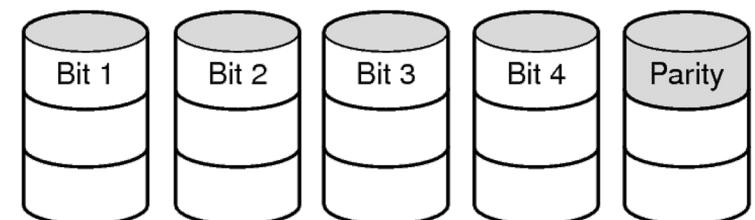
- **RAID 2** (*striping a livello di bit con ECC*):

- 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;



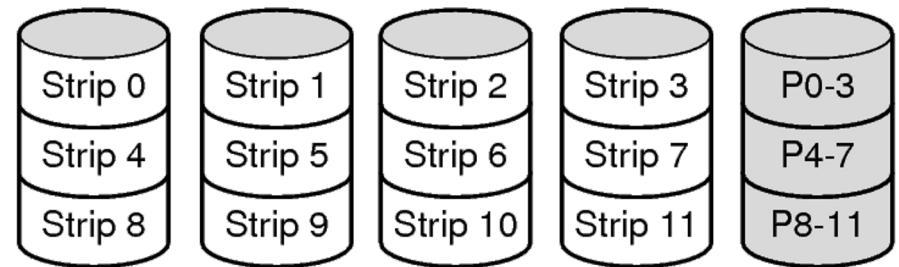
Il **bit di parità** è un codice di controllo utilizzato nei calcolatori per prevenire errori nella trasmissione o nella memorizzazione dei dati. Tale sistema prevede l'aggiunta di un bit ridondante ai dati, calcolato a seconda che il numero di bit che valgono 1 sia pari o dispari.



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 necessita 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 fatto sostituisce il RAID 4.

