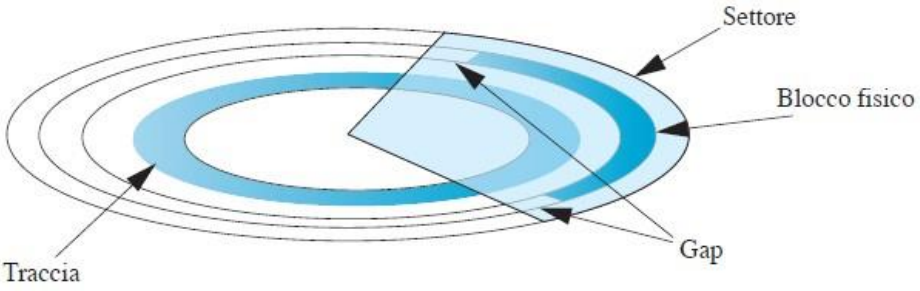
Memorie di massa e file system

Organizzazione fisica della superficie di un hard disk

L'accesso ai dati memorizzati su un hard disk è realizzato mediante le testine, trasduttori magnetici presenti su ogni faccia del disco, che si muovono in senso trasversale rispetto all'asse centrale intorno al quale il disco ruota e si posizionano sulle tracce concentriche sulle quali sono memorizzati i dati.



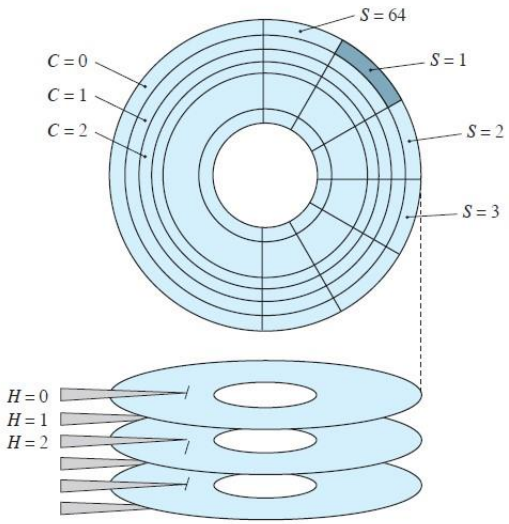
Le tracce sono suddivise in ***blocchi fisici***, la cui dimensione è generalmente di 512 byte, separati tra loro da uno spazio all'interno del quale non sono memorizzati dati, detto ***gap***. Un ***settore*** viene definito come uno spicchio circolare della superficie del disco, anche se spesso i termini settore e blocco fisico sono usati come sinonimi.
Il numero di tracce e di settori varia a seconda della capacità e delle caratteristiche costruttive del disco. Ad esempio, in un vecchio floppy disk dalla capacità di 1,44 MB, sulla superficie del disco erano create 80 tracce e 18 settori. Dato che il floppy disk è double sided, cioè i dati sono memorizzati su entrambe le superfici, la capacità totale di questo supporto può essere facilmente calcolata nel seguente modo:

2	*	80	*	18	*	512	=	1474560 (1,44 Mb)
Superfici		*Tracce*		*Settori*		*Byte/blocco*		*Capacità totale*

La creazione dei settori sulla superficie del disco viene realizzata con un'operazione preliminare detta ***formattazione fisica*** o formattazione di basso livello. Questa operazione non deve essere confusa con la ***formattazione logica***, che consiste nell'installazione dell'albero delle directory sul disco. La formattazione fisica è eseguita dal costruttore del disco, che in questo modo predispone il supporto alla memorizzazione dei dati, suddividendo la superficie di ogni faccia in tracce e settori, escludendo eventuali settori difettosi e numerando i settori utilizzabili per permettere in modo univoco la loro identificazione. La formattazione fisica è un'operazione che raramente deve essere nuovamente eseguita dopo la prima volta; infatti, la formattazione del disco eseguita dal sistema operativo, pur causando la perdita dei dati contenuti nel supporto, non va a modificare le caratteristiche a livello fisico dei settori ed è pertanto una formattazione di tipo logico.

Individuazione dei blocchi su un disco magnetico

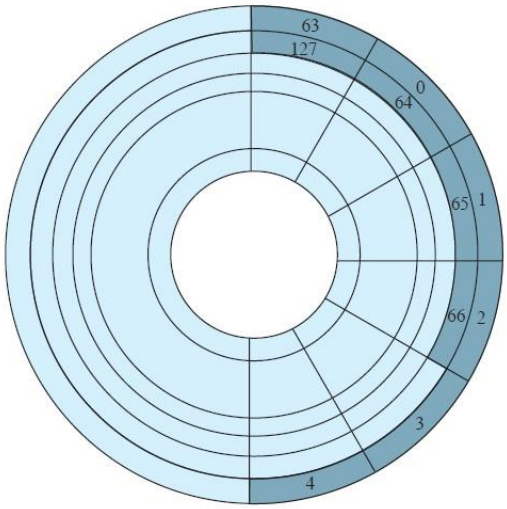
Il movimento delle testine di lettura/scrittura è comandato dai circuiti presenti a bordo dell'unità a disco che contengono un firmware che implementa la tecnica di reperimento del singolo blocco all'interno del disco.
I metodi utilizzati per individuare i blocchi sono i seguenti:



- ***CHS*** (*Cylinder*, *Head*, *Sector*): il blocco è individuato da una terna di coordinate, che costituiscono la geometria del disco: il ***cilindro*** (C), che rappresenta l'identificatore di un insieme di tracce appartenenti a tutti i piatti equidistanti dall'asse di rotazione, la ***testina*** (H), che individua la faccia contenente il blocco, e il ***settore*** (S), che è il numero sequenziale del blocco all'interno della traccia di appartenenza.
  Conoscendo la geometria di un disco e la dimensione in byte di un singolo blocco fisico, è possibile calcolarne la capacità, come abbiamo già visto per il floppy disk. Ad esempio, un disco che comprende 16 testine, 3148 cilindri, 63 settori per traccia e 512 byte per blocco fisico, avrà una capacità pari a circa 1,6 GB:

16	*	3148	*	63	*	512	=	1624670208 (1,6 Gb ca.)
Testine		*Cilindri*		*Settori*		*Byte/blocco*		*Capacità totale*

- ***LBA*** (*Linear Block Addressing*): in questa tecnica si assegna a ogni blocco di un disco un numero progressivo, a partire da 0, calcolato secondo la seguente equazione:



$$LBA = (C \times NH + H) \times SpT + S - 1$$

dove C, H, S rappresentano la terna di coordinate che individuano il blocco, NH il numero di testine totali del disco e SpT il numero di settori per traccia; la necessità di sottrarre 1 deriva dal fatto che i settori vengono contati a partire da 1 e non da 0.
Un blocco fisico contenuto in un disco che comprende 16 testine, 3148 cilindri, 63 settori per traccia e 512 byte per blocco fisico (capacità pari a circa 1,6 GB) con (C, H, S) pari a (1, 2, 5) avrà un valore LBA pari a 1133:

$$LBA = (1 \times 16 + 2) \times 63 + 5 - 1$$

Dato il valore LBA di un blocco è anche possibile calcolare la terna di coordinate (C, H, S) attraverso le seguenti relazioni:

$$C = LBA / (NH * SpT)$$
$$H = (LBA \% (NH * SpT)) / SpT$$
$$S = (LBA \% (NH * SpT)) \% SpT + 1$$

dove l'operatore / rappresenta il risultato della divisione intera, mentre l'operatore % è il resto della divisione intera.

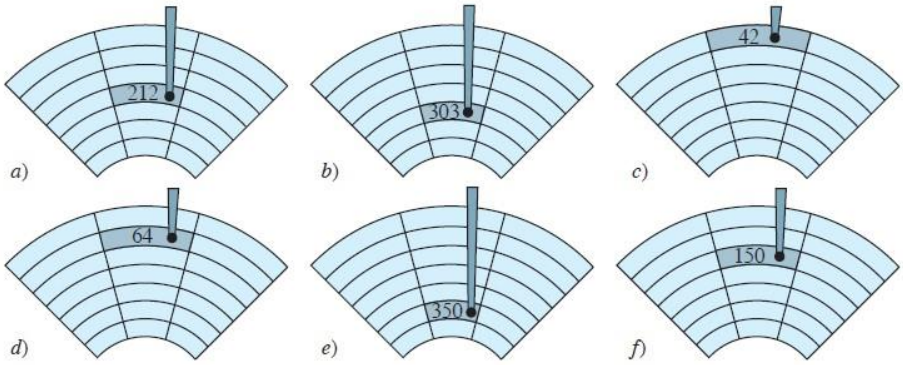
Posizionamento della testina su un blocco individuato

Una volta individuate le coordinate del blocco sul quale il sistema deve operare, la testina deve posizionarsi correttamente sull'area di disco interessata alla lettura o scrittura di dati. I movimenti dell'apparato devono essere effettuati in modo tale da minimizzare i tempi che occorrono per il posizionamento, per non penalizzare le prestazioni del sistema. I movimenti eseguiti dalla testina per spostarsi sulle aree del disco sono due:

- ***seek*** (*ricerca*): rappresenta lo spostamento in senso radiale della testina sulla traccia nella quale il dato deve essere letto o scritto;
- ***latency*** (*latenza*): è il movimento di rotazione del disco per far sì che la testina si posizioni sul blocco corretto.

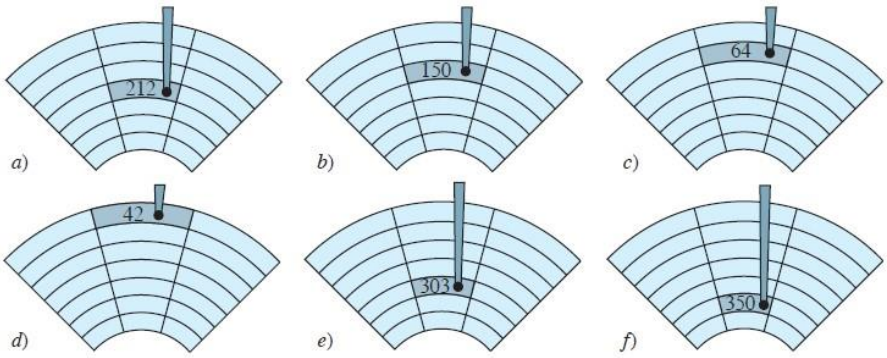
Il tempo di seek è quello che ha maggior incidenza sul tempo totale di trasferimento dei dati: una sua riduzione significa un sensibile miglioramento sia per quanto riguarda il ***throughput***, cioè la quantità di dati che vengono trasferiti nell'unità di tempo, sia in generale dal punto di vista delle prestazioni del sistema. A questo scopo gli spostamenti delle testine sono regolati da specifici algoritmi di scheduling (pianificazione), che devono organizzare gli spostamenti delle testine in modo tale che richieste di settori diversi siano esaudite con il minor spostamento possibile, e di conseguenza con tempi minori di elaborazione. Le principali tecniche di scheduling del disco sono:

FIFO
- ***FCFS*** (*First Come First Served*): è la tecnica più semplice, poiché esegue il posizionamento della testina tenendo conto soltanto dell'ordine di arrivo delle richieste e non della posizione relativa dei settori all'interno del disco; la facilità di implementazione di questo algoritmo di scheduling non è accompagnata tuttavia da buone prestazioni, dato che il tempo medio di seek è maggiore rispetto ad altre tecniche. Se ad esempio il sistema richiede la lettura di un blocco sulla prima traccia, un blocco sull'ultima e nuovamente un blocco sulla prima, applicando questa tecnica la testina deve spostarsi per uno spazio pari a due volte il raggio del disco, mentre se leggesse prima i due blocchi sulla prima traccia e poi il blocco sull'ultima il movimento di seek sarebbe esattamente dimezzato. Nella figura seguente simuliamo gli spostamenti che la testina deve effettuare per posizionarsi su una sequenza di blocchi e ne calcoliamo lo spostamento radiale misurato in numero di tracce:



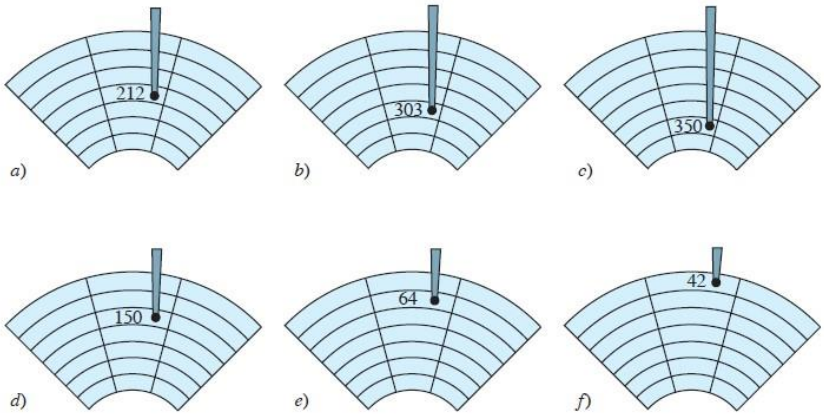
LBA blocco	Traccia	Spostamento
212	3	–
303	4	1
42	0	4
64	1	1
350	5	4
150	2	3
Totale spostamenti		15

- ***SSTF*** (*Shortest Seek Time First*): in base a questo algoritmo di scheduling, è eseguita la richiesta in attesa per la quale il blocco coinvolto nella lettura o nella scrittura ha la posizione più vicina alla posizione corrente della testina. Questo algoritmo ha migliori prestazioni dell'FCFS, ma un'operazione su un blocco può essere eseguita con notevole ritardo se ci sono richieste di operazioni su blocchi più vicini alla posizione della testina: tali richieste, anche se arrivate successivamente, sono eseguite prima. Nella figura seguente simuliamo gli spostamenti che la testina effettua per posizionarsi sulla stessa sequenza di blocchi del caso precedente e ne calcoliamo lo spostamento radiale misurato in numero di tracce; è evidente come sia variato l'ordine di visita dei settori, pur partendo dallo stesso ordine di arrivo delle richieste.



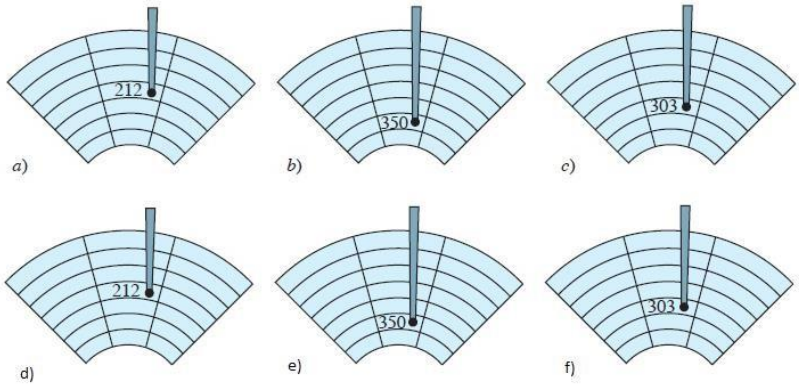
LBA blocco	Traccia	Spostamento
212	3	–
150	2	1
64	1	1
42	0	1
303	4	4
350	5	1
Totale spostamenti		8

- ***SCAN***: quando è implementato questo algoritmo, la testina percorre il disco dalle tracce esterne verso quelle interne e viceversa, esaudendo le richieste che man mano incontra lungo questo percorso. Questa tecnica garantisce un buon tempo medio di seek, ma penalizza le tracce esterne rispetto a quelle collocate nella parte centrale del disco, perché quando si inverte il verso dello spostamento la testina è posizionata sulle tracce appena percorse. Nella figura seguente simuliamo nuovamente gli spostamenti che la testina effettua per posizionarsi sulla stessa sequenza di blocchi dei casi precedenti e ne calcoliamo lo spostamento radiale misurato in numero di tracce:



LBA blocco	Traccia	Spostamento
212	3	–
303	4	1
350	5	1
150	2	3
64	1	1
42	0	1
Totale spostamenti		6

- ***CSCAN*** (*Circular Scan*): è un algoritmo molto simile allo SCAN, con la differenza che le richieste vengono esaudite solo nello spostamento della testina che va dai cilindri interni a quelli esterni: quando raggiunge l'ultima traccia, la testina ritorna a posizionarsi su quella più interna senza esaudire altre richieste. Nella figura seguente simuliamo nuovamente gli spostamenti che la testina effettua per posizionarsi sulla stessa sequenza di blocchi dei casi precedenti e ne calcoliamo lo spostamento radiale misurato in numero di tracce:



LBA blocco	Traccia	Spostamento
212	3	–
350	5	2
303	4	1
150	2	2
64	1	1
42	0	1
Totale spostamenti		7

Criticità nell'utilizzo delle memorie di massa

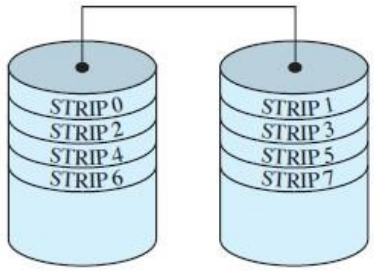
L'utilizzo delle memoria di massa, e in modo particolare degli hard disk, è critico rispetto a tre fattori:

- la ***capacità***: misurata in *GB*, determina la possibilità di memorizzazione di dati permanenti da parte del sistema; quando lo spazio libero risulta insufficiente rispetto alle richieste dei programmi applicativi o alle esigenze dell'utente, le soluzioni che possono essere adottate sono la compressione dei dati oppure l'aggiunta di nuove unità disco;
- la ***velocità di trasferimento***: misurata in Mbit al secondo, nel caso in cui sia troppo bassa può contribuire in modo determinante al degrado delle prestazioni del sistema;
- la ***sicurezza dei dati***: il rischio di errori o addirittura di perdite dei dati per cause accidentali o intenzionali deve essere ovviamente ridotto al minimo.

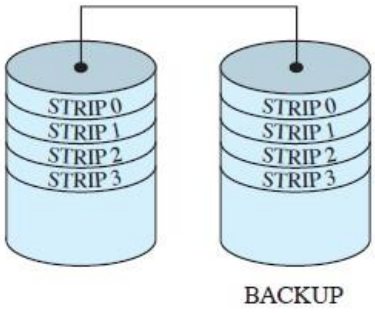
Tra le soluzioni adottate per migliorare le prestazioni degli hard disk, soprattutto in relazione a questi tre parametri, è particolarmente interessante la tecnologia ***RAID*** (*Redundant Array of Independent Disks*), utilizzata principalmente nei

dischi in dotazione agli elaboratori di tipo server, che garantisce un'elevata velocità di trasferimento e, grazie alla ridondanza dei dati, anche una buona affidabilità della periferica.
La tecnica RAID consente a due o più dischi, che normalmente sono scelti in modo che abbiano la medesima capacità, di lavorare in sincronia, permettendo trasferimenti in parallelo che garantiscono prestazioni più elevate grazie alla distribuzione dei dati tra le unità collegate. La sincronizzazione delle operazioni sui dati fa sì che le varie unità appaiono al sistema operativo come un disco unico: il vantaggio è che la distribuzione dei dati tra i vari dischi risulta così del tutto trasparente alle applicazioni che li utilizzano.
Esistono diverse tipologie di dischi RAID; di seguito ne esaminiamo brevemente alcune:

- ***RAID 0***: mediante una tecnica detta ***striping***, il disco virtuale risultante dai dischi fisici collegati tra loro è visto suddiviso in "strisce", ognuna composta di un certo numero di settori; come evidenziato dalla figura, le strisce consecutive sono memorizzate su unità diverse. Questa tecnica ha eccellenti prestazioni quando da parte del sistema ci sono grandi richieste di dati memorizzati su strisce appartenenti a dischi fisici diversi. In questo caso si realizza il parallelismo nel trasferimento con conseguente velocizzazione delle operazioni di lettura e scrittura; oltre a produrre un incremento delle prestazioni rispetto alla memorizzazione su un unico disco, è una tecnologia relativamente semplice da implementare. Nonostante sia classificata come RAID, questa tecnica non presenta la ridondanza dei dati;

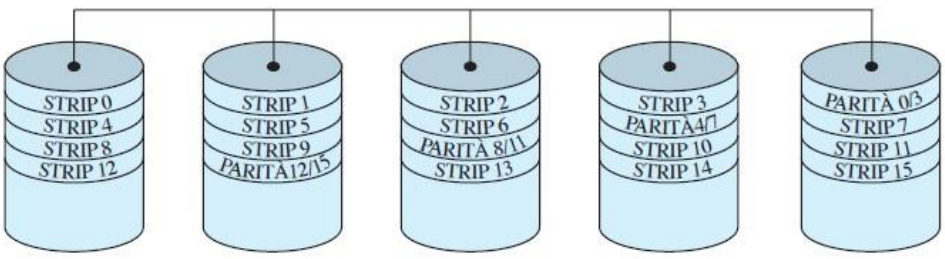


- ***RAID 1***: tutti i dischi presentano un'unità duplicata di backup. In fase di scrittura ogni striscia è memorizzata due volte, mentre in lettura è trasferita una delle due copie, mantenendo la distribuzione del carico su più unità fisiche. I vantaggi di questa tecnica risiedono nella velocità di lettura, che può essere fino a due volte superiore, e nella possibilità di ovviare a un malfunzionamento di una delle copie utilizzando l'altra copia;



Una semplice tecnica per proteggere i dati da errori accidentali è il ***controllo di parità***, che consiste nell'aggiungere ai dati un bit, detto ***bit di parità***, che viene calcolato in base alla modalità ***parità pari*** o ***parità dispari***.
Nel caso della parità pari, occorre avere nei dati, compreso il bit di parità, un numero di bit a 1. Se, per esempio, il dato è 10010010, il bit di parità che viene calcolato è 1, perché essendo il numero di 1 dispari per rendere pari il numero di 1 totale devo impostare il bit di parità a 1.
Esempi: 11010100 ➏ bit di parità = 0 (numero di 1 nel dato = 4)
10110011 ➏ bit di parità = 1 (numero di 1 nel dato = 5)
Nel caso della parità dispari, il numero totale di 1 deve essere dispari.
Per controllare la correttezza dei dati occorre contare il numero di bit a 1 compreso il bit di parità; nel caso della parità pari, se il numero totale di bit a 1 è dispari abbiamo rilevato un errore.
Non siamo purtroppo certi della correttezza di un dato perché un numero pari di errori si compensa.

Dato	Parità	Dato modificato	Parità	
11011001	1	10011101	1	il numero dei bit a 1 è sempre pari



- ***RAID 5:*** il RAID 5 lavora su strisce: vengono calcolati i bit del controllo di parità su ciascuna striscia e memorizzati uniformemente su tutte le unità.

Unità di memoria a stato solido (SSD)

Una unità di memoria a stato solido (***SSD***, *Solid State Drive*, talvolta impropriamente *disco a stato solido*), è un dispositivo di memoria di massa basato su semiconduttore, che utilizza memoria allo stato solido (in particolare *memoria flash*) per l'archiviazione dei dati. Il termine *disco a stato solido* è improprio perché all'interno dell'SSD non c'è alcun disco, né di tipo magnetico, né di altro tipo (anzi, non vi è contenuto alcun componente in movimento).
La maggior parte delle unità a stato solido utilizza la tecnologia delle memorie flash NAND, che permette di memorizzare in maniera non volatile grandi quantità di dati, senza l'utilizzo di organi meccanici (piatti, testine, motori ecc.) come fanno invece gli hard disk tradizionali, il che comporta notevoli vantaggi alla riduzione dei consumi elettrici e dell'usura.
Altra caratteristica vantaggiosa delle memorie flash risiede nelle ridotte dimensioni fisiche che consentono la realizzazione di unità SSD estremamente compatte e leggere, quindi facilmente integrabili anche all'interno dei dispositivi mobili ultrasottili.
Oltre alla memoria in sé, un'unità SSD dispone di diversi componenti necessari alla gestione del suo funzionamento:

- ***controller***,è costituito da un microprocessore che si occupa di coordinare tutte le operazioni di lettura/scrittura della memoria di massa, oltre all'error correcting code, la gestione della memoria cache, il garbage collection e la cifratura dei dati. Il software che governa questo componente è un firmware preinstallato dal produttore.: controllo e correzione degli errori in fase di lettura/scrittura
- ***Supercondensatore***: una novità introdotta dalle memorie a stato solido è la possibilità di terminare le operazioni di scrittura anche in caso di mancanza di tensione. Questo avviene grazie alla presenza di un supercondensatore o, più raramente, di una batteria di backup, che garantisce energia sufficiente per concludere l'operazione in corso. Questa tecnica permette di garantire una maggiore integrità dei dati ed evitare che il file system risulti corrotto.
- ***Memoria Cache***: è una memoria con una capacità solitamente nell'ordine di qualche MB o GB a seconda del tipo di sistema e generalmente proporzionale alla capienza dell'SSD, utilizzata dal processore per immagazzinare temporaneamente informazioni che verranno richieste in seguito dal sistema. Essa viene quindi riempita e svuotata molte volte.
- ***Interfaccia:*** la connessione può avvenire con cavi di tipo SATA, sia per quanto riguarda la connessione dati che per l'alimentazione. In definitiva è possibile collegare un SSD utilizzando una normale interfaccia SATA2 (3Gb/s) o SATA3 (6Gb/s). Vi sono inoltre SSD che utilizzano l'interfaccia PCI Express; quest'ultima può arrivare fino a una velocità di trasferimento di 20Gb/s.

I principali vantaggi rispetto alle unità tradizionali, determinati dalla totale assenza di parti meccaniche in movimento, sono i seguenti:

- Rumorosità assente, non essendo presente alcun componente (motore e disco magnetico) di rotazione, al contrario degli HDD tradizionali;
- le unità a stato solido hanno mediamente un tasso di rottura inferiore a quelli dei dischi rigidi che oscilla tra lo 0,5% e il 3%, mentre nei dischi rigidi può raggiungere il 10% (*l'MTBF - Mean Time Between Failure* - di un SSD di ultima generazione può raggiungere 2 000 000 di ore);
- minori consumi elettrici durante le operazioni di lettura e scrittura;
- tempi di accesso e archiviazione ridotti: si lavora nell'ordine dei decimi di millisecondo; il tempo di accesso dei dischi magnetici è oltre 50 volte maggiore, attestandosi invece tra i 5 e i 10 millisecondi;
- non necessitano di deframmentazione;
- maggior velocità di trasferimento dati;
- maggiore resistenza agli urti;
- minore produzione di calore;
- gli SSD SATA hanno la stessa identica forma, dimensione ed interfaccia di collegamento dei dischi rigidi SATA da 2,5" e sono pertanto interscambiabili con essi senza installare componenti hardware o software specifici (alcuni settaggi dell'UEFI potrebbero rendersi necessari per sfruttarne appieno la velocità di trasferimento dati).

A fronte di una maggiore resistenza agli urti e a un minor consumo, le unità a stato solido hanno due svantaggi principali:

- maggiore prezzo, ovvero una minore capacità di immagazzinamento a parità di costo rispetto ai dischi rigidi Serial ATA;
- peggiore permanenza dei dati quando non alimentati e in modo differente a seconda della temperatura d'esposizione.

Questi problemi sembrano però destinati a risolversi in futuro. Le nuove tecnologie stanno portando memorie flash in grado di garantire durata pari o superiore a quella di un disco rigido tradizionale e attualmente i produttori dichiarano 140 anni di vita con 50 GB di riscritture al giorno su un'unità da 250 GB. D'alrta parte, il costo della tecnologia SSD sta lentamente scendendo facendo facilmente presagire una futura sostituzione dei dischi tradizionali con unità a stato solido.

La frammentazione di un disco SSD non influisce sulle sue prestazioni, poiché il tempo d'accesso a qualunque cella è identico; i moderni sistemi operativi pertanto disattivano la deframmentazione del disco (in tutti i sistemi operativi Microsoft Windows, occorre però disattivare la schedulazione, altrimenti la deframmentazione sarà comunque pianificata), in quanto risulta non solo inutile, ma addirittura dannosa poiché influisce negativamente sulla vita del disco stesso. Infatti proprio per aumentare la durata del supporto si cerca di ridurre il sovraccarico sempre su una cella riscrivendola di continuo, grazie all'ausilio di un controllore che distribuisce i dati in fase di scrittura cercando di sfruttare tutto il disco ed evitando di lasciare parti inutilizzate. Al contrario la deframmentazione non farebbe altro che aumentare il numero di cicli scrittura di dati accorciando la vita del disco stesso.
Le caratteristiche delle unità SSD sono alla base di impieghi spesso specifici. Ad esempio, esiste la fascia dei desktop e notebook dotati di "doppio disco" (impropriamente in quanto l'SSD non è un disco) con la seguente configurazione:

- unità SSD di media capacità utilizzata per l'installazione del sistema operativo e delle applicazioni software in quanto la velocità di accesso in questo caso è più importante della capacità e affidabilità;
- tradizionale unità disco HDD di grande capacità impiegata come archivio dei contenuti, dato che l'affidabilità e la capienza sono più importanti della velocità e dell' efficienza.

Chiaramente, esistono le altre due configurazioni, quella più economica e quella più prestazionale:

- una sola unità, SSD, solitamente con due partizioni, una per sistema operativo e software installato, l'altra per i dati;
- due unità SSD.

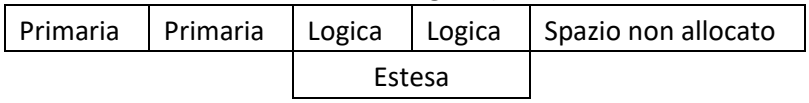
Organizzazione dei dati su disco

Il partizionamento del disco

Partizionare il disco vuol dire dividerlo in più dischi "virtuali": questo rende più flessibile l'uso del pc in quanto permette di usare le diverse partizioni come se fossero dischi distinti potendo così installare, per esempio, diversi sistemi operativi (uno in ogni diversa partizione), oppure conservare i dati in una partizione diversa da quella del sistema operativo.
Se si crea un'unica partizione sulla quale si installa un sistema operativo (nel caso fosse Windows, la vedrebbe come C:), qualora fosse necessario reinstallare il sistema operativo, tutto il precedente contenuto del disco verrebbe perso.
Se invece si creano due partizioni (Windows le leggerebbe come C: e D:) all'interno delle quali vengono installati rispettivamente il sistema operativo (C: di Windows) e i dati, nel caso in cui fosse necessario reinstallare il sistema operativo, verrebbero persi i contenuti della prima partizione (disco C:) ma non quelli della seconda (disco D:), che invece resterebbe intatta.
In generale, i motivi che giustificano la suddivisione del disco in partizioni possono essere i seguenti:

- separare dal punto di vista logico i file che costituiscono il sistema operativo e le sue impostazioni dai file dei programmi applicativi e dai dati degli utenti, memorizzandoli in partizioni diverse allo scopo di razionalizzare l'organizzazione dei dati contenuti nella memoria di massa;
- avere la possibilità di installare più sistemi operativi sullo stesso computer;
- avere sullo stesso disco più file system, anche di tipo diverso, che possono essere utilizzati dallo stesso sistema operativo;
- prevedere un'area di backup all'interno della quale salvare i dati di sistema;
- per i computer portatili, definire un'area all'interno della quale viene memorizzato lo stato del sistema quando, per risparmiare energia, la macchina passa nella modalità di ibernazione o standby.

Ogni supporto può contenere fino a quattro partizioni definite come ***partizioni primarie***, che possono a loro volta essere definite come ***bootable***, cioè avviabili. Impostare come avviabili le partizioni primarie permette di caricare sullo stesso computer più sistemi operativi e, tramite un apposito programma eseguito in fase di avvio della macchina, il ***boot loader***, far eseguire il bootstrap del sistema operativo desiderato.
La ***partizione estesa*** ha invece la funzione di contenere altre ***partizioni logiche*** che, a differenza delle partizioni primarie, non possono essere rese avviabili. Se in un disco creo quattro partizioni primarie, questo significa che non posso dividerlo ulteriormente, se invece vado a creare tre partizioni primarie ed una estesa, posso inserire, all'interno della partizione estesa molte altre partizioni logiche. In pratica, la partizione estesa è, semplicemente, un contenitore vuoto che può ospitare solamente partizioni logiche e quindi non può ospitate file (sistema operativo, musica, foto, documenti, ecc..), mentre le partizioni primarie e le partizioni logiche non possono ospitare altre partizioni ma solamente file (sistema operativo nelle primarie, musica, foto, documenti, ecc.. nelle logiche).



Esempio di partizionamento del disco

Per poter essere utilizzate correttamente, è indispensabile che le partizioni siano identificate univocamente dagli utenti dei sistemi operativi. Nei sistemi Windows a ogni partizione è assegnata una lettera progressiva a partire dalla C (*C:*, *D:*, …); le prime due lettere dell'alfabeto non vengono assegnate perché storicamente le lettere A e B erano dedicate ai due lettori di floppy disk. Nei sistemi Linux l'identificatore assegnato alla partizione è legato alla tipologia di disco sul quale la partizione è creata: ad esempio i dischi vengono indicati con */dev/hda*, */dev/hdb*, ecc., i dischi SCSI con */dev/sda*, */dev/sdb* e così via. Per fare riferimento alle partizioni, viene aggiunto un numero progressivo all'identificatore del disco: le partizioni del primo disco saranno pertanto */dev/hda1*, */dev/hda2*, ecc.

Il primo blocco del disco, localizzato nel cilindro 0, testina 0 e settore 1 ha una funzione fondamentale, perché fornisce al microprocessore le istruzioni necessarie per eseguire il caricamento del sistema operativo e permette al sistema operativo stesso di utilizzare correttamente il supporto di memoria. Questo blocco viene definito ***MBR*** (*Master Boot Record*) ed è letto dal BIOS quando il computer viene acceso.
Al suo interno sono memorizzate alcune linee di codice in linguaggio macchina che, eseguite dalla CPU, leggono la tabella delle partizioni alla ricerca della ***partizione attiva***. La partizione attiva è quella porzione di disco identificata come un'unità logica a sé stante, che nel suo primo blocco, o ***boot sector***, contiene una porzione di codice, il ***Master Boot Code***, che carica in memoria centrale la parte centrale (***kernel***) del sistema operativo.
Dal momento che il Master Boot Code è il primo programma a essere mandato in esecuzione, è uno dei bersagli preferiti dai virus che, corrompendolo, rendono inutilizzabile il computer.

<table>
<tr><td rowspan="2">Master Boot Record</td><td rowspan="2">Primaria</td><td rowspan="2">Primaria</td><td rowspan="2">Primaria</td><td>Logica</td><td>Logica</td></tr>
<tr><td colspan="2">Estesa</td></tr>
</table>

Al termine del Master Boot Record, cioè del primo blocco del disco che non è contenuto in nessuna partizione, è memorizzata la partition table o tabella delle partizioni, che contiene l'elenco della partizioni presenti sul disco e alcune informazioni a esse correlate. Ogni voce memorizzata della partition table che si riferisce a una specifica partizione ha una dimensione di 16 byte e contiene le seguenti informazioni: la ***terna di coordinate (C, H, S) del primo e dell'ultimo blocco*** della partizione, il ***valore LBA del primo blocco***, la ***dimensione totale della partizione***, se la partizione sia ***avviabile o meno***.
Ogni partizione ha al suo interno una particolare struttura che dipende dal file system presente nella partizione stessa.
Spesso per indicare una partizione è utilizzato il termine ***volume***; in generale, la definizione di volume si riferisce a un'unità logica di memorizzazione, cioè un insieme di dati indipendenti che possono essere memorizzati in una partizione, in un intero disco, oppure su unità fisiche distinte mediante tecniche di spanning. L'utente del sistema vede le informazioni appartenenti a un volume come facenti parte di una stessa unità, indipendentemente dalla loro collocazione fisica.

Organizzazione fisica dei file

All'interno di un sistema è fondamentale organizzare i file in una struttura, denominata ***file system***, che permetta il loro reperimento in modo semplice e immediato: l'elemento portante di questa struttura è la ***directory***, detta anche *folder* o *cartella*, a seconda del sistema operativo.
Una directory è un file memorizzato in memoria di massa che non contiene dati di alcun tipo, ma che svolge una funzione di servizio. Da un punto di vista logico, una directory è un contenitore di file, nel senso che attraverso la lettura della directory è possibile recuperare tutte le informazioni collegate ai file appartenenti a quella directory, come i dati e gli attributi.
Gli attributi di un file sono i seguenti:

- ***nome del file***: è l'identificatore del file; ha regole di lunghezza e composizione che variano a seconda del sistema operativo;
- ***tipo del file***: individua i dati contenuti nel file ed è spesso collegato all'estensione del nome del file;
- ***indirizzo del primo blocco fisico del file***: permette il reperimento del file nella memoria di massa;
- ***lunghezza corrente del file***: deve essere aggiornata ogni volta che avvengono modifiche sul contenuto del file;
- ***data dell'ultimo accesso al file, data dell'ultima modifica al file***: sono informazioni di servizio utili per identificare gli ultimi accessi compiuti e per effettuare correttamente le operazioni di backup;
- ***ID del proprietario del file***: è un'informazione fondamentale per la verifica dei diritti di accesso ai dati;
- ***protezione del file***: sono i criteri che regolano l'accesso al file.

La struttura delle directory è fondamentale all'interno del sistema: l'organizzazione delle directory può influenzare pesantemente l'efficienza del sistema ed eventuali errori di memorizzazione delle directory compromettono la struttura del file system; nel caso di una directory corrotta, non è più possibile accedere alle informazioni contenute nei file che le appartengono.

In generale, la directory è una tabella che contiene un certo numero di righe, ognuna delle quali fa riferimento a un file. Ogni riga contiene il nome del file e gli attributi che a esso si riferiscono, oppure un puntatore a un blocco in memoria di massa che contiene gli attributi, come avviene per il sistema operativo Linux.
La struttura viene delle directory definita ***gerarchica*** o ***ad albero***: i nodi dell'albero possono essere file o directory (cartelle o folder nella terminologia Windows). All'apice dell'albero è situata la ***root directory***, o *directory principale*; questa directory ha un certo numero di nodi figli, che possono essere file o directory; nel caso in cui siano directory, queste possono a loro volta essere padri di altri nodi. Ogni utente del sistema può essere proprietario di una porzione di albero, nella quale può liberamente definire una sua struttura di memorizzazione, creando sottocartelle annidate all'interno delle quali ordinare i suoi file; la directory all'apice del sottoalbero creato dall'utente è detta ***home directory***. Ogni file e ogni directory vengono identificati da un nome, che è ***/*** (*slash* o *barra*) per la directory root, e che deve essere unico all'interno di una directory; un file o una directory sono individuati mediante il ***pathname***, cioè l'elenco delle directory da attraversare per essere raggiunti.
Il pathname di un file, pur essendo un'informazione fondamentale, non viene memorizzato da nessuna parte, perché il reperimento dei dati avviene "navigando" la struttura gerarchica del file system. Il path o cammino utilizzato per identificare un oggetto (file o directory) può essere ***assoluto***, cioè comprendere l'elenco delle directory che, partendo dalla root, vengono attraversate per raggiungere l'oggetto, o ***relativo***, cioè contenere le directory che devono essere attraversate per raggiungere l'oggetto a partire da una determinata directory.
La ***current working directory*** è la directory all'interno della quale l'utente è "posizionato" ed è indicata dal carattere "**.**", mentre la directory genitrice della directory di lavoro viene indicata con "**..**".
Impartendo un comando che elenca i file senza specificare alcun path, si ottiene la visualizzazione della lista dei file contenuti nella current working directory.
Il comando ***cd*** (*change directory*), previsto sia dal sistema operativo DOS sia da Linux, permette di cambiare la current working directory, sia attraverso l'indicazione di un path assoluto, sia attraverso un path relativo. Il comando:

$ cd /scuola

cambia la current working directory in modo che diventi */scuola*. Il path è assoluto perché inizia con una barra che rappresenta la directory radice. In DOS il comando:

$ cd

senza argomenti, permette di verificare la current working directory attuale. L'analogo in Linux è il comando ***pwd***, sempre senza argomenti.
Indicando un percorso che non inizia con una barra obliqua, allora si specifica un percorso che ha origine dalla current working directory. Esaminiamo ad esempio la seguente sequenza di comandi (in Linux):

```
$ pwd
        /scuola
$ cd classi
$ pwd
        /scuola/classi $
cd ..
$ pwd
        /scuola
```

Il comando *cd classi* ci permette di passare dalla directory corrente *scuola* alla sua sottodirectory *classi*, il comando *cd ..* ci fa tornare alla directory di livello superiore, cioè da *classi* a *scuola*.

Il file system manager

Il ***file system manager*** è un modulo del sistema operativo cui spetta l'organizzazione e la gestione dei file presenti nella memoria di massa. Il file system manager offre le seguenti funzioni principali:

- fornire agli utenti, ai programmatori e ai gestori del sistema ***strumenti per memorizzare e gestire i dati***. Molte applicazioni interagiscono con la memoria di massa per prelevare i dati sui quali lavorano e per memorizzare i risultati delle elaborazioni: la struttura ad albero delle directory permette un facile reperimento dei file, senza dover conoscere come fisicamente sono memorizzati sul supporto;
- ***ottimizzare il throughput***, cioè trasferire la maggior quantità di dati possibile tra la memoria di massa e la memoria centrale e viceversa;
- ***minimizzare i tempi di risposta delle periferiche***, in particolare se le informazioni sono richieste da applicazioni che lavorano in tempo reale;
- ***rendere trasparenti alle applicazioni le caratteristiche fisiche*** dei dispositivi di memoria di massa;

- ***adottare politiche di sicurezza*** al fine di evitare la perdita accidentale dei dati o l'accesso non autorizzato da parte di utenti o applicazioni privi dei diritti necessari per eseguire le operazioni richieste;
- ***fornire un insieme standard di routine di I/O***, cioè procedure che consentano di trasferire blocchi di byte tra i dispositivi e la memoria centrale;
- ***rendere nuovamente disponibili le aree utilizzate*** dalle informazioni che sono state cancellate;
- ***gestire in modo automatico le copie di riserva o backup***. Nel momento in cui un file viene aperto per essere modificato, il file system manager può copiarlo mantenendo in questo modo una copia corretta. Questa operazione consente all'utente di ritornare all'ultima versione salvata del file nel caso in cui le modifiche apportate non siano ritenute corrette, oppure nel caso in cui il file si corrompa e non sia più leggibile. In alcuni file system, anziché una sola copia di riserva viene mantenuto lo storico delle versioni precedenti, che sono numerate progressivamente in modo da poterle distinguere in base al momento di creazione; sarà poi cura dell'utente la cancellazione dei file non necessari;
- • nel caso di sistemi multiprogrammati, ***regolare l'accesso contemporaneo di più applicazioni*** allo stesso insieme di dati.

La presenza di un modulo del sistema operativo che gestisce le operazioni sui file permette quindi al programmatore di non tenere conto delle modalità di accesso ai dispositivi fisici sui quali sono effettivamente memorizzati i dati. Il programmatore utilizza strutture astratte, come i descrittori di file, e operazioni che sono totalmente indipendenti dalla particolare tipologia di memoria di massa con la quale il programma interagisce.
Inoltre, le applicazioni si riferiscono ai file tramite un *nome simbolico*, che viene tradotto dal file system manager nei corretti riferimenti al disco fisico, e possono accedere ai dati con operazioni come read, write, append. In modo del tutto trasparente all'applicazione, il file system manager si occupa di reperire il file e di eseguire le operazioni su di esso, anche se è allocato su dispositivi fisici diversi.

Operazioni del file system manager

Le principali operazioni che il file system manager può effettuare sia sulla struttura gerarchica dei file sia sui singoli file e directory sono le seguenti:

- ***Inizializzazione del file system***

  La prima operazione che deve essere effettuata da un file system manager su un volume vuoto è la creazione della struttura del file system, che consiste sostanzialmente nella preparazione di una struttura, pronta a contenere file e directory.
  In questa fase, il sistema operativo calcola il numero di blocchi liberi del volume e, in base a questo valore, memorizza in un'area nota della memoria di massa alcune informazioni quali il nome del volume, la sua dimensione, se è integro o danneggiato, il riferimento ai blocchi liberi e altri dati fondamentali per la gestione dei file che verranno memorizzati.
  Viene inoltre creata la root directory, che in questa fase iniziale è ovviamente vuota; nel caso in cui la root non sia memorizzata in una posizione prefissata del volume, deve essere registrato il riferimento alla posizione fisica in cui è collocata. Affinché il file system possa essere effettivamente utilizzato dagli utenti e dagli applicativi occorre eseguire l'operazione di mount.

- ***Mount e unmount del file system***

  L'operazione di mount ha lo scopo di rendere il file system disponibile all'accesso da parte di utenti e applicazioni. Prima di tutto, è importante che il file system che viene "montato" sia integro; nel caso in cui si rilevi uno stato del file system non coerente, come può accadere in seguito a uno shutdown del sistema non eseguito correttamente, bisogna eseguire un controllo sull'integrità del file system ed eventualmente riparare il problema. Un modulo separato dal file system manager, che si chiama system check program, ha il compito di svolgere il test di integrità, operazione che presenta una certa complessità di esecuzione. Se il file system è integro, al fine di velocizzare le operazioni sui file, sono caricate in memoria centrale le strutture memorizzate su disco che permettono l'accesso ai dati del volume. Dal momento in cui queste informazioni sono memorizzate in memoria centrale, il volume montato è accessibile agli utenti e alle applicazioni.
  Nei sistemi operativi attuali il mount di un volume avviene in modo automatico; nei sistemi Linux esiste anche il comando *mount* per eseguire manualmente questa operazione.
  Perfettamente speculare all'operazione di mount, l'operazione di *unmount* può essere eseguita in modo automatico allo shutdown del sistema operativo oppure manualmente con il comando umount.
  L'operazione di unmount non rende più disponibili i file del volume smontato; in seguito a una richiesta di unmount, il file system manager memorizza nella memoria di massa le informazioni relative al file system e lo marca come *clean*, in modo che un mount successivo non richiederà più un test completo dell'integrità dei dati.

    - ***Creazione di file e di directory***

Un volume appena formattato e del quale viene eseguito il mount si presenta come una struttura vuota, all'interno della quale devono essere creati file e directory, che a loro volta possono contenere altri file e directory.
Quando arriva la richiesta di creazione di un file, il file system manager crea un descrittore, cioè una struttura dati agganciata alla directory che deve contenere il file. Questa struttura memorizza alcune informazioni che si riferiscono al file, tra le quali:

- la data di creazione;
- l'indicazione che il file è vuoto, quindi che la sua dimensione è uguale a 0;
- l'identità dell'owner, cioè di chi ha creato il file;
- i diritti di accesso al file da parte degli utenti del sistema operativo.

Occorre evidenziare il fatto che, quando un file viene creato, sono memorizzate soltanto le informazioni che permettono la sua gestione da parte del file system manager: lo spazio fisico su disco che contiene i dati del file è allocato solo quando è richiesta la scrittura di un blocco di dati. Un file può essere creato:

- da un programma che deve memorizzare su memoria di massa il risultato delle sue elaborazioni;
- da un utente che utilizza un applicativo, ad esempio un pacchetto di automazione d'ufficio che consente di creare file contenenti testi, fogli elettronici o presentazioni;
- dal sistema operativo o da altre applicazioni che creano un *log*, cioè un diario, delle operazioni che hanno eseguito.

La creazione di una directory è fondamentale per la struttura del file system. Da un punto di vista pratico, questa operazione è molto simile alla creazione di un file, con la differenza che il descrittore della directory deve essere inizializzato. La struttura deve contenere un riferimento alla directory che gerarchicamente la precede immediatamente e che è identificata come *parent directory*.
Questo riferimento permette una navigazione molto più semplice attraverso l'albero delle directory, che può avvenire sia verso le sottodirectory sia verso le directory di livello superiore; come abbiamo visto, il riferimento alla directory superiore è convenzionalmente indicato con "**..**" .

- ***Apertura, scrittura, lettura, cancellazione e spostamento di un file.***

L'apertura di un file è un'operazione che il file system manager esegue in due fasi distinte:

- ***ricerca del file***: seguendo i riferimenti memorizzati all'interno dei descrittori delle directory che mantengono la struttura gerarchica del file system, il file system manager naviga lungo l'albero delle directory e, se il nome del file esiste, viene letto in memoria centrale il descrittore associato al file;
- ***verifica dei diritti di accesso al file***: questa seconda fase avviene nei sistemi operativi che prevedono un controllo sulle operazioni richieste da utenti e applicazioni nei confronti del file. Se l'accesso è consentito, sono memorizzate nel descrittore quali operazioni sono permesse all'applicazione che ha effettuato l'apertura del file, che possono essere, ad esempio, lettura, scrittura, aggiunta alla fine del file. Se il tentativo di apertura del file ha esito positivo, il file system manager restituisce ai livelli superiori del sistema operativo un ***handle***, termine che letteralmente significa *maniglia*, che costituisce il riferimento che consentirà all'applicazione che ha richiesto l'apertura del file di effettuare su di esso le operazioni di input/output.

L'operazione di scrittura su un file deve prevedere, da parte del programma che la richiede, che siano specificati i seguenti argomenti:

- *l'handle* del file;
- la *lunghezza dei dati* che devono essere scritti; • il riferimento al buffer che contiene i dati;
- la *posizione all'interno del file* a partire dalla quale deve iniziare la scrittura.

La ***scrittura*** è eseguita leggendo il contenuto dell'area di memoria che contiene i dati da memorizzare nel file e scrivendo i dati nel file a partire dalla posizione specificata. Nel caso in cui la posizione coincida con la fine del file, il file system manager dovrà allocare nuovi blocchi in memoria di massa scegliendoli dalla lista dei blocchi liberi e agganciandoli fisicamente al file. Una volta riservato lo spazio sufficiente a contenere i dati, il file system manager deve calcolare l'indirizzo fisico dei blocchi allocati a partire dagli indirizzi logici e quindi scrivere fisicamente i dati sul file. Al termine di un'operazione di scrittura, il riferimento alla posizione corrente del file, conservato dal sistema operativo, è aggiornato sommando al valore precedente il numero di byte che sono stati scritti in memoria di massa.

La ***lettura*** da un file è un'operazione identica a quella di scrittura per quanto riguarda gli argomenti che devono essere specificati: l'handle, il numero di byte che devono essere letti, il riferimento al buffer in memoria centrale nel quale i dati vengono copiati e la posizione iniziale dalla quale iniziare la lettura. La posizione corrente all'interno del file è modificata come nell'operazione di scrittura. La differenza sostanziale tra le due operazioni consiste nel fatto che in quella di lettura nessun dato su disco è modificato.

La ***cancellazione*** di un file deve avvenire in due fasi distinte per evitare che, nel caso di una richiesta di cancellazione, vi siano altre applicazioni che, avendo aperto lo stesso file, eseguano operazioni di lettura o scrittura su di esso. Se in un caso del genere il file fosse immediatamente cancellato, le operazioni eseguite da altre applicazioni genererebbero errori derivati dal tentativo di eseguire operazioni su blocchi non validi.
Nella prima fase della cancellazione il file viene dunque solamente marcato come "cancellato" nel suo descrittore. Nella seconda fase, il file system manager verifica che altre applicazioni non stiano utilizzando il file, controllando che non esistano altri handle per lo stesso file. Nel caso in cui questo non si verifichi, il descrittore del file e i blocchi di dati occupati sono rilasciati, rendendoli in questo modo disponibili per la memorizzazione di altre informazioni. Utilizzando questa strategia è garantito che, una volta aperto un file, tutte le operazioni siano eseguite fino all'effettiva chiusura del file da parte dell'applicazione.

L'operazione di ***spostamento*** risulta molto complessa per la grande quantità di controlli che devono essere fatti per garantire l'integrità del file system. Gli argomenti dell'operazione sono l'handle della directory e il nome del file di partenza con l'handle della directory e il nome del file di destinazione;
I controlli che devono essere effettuati dal file system manager per garantire la correttezza dell'operazione sono i seguenti:

- il nome del file sorgente e del file destinazione possono coincidere se le directory sorgente e destinazione sono diverse; viceversa, i due nomi devono essere diversi se le due directory coincidono. Nel secondo caso si parla di rename del file;
- nel caso in cui il nome del file sorgente faccia riferimento a una directory, occorre che la destinazione non sia una sottodirectory della directory sorgente; ciò significherebbe infatti far diventare una directory figlia di se stessa. Questo controllo è effettuato navigando nel file system a ritroso partendo dalla directory destinazione verso la root e verificando di non incontrare la directory sorgente.

Se le verifiche danno esito positivo, viene modificato il descrittore del file cambiando il nome del file e collegando il suo descrittore alla directory destinazione.

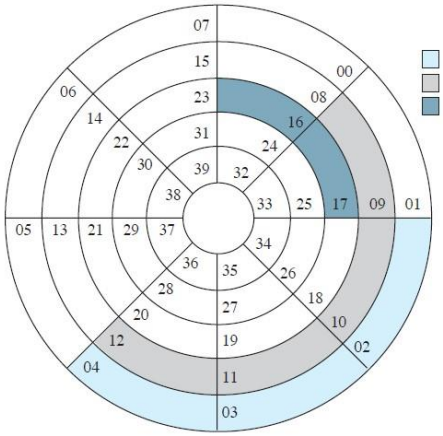
Tecniche di allocazione dei file

Quando sulla memoria di massa sono memorizzati file le cui dimensioni superano quelle di un record fisico, occorre che il file system manager adotti tecniche di allocazione dei file, in base alle quali selezionare i blocchi al cui interno il file sarà salvato. Esistono fondamentalmente tre tecniche di allocazione dei file su hard disk:

- allocazione ***contigua***; • allocazione ***concatenata***;
- allocazione ***indicizzata***.

Allocazione contigua

La tecnica di allocazione contigua è quella che presenta la minore complessità, in quanto ogni file è memorizzato in blocchi consecutivi. Il reperimento del file diventa estremamente semplice, perché conoscendo l'indirizzo iniziale del primo blocco del file e i byte totali, è sufficiente leggere a partire dal primo blocco un numero di blocchi la cui dimensione totale è pari al numero di byte occupati dal file.
Questa tecnica presenta i seguenti vantaggi:



FILE	START	LUNGHEZZA
UNO.TXT	02	3
DUE.TXT	09	4
TRE.TXT	16	2

guardare appunti

- lo spazio su disco riservato alle informazioni che devono essere memorizzate per poter accedere al file è estremamente ridotto: infatti, come abbiamo visto, sono sufficienti la posizione del primo blocco e la dimensione totale del file;
- una volta che la testina di lettura del disco si posiziona all'inizio del file, l'accesso è semplice e veloce.

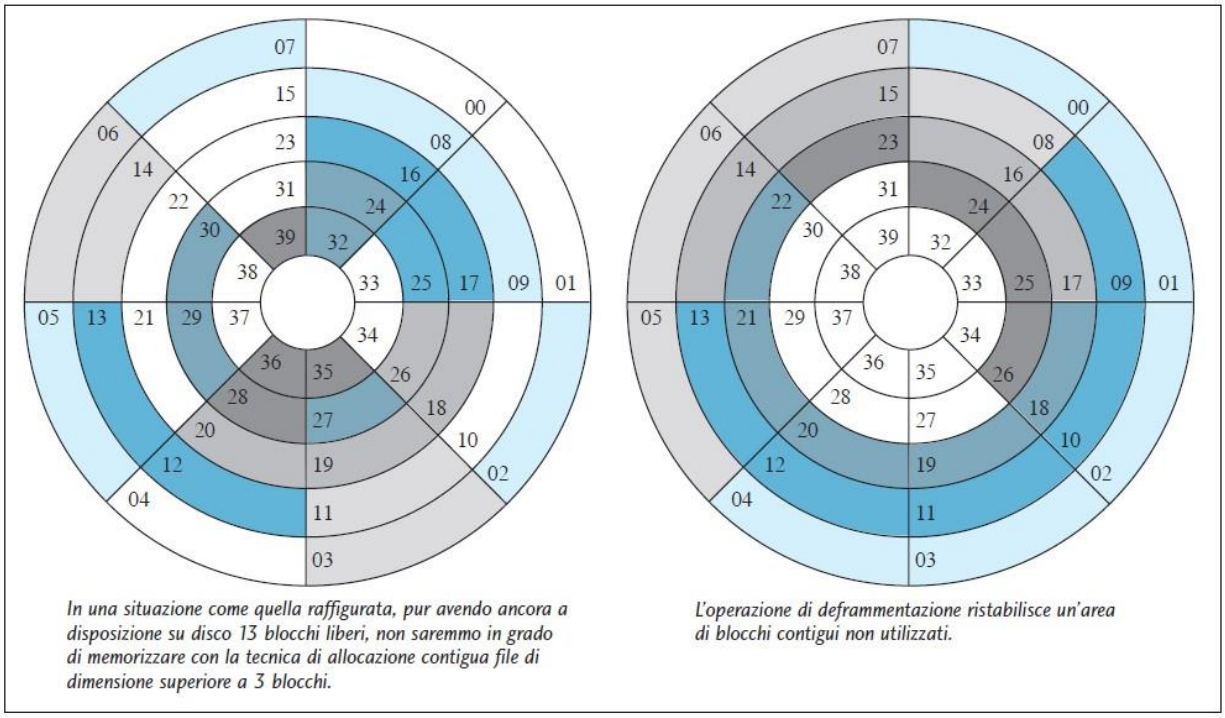
La semplicità della tecnica di allocazione contigua comporta però alcune problematiche:

- quando si vuole memorizzare un file, è necessario trovare su disco uno spazio libero contiguo abbastanza grande per contenerlo; per questo motivo potrebbe non essere possibile memorizzare un file anche se lo spazio libero complessivo sulla memoria di massa è superiore alla dimensione del file stesso, come mostrato in figura. La situazione in cui un disco presenta piccole aree libere disordinatamente distribuite è nota come ***frammentazione del disco***, ed è dovuta al fatto che, nel corso del tempo, le successive creazioni e cancellazioni di file determinano aree libere sulla

superficie del disco sempre più numerose ma sempre più limitate nelle dimensioni. La frammentazione della superficie del disco costituisce un problema anche se si usano altre tecniche di allocazione dei file, perché causa un notevole rallentamento nelle prestazioni del sistema. Per questo motivo i sistemi operativi prevedono ut***ilità di deframmentazione*** che, spostando fisicamente i blocchi che costituiscono i file, compattano le aree libere fino a formare un'unica area di settori contigui non utilizzati;

- quando un file aumenta di dimensione, è necessario, nel caso in cui il blocco immediatamente successivo al suo ultimo record fisico sia occupato, trovare un'area contigua libera sufficientemente grande e trasferirvi tutto il file. Questa operazione è ovviamente molto costosa in termini di tempo di elaborazione. Un'alternativa sarebbe quella di sovradimensionare il file al momento della creazione, ma, oltre ad avere un notevole spreco di spazio, non possiamo essere comunque certi che il file rimanga effettivamente nelle dimensioni stabilite.

A fronte di queste considerazioni, risulta che la tecnica di allocazione contigua è impossibile da applicare, se non si conoscono a priori le dimensioni definitive che i file possono assumere.

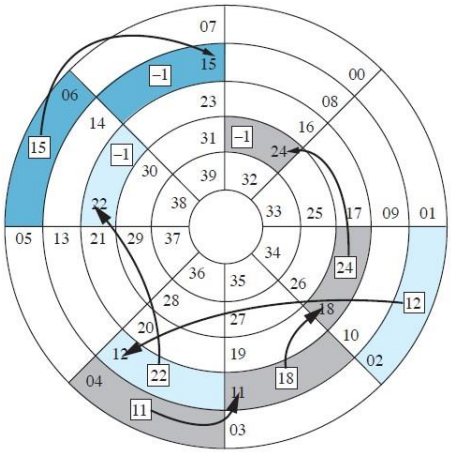


In una situazione come quella raffigurata, pur avendo ancora a disposizione su disco 13 blocchi liberi, non saremmo in grado di memorizzare con la tecnica di allocazione contigua file di dimensione superiore a 3 blocchi.

L'operazione di deframmentazione ristabilisce un'area di blocchi contigui non utilizzati.

Allocazione concatenata

Nella tecnica di allocazione concatenata, non necessariamente un file è memorizzato in blocchi consecutivi di memoria di massa. Per mantenere la sequenza corretta, ogni blocco contiene il riferimento al successivo blocco in cui è memorizzato il file; in questo modo, per reperire il file è sufficiente memorizzare le coordinate del blocco iniziale e seguire il riferimento al blocco successivo, che è registrato negli ultimi byte di ogni blocco. L'ultimo blocco del file è identificato memorizzando un numero negativo nello spazio riservato al riferimento al successivo.



FILE	START	END
UNO.TXT	02	22
DUE.TXT	04	24
TRE.TXT	06	15

Il vantaggio evidente portato da questa tecnica è il possibile utilizzo di qualsiasi blocco libero: se un file non può essere memorizzato è perché effettivamente la somma dello spazio dei blocchi non occupati è inferiore alle dimensioni del file. La frammentazione non pregiudica quindi la possibilità di memorizzare file, anche se è in ogni caso preferibile eseguire periodicamente una deframmentazione, perché un'eccessiva dispersione dei blocchi liberi sulla superficie del disco determina un degrado delle prestazioni del sistema nelle operazioni di lettura e di scrittura dei file, a causa dei troppi spostamenti che devono effettuare le testine del disco.

Le problematiche legate alla tecnica dell'allocazione concatenata sono le seguenti:

- l'accesso a un blocco del file può avvenire soltanto leggendo tutti i blocchi che lo precedono; se si vuole leggere o scrivere il centesimo blocco di un file, è necessario effettuare 100 accessi a disco per ricostruire la catena di blocchi attraverso i riferimenti al blocco successivo;

- se un blocco è corrotto, non è più possibile recuperare la porzione di file a partire da quel blocco in avanti.

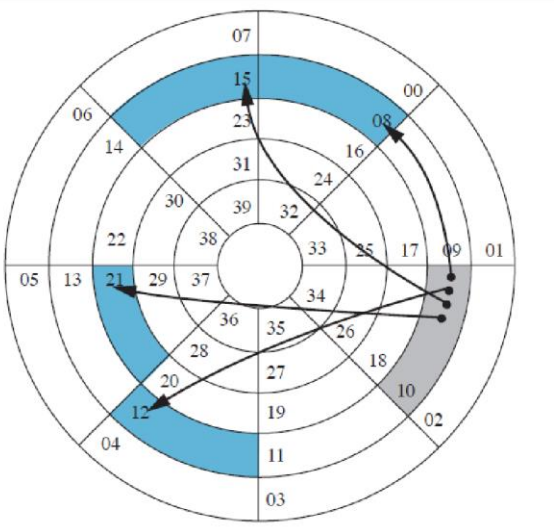
Per risolvere almeno in parte il problema della perdita di informazioni quando un blocco risulta danneggiato, possono essere adottate alcune soluzioni:

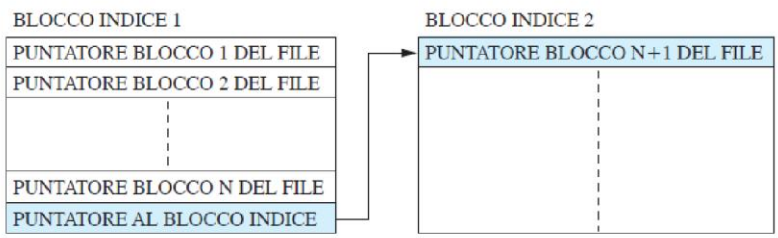
- ogni blocco può contenere anche il riferimento al blocco precedente: se inizialmente si conosce anche l'indirizzo dell'ultimo blocco, ripercorrendo a ritroso i riferimenti a partire dall'ultimo si può ricostruire la sequenza dei blocchi successivi a quello danneggiato;
- all'interno di ogni blocco può essere memorizzato il nome del file al quale appartiene e la sua posizione relativa all'interno del file: in questo modo, la scansione completa del disco permette la ricostruzione della sequenza di blocchi che erano stati persi.

Un esempio di allocazione concatenata è stato realizzato nel file system ***FAT***, in cui la *File Allocation Table* che contiene i riferimenti ai blocchi è mantenuta in memoria centrale: in questo modo i tempi di accesso al disco sono notevolmente inferiori, poiché non è più necessario leggere dalla memoria di massa tutti i blocchi che precedono quello richiesto. L'uso della FAT diventa però problematico quando gli hard disk sono di grosse dimensioni, perché la tabella occupa molto spazio in memoria centrale.

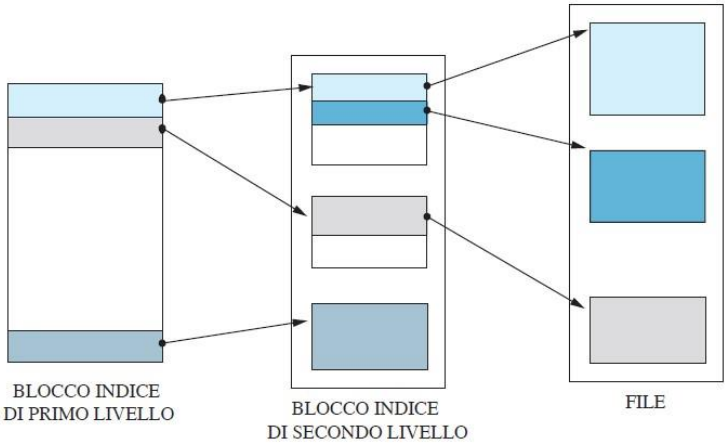
Allocazione indicizzata

Nell'allocazione indicizzata è riservato un blocco di memoria che contiene tutti i riferimenti ai blocchi che compongono il file e che prende il nome di ***blocco indice***.
Anche nel caso dell'allocazione indicizzata non è necessario che lo spazio all'interno del quale è memorizzato un file sia costituito da blocchi contigui; la lettura del blocco indice permette l'accesso diretto a tutti i blocchi del file, e conseguentemente anche l'accesso sequenziale, in modo molto efficiente.
In caso di file molto piccoli, è però in gran parte sprecato lo spazio riservato al blocco indice, che risulta praticamente vuoto. Se invece il file è di grosse dimensioni e un blocco indice non è sufficiente per contenere tutti i riferimenti ai blocchi che compongono il file, è possibile adottare le seguenti soluzioni:





- ***concatenazione di blocchi indice***: si tratta in pratica di applicare la tecnica dell'allocazione concatenata ai blocchi indice: se un blocco indice non può contenere tutti i riferimenti ai blocchi del file, l'ultimo puntatore non indica il successivo blocco del file, ma un ulteriore blocco indice;



- ***indici su più livelli***: in questo caso il primo blocco indice contiene solo i riferimenti ad altri blocchi indice, i quali poi contengono i riferimenti ai blocchi contenenti i dati.

Gestione dello spazio libero

All'interno di un file system il numero di file presenti varia dinamicamente al passare del tempo. Nuovi file possono essere creati da utenti e applicazioni, file già esistenti in seguito a modifiche possono aumentare o diminuire il numero dei settori occupati, altri file che non sono più utili possono essere cancellati. Il file system manager deve pertanto essere in grado di gestire le parti di disco che non sono occupate dai file al fine di poter allocare nuovi blocchi per i dati che devono essere trasferiti sulla memoria di massa e per poter liberare, e quindi rendere nuovamente disponibili, i blocchi che appartengono a file che sono stati cancellati.
La gestione degli spazi liberi permette inoltre che all'interno del sistema siano implementate politiche di riorganizzazione degli spazi, al fine di evitare fenomeni di frammentazione della superficie del disco che possono portare a un degrado delle prestazioni.
Le tecniche più utilizzate nella gestione dei blocchi liberi sono la ***free list*** e la ***bit map***:

- ***free list***: il metodo della free list prevede una tabella che elenca i gruppi di blocchi contigui liberi. Nella tabella sono indicati per ogni gruppo il numero di traccia e di settore del primo blocco del gruppo e quanti blocchi successivi sono liberi;
- ***bit map:*** la tecnica della bit map consiste in una mappa di bit, dove ogni bit si riferisce a un blocco fisico e una sequenza di n bit si riferisce agli n blocchi appartenenti a una stessa traccia. Il valore 1 assegnato a un bit segnala che il blocco corrispondente è già occupato dalla porzione di un file, mentre il valore 0 indica che è disponibile.

Il metodo della free list consente più facilmente l'individuazione di uno spazio libero di una certa dimensione, ma la sua gestione risulta più complessa, anche per il fatto che la dimensione della tabella varia al passare del tempo. La bit map ha invece una dimensione fissa, con un numero di bit pari al numero di blocchi della memoria di massa, ed è molto più facilmente gestibile, in quanto per segnalare un blocco come libero o occupato è sufficiente impostare un bit a 0 o 1. Per entrambi i metodi, quando un file viene creato, modificato o rimosso è necessario che il file system manager aggiorni anche la lista dei blocchi liberi o la mappa dei bit.

Free list

Traccia	*Settore*	*N. settori liberi*
0	0	2
0	4	1
1	2	1
1	7	1
2	5	3
3	7	1
4	1	2
4	5	2

Bit map

Tracce \ *Settori*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	0	0	1	1	0	1	1	1
1	1	1	0	1	1	1	1	0
2	1	1	1	1	1	0	0	0
3	1	1	1	1	1	1	1	0
4	1	0	0	1	1	1	1	1
5	1	1	1	1	1	0	0	1

Le tipologie di file system

Come già affermato in precedenza, un ***file system*** è una struttura che permette di organizzare i file in modo tale che il loro reperimento sia semplice e immediato.
Esistono numerosi file system, che possono essere raggruppati nelle seguenti tre tipologie:

- ***file system del disco***: progettati per memorizzare file su un'unità a disco, che può essere collegata direttamente o indirettamente al computer;
- ***file system distribuiti***: permettono di accedere ai file contenuti su un computer remoto tramite la rete, anche in simultanea da diversi computer;
- ***file system per compiti speciali***: utilizzati per compiti che non rientrano direttamente nelle prime due categorie. Molti non hanno alcuna relazione con un supporto di memorizzazione permanente dei dati, ma vengono utilizzati dal sistema operativo per dare accesso ad alcune funzionalità.

Tra i file system del disco ricordiamo i seguenti:

- ***FAT*** (*File Allocation Table*): è stato utilizzato dai sistemi operativi Microsoft, DOS e Windows. La versione FAT12 (prime versioni di DOS) utilizza 12 bit per l'indirizzamento dei file, mentre la versione FAT16 (16 bit per l'indirizzamento) è stata presente nelle ultime versioni del DOS fino alla prima versione di Windows 95. Con lo sviluppo di applicazioni sempre più evolute, che richiedono grossi volumi di memoria di massa, si è evidenziato un grosso limite nella FAT16, che consiste nella dimensione massima della partizione definita in 2 GB. Questa limitazione è stata superata dall'adozione della FAT32 nelle successive versioni di Windows 95 e in Windows 98;
- ***NTFS*** (*New Technology File System*): è il file system utilizzato dai sistemi Windows NT e da Windows 2000 in poi. Oltre a migliorare molti aspetti del file system FAT, prevede alcune funzionalità aggiuntive, come la gestione degli attributi dei file e dei diritti di accesso alle risorse del sistema.
- ***Ext2, Ext3, Ext4*** (*Extended File System*): sono i file system utilizzati da Linux a partire dal 1992 (Ext2) e si caratterizzano per la loro stabilità;

- ***HFS*** (*Hyerarchical File System*): utilizzato dal sistema operativo dei computer Maintosh, è conosciuto anche come *MAC OS Standard* o, nella versione successiva denominata ***HFS Plus***, come *MAC OS Extended*. HFS consente la gestione di volumi di capacità massima pari a 2 TB contenenti un massimo di 65535 file, dove ogni file non può avere dimensioni superiori a 2 GB. La particolarità di questo file system risiede nel fatto che l'organizzazione dei file si basa sulla struttura ***BTree***, che permette ricerche particolarmente veloci anche quando il numero di file è molto grande.

File system FAT

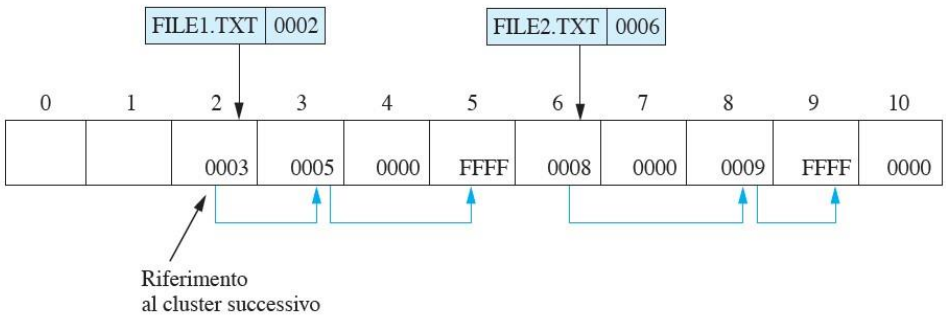
Il file system ***FAT*** (*File Allocation Table*) è stato progettato per dischi di modeste capacità, ed è per questo motivo che Microsoft per i più recenti sistemi Windows utilizza il più potente e versatile NTFS, pur mantenendo ancora la compatibilità con il vecchio file system. Una partizione all'interno di un volume formattato secondo il file system FAT ha la seguente struttura:

BOOT SECTOR DELLA PARTIZIONE	*FILE ALLOCATION TABLE*	*DUPLICATO DELLA FILE ALLOCATION TABLE*	*DIRECTORY ROOT*	*DIRECTORY E FILE*

La struttura che il file system utilizza per la gestione dei file è la ***File Allocation Table***, memorizzata come un file all'inizio del volume, la cui funzione è quella di mantenere i riferimenti indispensabili per reperire i file appartenenti all'albero delle directory. Per assicurare la funzionalità del sistema ed evitare problemi di reperimento dei file nel caso in cui la tabella sia soggetta a errori di memorizzazione, è presente sul volume una copia ulteriore della File Allocation Table, alla quale si fa riferimento nel caso in cui la tabella principale non sia utilizzabile a causa della corruzione dei dati contenuti al suo interno. Lo spazio all'interno del quale sono memorizzati i dati sul volume è suddiviso in ***cluster***, gruppi di byte contigui la cui dimensione fissa è determinata dalla dimensione totale del volume ed è compresa tra 4 e 64 kbyte. Ogni cluster è identificato da un numero che deve essere contenuto in 12 bit nel caso della FAT12, in 2 byte nel caso della FAT16 e in 4 byte nel caso della FAT32; il numero di cluster disponibili nel volume è 4.086 per la FAT12, fino a 65.526 per la FAT16 e 268.435.456 per la FAT32. Il riferimento ai file è mantenuto in una struttura denominata ***folder entry***, che comprende le seguenti informazioni:

- ***Nome*** del file: 11 caratteri
- ***Attributi***: con 4 bit che specificano se il file è rispettivamente un archivio, di sistema, nascosto o di sola lettura; • ***Ora e data di creazione***;
- ***Data ultimo accesso***;
- ***Ora e data ultima modifica***;
- ***Identificatore del primo cluster*** del file;
- ***Dimensione*** del file.

L'identificatore del primo cluster è l'indirizzo del primo cluster su disco occupato dal file; se il file ha dimensioni superiori a quelle di un cluster, il suo contenuto è frammentato su blocchi diversi, che possono anche non essere contigui. Il file system FAT utilizza la tecnica di *allocazione concatenata*: infatti, al fine di mantenere il riferimento alla porzione successiva del file, all'interno di ogni cluster è memorizzato l'identificatore del successivo blocco del file. Il cluster che contiene il valore esadecimale FFFF indica la fine del file, o EOF (End Of File).



In ogni cluster possono essere memorizzati i seguenti valori:

- 0, se il cluster non è occupato da nessun file;
- il riferimento al successivo cluster del file, se il cluster non è l'ultimo; • il valore 65527, se il cluster è corrotto;
- il valore esadecimale FFFF, se il cluster è l'ultimo del file.

La struttura ad albero delle directory è mantenuta nel ***root folder entry***, che contiene il riferimento a ogni file e directory contenuti nella root. La differenza rispetto agli altri folder entry risiede nel fatto che il root folder entry è memorizzato in una posizione fissa all'interno del disco, mentre gli altri folder entry possono essere memorizzati in qualsiasi posizione. Le informazioni contenute nel folder possono essere utilizzate da tutti i sistemi operativi che utilizzano il file system FAT.

File system NTFS

Il file system NTFS rappresenta un notevole passo avanti rispetto al file system FAT. Le sue principali caratteristiche sono le seguenti:

- ***affidabilità***: nel caso di eventi che potrebbero compromettere i dati memorizzati nel file system (ad esempio i blackout), NTFS permette il recupero di una situazione corretta grazie alla caratteristica di essere un sistema transazionale. Nel caso di un'operazione non terminata con successo, i file sono recuperati dall'ultima situazione coerente: in sostanza sono perse le modifiche dell'ultima operazione eseguita ma non è compromessa l'integrità del file system;
- ***controllo di accesso***: è possibile assegnare agli utenti del sistema operativo diritti di accesso distinti di lettura, scrittura, modifica, cancellazione sul singolo file;
- ***lunghezza nomi dei file***: i nomi dei file e delle cartelle sono direttamente gestiti fino a una lunghezza di 255 caratteri e possono contenere caratteri di tutte le lingue del mondo grazie alla codifica *Unicode*;
- ***dimensioni***: la dimensione di un volume può raggiungere 256 Terabyte (pari a 2^{32} clusters – 1), il numero massimo di files è circa 4,3 miliardi ($2^{32} – 1$), mentre la dimensione di un singolo file può raggiungere 16 Terabyte, a fronte dei 4 GigaByte di FAT32.

La struttura principale di un file system NTFS è la ***Master File Table*** (*MFT*), una tabella strutturata in blocchi che possono avere una dimensione compresa tra 1 kB e 4 kB. Quando un file o una directory sono creati all'interno di un volume, in questa struttura è aggiunta una nuova voce contenente alcune informazioni che si riferiscono al file che è stato creato, definite ***attributi***. Poiché anche il contenuto del file è considerato un attributo, nel caso in cui la sua dimensione sia piccola anch'esso è memorizzato all'interno della voce della Master File Table, migliorando così sensibilmente le prestazioni del sistema.
Dal momento che all'interno del file system possono essere creati molti file e directory, il sistema operativo riserva lo spazio immediatamente successivo alla MFT per una quantità pari al 12,5% dello spazio totale del disco per contenere le eventuali voci aggiuntive della Master File Table, creando in questo modo la ***MFT Zone***. L'assegnazione di questo spazio, che costituisce una percentuale rilevante della quantità di memoria del disco, viene gestita da NTFS in modo dinamico: nel caso in cui tutto lo spazio rimanente sia occupato, nuovi file possono essere allocati nelle aree libere della MFT Zone. Le informazioni memorizzate nella Master File Table sono le seguenti:

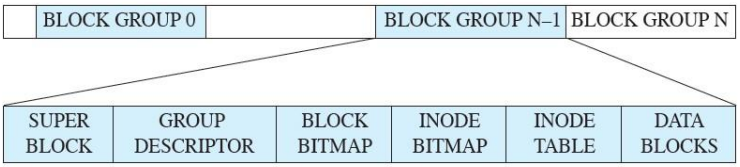
- ***Header***: contiene le informazioni che NTFS utilizza per gestire le directory, tra le quali gli identificatori per i file usati internamente dal sistema operativo, i riferimenti agli altri attributi e la quantità di spazio rimasto libero all'interno della voce della MFT;
- ***Standard Information Attribute***: all'interno di questo attributo sono memorizzate informazioni fondamentali, come la data e l'ora di creazione e dell'ultima modifica del file, quando si è verificato l'ultimo accesso, se il file è nascosto e se è possibile modificarlo;
- ***nome del file***: un file può avere anche più nomi, per permettere ad esempio di assegnare a esso anche una sequenza di caratteri che segua le regole dei nomi brevi del DOS;
- ***informazioni di sicurezza***: è l'attributo che contiene le regole di sicurezza che limitano l'accesso al file per ogni utente del sistema. Queste regole sono memorizzate nelle Access Control List (ACL) e specificano i seguenti permessi:
    - *nessun tipo di accesso*;
    - *possibilità di elencare i file* di una directory; ○ *lettura del contenuto* del file; ○ *aggiunta* di un file; ○ *aggiunta* e *lettura*; ○ *modifica del contenuto* del file; ○ *controllo completo*.

Le prestazioni e le opportunità offerte dal file system NTFS sono nettamente superiori a quelle dei precedenti file system FAT. Una funzionalità interessante offerta dal file system manager è la possibilità di comprimere i file, anche se con una percentuale di compressione inferiore a quella dei programmi specifici di compressione dati, con la possibilità di poter accedere in modo immediato a qualunque punto del file senza dover operare una decompressione.
In NTFS sono stati aggiunti i cosiddetti ***punti di reparse***, ovvero meccanismi che consentono le giunzioni, o ***hard link***, tra directory, già previsti, come vedremo, nei file system utilizzati dal sistema operativo Linux. In pratica gli hard link offrono la possibilità di mantenere una sola copia di un file che però è visibile direttamente in più directory.
Rispetto ai file system FAT, NTFS è più rispondente alle esigenze di utenti e programmatori che utilizzano e sviluppano applicativi sempre più complessi: per questo motivo risulta molto più completo rispetto ai predecessori, ma anche più complicato da gestire. Questa difficoltà, legata al fatto che NTFS è un file system proprietario, non permette ancora la

compatibilità diretta con i sistemi operativi non sviluppati da Microsoft: l'utilizzo di questo file system da parte di altri sistemi operativi è possibile solo con l'utilizzo di programmi che agiscano da intermediari.

I file system Ext2, Ext3 e Ext4

Il file system Ext sono stati progettati in modo specifico per i sistemi operativi Linux e, come nella quasi totalità degli altri file system, i file sono memorizzati all'interno della memoria di massa in blocchi di lunghezza fissa, che viene definita dalla particolare versione di file system; se la dimensione di un file non è esattamente un multiplo della dimensione del blocco, una parte dell'ultimo blocco non viene utilizzata, con conseguente spreco di spazio. Alcuni blocchi sono utilizzati per mantenere le informazioni relative all'organizzazione del file system: come vedremo in seguito, la struttura utilizzata a questo scopo è l'***inode***.
Nella seguente figura è mostrato come l file system Ext vedono la superficie di un disco, suddivisa in ***gruppi di blocchi***, definiti anche come ***cilynder groups***.

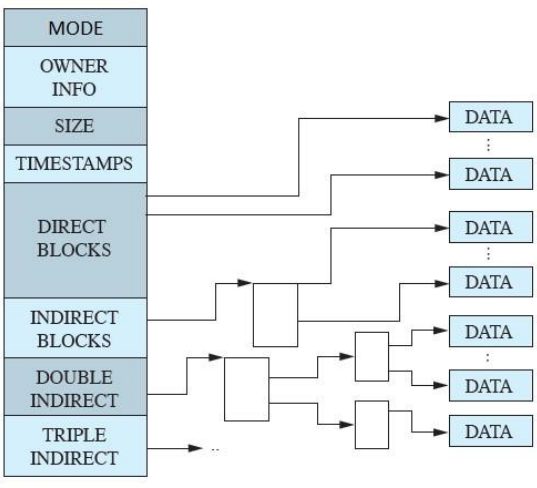


La suddivisione in gruppi di blocchi ha lo scopo di suddividere il file system in entità autonome, sia per far sì che eventuali errori all'interno di un blocco di record non compromettano l'intero file system, sia per ridurre i tempi di accesso ai dati, memorizzando i file nello stesso gruppo di blocchi delle directory che li contengono. La struttura di ogni gruppo di blocchi è la seguente:

- ***Superblock***: è l'informazione che consente al sistema operativo di usare e mantenere il file system. Quando il sistema operativo esegue il mount del file system, viene letto il superblock contenuto nel primo gruppo di blocchi fisici, corrispondente al blocco 0; per preservare l'integrità dei dati in caso di corruzione dei dati stessi, ogni gruppo di blocchi contiene un duplicato del superblock. Tra le informazioni contenute nel superblock troviamo diverse informazioni tra le quali la versione di file system utilizzata, il riferimento al gruppo di blocchi che conserva la copia del superblock, la dimensione del singolo blocco (normalmente è di 1024 byte), il numero di blocchi e di inode liberi all'interno del gruppo;
- ***Group descriptor***: ogni group descriptor contiene informazioni relative al gruppo di blocchi cui si riferisce, come ad esempio il numero dei blocchi liberi, quello degli inode liberi e delle directory presenti nel gruppo;
- ***Data block bitmap***: è una sequenza di bit, ognuno dei quali indica se il corrispondente data block del gruppo è libero o già occupato da un file. La sua dimensione è quella di un blocco e quindi, nel caso in cui la dimensione del blocco sia di 1024 byte, è possibile fare riferimento a 1024 x 8 = 8192 blocchi per ogni gruppo;
- ***Inode bitmap***: è la bitmap che si riferisce agli inode e ha modalità del tutto analoghe a quelle della data block bitmap;
- ***Inode table***: è l'area all'interno del file system che contiene gli inode del gruppo;
- ***Data blocks***: sono i blocchi del gruppo che memorizzano le informazioni contenute all'interno dei file.

Per ogni file presente nel file system viene memorizzato in memoria di massa un ***inode***, che costituisce la struttura fondamentale sulla quale poggia l'organizzazione dei file system Ext. Ogni inode è identificato in modo univoco da un numero ed è contenuto in tabelle accessibili dal sistema operativo e dalle applicazioni. Al suo interno sono memorizzate alcune informazioni relative al file cui si riferisce, quali i puntatori ai blocchi che contengono i dati del file, i diritti di accesso, il tipo del file e quando il file è stato modificato. Anche le directory, che sono viste come file speciali, hanno un inode associato, che contiene al suo interno i puntatori agli inode dei file e delle sottodirectory che fanno parte della directory considerata.
Tutti gli inode sono contenuti in una tabella che forma una bitmap, all'interno della quale il valore di ogni bit indica se l'inode corrispondente è occupato o può essere utilizzato per memorizzare la porzione di un file.
Nel dettaglio, le informazioni contenute negli inode sono le seguenti:

- ***Mode***: memorizza qual è l'oggetto al quale l'inode si riferisce (ad esempio file o directory) e i diritti di accesso che hanno gli utenti su di esso;
- ***Owner Information***: contiene l'identificatore dell'utente che è il proprietario del file o della directory;
- ***Size***: dimensione del file in byte;



- ***Timestamps***: memorizza quando il file è stato creato e quando è stato modificato per l'ultima volta;
- ***Data blocks***: sono i puntatori ai blocchi che contengono effettivamente i dati del file. Come mostrato nella figura, solo i primi dodici puntatori si riferiscono direttamente ai blocchi che contengono fisicamente i file, mentre gli ultimi tre fanno riferimento ad altri blocchi di puntatori: nel caso del tredicesimo puntatore si ha una *redirezione*, cioè il puntatore punta a un blocco di puntatori che a loro volta puntano ai blocchi dei file. Nel penultimo e nell'ultimo blocco viene realizzata rispettivamente una doppia e una tripla redirezione.

Oltre a rendere più facile e veloce la lettura e la scrittura dei file su disco, il file system Ext3 introduce rispetto a Ext2 l'importante novità del ***journaling***, una tecnica che, mantenendo una traccia delle modifiche effettuate sui file, è in grado di recuperare situazioni di errore, quali malfunzionamenti hardware o lo spegnimento del PC senza la preventiva chiusura del sistema operativo.
Le principali caratteristiche dei tre file system sono le seguenti:

ext2:

- non ha funzionalità di journaling e per questo motivo è raccomandato su unità flash, in quanto non è necessario eseguire il sovraccarico del journaling;
- le dimensioni massime dei singoli file possono variare da 16 GB a 2 TB;
- le dimensioni complessive del file system ext2 possono variare da 2 TB a 32 TB.

ext3

- consente l'inserimento nel journal;
- il journaling ha un'area dedicata nel file system, in cui vengono monitorate tutte le modifiche. Quando il sistema si arresta in modo anomalo, la possibilità di corruzione del file system è inferiore a causa del journaling;
- le dimensioni massime dei singoli file possono variare da 16 GB a 2 TB;
- le dimensioni complessive del file system ext3 possono variare da 2 TB a 32 TB; • esistono tre tipi di journaling disponibili nel file system ext3:
  - *diario*: i metadati e il contenuto vengono salvati nel diario;
  - *ordinato*: nel diario vengono salvati solo i metadati. I metadati vengono registrati su giornale solo dopo aver scritto il contenuto sul disco. Questo è il valore predefinito;
  - *writeback*: solo i metadati vengono salvati nel journal. I metadati potrebbero essere registrati su giornale prima o dopo che il contenuto è stato scritto sul disco;
- è possibile convertire direttamente un file system ext2 in file system ext3 (senza backup / ripristino).

ext4

- Supporta enormi dimensioni di file individuali e dimensioni complessive del file system: le dimensioni massime dei singoli file possono variare da 16 GB a 16 TB e la dimensione massima complessiva del file system ext4 è 1 EB (exabyte). NB: 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte);
- la directory può contenere un massimo di 64.000 sottodirectory (invece di 32.000 in ext3);
- è possibile montare un file system ext3 esistente come file system ext4 (senza doverlo aggiornare);
- sono state introdotte diverse altre nuove funzionalità: allocazione multiblocco, allocazione ritardata, checksum journal. fast fsck, ecc. , che hanno migliorato le prestazioni e l'affidabilità del filesystem rispetto a ext3;
- esiste anche la possibilità di disattivare la funzione di journaling.