Università degli Studi di Cassino e del Lazio Meridionale

Corso di Fondamenti di Informatica

Files

Anno Accademico 2016/2017

Francesco Tortorella

File e file system

Le unità di memoria di massa forniscono il supporto fisico per la memorizzazione permanente dei dati, e presentano caratteristiche estremamente diverse a seconda della casa costruttrice e del tipo di unità.

Il File System offre una visione logica uniforme della memorizzazione dei dati basata su un'unità di memoria logica, il file, definita indipendentemente dalle caratteristiche fisiche delle particolari unità.

Il file

Il file è un insieme di informazioni, correlate e registrate nella memoria di massa, identificato da un nome, che può essere formato da più sottoparti.

- nome: si riferisce ai contenuti del file
- estensione: si riferisce al tipo del file

Dal punto di vista dell'utente, un file è la più piccola porzione (logica) di memoria secondaria: i dati, cioè, possono essere scritti nella memoria secondaria solo all'interno di un file.

Contenuti di un file

Le informazioni registrate all'interno di un file sono di due tipi:

- **dati veri e propri**
  - programmi eseguibili
  - testi
  - immagini
  - dati numerici
  - …
- **attributi di interesse per l'utente**
  - dimensione del file
  - data di creazione e/o ultima modifica
  - permessi di accesso

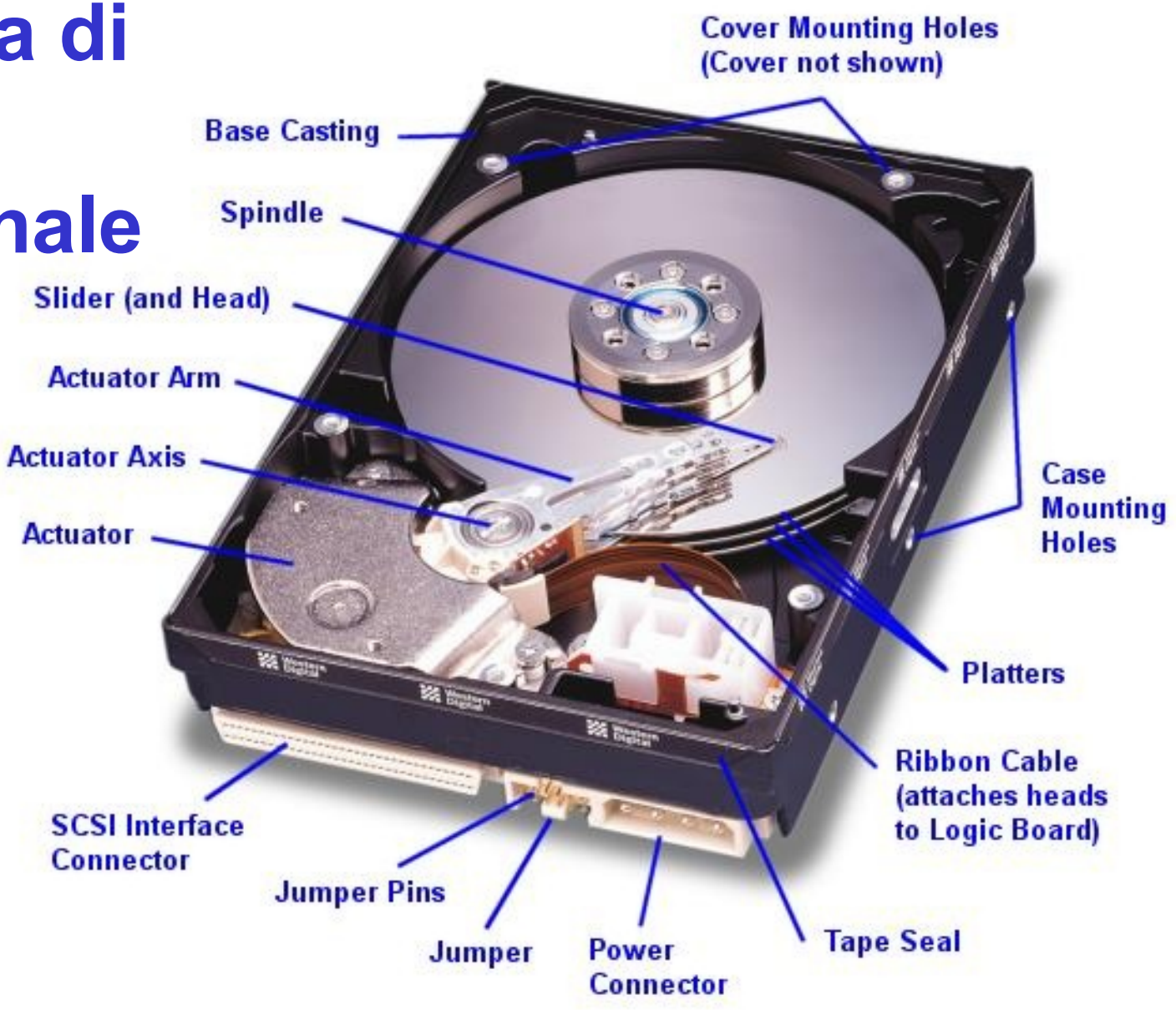
Organizzazione logica dei files

- L'insieme dei file presenti in memoria di massa è organizzato secondo una struttura gerarchica ad albero, in cui i nodi intermedi costituiscono le directory (che raggruppano altri files e directory secondo un criterio di omogeneità), mentre le foglie rappresentano i file.
- All'interno di tale struttura, un particolare file è univocamente identificato dal path (o percorso) che localizza la directory in cui il file è memorizzato.

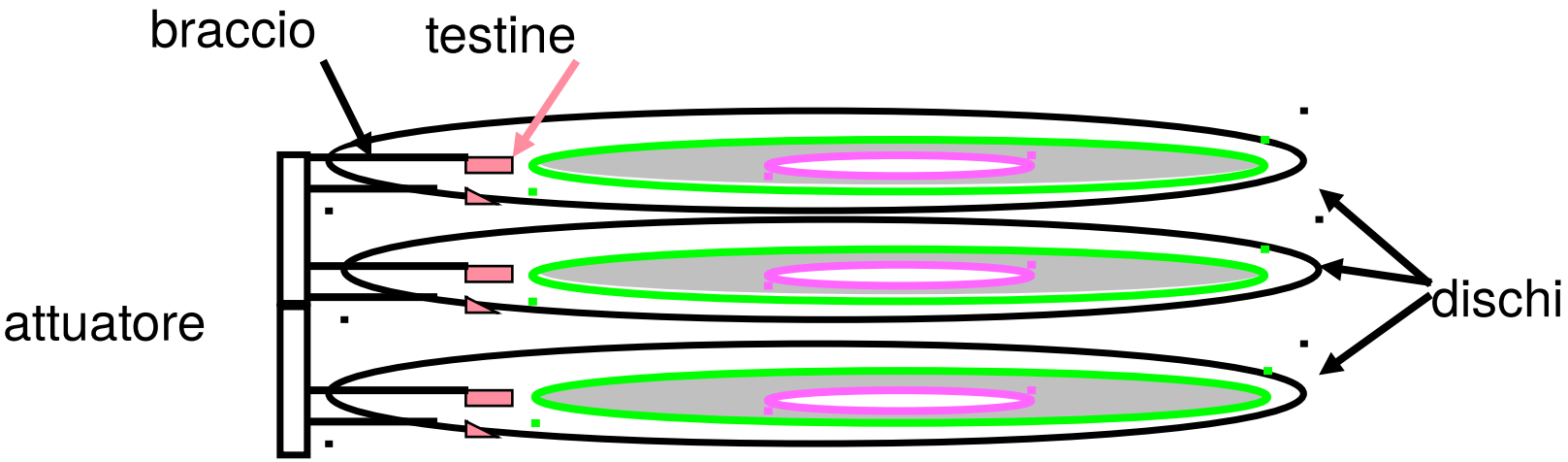
Organizzazione fisica dei files

- Da un punto di vista fisico, la registrazione del file sul disco viene realizzata dal sistema operativo disponendo il contenuto del file su un insieme di cluster possibilmente contigui.
- La registrazione dei dati è organizzata in maniera sequenziale, per cui le operazioni di lettura e scrittura possono avvenire solo a partire dall'inizio e procedendo verso la fine.
- La successione dei blocchi nei quali sono memorizzati i bytes che lo compongono può essere strutturata come:

Struttura di un HD tradizionale

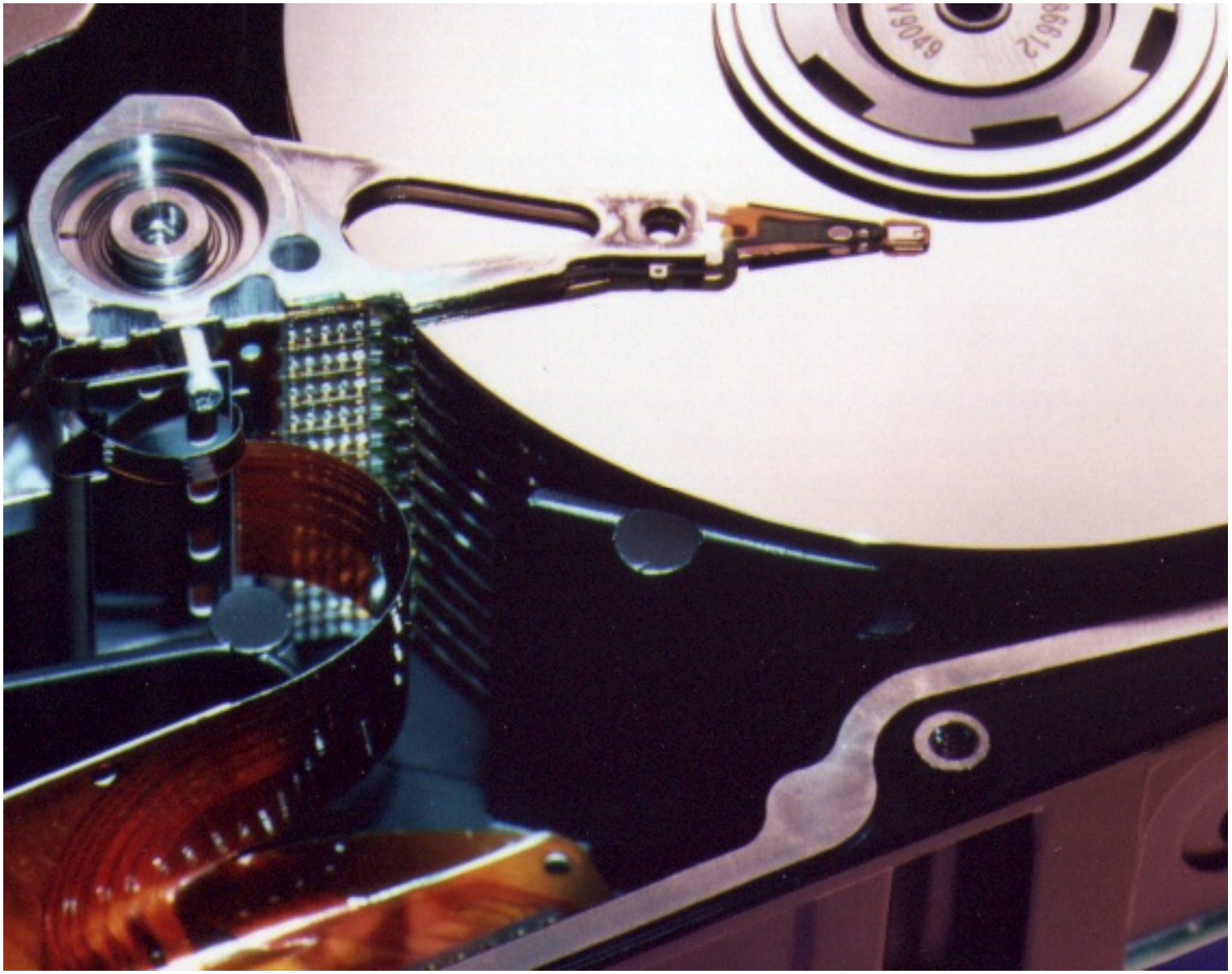


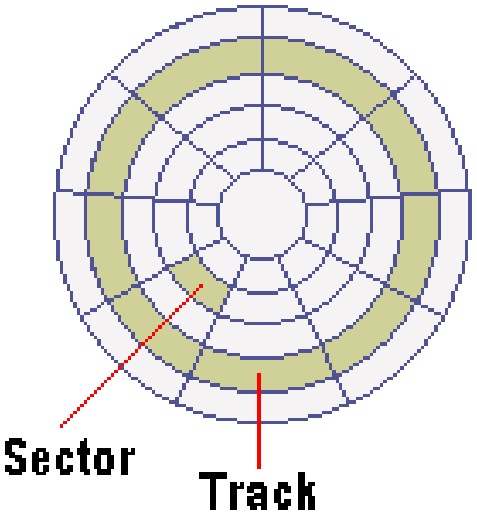
Struttura di un HD tradizionale



L'unità è in realtà costituita da diversi dischi. Entrambe le superfici di ogni disco sono rivestite di materiale magnetico sul quale vengono memorizzate le informazioni.

Le operazioni di letture e scrittura sono realizzate da testine, poste su bracci e movimentate da un attuatore.



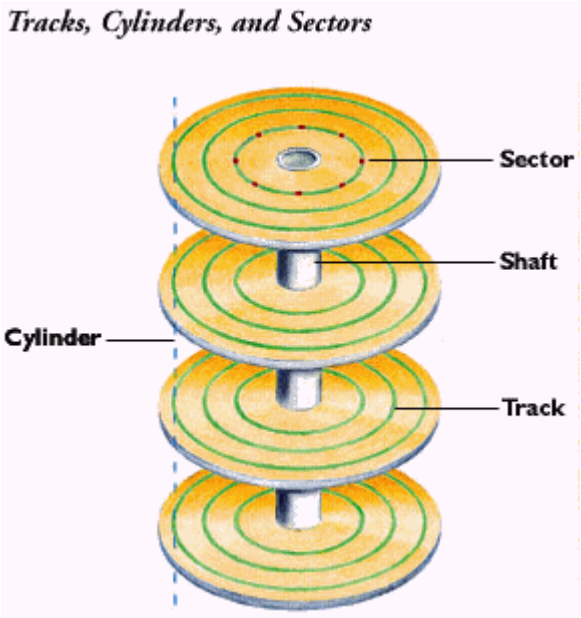


Tutte le informazioni memorizzate sul disco sono organizzate in tracce (corone circolari concentriche disposte sulla superficie del disco).

Le tracce sono numerate a partire da zero dal bordo del disco e procedendo verso l'interno.

Ogni traccia è divisa in più blocchi (da 512 byte) denominati settori, che sono le più piccole unità di memorizzazione sul disco.

Siccome l'unità è formata da più dischi, ad ogni traccia su un disco corrispondono tracce omologhe sugli altri dischi, che, nell'insieme, formano un *cilindro.*



Operazioni sul disco

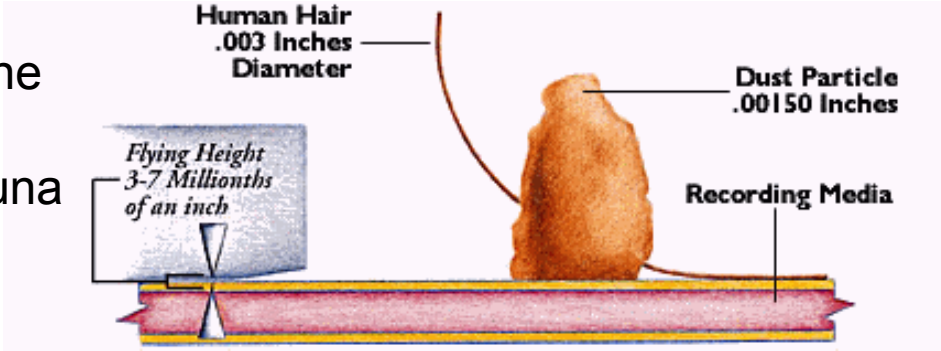
Le informazioni memorizzate sul disco sono codificate sotto forma di stati di memorizzazione di zone del materiale magnetico disposto sulla superficie del disco.

Le operazioni di lettura/scrittura sono realizzate dalle testine tramite le seguenti fasi:

1. Posizionamento della testina sulla traccia (cilindro) di interesse;
2. Attesa del passaggio del settore di interesse;
3. Lettura o scrittura del dato.

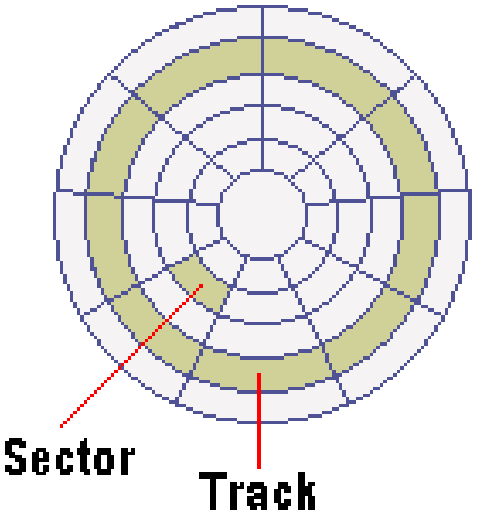
Accesso ai dati di tipo *random*

Date le alte velocità di rotazione, le testine non toccano la superficie del disco, ma "planano" su di essa, mantenendosi ad una distanza dell'ordine di 10^{-4} mm.



Organizzazione di un file sul disco

- Un file è quindi composto da un insieme di settori distribuiti sul disco
- In una zona speciale risiedono i metadati del singolo file (nome, ecc.) insieme con l'elenco dei settori che lo formano
-

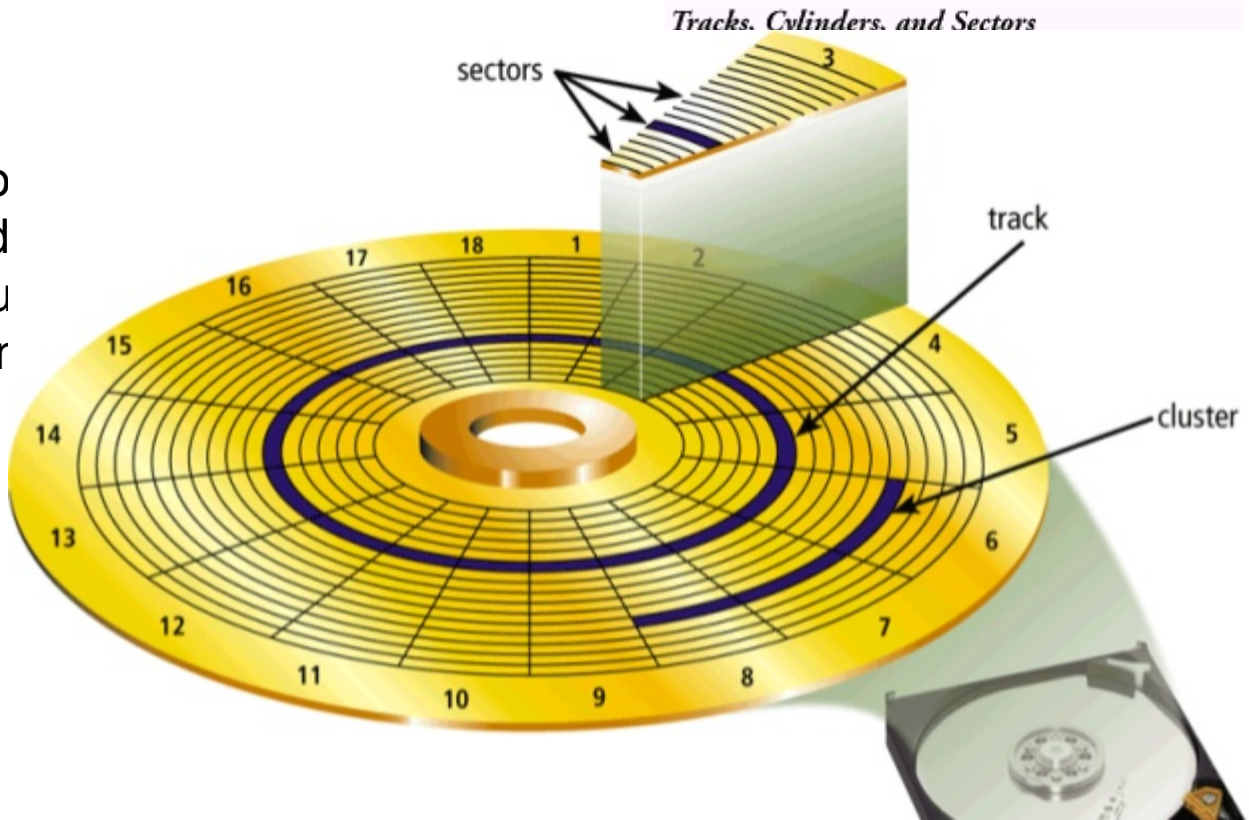


Tutte le informazioni memorizzate sul disco sono organizzate in tracce (corone circolari concentriche disposte sulla superficie del disco).

Le tracce sono numerate a partire da zero dal bordo del disco e procedendo verso l'interno.

Ogni traccia è divisa in più blocchi (da 512 byte) denominati settori, che sono le più piccole unità di memorizzazione sul disco.

Siccome l'unità è fo
ogni traccia su un d
tracce omologhe su
nell'insieme, formar



Organizzazione fisica dei files

- Da un punto di vista fisico, la registrazione del file sul disco viene realizzata dal sistema operativo disponendo il contenuto del file su un insieme di cluster possibilmente contigui.
- La registrazione dei dati è organizzata in maniera sequenziale, per cui le operazioni di lettura e scrittura possono avvenire solo a partire dall'inizio e procedendo verso la fine.

SSD vs. HDD

2.5" SATA 3.0Gbps SSD		2.5" SATA 3.0Gbps HDD
Solid NAND flash based	**Mechanism type**	Magnetic rotating platters
64GB	**Density**	80GB
73g	**Weight**	365g
Read: 100MB/s, Write :80MB/s	**Performance**	Read: 59MB/s, Write: 60MB/s
1W	**Active Power consumption**	3.86W
20G (10~2000Hz)	**Operating Vibration**	0.5G (22~350Hz)
1,500G for 0.5ms	**Shock resistance**	170G for 0.5ms
0°C to 70°C	**Operating temperature**	5°C to 55°C
None	**Acoustic Noise**	0.3 dB
MTBF >2M hours	**Endurance**	MTBF < 0.7M hours

Il concetto di *stream*

- Uno stream è un'astrazione che rappresenta un dispositivo sul quale si possono realizzare operazioni di input e output.
- Uno stream può essere sommariamente rappresentato come una sorgente o una destinazione di sequenze di caratteri di lunghezza indefinita.
- Lo stream costituisce un'interfaccia verso il file fisico e consente di realizzare da programma operazioni di I/O su di esso.

La libreria `fstream`

- Il C++ gestisce gli stream attraverso oggetti di tipo `filestream`.
- La libreria `<fstream>` contiene le definizioni per tali oggetti
- Gli oggetti filestream sono di tre tipi:
  - **`ifstream`** per leggere dati da file (input file stream)
  - **`ofstream`** per scrivere dati su file (output file stream)
  - **`fstream`** per leggere e scrivere dati su file (in momenti diversi)

Definizione di un filestream

- È necessario includere l'header file `fstream`
- Si definisce quindi il filestream

```
#include <fstream>

ifstream f_in; //stream in lettura
ofstream f_out; //stream in scrittura
```

Connessione ad un file

- Per poter compiere operazioni su un file è necessario connettere il filestream al file fisico che si trova sulla memoria di massa
- La connessione si realizza tramite la funzione membro `open`

```
void open(const char *filename,
          ios::openmode mode);
```

`filename:` nome e locazione del file

`mode:` modalità di apertura

Modalità di apertura

`ios::in` apertura in lettura

`ios::out` apertura in scrittura

`ios::app` apertura in “append”.
L’output verso il file è accodato al contenuto già esistente.

`ios::trunc` se il file esiste, il suo contenuto viene cancellato prima della connessione.

Esempi di modalità di apertura

```
ofstream fs;
fs.open(“pippo.txt”,ios::out);
```

- È possibile unire più modalità mettendole in OR

```
ofstream fs;
fs.open(“pippo.txt”,ios::out|
  ios::append);
```

Sconnessione da un file

- Una volta terminate le operazioni, è possibile sconnettere il filestream dal file fisico tramite la funzione membro **`close`**

```
ofstream fs;
fs.open("pippo.txt",ios::out);
. . .
fs.close()
```

- Dopo la sconnessione, il filestream può essere connesso ad un altro file

Controlli sul filestream

- Esistono diverse funzioni che restituiscono un valore booleano e informano sullo stato del filestream

`bool good()` operazione andata a buon fine

`bool bad()` condizione di errore (non recuperabile)

`bool fail()` condizione di errore (da controllare)

`bool eof()` raggiunta la fine del file

Controlli sul filestream

```
ifstream in;
char buffer[256];
in.open("elenco.txt", ios::in);
if(!in.good()){
        cout<<"Problemi di apertura del file\n";
       exit(1);
}
in.getline(buffer);
while(!in.eof()){
   cout << buffer << endl;
   in.getline(buffer);
}
in.close();
```

Filestream come parametri di funzioni

- È possibile avere filestream come parametri di una funzione, ma sempre passati by reference

```
void lineread(ifstream &fin, char nome[], char
cognome[]){
char appo[MAX_LEN];

fin.getline(nome, MAX_LEN, ';');
fin.getline(cognome, MAX_LEN, ';');
// Elimina il \n alla fine di ogni riga
fin.getline(appo, MAX_LEN);
}
```