Sistemi Operativi: Filesystems

Amos Brocco, Ricercatore, DTI / ISIN

Basato su:
[STA09] “Operating Systems: Internals and Design Principles”, 6/E, William Stallings, Prentice Hall, 2009
[TAN01] “Modern Operating Systems”, 2/E, Andrew S. Tanenbaum, Prentice Hall, 2001
[TAN09] “Modern Operating Systems”, 3/E, Andrew S. Tanenbaum, Prentice Hall, 2009

Perché memorizzare (su disco)

- Perché la memoria (RAM) non è sufficiente
- Per mantenere informazione a lungo termine
  - ...dopo aver terminato un'applicazione
  - ...dopo aver spento il computer
- Per permettere la condivisione
  - ...tra processi
  - ...tra sistemi

Files

- **File:** astrazione che permette di salvare informazione su una memoria a lungo termine (es. disco rigido) e leggerla successivamente

- I dettagli di “come” il file è scritto sul disco sono nascosti dal sistema operativo

- **Attributi (metadati) di un file:** nome, estensione, data di creazione, data di modifica, proprietario, permessi,...

Tipi di files

- File dati: es. Eseguibili, immagini, audio,...

  Nota: per la gestione del file system tutti questi files rappresentano la stessa cosa: una sequenza di bytes!
- Directory
- File speciali (*NIX):
  - Periferiche a carattere
  - Periferiche a blocchi

File system

- È principalmente un'astrazione, di cui fanno parte i file e le directory

- È una componente del sistema operativo che si occupa di gestire i files
  - Operazioni sui file (lettura, scrittura,...)
  - Gestione degli attributi
  - Gestione dello spazio libero sul disco
  - Gestione dei dati su disco
  - Gerarchia dei files (directories)

Accesso ai files

- **Sequenziale**
  - I dati vengono letti/scritti uno dopo l'altro, a partire dall'inizio
- **Casuale**
  - È possibile accedere a una qualsiasi posizione del file

Operazioni sui files

- Crea (Create)
- Elimina (Delete)
- Apri (Open)
- Chiudi (Close)
- Leggi (Read)
- Scrivi (Write)
- Aggiungi (Append)
- Cerca (Seek)
- Leggi attributi (Get attributes)
- Scrivi attributi (Set attributes)
- Rinomina (Rename)

Esempio: copia di un file (1)

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
      int in_fd, out_fd, rd_count, wt_count;
      char buffer[BUF_SIZE];

      if (argc != 3) exit(1);           /* syntax error if argc is not 3 */
```

Da A. Tanenbaum, “Modern Operating Systems”, 2ª Edizione

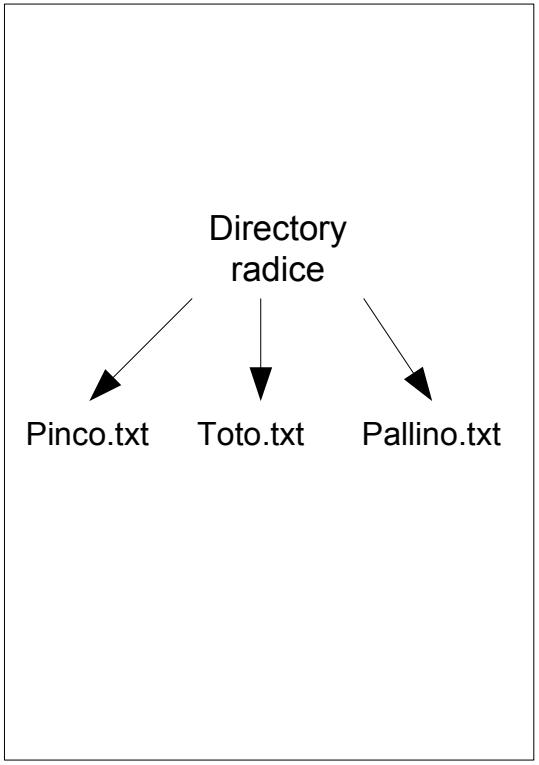
Esempio: copia di un file (2)

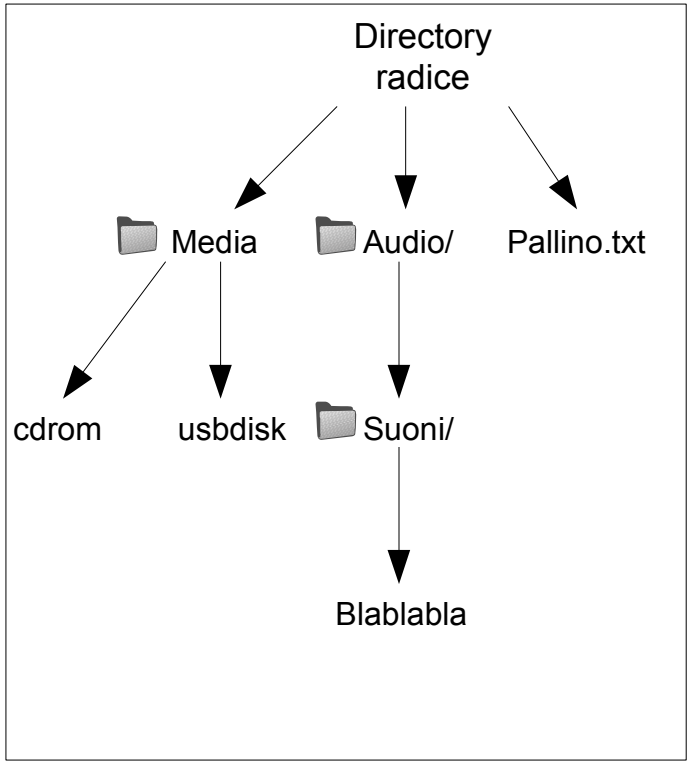
```
/* Open the input file and create the output file */
in_fd = open(argv[1], O_RDONLY);     /* open the source file */
if (in_fd < 0) exit(2);              /* if it cannot be opened, exit */
out_fd = creat(argv[2], OUTPUT_MODE);  /* create the destination file */
if (out_fd < 0) exit(3);             /* if it cannot be created, exit */

/* Copy loop */
while (TRUE) {
      rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
if (rd_count <= 0) break;            /* if end of file or error, exit loop */
      wt_count = write(out_fd, buffer, rd_count); /* write data */
      if (wt_count <= 0) exit(4);    /* wt_count <= 0 is an error */
}

/* Close the files */
close(in_fd);
close(out_fd);
if (rd_count == 0)                   /* no error on last read */
      exit(0);
else
      exit(5);                       /* error on last read */
}
```

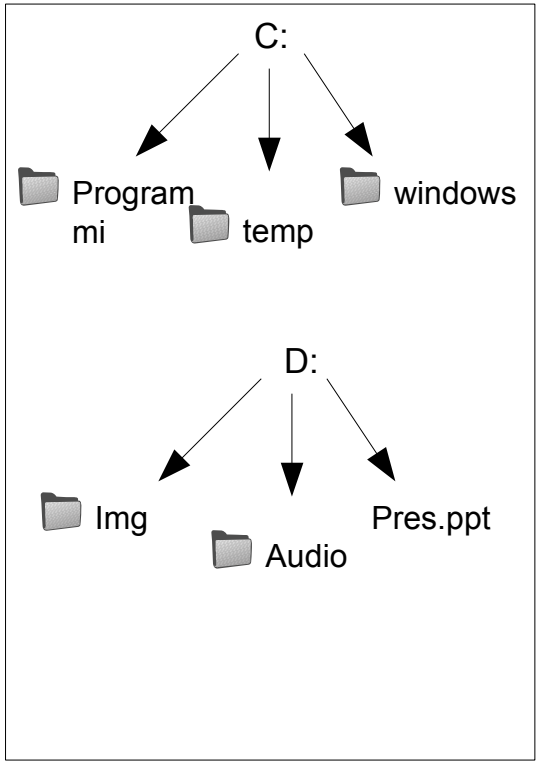
Da A. Tanenbaum, “Modern Operating Systems”, 2ª Edizione

Directory

- Astrazione che permette di organizzare i files
  - A un solo livello (una sola “directory” per tutti i files)
  - Gerarchico





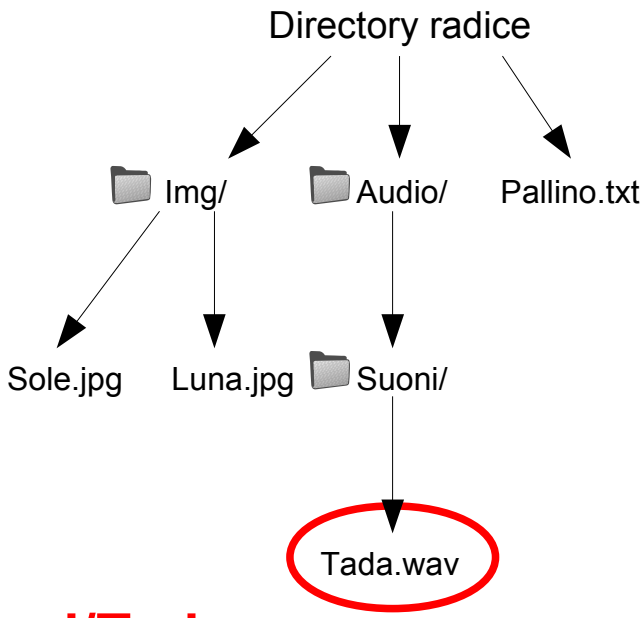


Operazioni su una directory

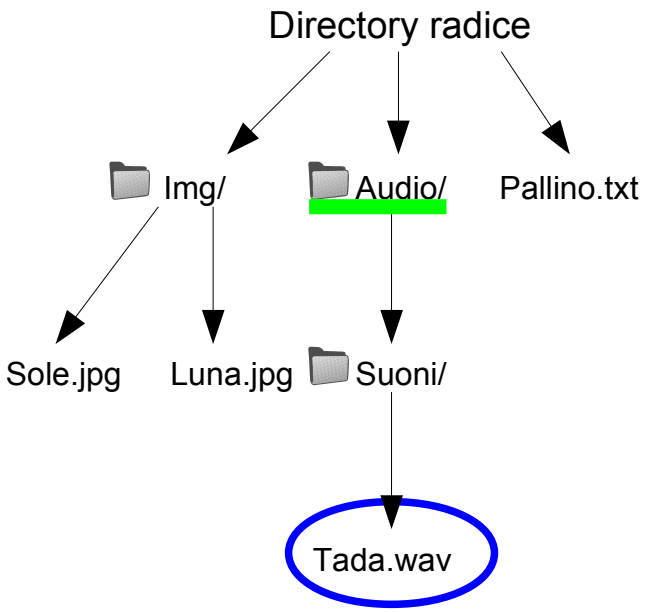
- Crea (Create)
- Elimina (Delete)
- Apri (Opendir)
- Chiudi (Closedir)
- Leggi contenuto (Readdir)
- Rinomina (Rename)
- Crea collegamento (Link)
- Rimuovi collegamento (Unlink)

Percorso

- In una struttura gerarchica è possibile specificare il file tramite
  - **Percorso assoluto**
  - **Percorso relativo** (alla directory corrente)

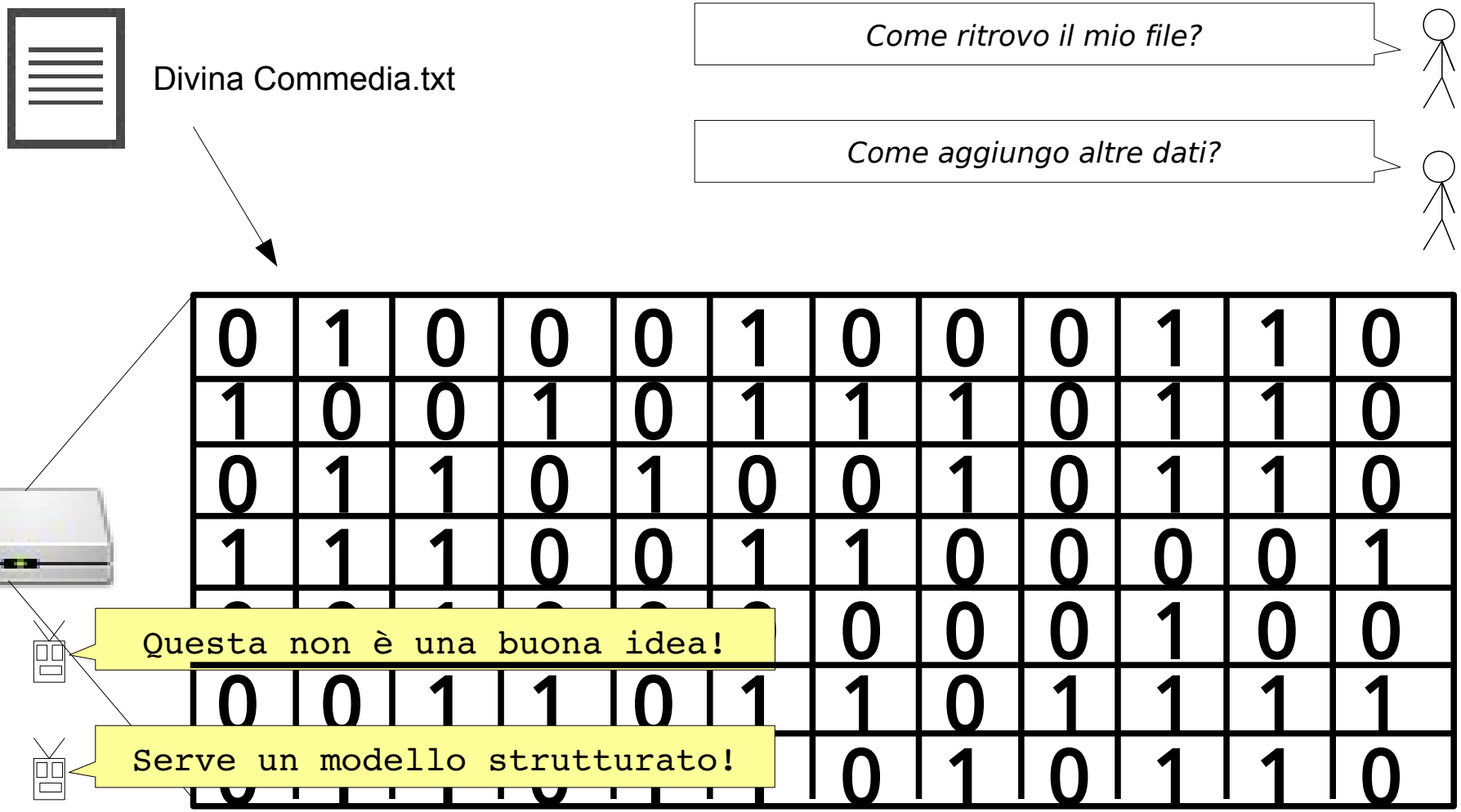


/Audio/Suoni/Tada.wav

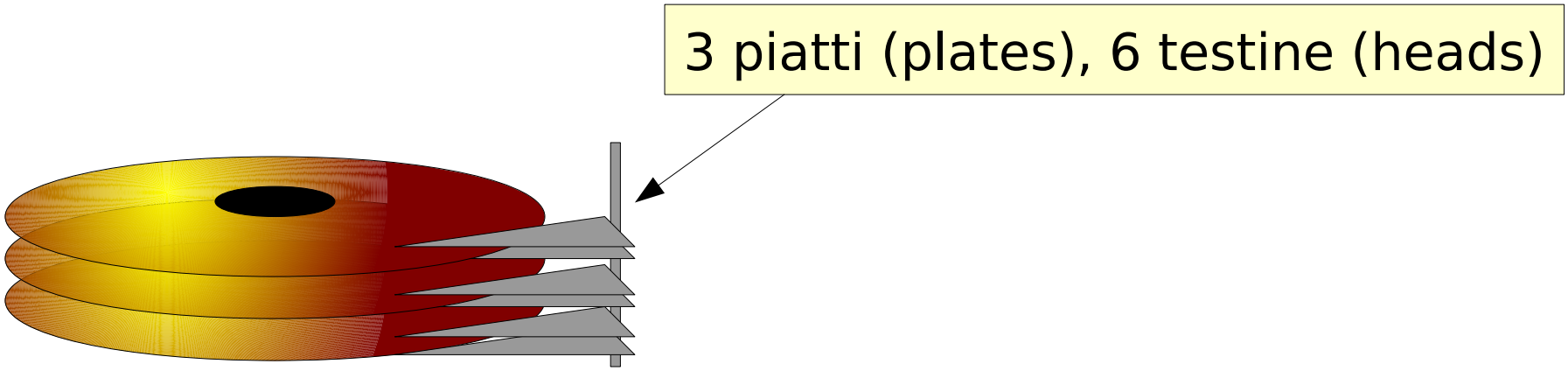


Suoni/Tada.wav

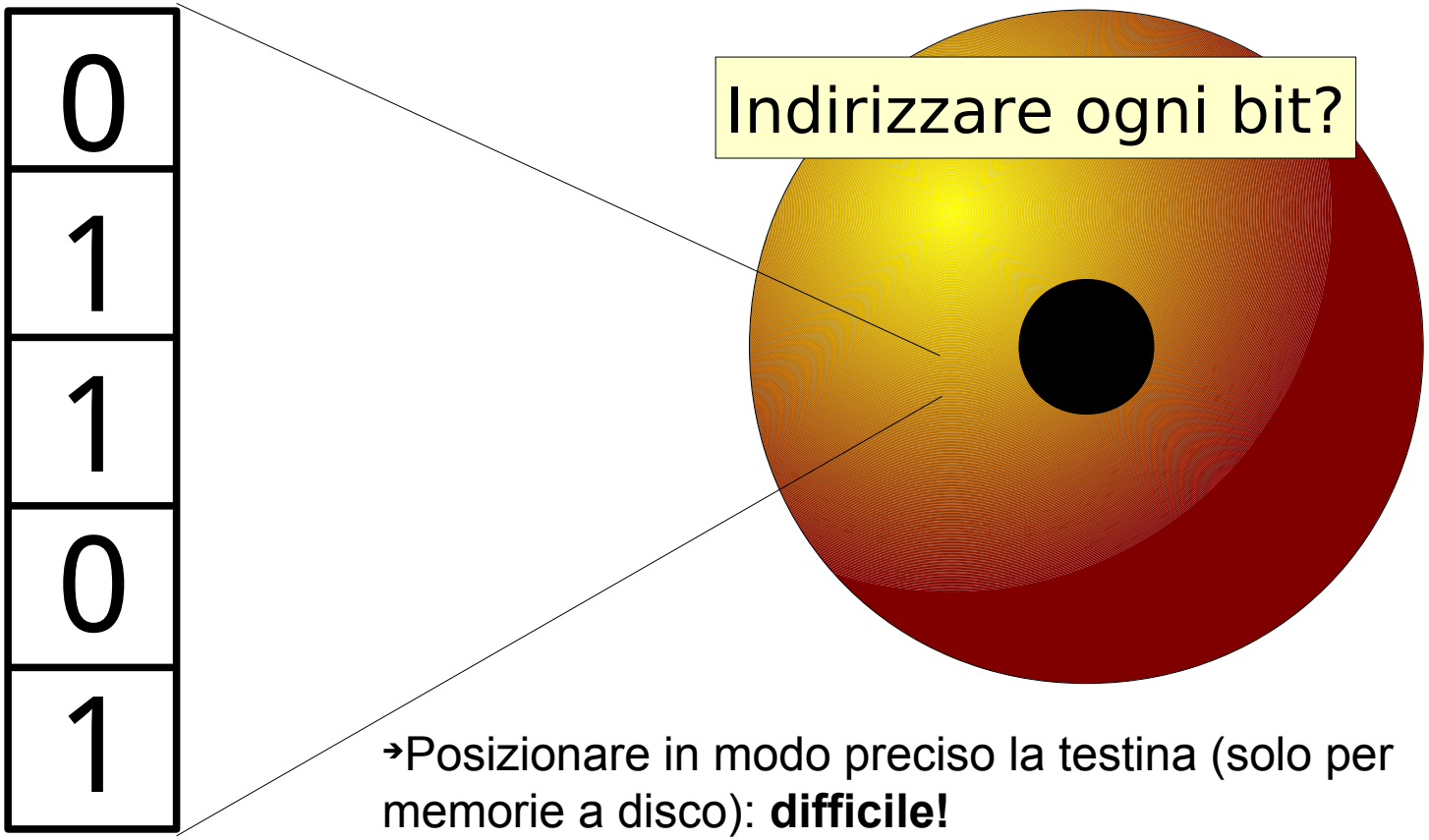
Memorizzare i files... come sequenza di bit sul disco?



Disco rigido

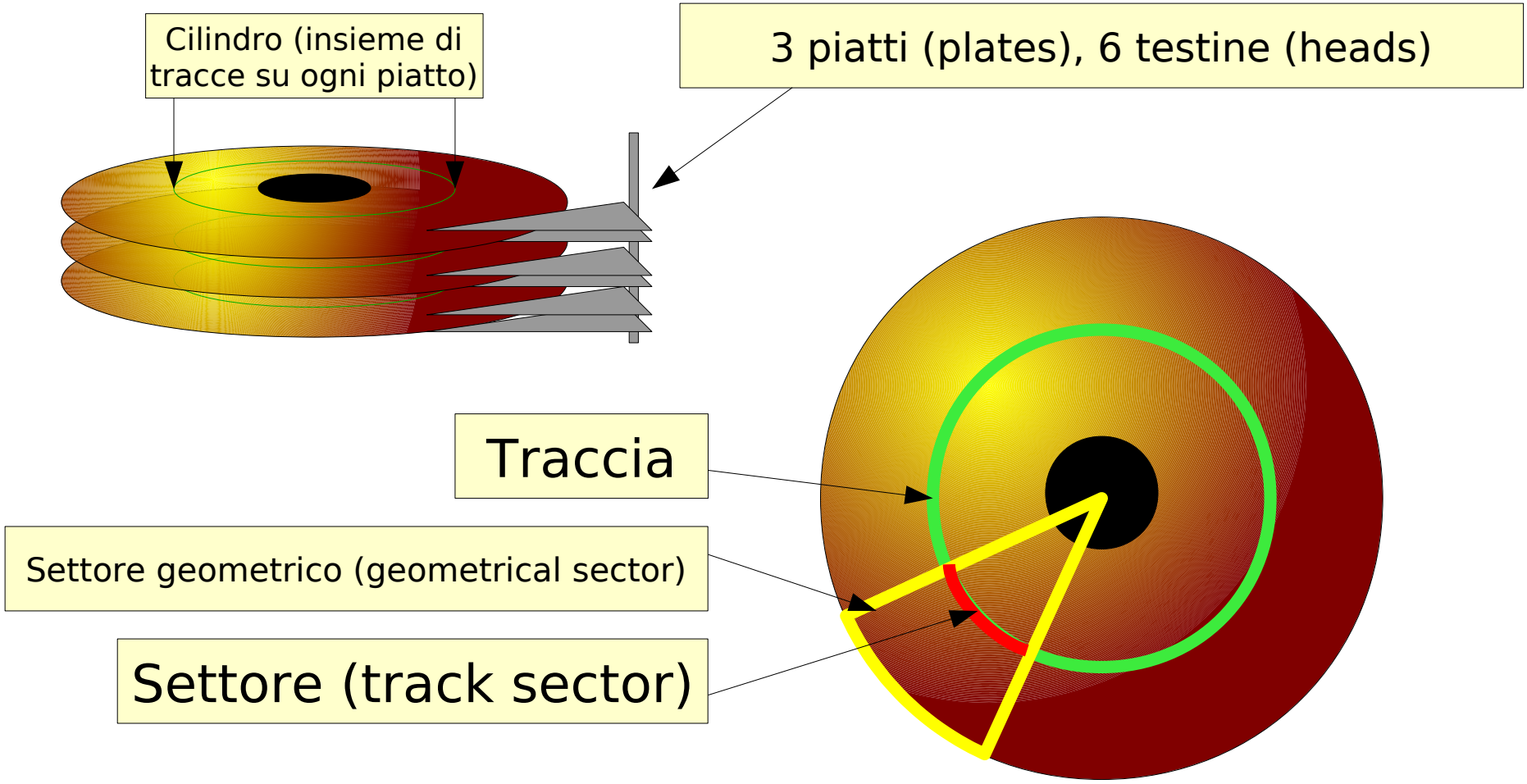


Indirizzamento sul disco (1)



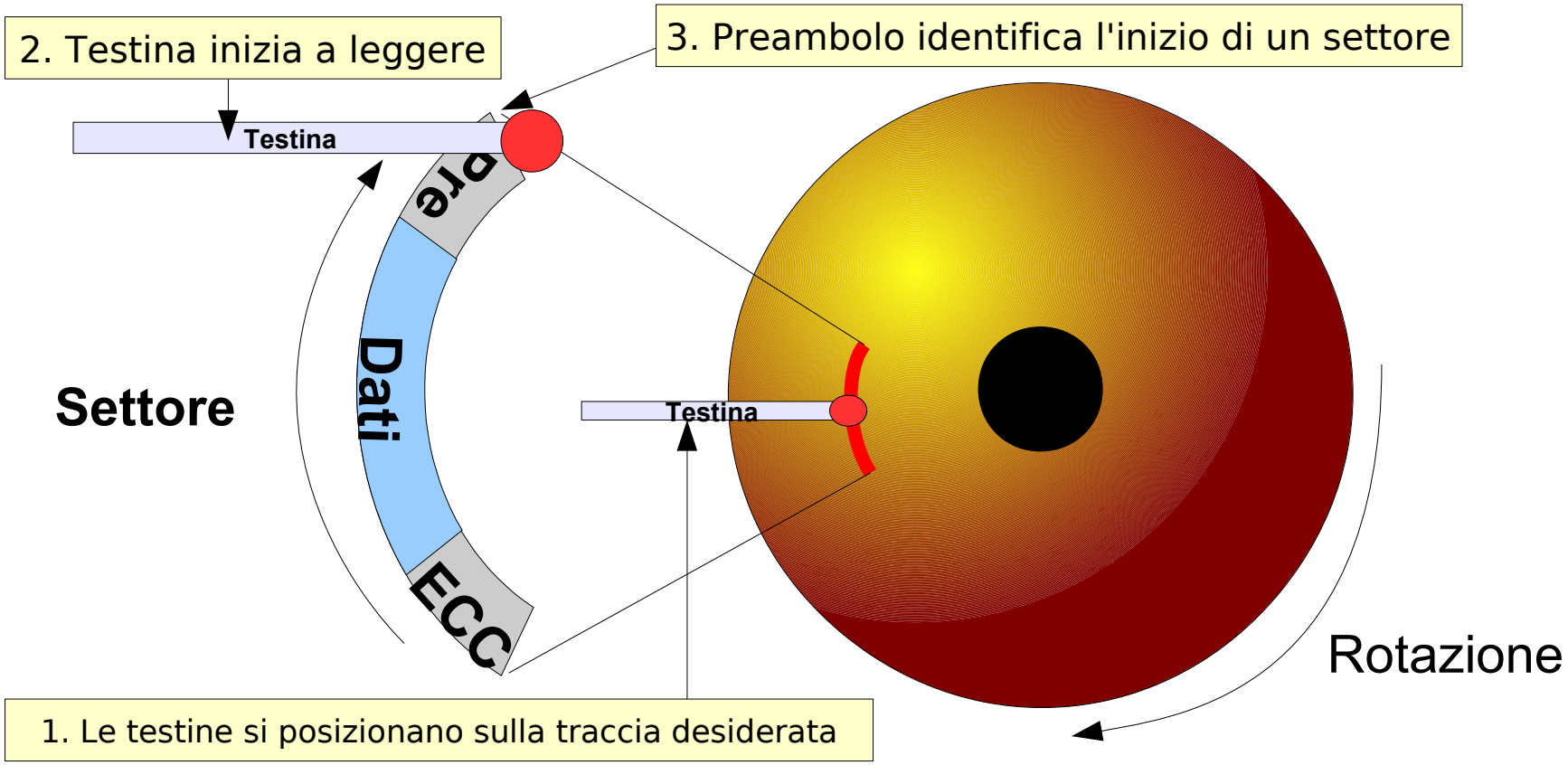
- Posizionare in modo preciso la testina (solo per memorie a disco): **difficile!**
- La maggior parte delle operazioni vengono effettuate su più bit/byte: **inefficiente!**
- Richiede un ampio spazio di indirizzi: **dimensione limitata dello spazio di archiviazione!**

Indirizzamento sul disco (2)

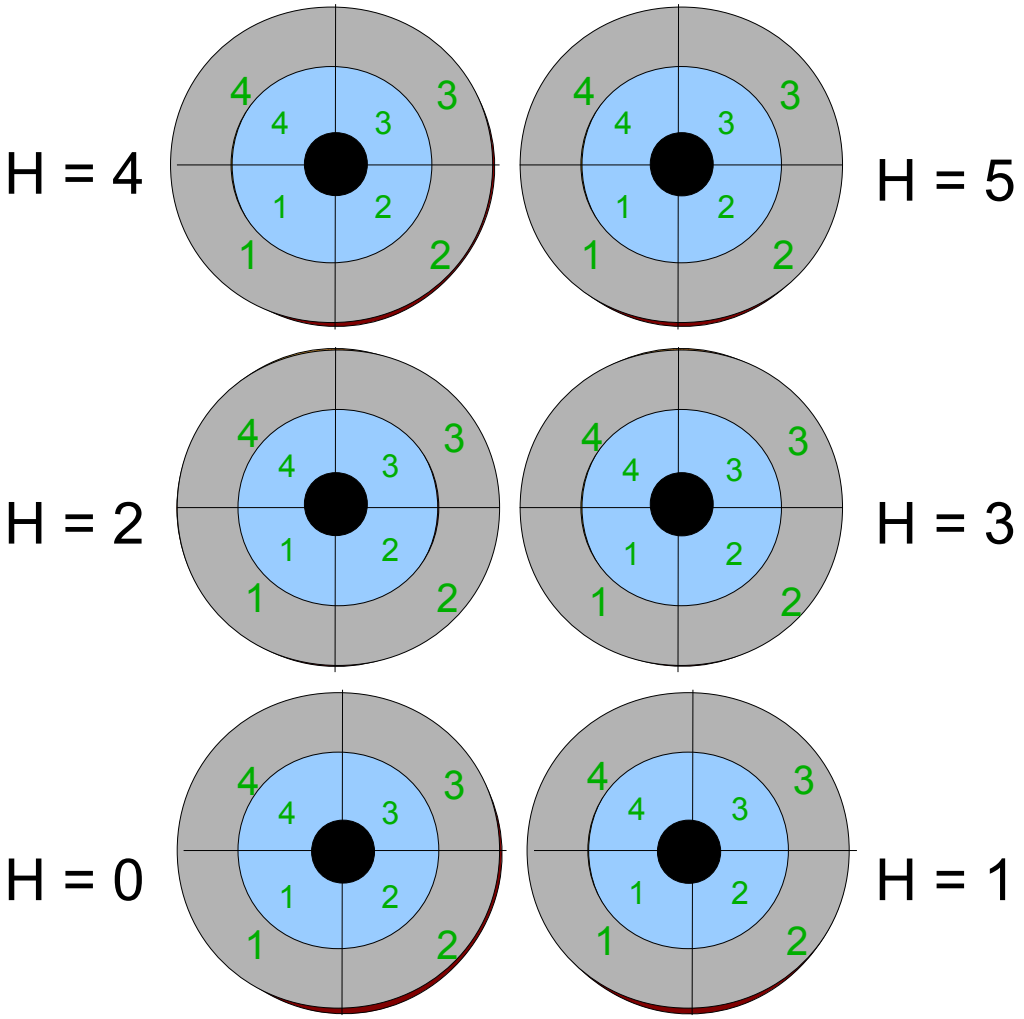


Lettura di un settore

Settore: più piccola unità indirizzabile sul disco, più piccola unità di allocazione (tipicamente 512B o 4096B)
Cluster: insieme di unità di allocazione

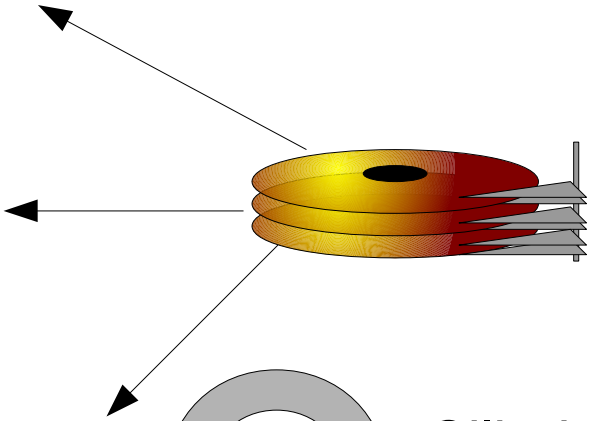


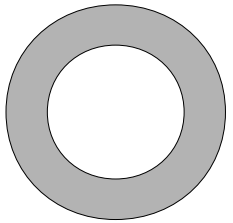
Indirizzamento CHS (Cylinder-head-sector)



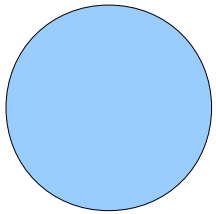
Geometria:

6 testine per cilindro
2 cilindri
4 settori per traccia





Cilindro 0



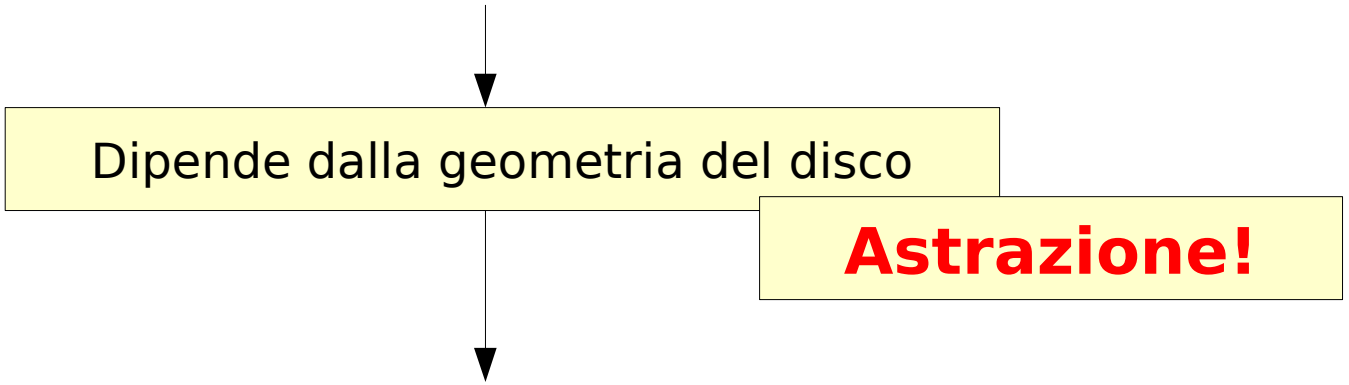
Cilindro 1

Attenzione! I settori sono numerati a partire da 1

CHS → LBA

Indirizzamento CHS (Cylinder-head-sector)

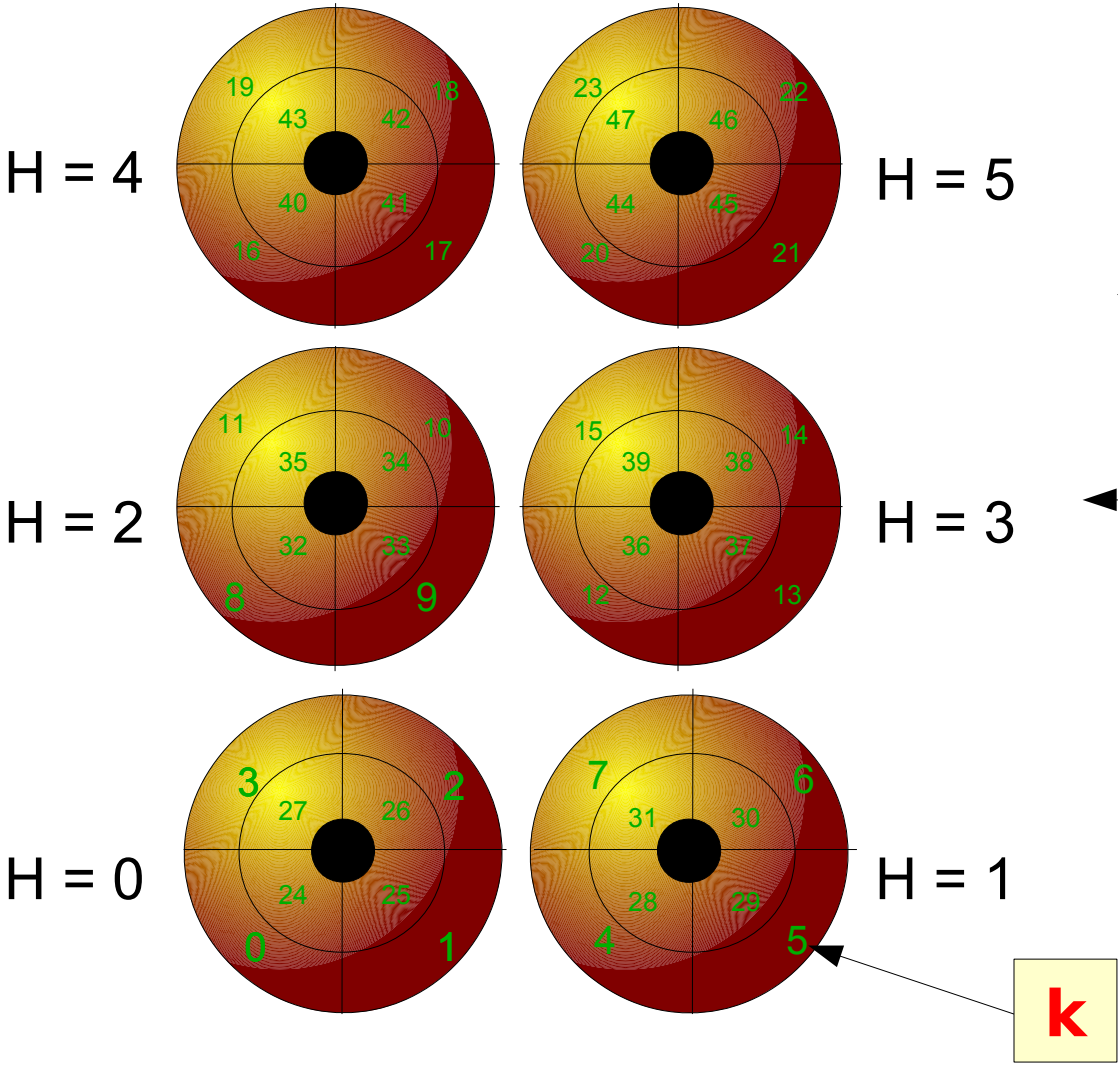
ogni settore è univocamente indirizzato da (C,H,S)



Indirizzamento LBA (Logical Block Address):

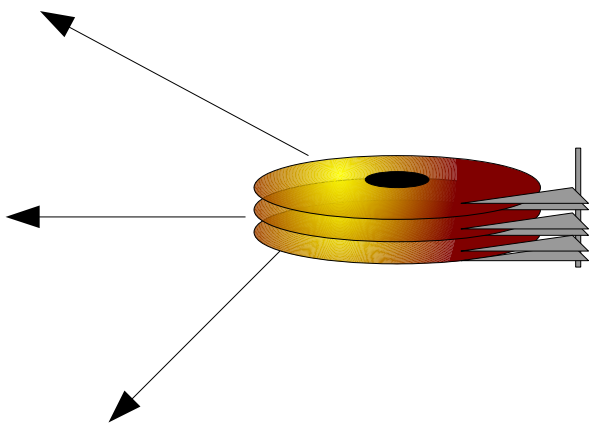
i settori sono numerati sequenzialmente, il controller del disco si occupa di convertire LBA in CHS

Indirizzamento LBA (Logical Block Address)



Geometria:

6 testine per cilindro
2 cilindri
4 settori per traccia



CHS → LBA

$$k_{LBA} = \underbrace{\mathbf{C} * \mathit{Settori}_{Cilindro}}_{\text{Settori per cilindri completi}} + \underbrace{\mathbf{H} * \text{Settori}_{Traccia}}_{\text{Settori per tracce complete nel cilindro corrente}} + \underbrace{\mathbf{S} - 1}_{\text{Settori rimanenti (offset nell'ultima traccia)}}$$

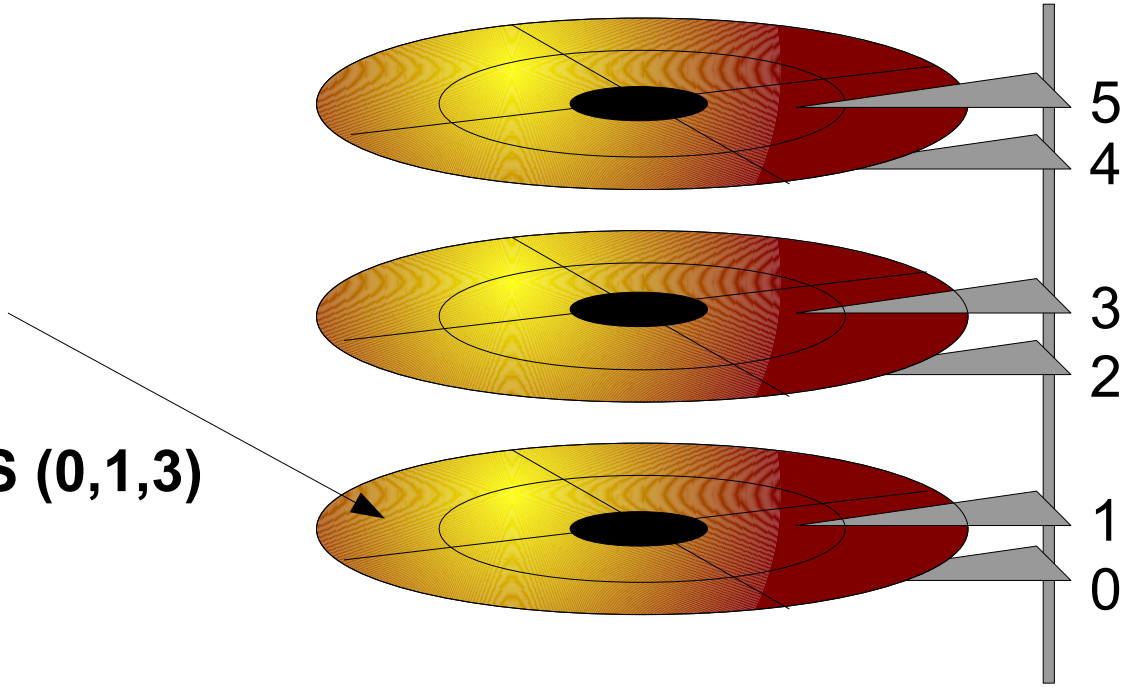
$$\underbrace{C * N_{testine} * \text{Settori}_{Traccia}}_{\mathbf{C} * \mathit{Settori}_{Cilindro}}$$

$$k_{LBA} = ((\mathbf{C} * N_{testine} + \mathbf{H}) * \text{Settori}_{Traccia}) + \mathbf{S} - 1$$

Esempio

Geometria:

6 testine per cilindro
2 cilindri
4 settori per traccia



Calcolare LBA per CHS (0,1,3)

LBA = C * 24 + H * 4 + S – 1 =

= 0 * 24 + 1 * 4 + 3 – 1 = 6

LBA → CHS

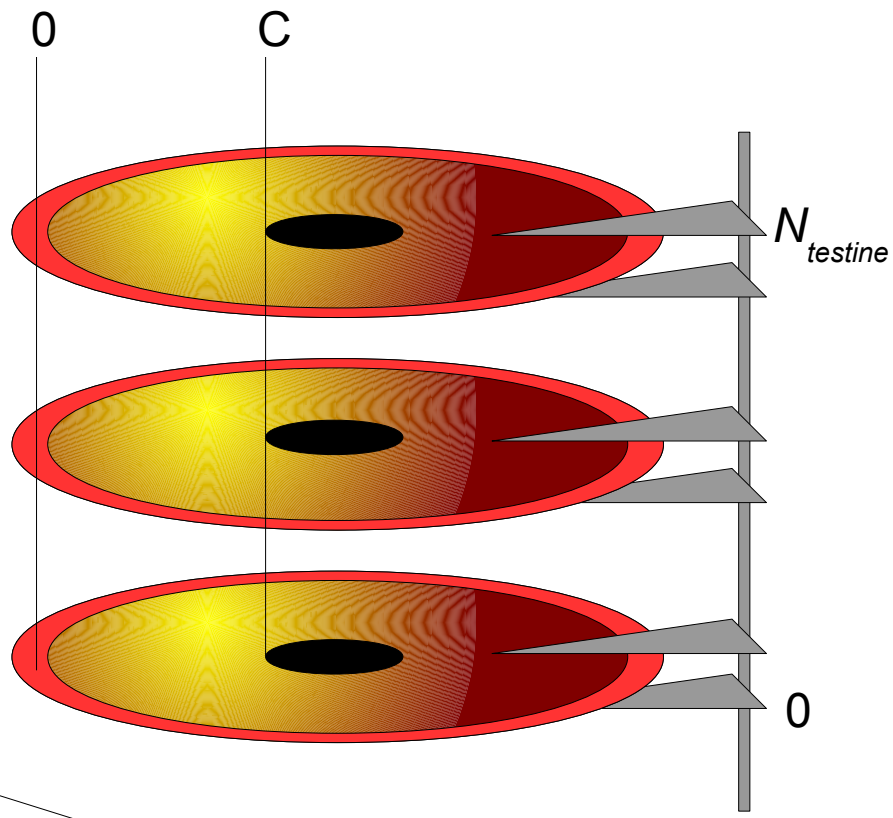
$Settori_{Cilindro} = (N_{testine} * Settori_{Traccia})$

Cylinder = LBA / $Settori_{Cilindro}$

R = LBA % $Settori_{Cilindro}$

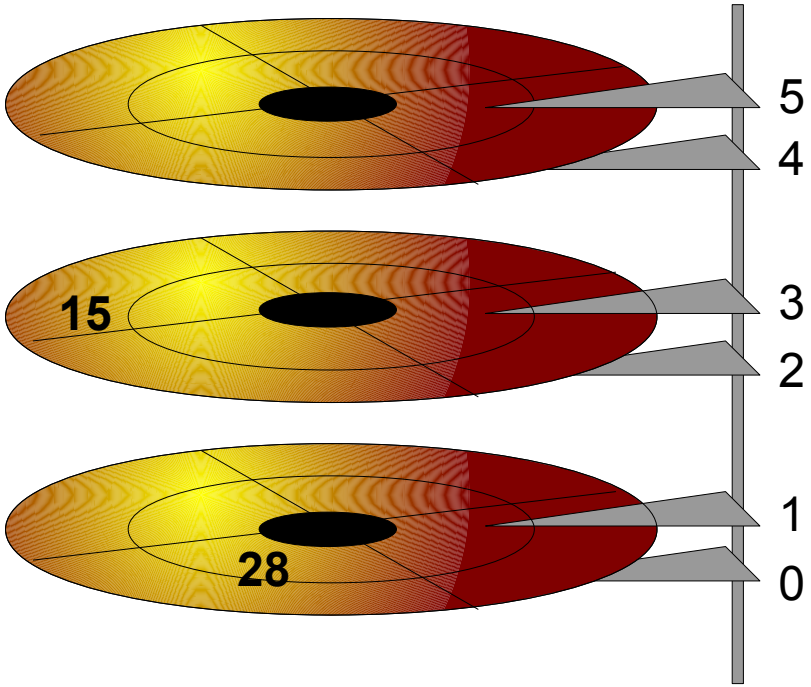
Head = R / $Settori_{Traccia}$

Sector = R % $Settori_{Traccia}$ +1



+ 1 perché in LBA i blocchi sono numerati a partire da 0!

Esempio



Geometria:

6 testine per cilindro
2 cilindri
4 settori per traccia

Calcolare CHS per blocco LBA k = 15

C = LBA / *Settori*$_{Cilindro}$ = 15 / (6 * 4) = 0
R = LBA % Settori$_{Cilindro}$ = 15 % 24 = 15
H = R / Settori$_{Traccia}$ = 15 / 4 = 3
S = R % Settori$_{Traccia}$ +1 = 15 % 4 + 1 = 4

Calcolare CHS per blocco LBA k = 28

C = LBA / *Settori*$_{Cilindro}$ = 28 / (6 * 4) = 1
R = LBA % Settori$_{Cilindro}$ = 28 % 24 = 4
H = R / Settori$_{Traccia}$ = 1 / 4 = 1
S = R % Settori$_{Traccia}$ +1 = 4 % 4 + 1 = 1

Riassumendo

Un disco/memoria di massa è...

Una periferica che gestisce una sequenza lineare di blocchi di dimensione fissa e permette due operazioni:

Leggi blocco k

Scrivi blocco k

Per questo si parla di **periferica a blocchi *(block devices)*** *

* in contrapposizione alle periferiche a carattere – *char devices* – in cui i dati sono letti in maniera sequenziale

Blocchi fisici / logici

Blocco fisico

l'unità di memorizzazione più piccola supportata dalla periferica (es. settore di un disco rigido). Tipicamente 512 bytes (o 4096 sui nuovi dischi)

≠

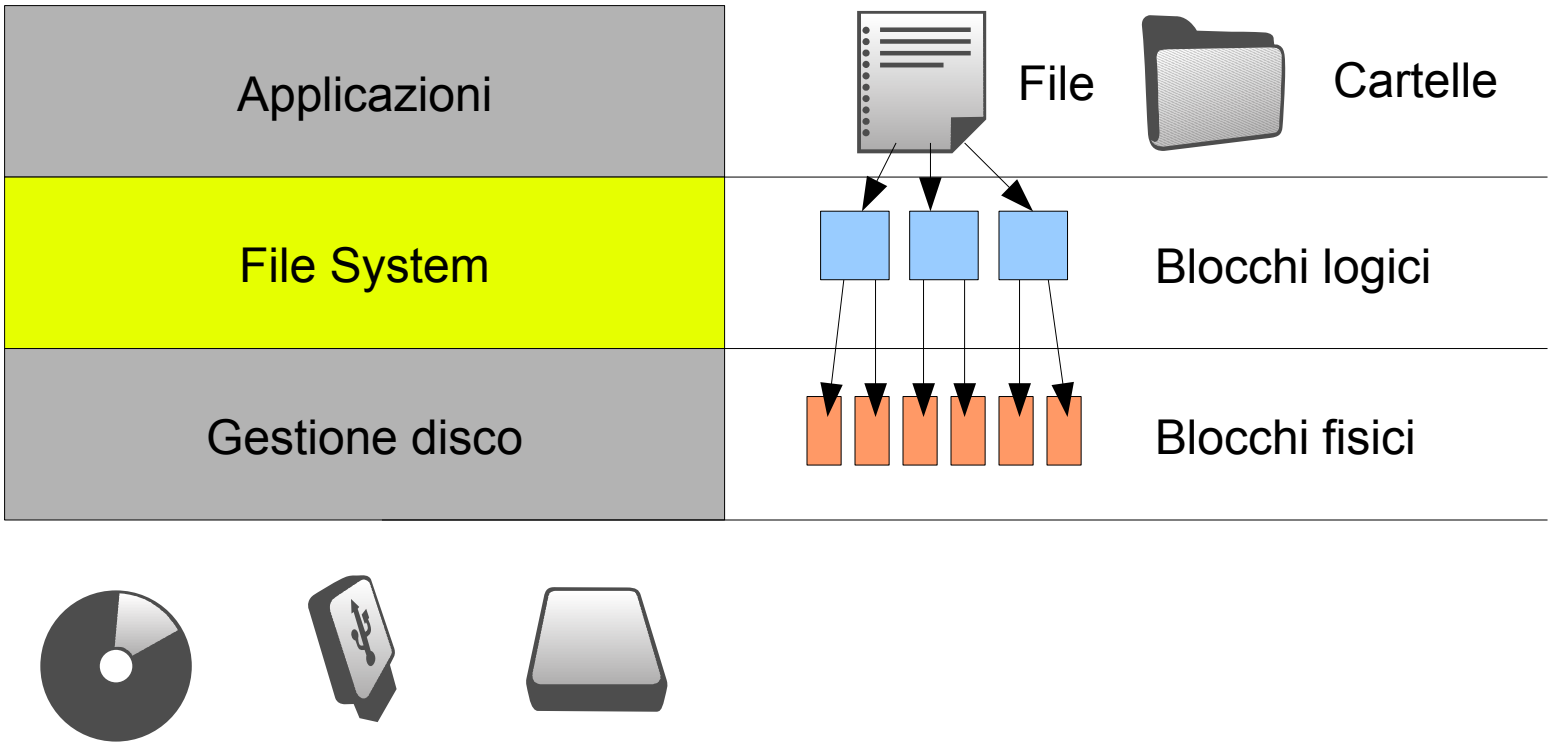
Blocco logico

o *cluster*, l'unità di memorizzazione più piccola supportata dal file system (può essere un multiplo della dimensione di un blocco fisico). Da 512 bytes a 64 Kbytes.

Partizioni

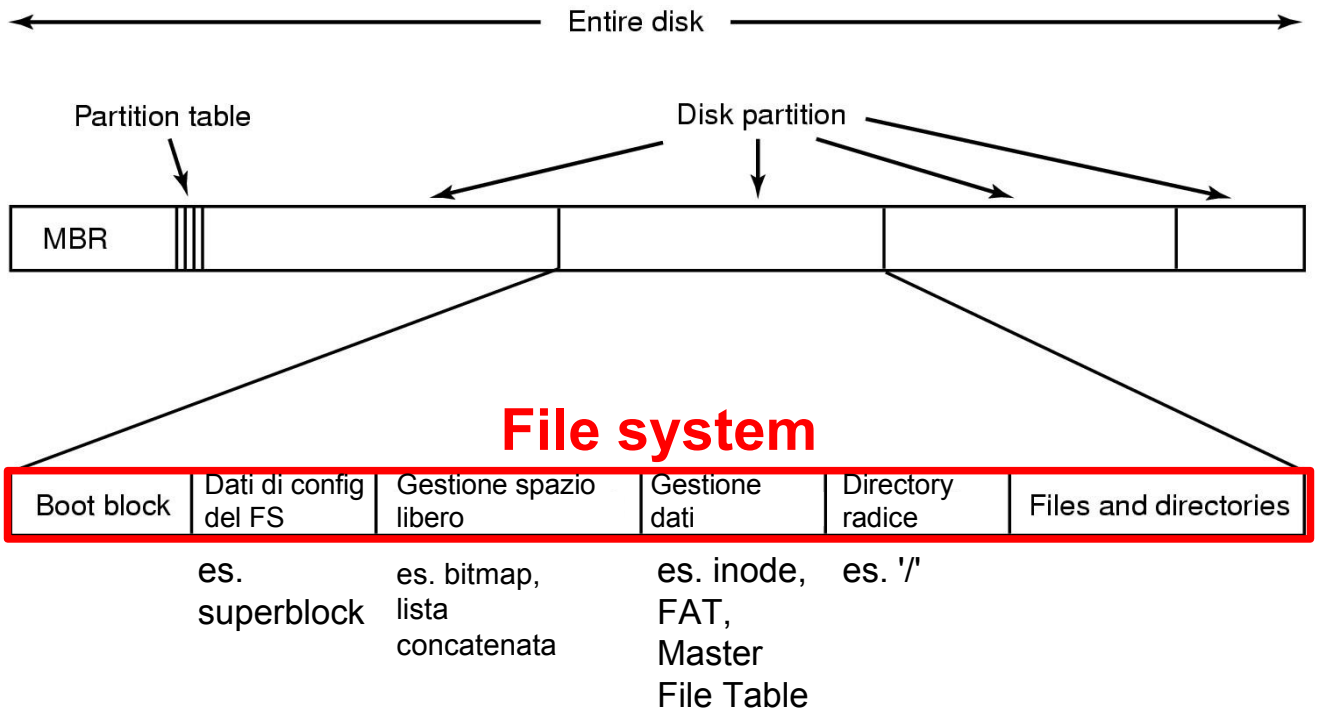
- Un disco può essere diviso in più partizioni, ognuna con il suo filesystem
  - Suddividere i dati
  - Installare più sistemi operativi

Livelli e astrazioni

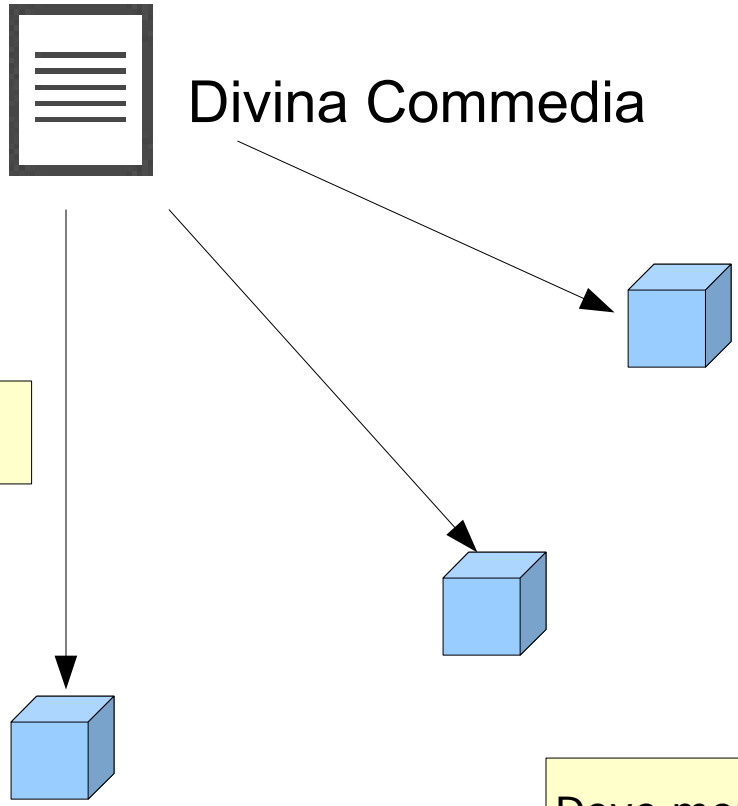


File system e dischi

- Il file system è memorizzato sul disco
  - Se il disco ha più partizioni, ognuna ha il suo file system
  - Il primo settore del disco è detto MBR (Master Boot Record) e contiene la tabella delle partizioni



Suddivisione in blocchi... qualche problematica

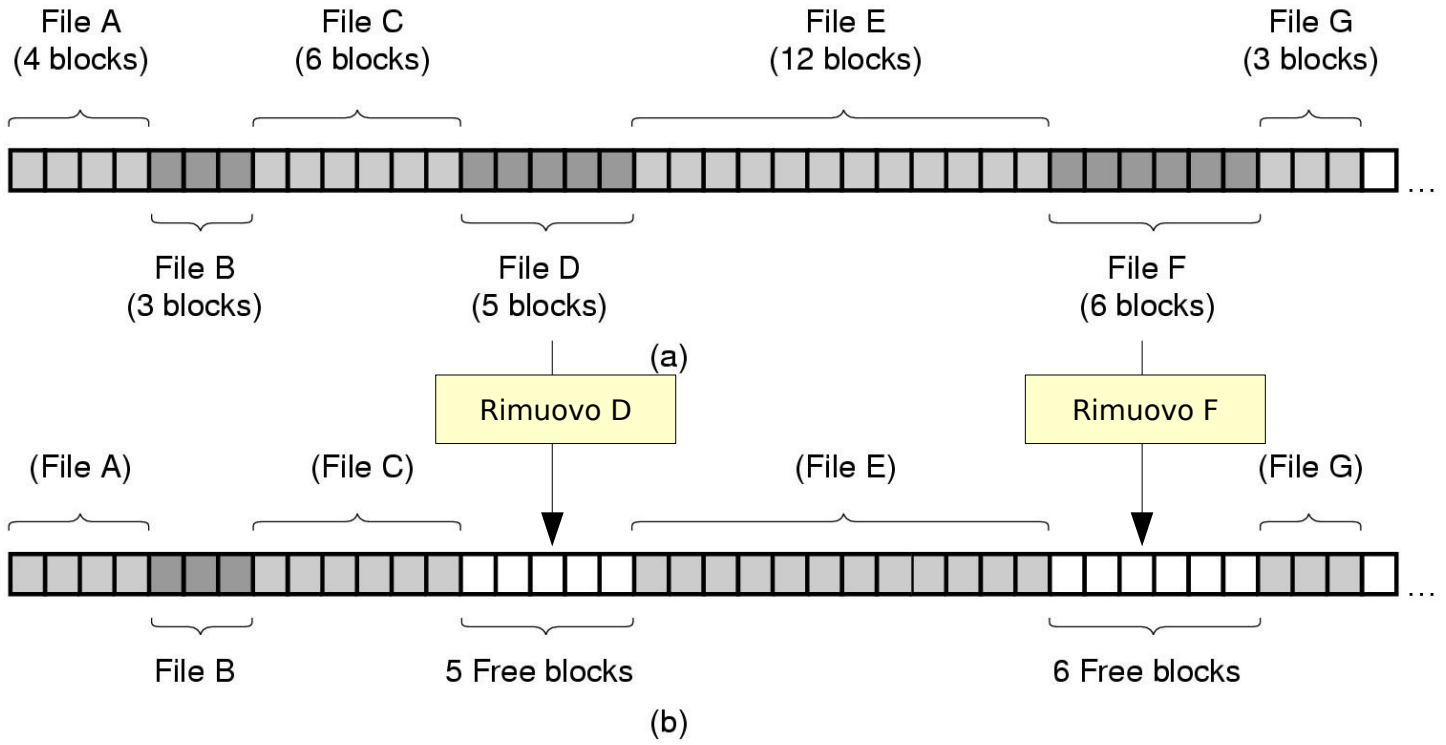


Come organizzare i blocchi?

Dove memorizzare gli attributi?

Come ritrovo i blocchi?

1. Allocazione contigua dei file



+ Semplice

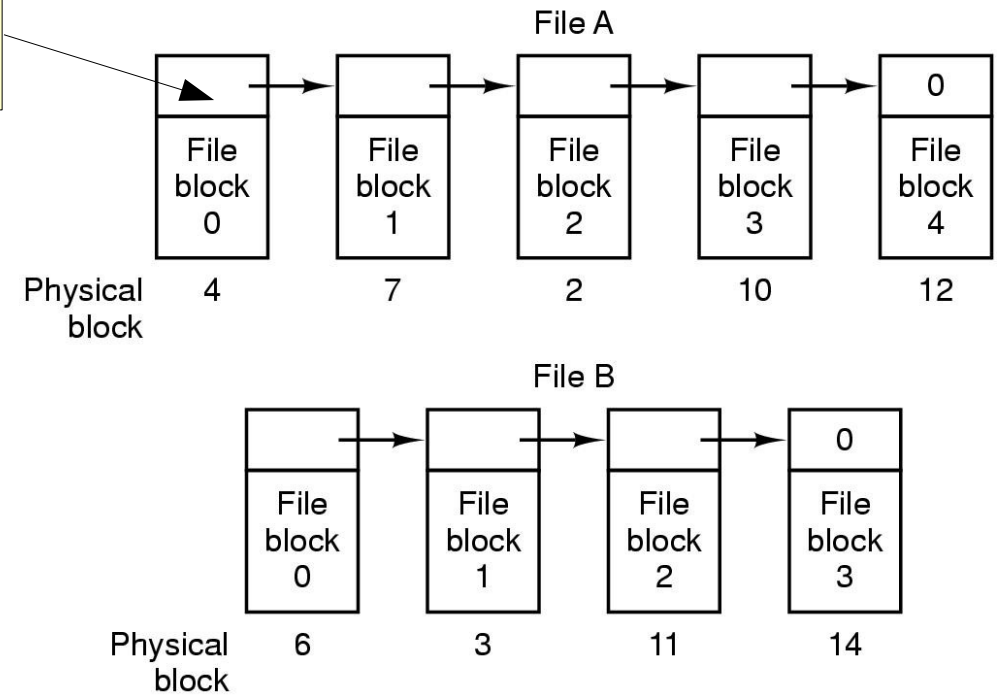
+ Veloce (file possono essere letti in una sola operazione)

- Frammentazione (non è un problema se supporto è a sola lettura)

Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

2. Allocazione con lista concatenata

All'inizio di ogni blocco puntatore al successivo



+ Nessuna struttura esterna (basta puntatore al primo blocco della radice)

- Lettura sequenziale lenta, lettura causale molto lenta

- Non affidabile (se un blocco non è più leggibile, seguenti sono persi)

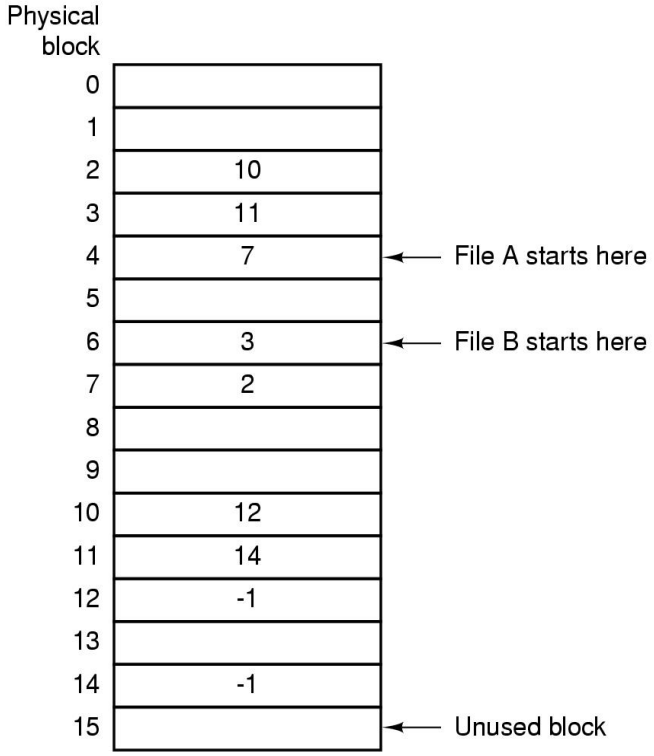
- Spreco di spazio in ogni blocco

Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

3. Allocazione con lista concatenata e tabella in memoria

Il puntatore per ogni blocco è salvato in memoria

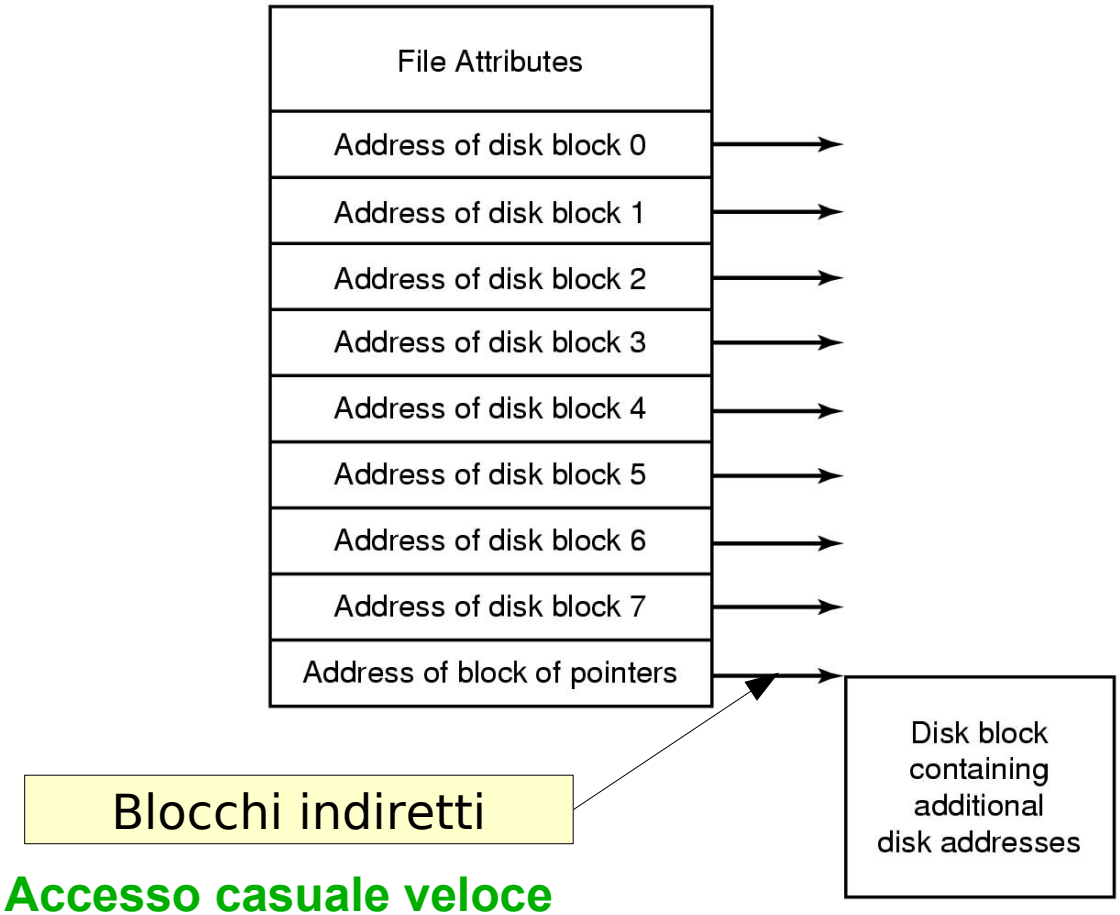
FAT (File Allocation Table)



- Accesso lento se la tabella non è in memoria
- Occupazione di memoria (non adatto per dischi di grandi dimensioni)

Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

4. Inode (index node)

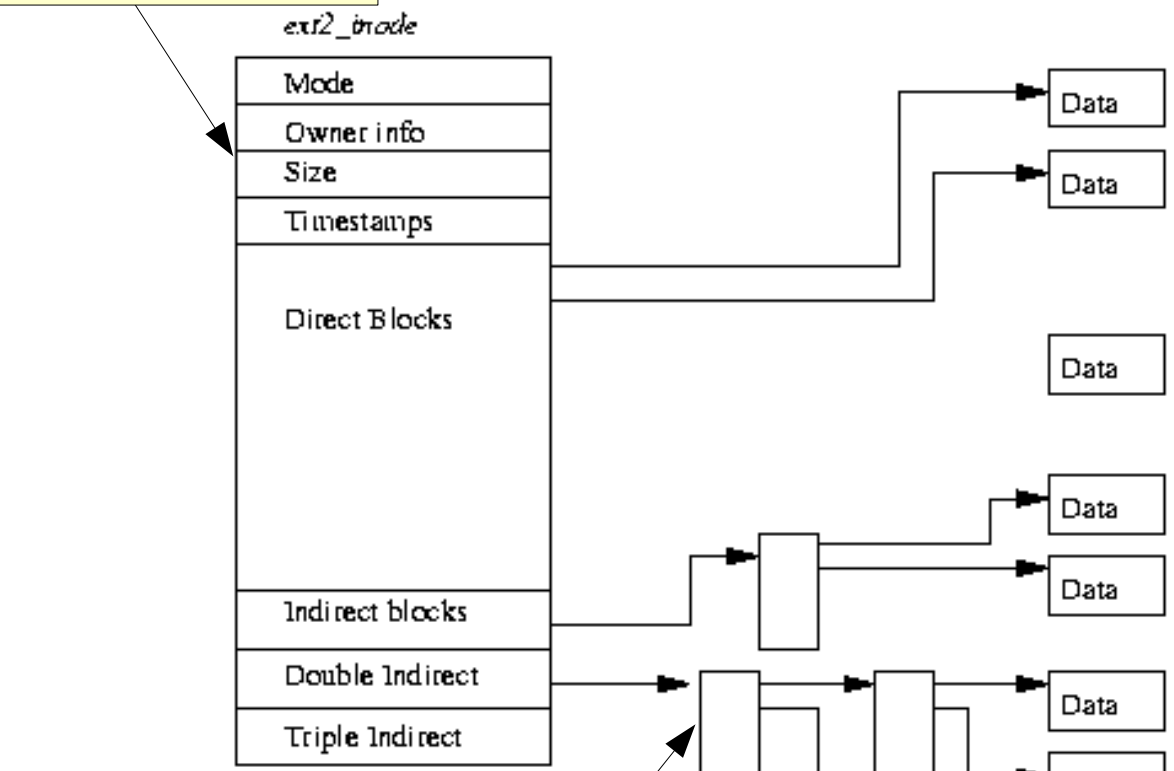


- **Accesso casuale veloce**
- **Supporta facilmente file con dimensioni crescenti**
- **Se abbiamo tanti blocchi dobbiamo usare indirezione**

Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

Esempio: Inode ext2 (Linux)

Il nome è salvato nella directory!



I blocchi indiretti sono salvati insieme agli altri dati

Da “The Linux Kernel”, © 1996-1999 David A Rusling

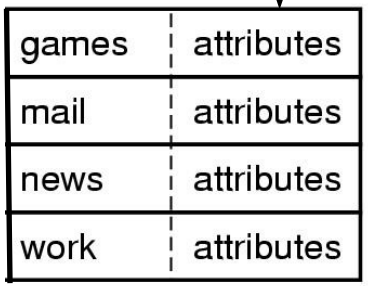
Implementazione delle directory

- Le directory sono dei file speciali
  - Contengono la lista dei file e delle sotto-cartelle

Implementazione delle directory

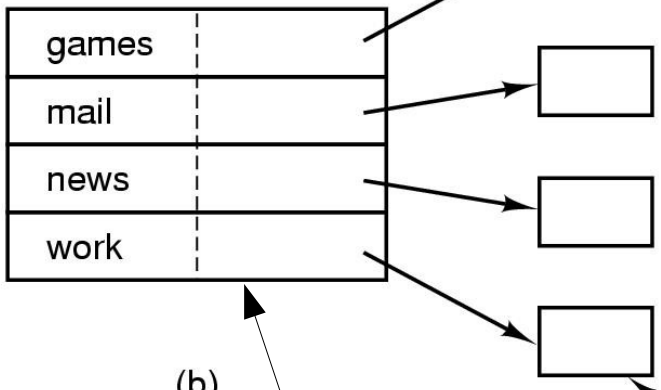
Come associare un nome a un i-node ?

Gli attributi includono gli indirizzi dei blocchi dati o un puntatore ad un elemento di una FAT



(a)

FAT



(b)

UNIX (es. ext2)

Data structure containing the attributes

Una directory associa dei nomi con gli inode corrispondenti

Da A. Tanenbaum, "Modern Operating Systems", 2ª Edizione

Link

Quindi posso avere un file in due directory che fa riferimento allo stesso i-node?

- Un link (collegamento) è un file speciale che fa riferimento a un altro file
- Hard link vs Soft link
  - **Hard link:** più file puntano allo stesso i-node (e quindi agli stessi dati su disco)
  - **Soft link (link simbolico):** è un file speciale che contiene il percorso di un'altro file
- Windows (FAT)
  - Solo soft-link
- Unix
  - Soft link

[X] bash

```
utente@host:~/Documenti/Privato$ ln -s fileOrigine linkSimbolico
```

  - Hard link

[X] bash

```
utente@host:~/Documenti/Privato$ ln fileOrigine linkSimbolico
```

Hard vs Soft Link

- Il comando `ls -l` mostra i collegamenti simbolici e il numero di file che puntano allo stesso i-node

```
[X] bash

utente@host:~/Documenti$ ls -al
drwxr-xr-x  2 utente gruppo  4096 2011-09-02 10:18 .
drwx------ 36 utente gruppo 12288 2011-09-02 09:46 ..
drwx------  3 utente gruppo  4096 2011-09-01 10:24 Private
-rw-r--r--  2 utente gruppo 83443 2011-09-02 10:18 Spese.odt
-rw-r--r--  2 utente gruppo 83443 2011-09-02 10:18 Costi.odt
-rw-r--r--  1 utente gruppo 96432 2011-07-15 18:12 Vacanze.ppt
-rw-r--r--  1 utente gruppo  5674 2010-01-03 22:01 Mappa.jpg
-rw-rw-rw-  1 utente gruppo  8    2010-01-07 12:23 Mappa2.jpg -> Mappa.jpg
```

`ln Spese.odt Costi.odt`

`ln -s Mappa.jpg Mappa2.jpg`

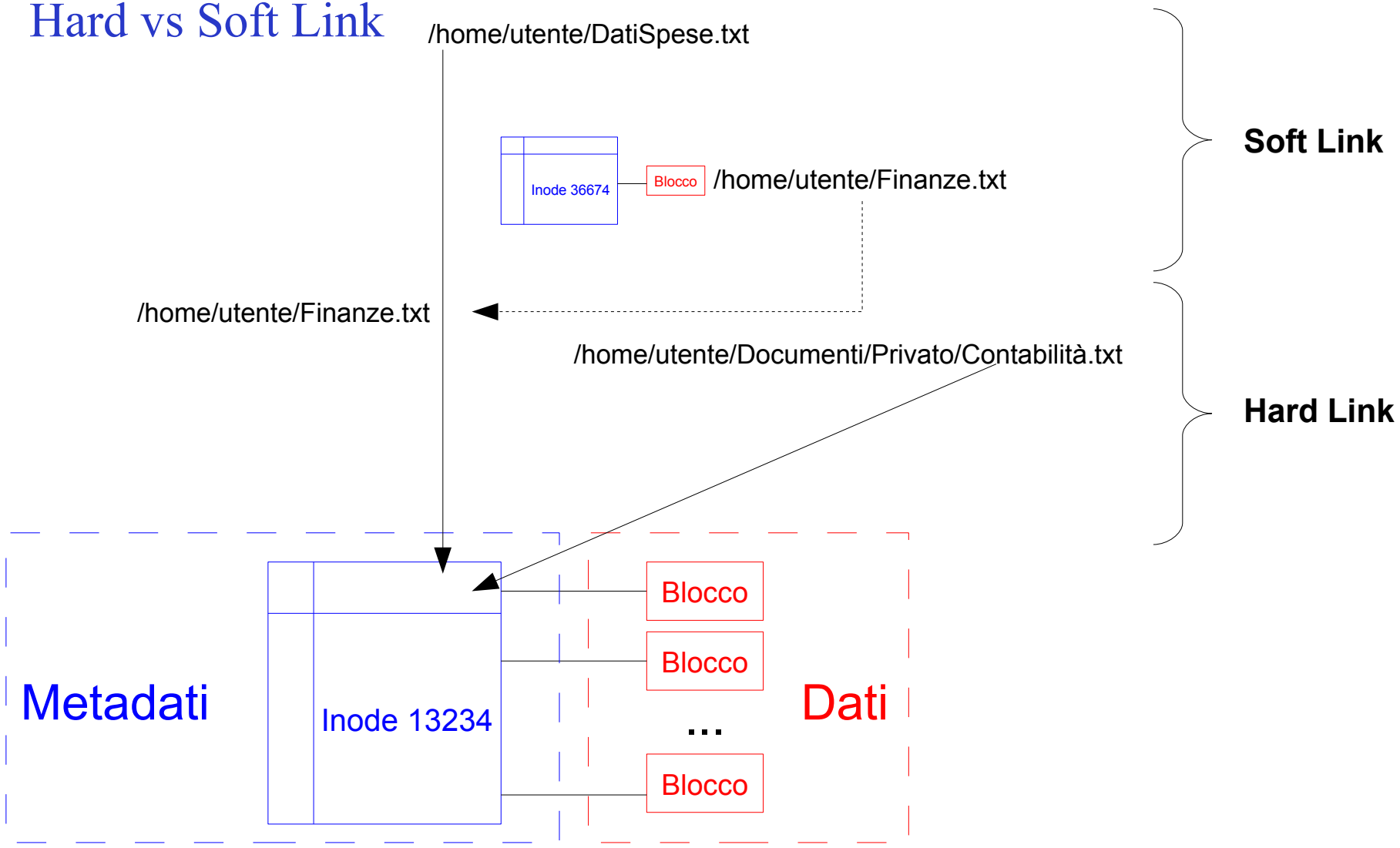
Perché un hard link ha le stesse proprietà del file originale mentre il soft link può essere diverso?

Le proprietà sono memorizzate nell'i-node, e gli hard link puntano agli stessi i-node. I soft-link hanno i-node diversi.

Perché non posso avere hard link a una directory?

Perché la gerarchia non sarebbe più un grafo aciclico e le operazioni ricorsive non funzionerebbero correttamente

Hard vs Soft Link



Link count

[X] bash

```
utente@host:~/Documenti$ ls -al
drwxr-xr-x  2 utente gruppo  4096 2011-09-02 10:18 .
drwx------ 36 utente gruppo 12288 2011-09-02 09:46 ..
drwx------  3 utente gruppo  4096 2011-09-01 10:24 Private
-rw-r--r--  2 utente gruppo 83443 2011-09-02 10:18 Spese.odt
-rw-r--r--  2 utente gruppo 83443 2011-09-02 10:18 Costi.odt
-rw-r--r--  1 utente gruppo 96432 2011-07-15 18:12 Vacanze.ppt
-rw-r--r--  1 utente gruppo  5674 2010-01-03 22:01 Mappa.jpg
-rw-rw-rw-  1 utente gruppo  8    2010-01-07 12:23 Mappa2.jpg -> Mappa.jpg
```

`ln Spese.odt Costi.odt`

`ln -s Mappa.jpg Mappa2.jpg`

La colonna che indica la dimensione (numero di elementi) di una directory, è usata per indicare il numero di file che puntano allo stesso i-node

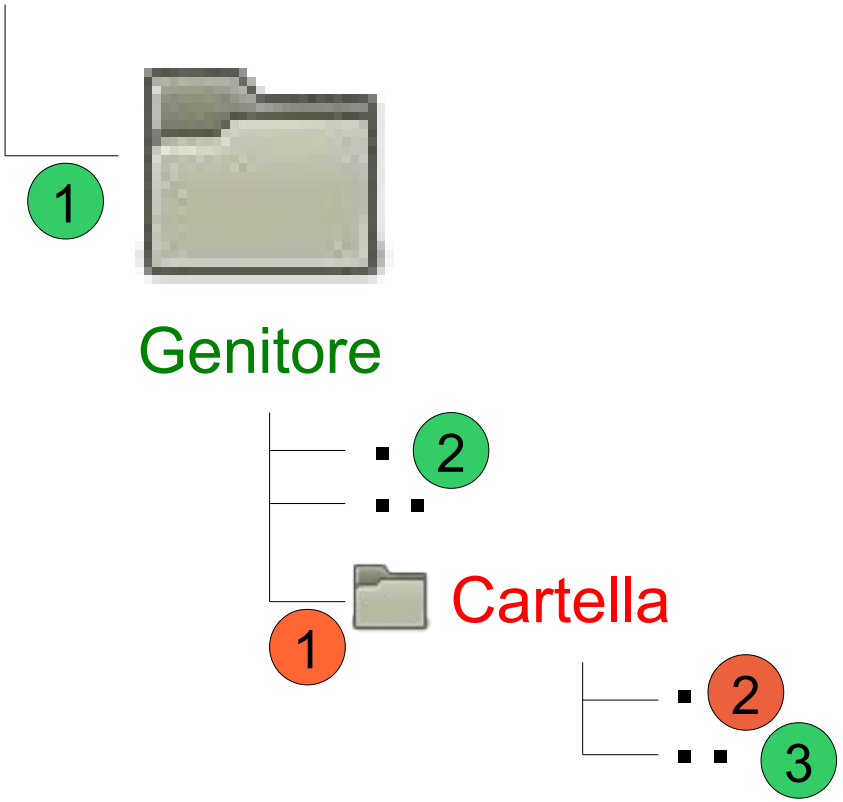
Qual'è l'i-node di un file?

`ls -i`

Quali file sono collegati con l'inode 123456 ?

`find . -inum 123456`

Perché per ogni directory il numero di link è sempre almeno 2

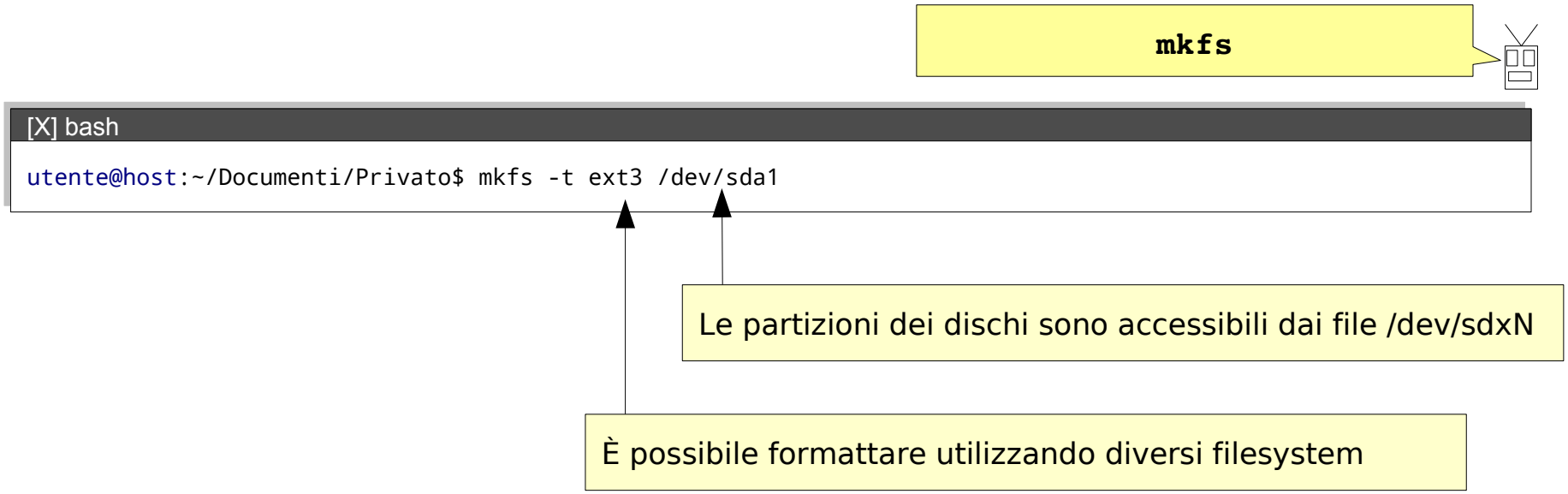


Semantica dei link

- Hard e soft link si comportano come i rispettivi file di origine per quanto riguarda le operazioni di lettura e scrittura
- Eccezione: rimozione (con `rm`)
  - **Hard-link**
    - Il numero di link nell'i-node viene decrementato. Quando raggiunge 0 l'i-node e i blocchi vengono cancellati dal disco
  - **Soft-link**
    - Cancella il link simbolico, il file a cui il link faceva riferimento non viene toccato

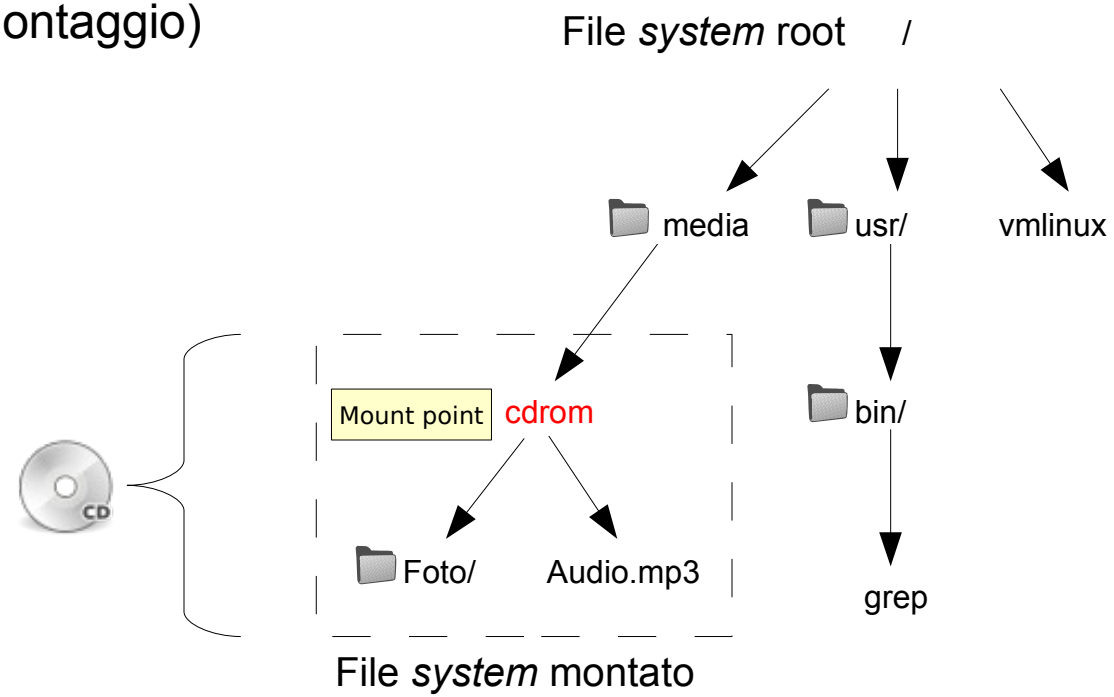
Formattazione di un filesystem

- Per poter memorizzare dei file su un supporto dati è necessario formattare il supporto
  - Crea la struttura di base del filesystem (cf. Slide 28)



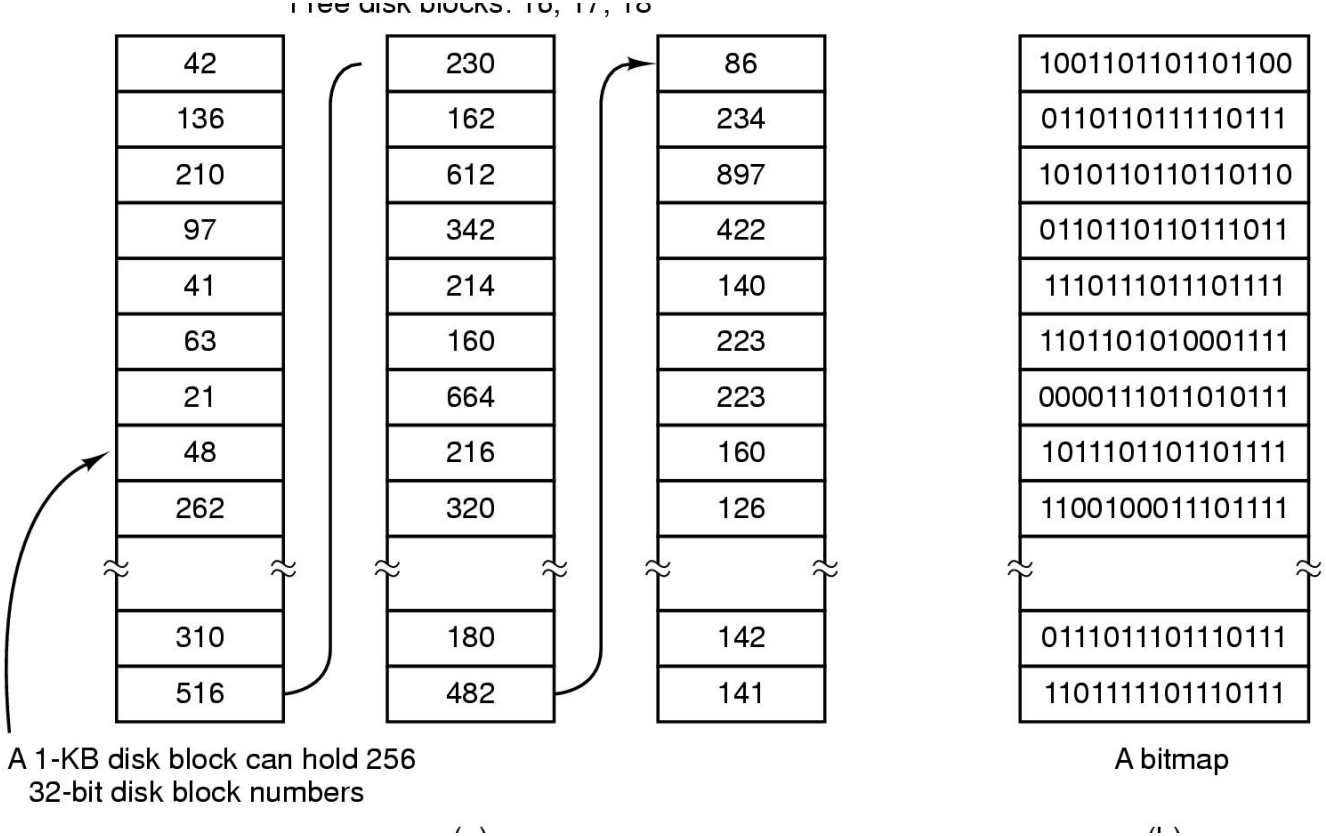
Montaggio di file system Unix

- Prima di poter accedere ai file archiviati in un filesystem è necessario “montarlo”
- Il montaggio (**mount**) di un file system consiste nel collegare una nuova gerarchia di directory, tipicamente di un altro file system alla gerarchia principale (file system root)
- La directory dove viene aggiunto il nuovo file system è detto **mount point** (punto di montaggio)



Spazio libero

Quali sono i blocchi disponibili?



[X] bash

```
utente@host:~/Documenti/Privato$ df
```

Quale filesystem utilizzare?

- Diverse possibilità:
  - Linux: ext2, ext3, ext4, jfs, XFS, Reiser, btrfs,...
  - Windows: FAT, NTFS
  - Altri: ISO9660, UDF,...
- Scelta dipende da:
  - Tipo di applicazione
    - Utente desktop
    - Web server
    - Server multimedia
    - Database
    - WORM (Write Once, Read Many)
  - Compatibilità
  - Limitazioni del filesystem
    - Dimensione massima del supporto
    - Dimensione massima del file
    - Case-sensitive
    - Spazio su disco