

Università di Roma Tor Vergata  
Corso di Laurea triennale in Informatica  
**Sistemi operativi e reti**  
A.A. 2020-2021

Pietro Frasca

## Lezione 20

Martedì 15-12-2020

# Scheduling del disco

- In un sistema multiprogrammato con molti processi attivi, la coda del disco spesso può avere diverse richieste in sospeso. Così, quando una richiesta è completata, il firmware del HDD sceglie quale richiesta in attesa conviene servire prima. Per tale scelta si ricorre a diversi algoritmi di scheduling del disco.
- Esamineremo tre noti algoritmi di pianificazione del disco: FCFS, SSTF e SCAN.

## Scheduling in ordine di arrivo - FCFS

- L'algoritmo più semplice di scheduling del disco è, naturalmente, **FCFS (First Came First Served)**, basato sull'ordine di arrivo. Questo algoritmo ha un comportamento equo, ma in genere non fornisce il servizio più veloce.
- Consideriamo, per esempio, una coda di richieste per un disco con relative a settori sui seguenti cilindri e in questo ordine: 14, 40, 23, 47, 7.

- Supponiamo, inoltre, e che le testine siano correntemente posizionate sul cilindro 20.
- Tipicamente, con FCFS, il percorso che la testine compiono ha una forma altalenante. In questo caso ciascuna testina esegue un percorso totale, misurato in numero di cilindri attraversati, di 113 cilindri.
- Questa pianificazione è schematizzata nella figura seguente

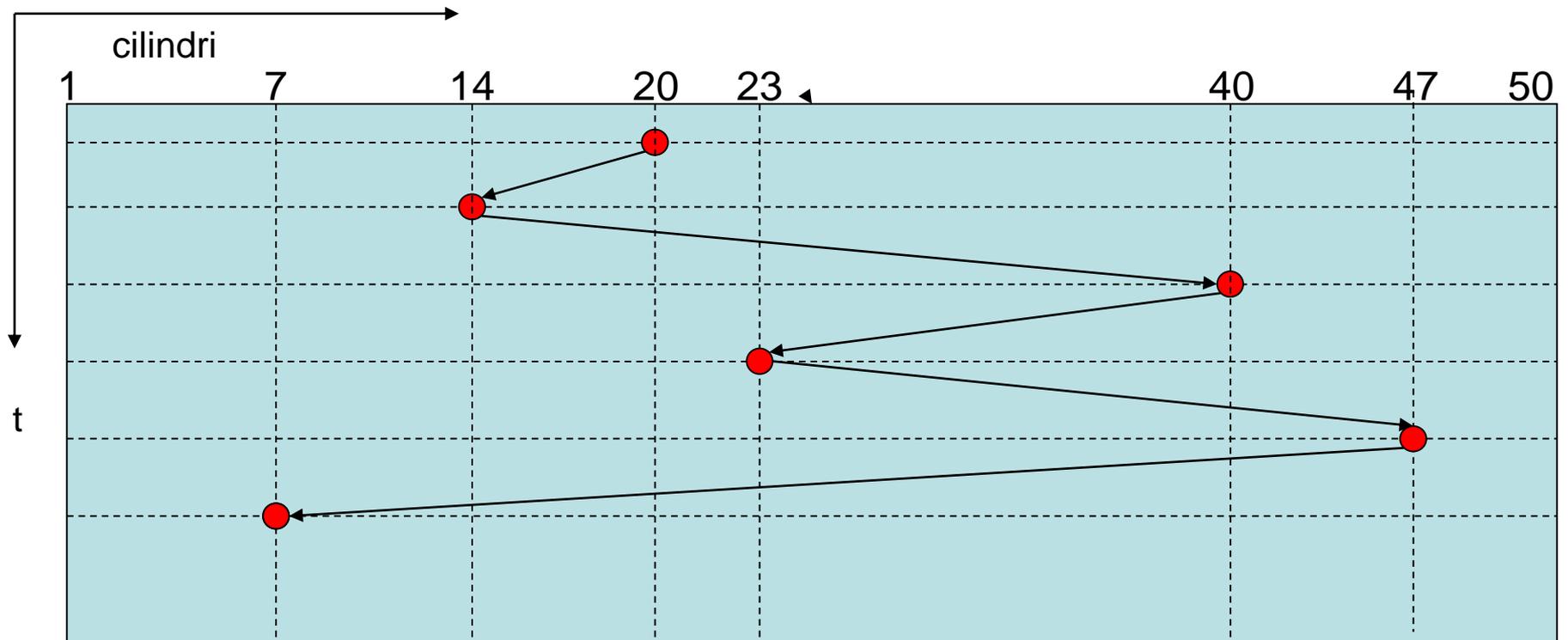
- **Sceduling in ordine di arrivo - FCFS**

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 14, 40, 23, 47, 7

Percorso: 113 cilindri



## Scheduling per brevità - SSFT

- L'algoritmo **SSTF (Short Seek Time First)** si basa sulla ragionevole evidenza che è più conveniente servire tutte le richieste vicine alla posizione attuale della testina prima di spostarla lontano per servire altre richieste.
- Pertanto, SSTF sceglie la richiesta in sospeso relativa al cilindro più vicino al cilindro sui cui è posizionata attualmente la testina.
- Per il nostro esempio, questo metodo di pianificazione determina un movimento totale della testina di soli 59 cilindri, poco più della metà della distanza percorsa da FCFS.
- Chiaramente, questo algoritmo fornisce un sostanziale miglioramento delle prestazioni.
- SSTF è essenzialmente una forma di scheduling per brevità, come l'algoritmo SJF (Short Job First) descritto nello scheduling della CPU. Come SJF, anche SSFT può causare problemi di attesa indefinita (starvation).

- Infatti, se sono presenti in coda molte richieste per accedere a cilindri tra loro vicini e una richiesta distante da questi la richiesta del cilindro lontano rischia di non essere soddisfatta per un tempo eccessivo. Questo scenario diventa sempre più probabile se la coda delle richieste si allunga.

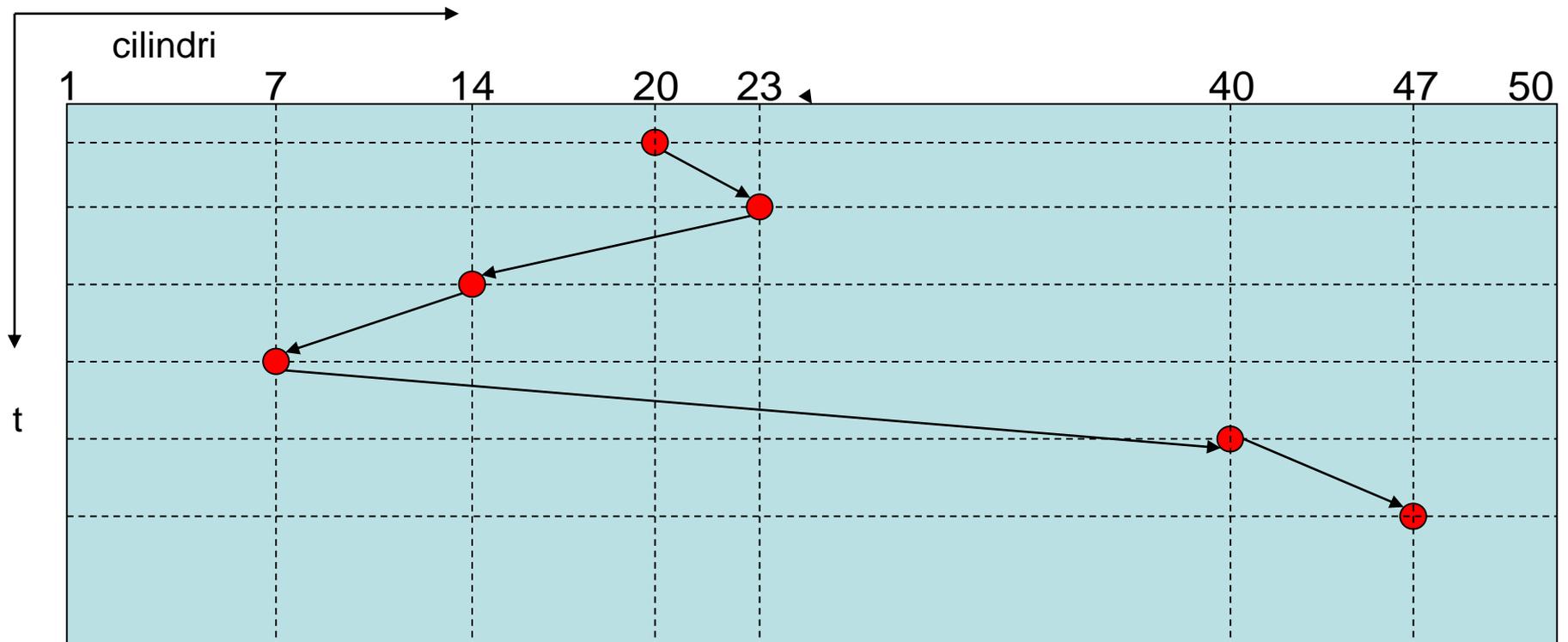
- **Sceduling per brevità – SSTF (Shortest Seek Time First)**

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 23, 14, 7, 40, 47

Percorso: 59 cilindri



# Scheduling per scansione - SCAN

- Con l'algoritmo SCAN, il braccio del disco si muove partendo da un estremità del disco all'altra servendo le richieste durante l'attraversamento dei cilindri.
- Raggiunto il cilindro con numerazione estrema, il braccio è mosso nella direzione opposta. La testina si muove continuamente avanti e indietro attraverso il disco.
- Per questo funzionamento, l'algoritmo SCAN è spesso chiamato algoritmo dell'ascensore, in quanto il braccio del disco si comporta come un ascensore con prenotazione in un edificio.
- Per il nostro esempio il percorso con SCAN è mostrato nella figura seguente. Il percorso è di 53 cilindri.
- Questo algoritmo elimina la possibilità che si verifichi starvation.

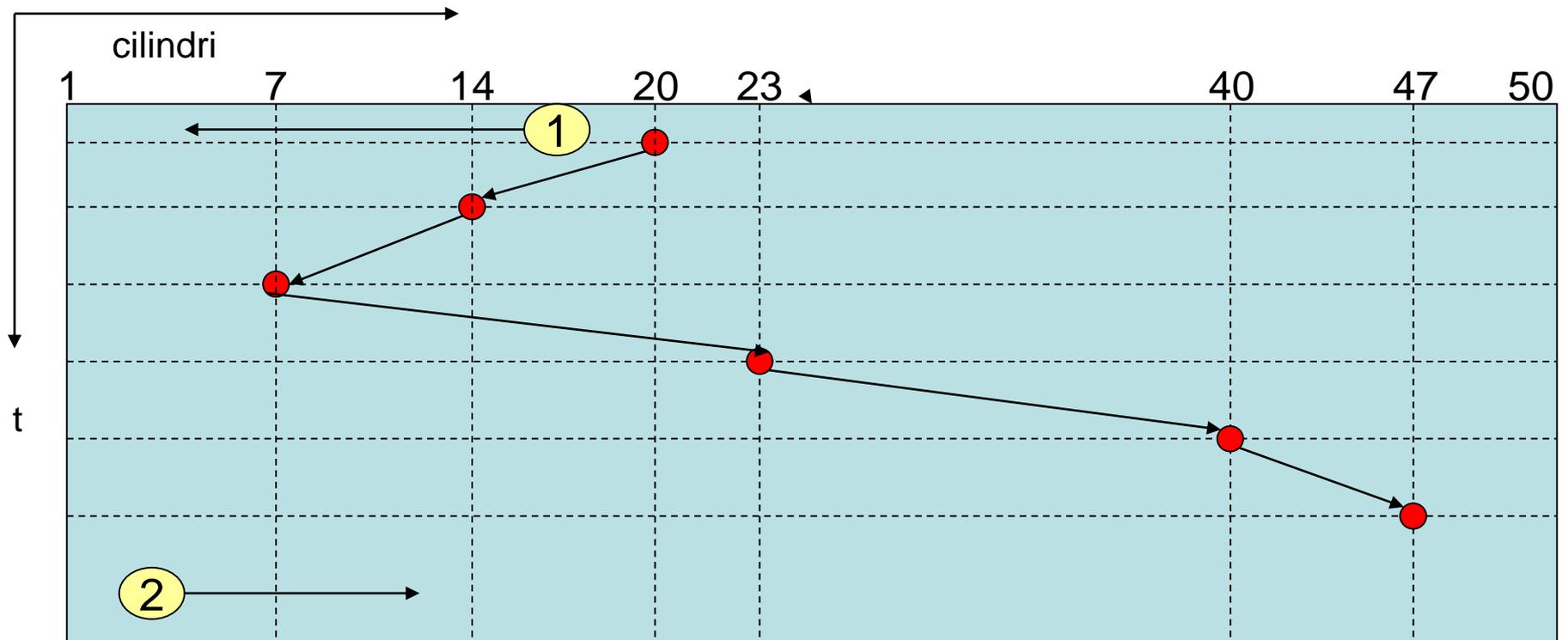
# Sceduling per scansione – SCAN

Coda delle richieste: 14, 40, 23, 47, 7

Posizione iniziale: 20

Scheduling: 14, 7, 23, 40, 47

Percorso 53 cilindri

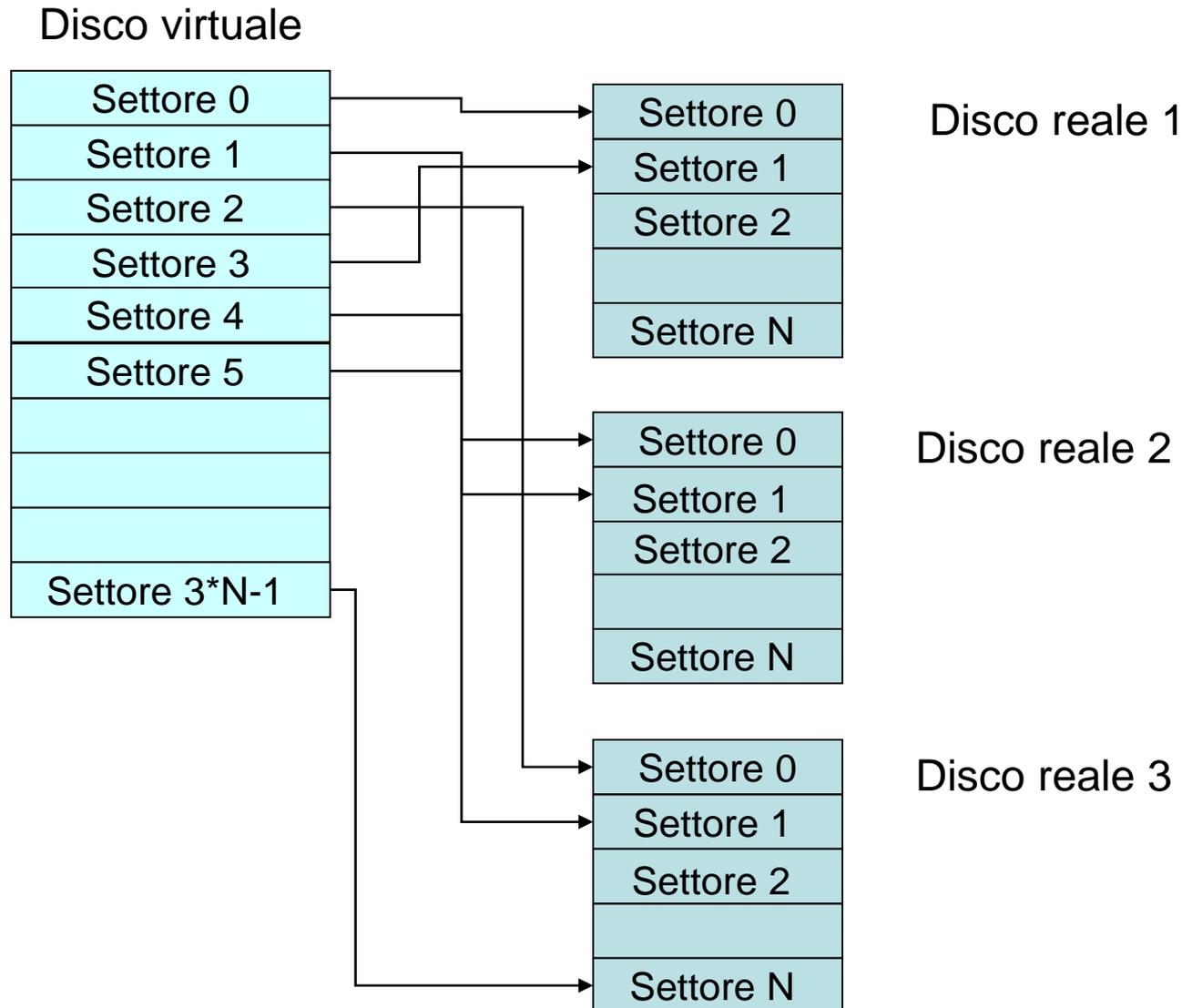


# Dischi RAID (Redundancy Array of Independent Disk)

- E' possibile migliorare le prestazioni dei dischi mediante particolari tecnologie.
- Uno standard molto diffuso è **RAID** (Redundancy Array of Independent Disk), che consente di memorizzare i dati su array di dischi, per migliorare l'affidabilità e la velocità degli accessi.
- Con questa tecnologia, in un sistema con **N** dischi, ciascuno di capacità **disk\_size**, è possibile realizzare un disco virtuale di capacità **N\*disk\_size** e con tempo di accesso notevolmente inferiore.
- Lo standard RAID ha 7 varianti indicate con ***varianti di livello 0,1,2,3,4,5 e 6***. Ogni livello ha caratteristiche diverse, in dipendenza della ridondanza di dati e della velocità di accesso che si vogliono ottenere.

- La variante del livello 0, il cui schema è mostrato nella figura seguente, non produce alcuna ridondanza dei dati.
- La variante di livello 1 (**mirroring**) è come il livello 0 ma con tutti i dischi duplicati (in relazione al caso di figura sarebbero necessari 6 dischi). E' uno schema ottimo sia per quanto riguarda l'affidabilità che la velocità di accesso.
- La presenza delle copie consente di effettuare operazioni di scrittura in parallelo mentre le operazioni di lettura si possono ottenere referenziando il disco che richiede il minor tempo di trasferimento. Questo schema è il più costoso in quanto richiede un numero elevato di dischi.
- Ci sono altre varianti che vanno dai livelli 2 al 6, che differiscono dal livello 1 per il livello di ridondanza dei dati e per le capacità di rilevazione e correzione degli errori, che utilizzano schemi per il controllo di parità o codifica di hamming.

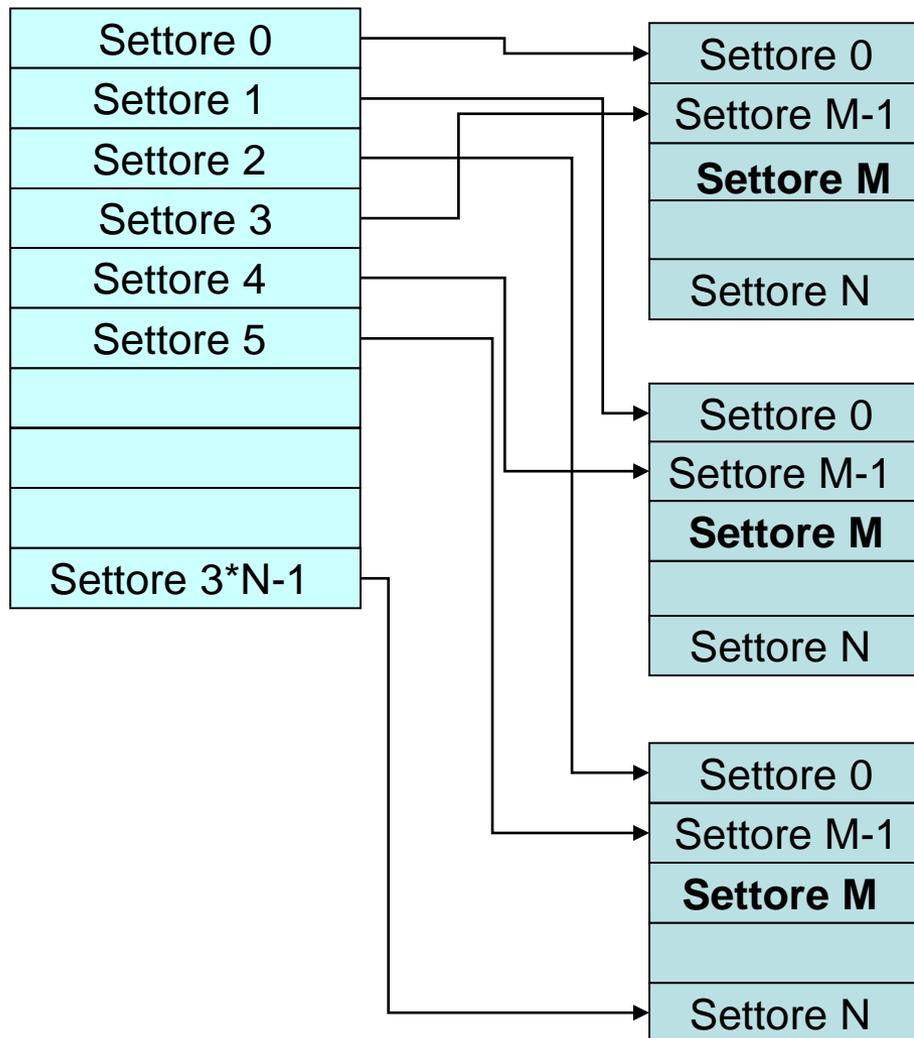
- Dischi RAID. Schema di livello 0 (senza ridondanza dei dati)



- Il livello 5 ad esempio, presenta un livello di ridondanza nettamente inferiore al mirroring (livello 1). Un determinato numero di blocchi su ogni disco è usato per contenere dati aggiuntivi per il controllo di parità. Ad esempio, nel caso di un array di  $N$  dischi, per ogni gruppo di  $M$  settori consecutivi, viene calcolato un settore  $M+1$  per contenere bit di parità.

- Dischi RAID. Schema di livello 5

Disco virtuale



Disco reale 1

Blocco di ridondanza per rilevazione e correzione errori

Disco reale 2

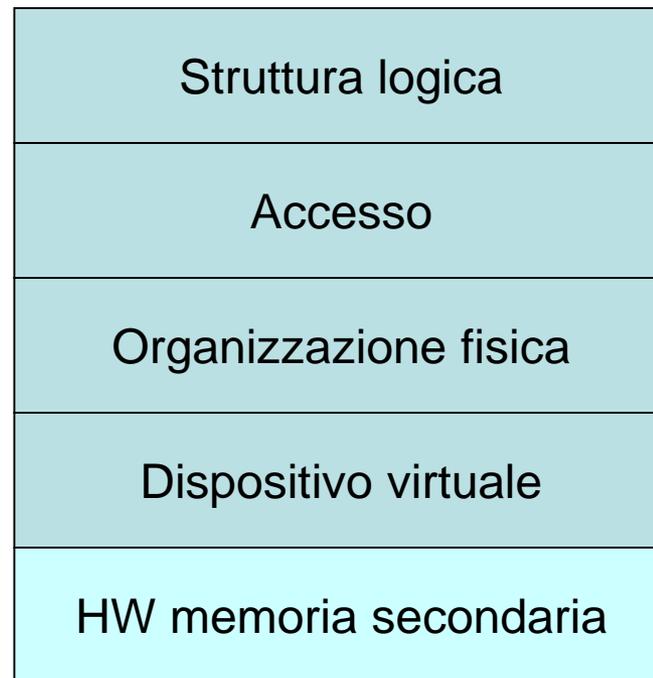
Blocco di ridondanza per rilevazione e correzione errori

Disco reale 3

Blocco di ridondanza per rilevazione e correzione errori

# Il file system

- Il file system è la parte del SO che realizza le astrazioni e le tecniche che consentono la rappresentazione, l'archiviazione e l'accesso ai dati memorizzati sulla memoria secondaria.
- La struttura di un file system è formata da vari componenti organizzati in vari livelli:



# La struttura logica del file system

La struttura logica consente all'utente di vedere i dati memorizzati sulla memoria secondaria a prescindere dalle caratteristiche dei dispositivi e dalle tecniche di allocazione. I dati sono visti sotto forma di **file e directory (cartelle)**.

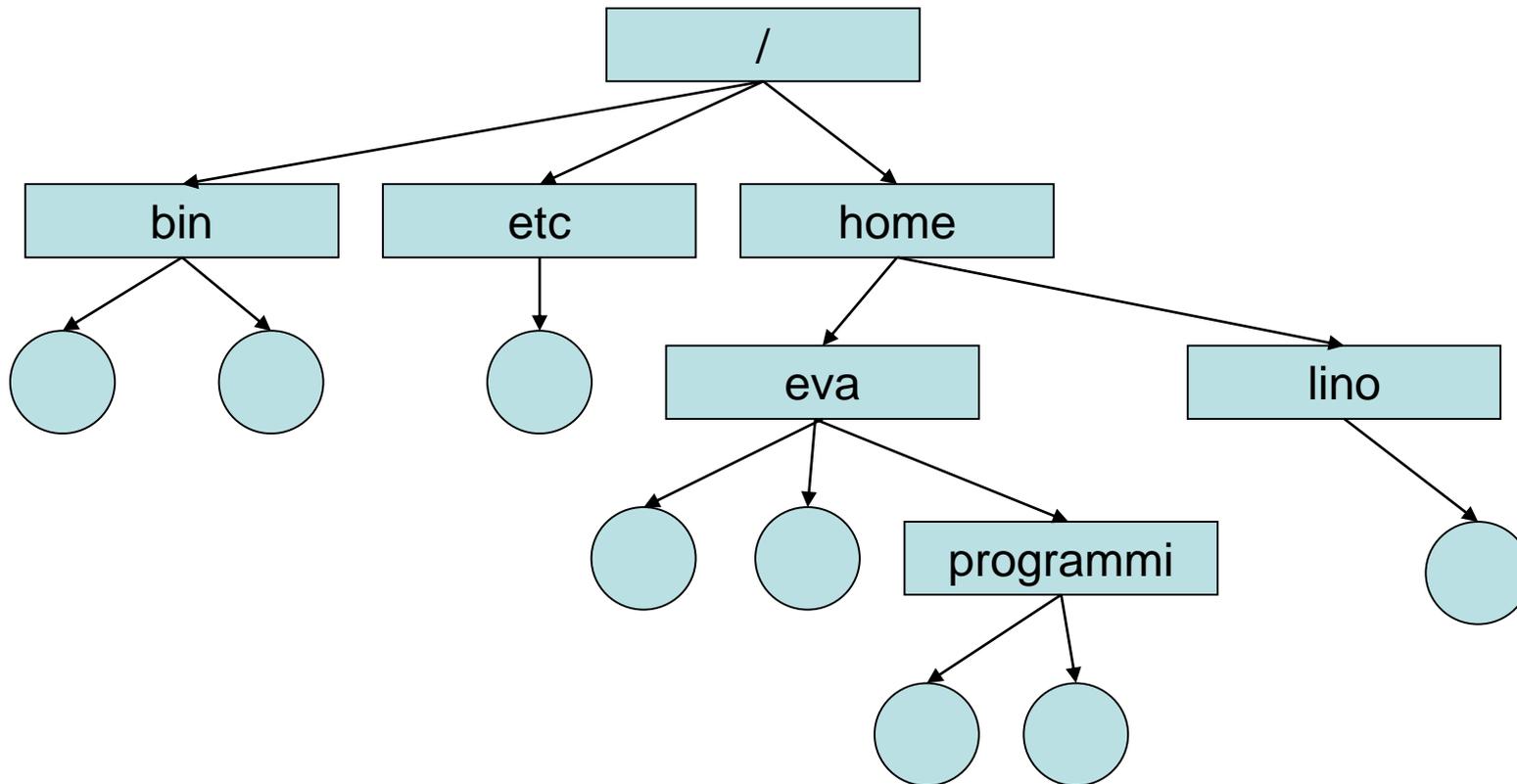
## Il file

- il file è l'unità logica di memorizzazione all'interno del file system.
- I file possono contenere informazione di vario genere: testo, programma eseguibile, audio, video, immagini etc.
- Un file è identificato da un **nome** ed è descritto da un insieme di attributi, come ad esempio
  - data ultima modifica
  - dimensione
  - tipo

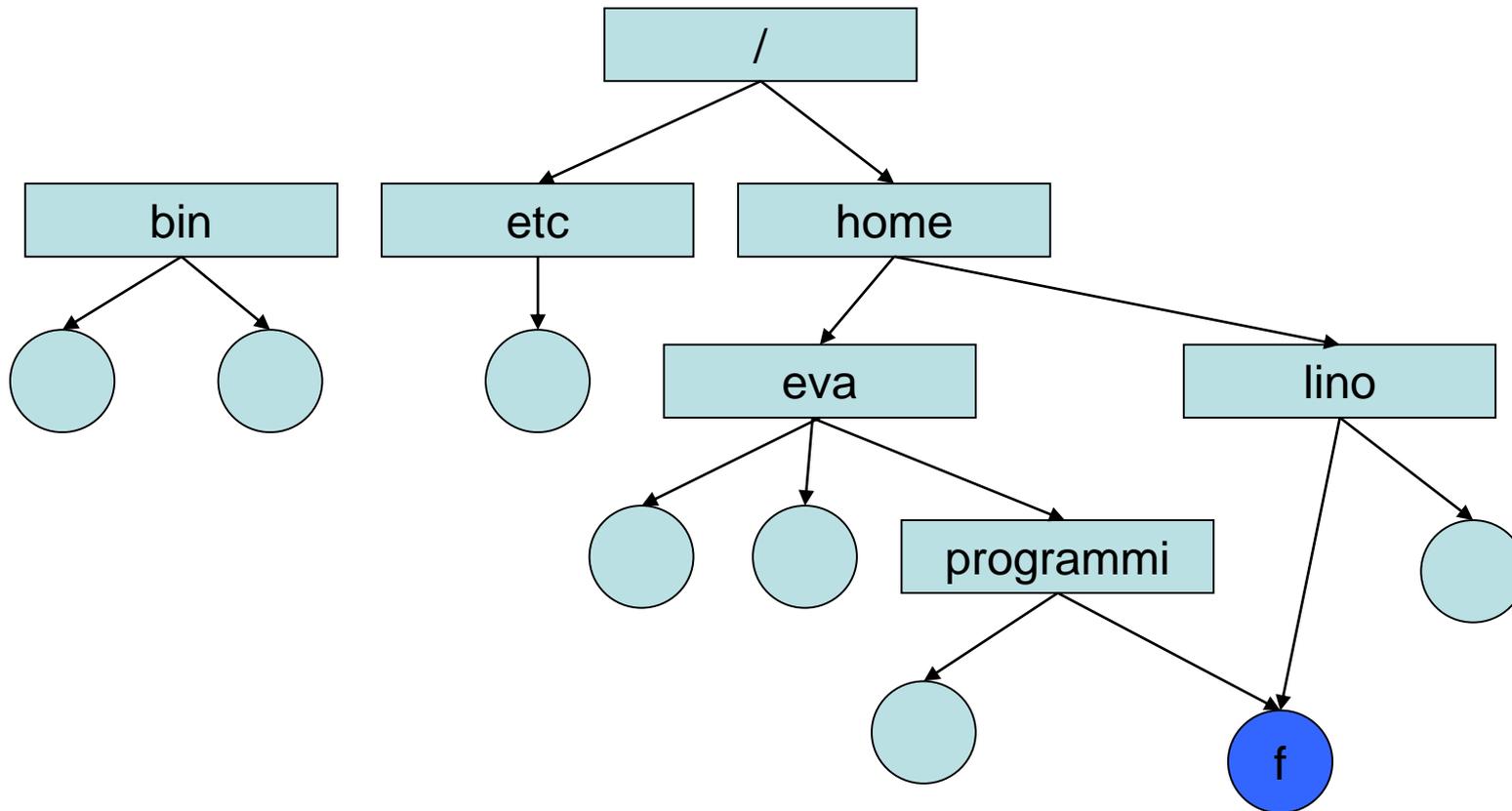
- Nei SO multiutente sono molto importanti:
  - proprietario
  - gruppo
  - diritti di accesso

## La directory

- La directory è un'astrazione che consente di raggruppare più file. E' un contenitore di file.
- Una directory può contenere nel suo interno altre directory (directory nidificate).
- La struttura del file system è quindi un insieme di file e directory.
- Generalmente i file system sono strutturati ad albero o a grafo.



## File system con struttura ad albero



## File system con struttura a grafo

# Gestione della struttura logica del file system

- Il SO offre un insieme di chiamate di sistema per la gestione del file system. Le principali operazioni sono:
  - Creazione, rinomina e cancellazione di directory
  - Creazione, rinomina e cancellazione di file
  - Creazione, rinomina e cancellazione di link
  - Listato di una directory
  - Navigazione del file system
  - Spostamento di file e directory
- E' possibile eseguire queste operazioni sia mediante interfaccia grafica (GUI) del SO che attraverso *comandi* digitati mediante l'uso di una shell.
- Operazioni più complesse si possono ottenere realizzando script di shell o applicazioni, scritte con un linguaggio, che fanno uso di chiamate di sistema relative al file system.

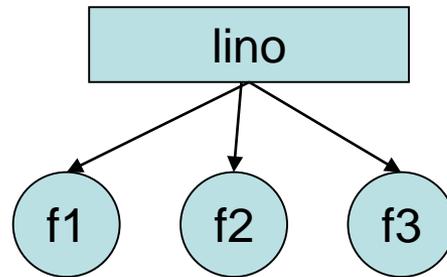
# Il livello di accesso al file system

- Questo livello definisce e realizza le **funzioni e le tecniche** attraverso le quali è possibile accedere ai file.
- Le operazioni di lettura e scrittura sono due tipi di accesso.
- In questo livello, i file sono formati da un insieme di **record logici**.
- Il record logico costituisce l'**unità di trasferimento** tra file e processo ed è caratterizzato da un insieme di proprietà come ad esempio il formato e la dimensione.
- Esistono vari tipi di accesso ai file, i più comuni sono l'**accesso sequenziale** ed l'**accesso diretto (random)**.
- Inoltre, questo livello implementa, in un SO multiutente, le tecniche e le **politiche di protezione dei file** che stabiliscono i diritti di accesso dei file per gli utenti.

# Strutture dati e operazioni di accesso ai file

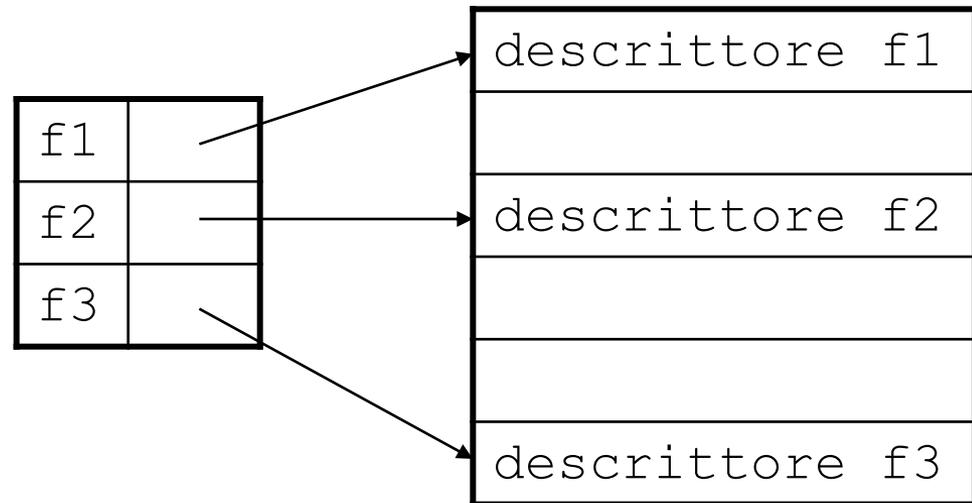
- Un file è descritto mediante una struttura dati, detta ***descrittore del file (FCB, File Control Box)***.
- Il descrittore deve avere caratteristiche di persistenza, ed è quindi memorizzato su dispositivo di memoria secondaria.
- Il descrittore del file contiene oltre le proprietà precedentemente descritte (nome, tipo, dimensione, proprietario, etc.) anche gli indirizzi dei blocchi su disco o dispositivo NVM che ne specificano l'allocazione fisica.
- Ogni directory è collegata ai descrittori di file in essa contenuti.
- i processi possono accedere ai file in vario modo:
  - Lettura
  - Scrittura
  - Scrittura accodata (append)
- Per eseguire un'operazione su un file è necessario leggere prima il suo descrittore.

- Per velocizzare le operazioni sui file, il SO mantiene in memoria una struttura dati detta **tabella dei file aperti** che contiene alcune informazioni associate ai file in uso.
- Generalmente, un elemento della tabella dei file aperti contiene:
  - Una copia del descrittore del file
  - Informazioni relative al processo che accede al file
  - Il puntatore al prossimo record logico da leggere o scrivere (nel caso di accesso sequenziale)
- L'operazione di **apertura (open)** di un file produce l'inserimento di un nuovo elemento nella tabella dei file aperti.
- E' quindi conveniente chiudere un file, quando non lo si deve più usare, in quanto comporta l'eliminazione del relativo elemento dalla tabella dei file aperti.



f1	descrittore
f2	descrittore
f3	descrittore

Struttura directory: il descrittore del file fa parte della directory (windows)



Struttura directory: i descrittori dei file sono nella tabella dei descrittori (unix)

- Per velocizzare le operazioni sui file, molti sistemi operativi utilizzano la tecnica **memory mapping** che consiste nel copiare il file aperto in memoria. In questo modo le operazioni si compiono sulla copia in memoria con notevole aumento di velocità.
- Quando si chiude il file, il SO salva automaticamente la **copia in memoria** del file su disco e elimina l'elemento corrispondente dalla tabella dei file aperti.

# Metodi di accesso

- I metodi di accesso determinano le operazioni che i processi possono eseguire sui file. I metodi di accesso più diffusi sono:
  - Accesso sequenziale
  - Accesso diretto (random)
  - Accesso indicizzato
- Ogni metodo di accesso dipende dalla particolare struttura del file, che a questo livello è visto come un insieme di record logici.

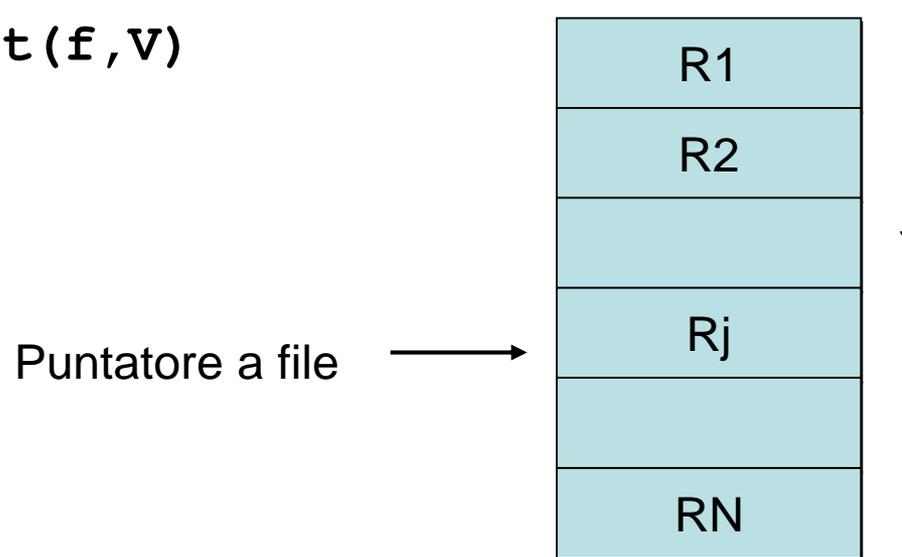
F: {R1, R2,...Rn}



- Il record logico è l'unità di lettura/scrittura sul file.

# Accesso sequenziale

- Ogni file è formato da una sequenza di record logici.
- I record possono essere scritti e letti solo in sequenza, quindi, ad esempio, per leggere un record che si trova nella posizione  $j$  è necessario leggere tutti i  $j-1$  precedenti.
- Il sistema utilizza un puntatore al file che indica il prossimo byte su cui leggere o scrivere.
- Le chiamate di sistema per l'accesso sequenziale sono del tipo:
  - `readnext (f, V)`
  - `writenext (f, V)`



- Per accedere al record  $j$  è necessario eseguire nel modo seguente:

```
for (i=1; i<j;i++)
```

```
    readnext(f, V); //lettura dei record precedenti
```

```
readnext(f ,V); //lettura del record j
```

# Accesso diretto

- L'accesso diretto permette di leggere/scrivere un qualunque record del file con operazioni del tipo:
  - **readd(f, j, V)**
  - **writed(f, j, V)**
- Si può accedere ad un determinato record specificandone l'indice **j**. Non è necessario quindi dover scorrere i record precedenti.

# Accesso a indice

Con l'accesso a indice la struttura del record logico è composta di almeno due campi, uno dei quali, la **chiave**, identifica univocamente un determinato record. Inoltre ad ogni file viene associata una tabella, detta **indice**, nella quale è presente una riga per ogni chiave che contiene un campo di riferimento al record corrispondente nel file. In questo modo è possibile accedere a un particolare record del file specificandone la chiave, mediante le operazioni:

- **readk(f, key, V)**
- **wriek(f, key, V)**

La chiave consente di accedere direttamente al record cercato, previa ricerca associativa nell'indice del file.

