Implementazione del file system

Implementazione del file system

- Struttura del file system
- Implementazione delle directory
- Metodi di allocazione
  - Allocazione contigua
  - Allocazione concatenata
  - Allocazione indicizzata
- Gestione dello spazio libero
- Esempio di file system distribuito (NFS)

Struttura del file system

- Struttura dei file
  - Unità logica di memoria
  - Collezione di informazioni correlate
- Un file è memorizzato e trasferito a blocchi.
- Il file system risiede su memoria secondaria (dischi).
- Il file system è organizzato a livelli funzionali.

Livelli di un file system

Programmi applicativi, script, comandi (ls, dir, . . .)

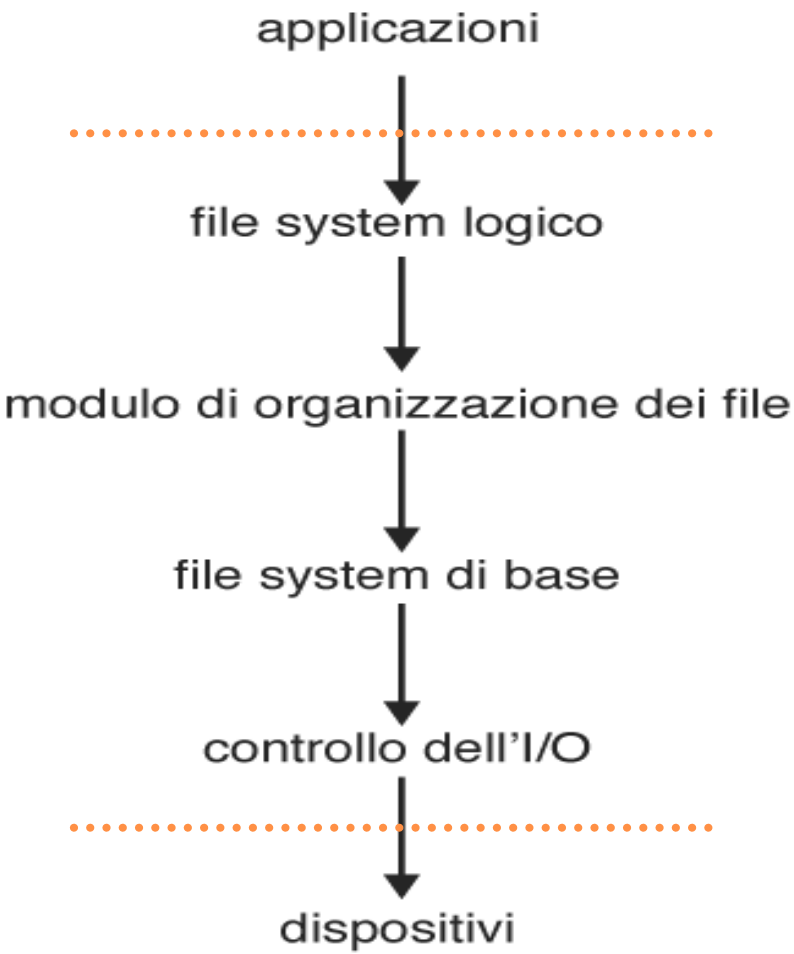
file system logico: presenta il file system come un'unica struttura; implementa i controlli di protezione

organizzazione dei file: controlla l'allocazione dei blocchi fisici e la loro associazione con quelli logici: trad. da indirizzi logici a fisici.

file system di base: usa i driver per accedere ai blocchi fisici sul dispositivo.

controllo dell'I/O: i driver dei dispositivi

dispositivi: i controller hardware dei dischi, nastri, ecc.



File Control Block

File Control Block: struttura di memoria contenente un insieme di informazioni riguardanti un file.

Nei file sistem basati su UNIX si chiama **inode** (*index node*)

permessi per il file
data e ora di creazione, di ultimo accesso e di ultima scrittura
proprietario del file, gruppo, ACL
dimensione del file
blocchi di dati del file o puntatori a blocchi di dati del file

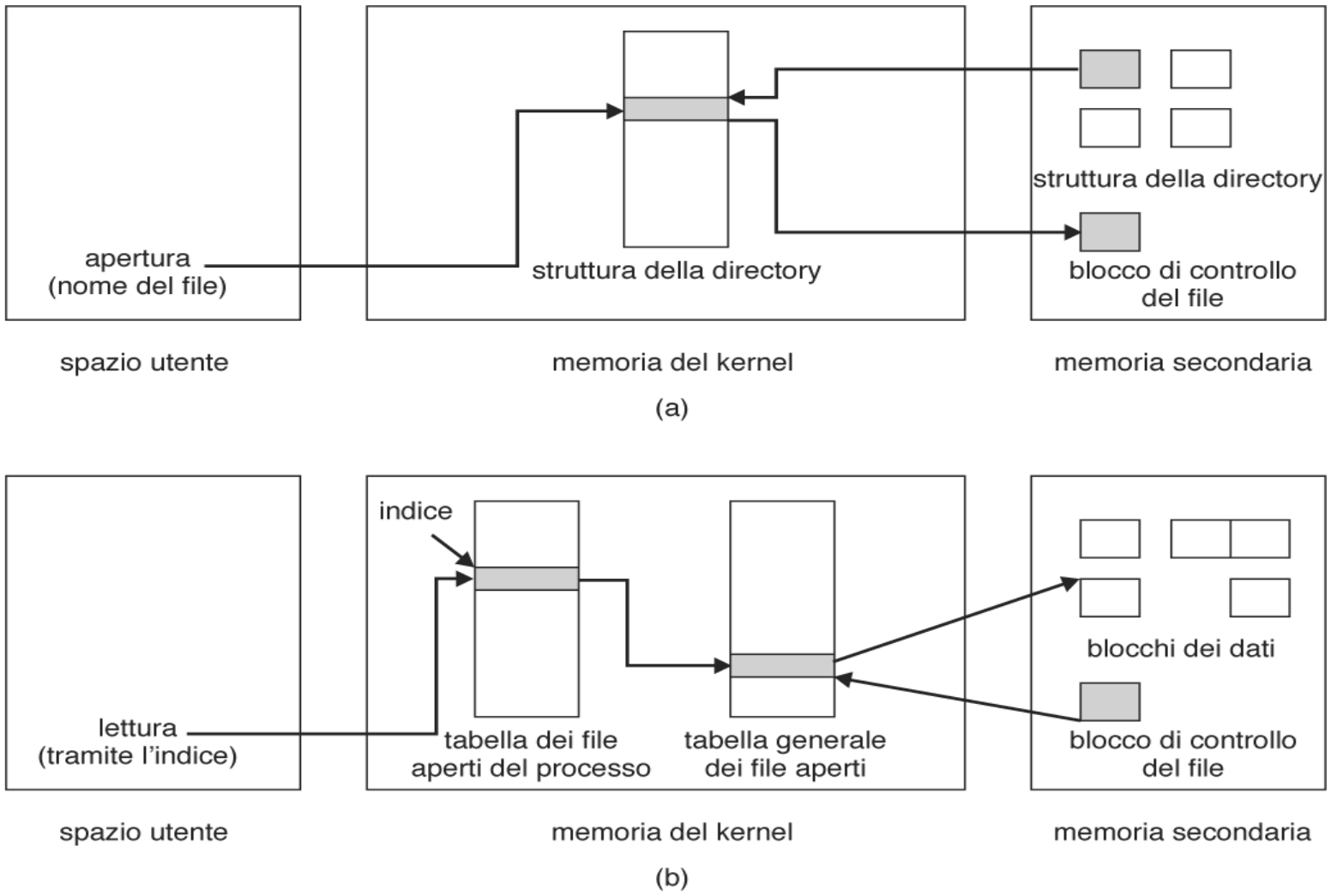
Strutture del file system in memoria

- Le due figure nella prossima slide illustrano le strutture necessarie al file system fornite dal sistema operativo.

- La Figura (a) descrive l'operazione di apertura di un file (***open***).
  - Accesso alla directory in memoria
  - Accesso alla directory sul disco (se necessario)
  - Accesso al file control block su disco

- La Figura (b) descrive l'operazione di lettura (***read***) di un file.
  - Accesso alle tabelle dei file in memoria
  - Accesso al file control block su disco
  - Accesso ai blocchi dati dei file.

Strutture del file system in memoria

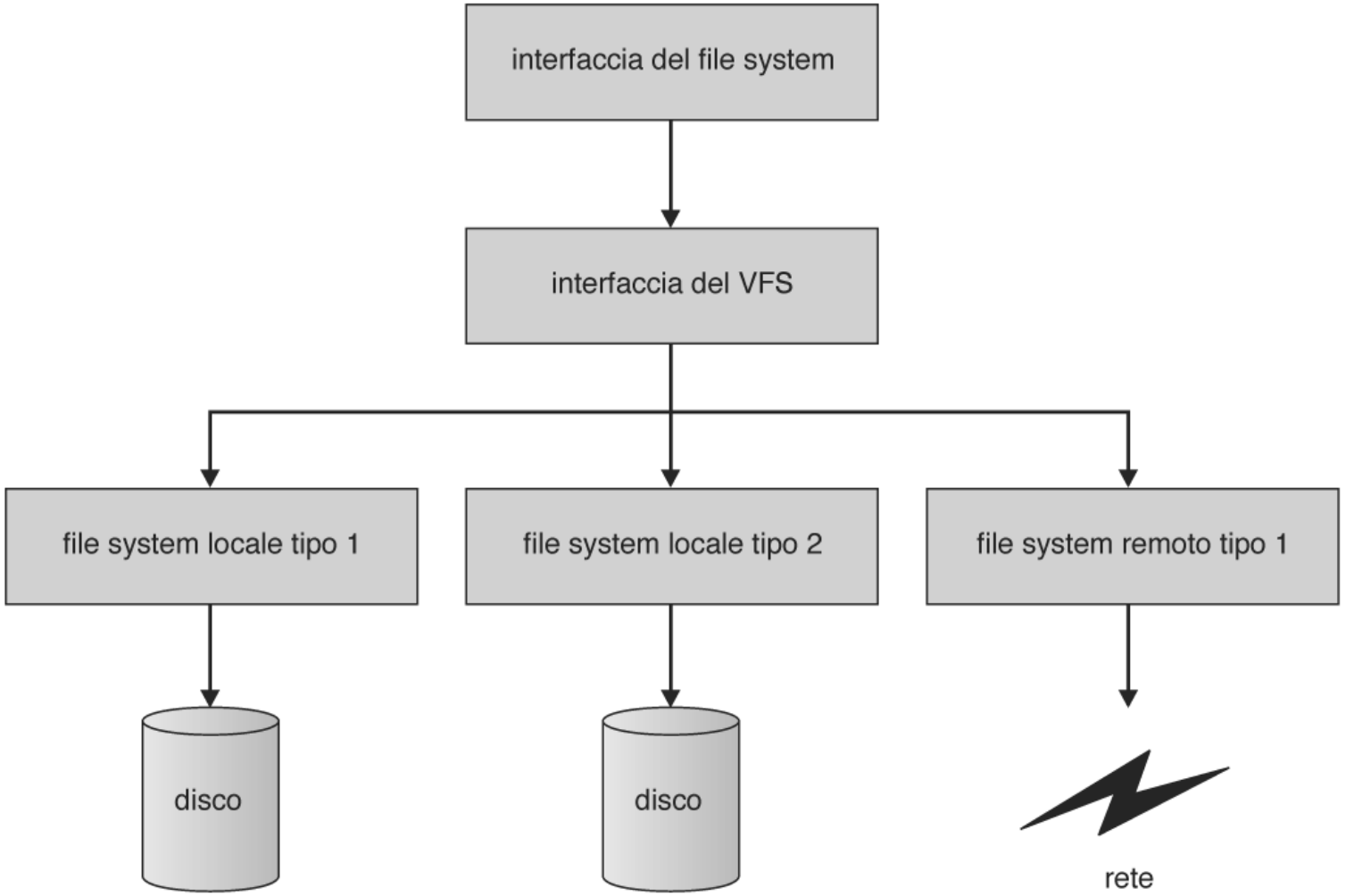


(a) Apertura di file; (b) Lettura di file.

File system virtuali

- Alcuni sistemi operativi, come UNIX, permettono di gestire in maniera integrata diversi tipi di file system.
- Questo è fatto tramite un Virtual File System (VFS) che fornisce una rappresentazione object-oriented del file system.
- Un VFS permette di usare una stessa interfaccia (API) per differenti tipi di file system.
- L'interfaccia opera verso il VFS che "nasconde" i diversi tipi di file system sottostanti.

Schema di un file system virtuale



Implementazione delle directory

- La scelta del metodo di allocazione e gestione delle directory ha un impatto sull'efficienza e sull'affidabilità del file system. Due sono i metodi principali:

- **Lista lineare** dei nomi dei file con i puntatori ai blocchi dei dati (contenuto dei file)
  - semplice da programmare
  - non molto efficiente nella ricerca (lineare)

- **Tabella hash** – lista lineare con struttura hash per la ricerca.
  - Diminuisce il tempo di ricerca
  - *Collisioni* – situazioni dove due nomi di file portano alla stessa locazione della tabella.
  - Dimensione fissata legata alla funzione hash.

Metodi di allocazione

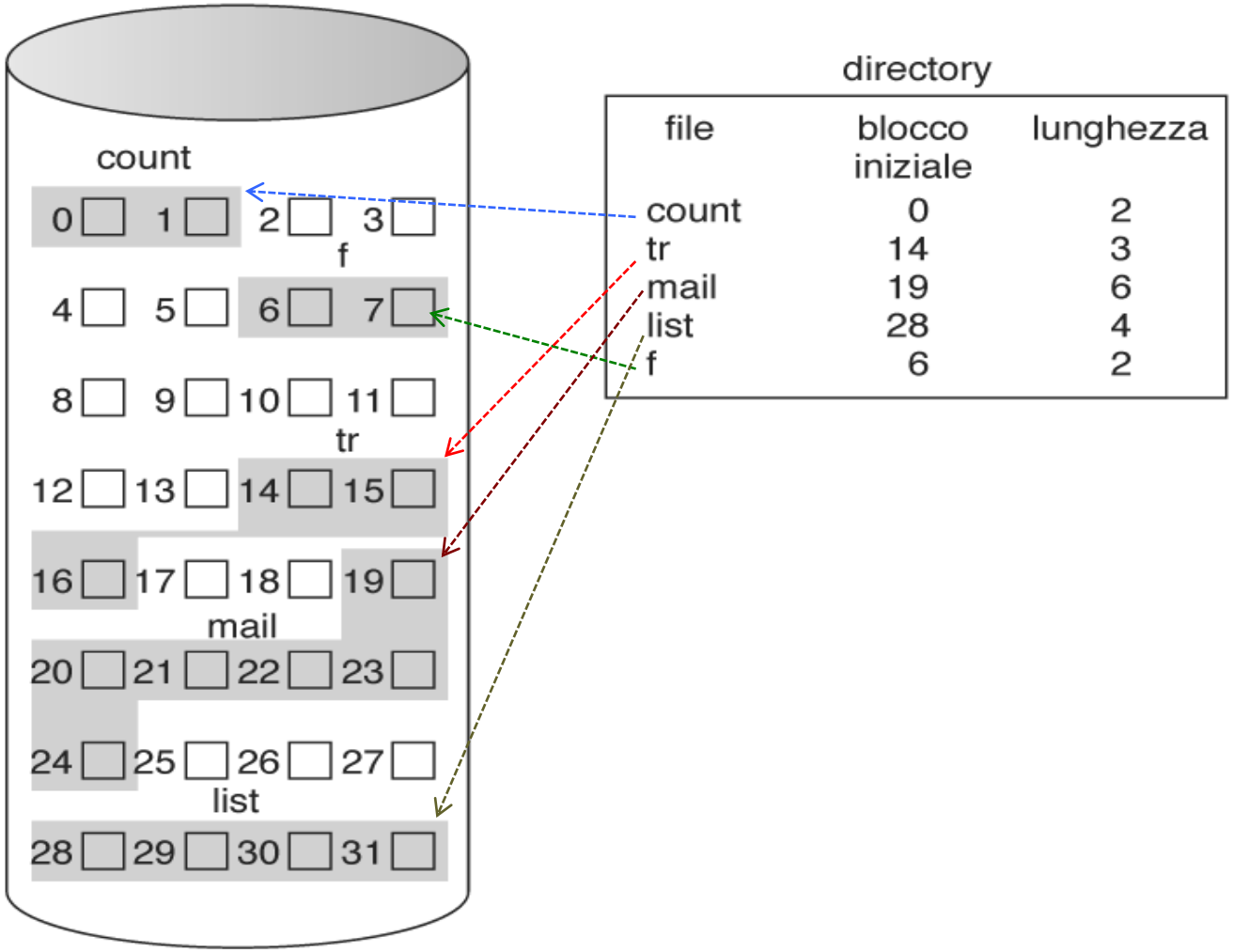
- Un metodo di allocazione si occupa di come allocare sulla memoria secondaria i blocchi di un file.

- Tre metodi principali:
  - **Allocazione contigua**
  - **Allocazione concatenata**
  - **Allocazione indicizzata**

Allocazione contigua

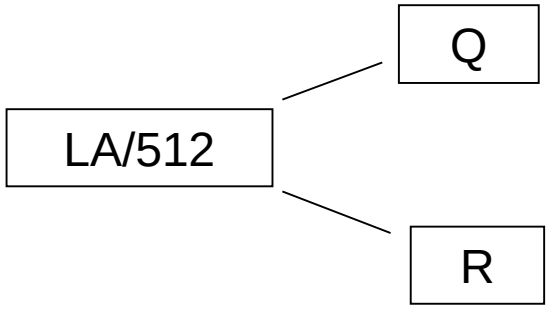
- Ogni file occupa un insieme contiguo di blocchi sul disco.
- E’ necessario conoscere la locazione di partenza (indirizzo del primo blocco) e la lunghezza (numero di blocchi).
- Supporta l’accesso diretto.
- Spreco di spazio.
- Occorre trovare lo spazio sufficiente (problema di allocazione dinamica di memoria).
- In determinati casi, i file non possono crescere di dimensione.

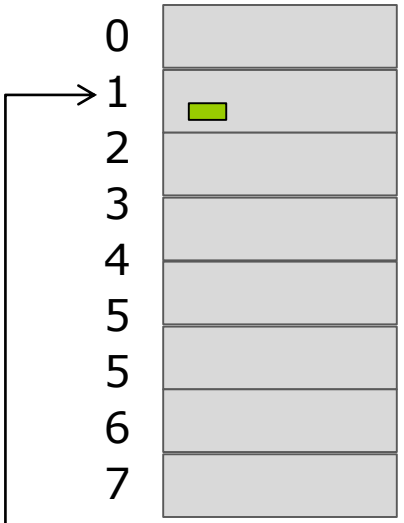
Esempio di allocazione contigua



Allocazione contigua

- Mapping da indirizzo logico (LA) a indirizzo fisico, con una dimensione dei blocchi di 512 words:





  - Blocco da accedere = Q + indirizzo di partenza
  - Spiazzamento nel blocco = R

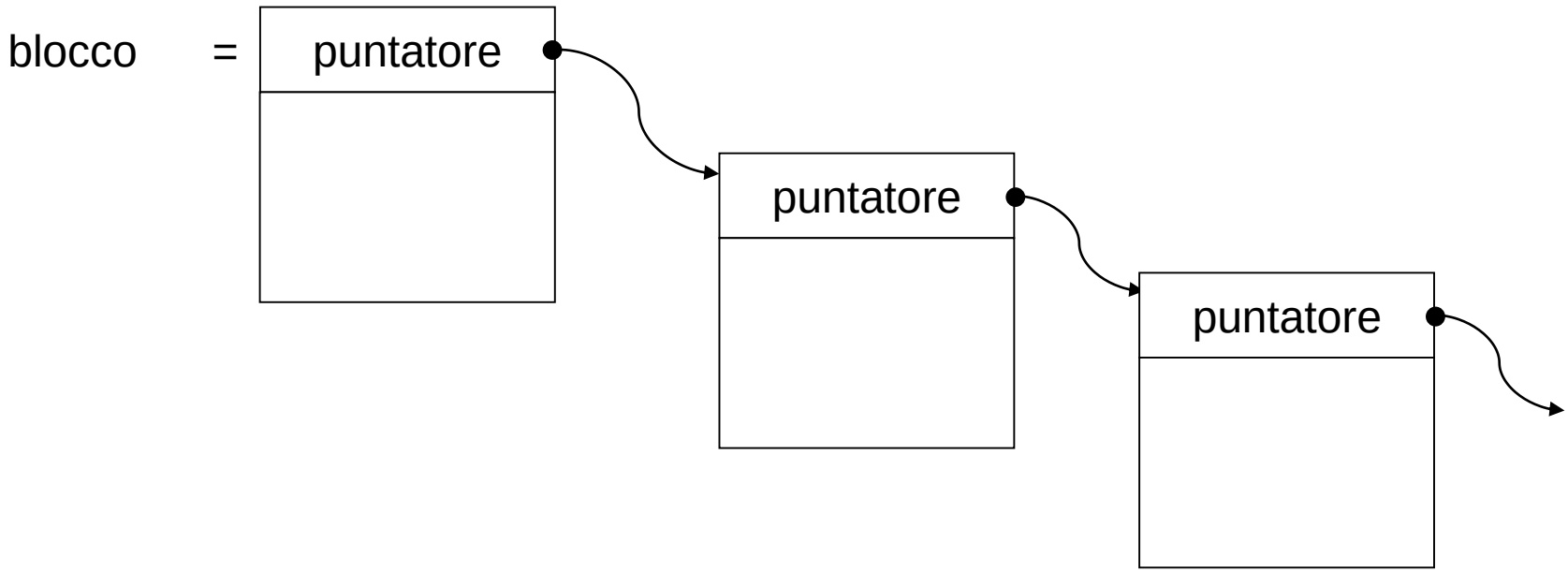
- Esempio:
  - LA= 864
  - Q = 1 (Blocco)
  - R = 352 (Spiazzamento nel Blocco)

Allocazione contigua modificata

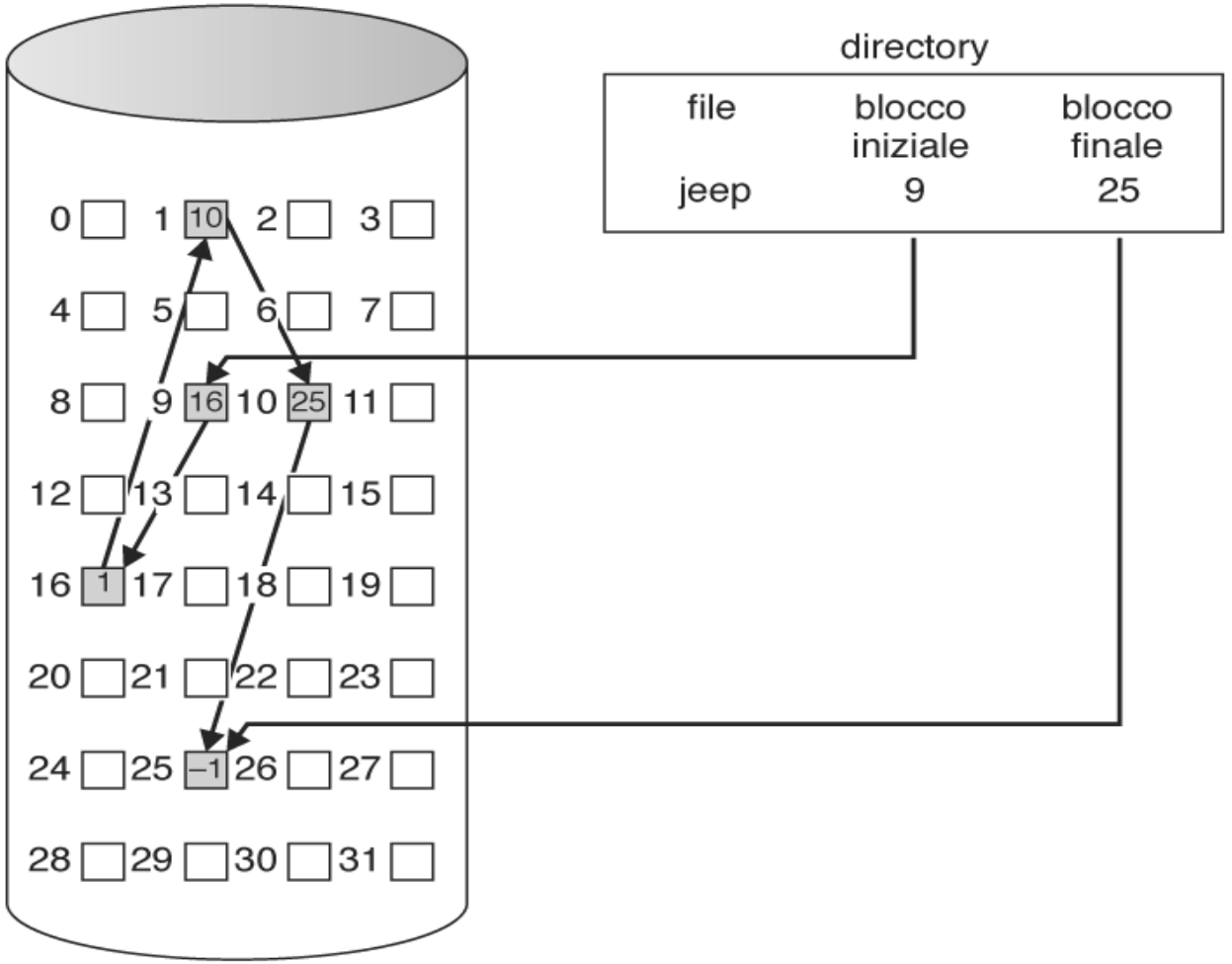
- Alcuni sistemi operativi usano uno schema modificato dell'allocazione contigua
- Se al file non basta lo spazio di memoria contigua allocata si alloca un ulteriore spazio contiguo (**extent**) su una parte libera del disco.
- Un file quindi può essere composto da uno o più *extent*.
- E' necessario avere un contatore degli extent e l'indirizzo del primo *extent*.

Allocazione concatenata

- Ogni file è gestito tramite una lista concatenata di blocchi di disco: i **blocchi non devono essere necessariamente contigui** e quindi possono trovarsi in punti diversi del disco.

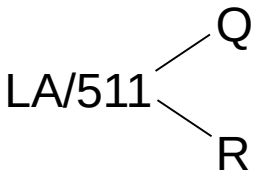


Esempio di allocazione concatenata



Allocazione concatenata

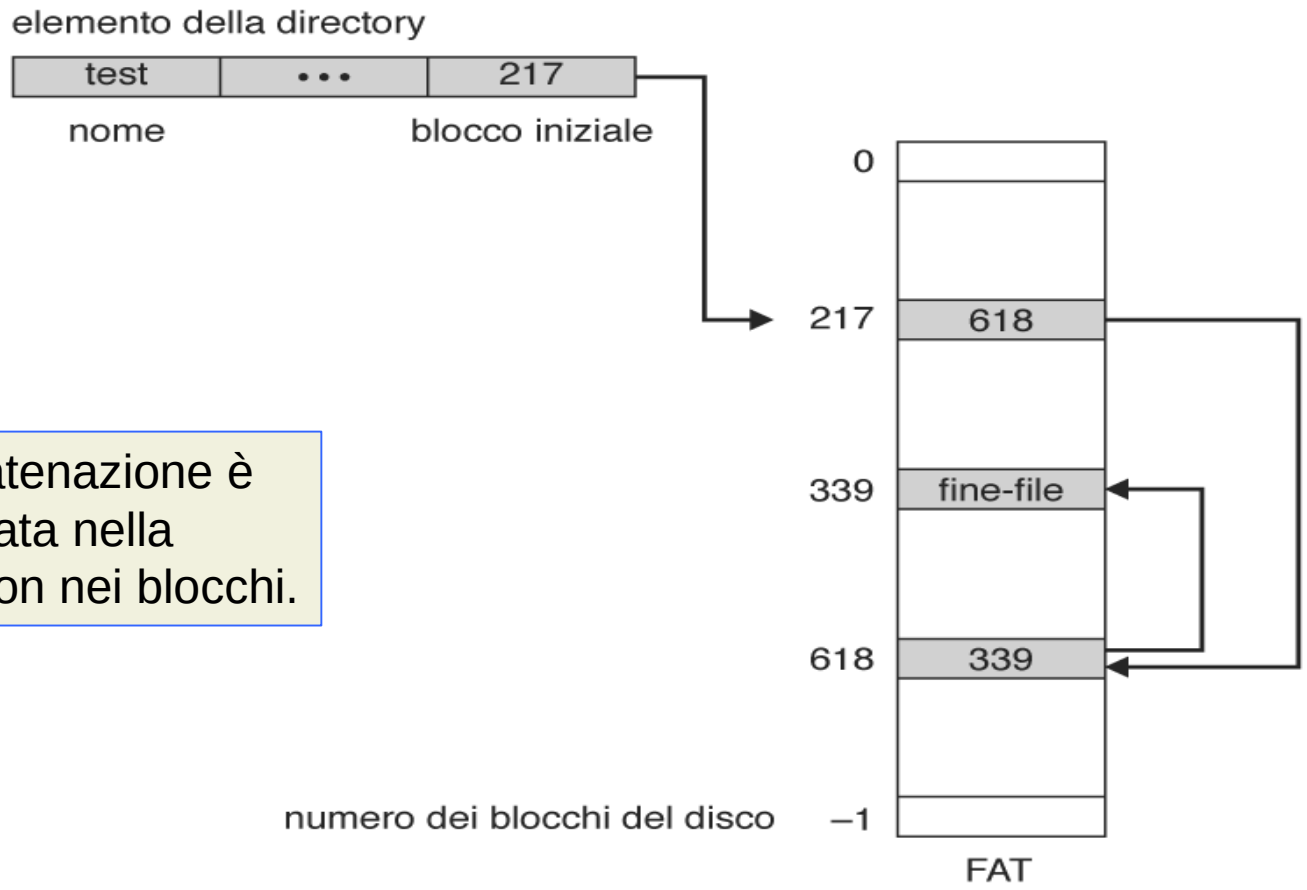
- Semplice: serve solo l'indirizzo di partenza.
- Non c'è spreco di spazio.
- Non supporta l'accesso diretto (solo accesso sequenziale).
- I file possono crescere.
- Mapping (con blocco di 512 word):



  - Il blocco a cui accedere è il Q-esimo nella lista concatenata di blocchi che rappresenta il file.
  - Spiazzamento nel blocco = R + 1

- Una variante usata in MS-DOS e OS/2 è la **File Allocation Table (FAT):**
  - Tabella con tanti elementi per quanti sono i blocchi sul disco.
  - Ogni elemento contiene l'indice del prossimo blocco.

File Allocation Table



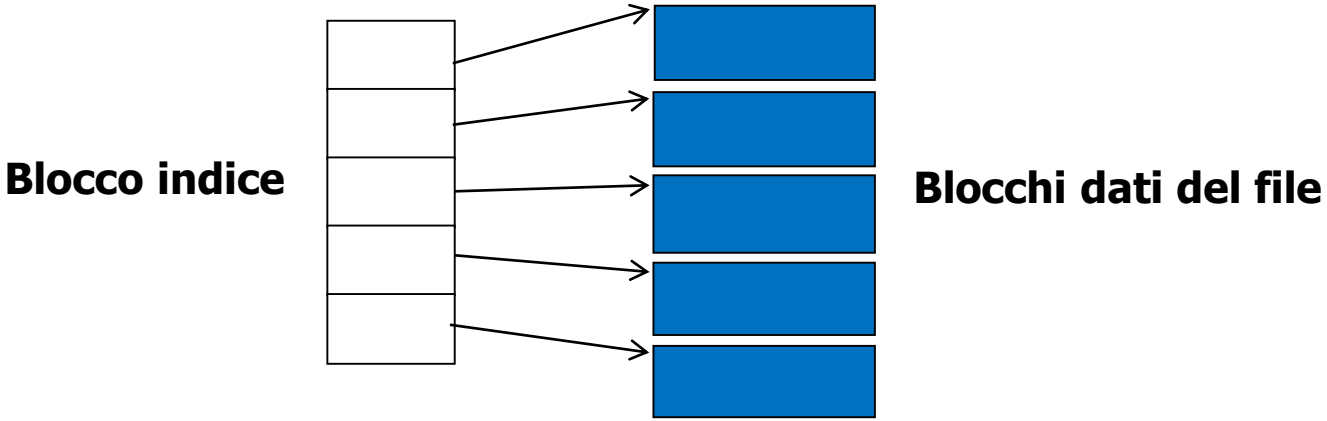
La concatenazione è organizzata nella tabella non nei blocchi.

Usata nel file system FAT32 e FAT64 (exFAT)

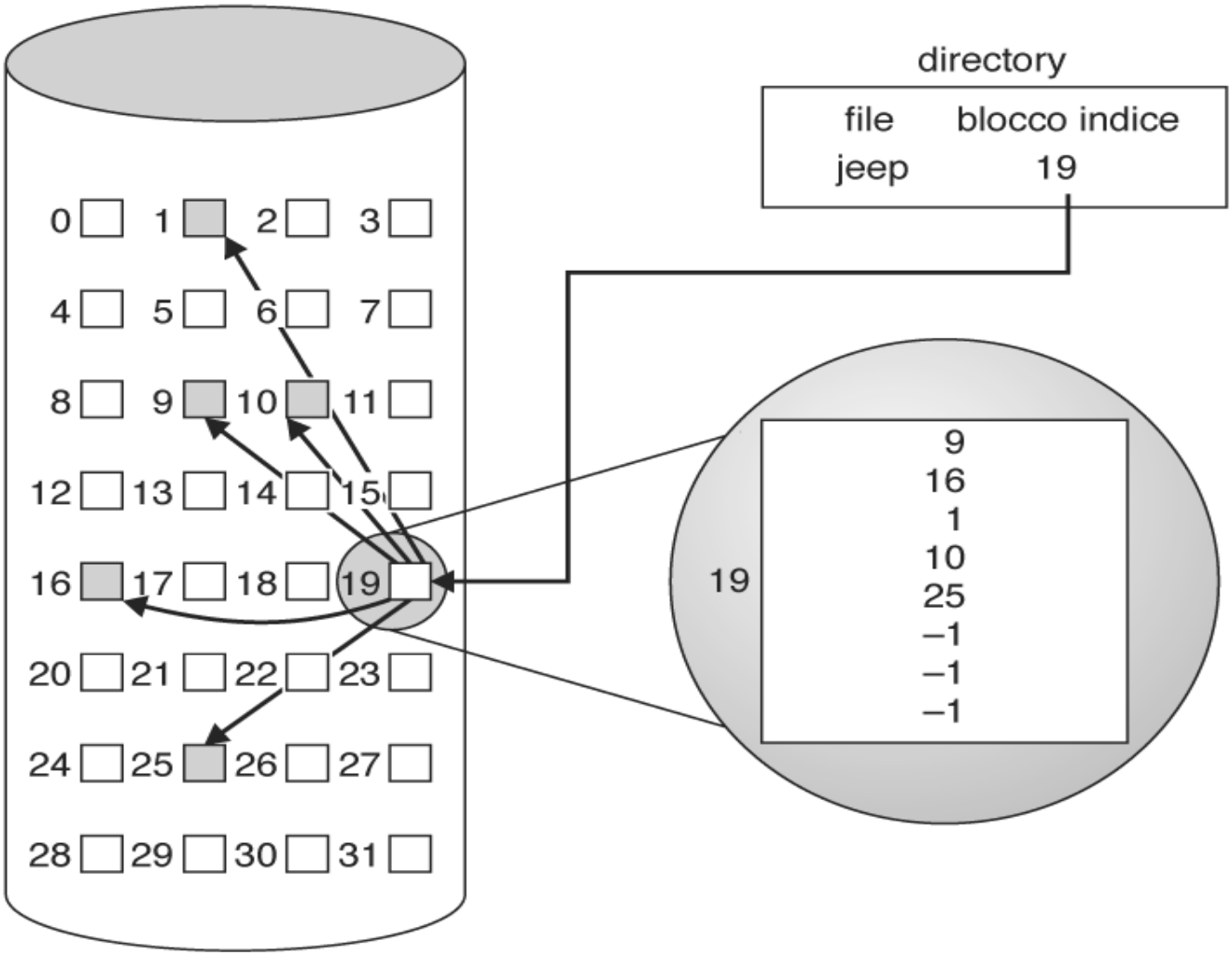
Allocazione indicizzata

- Mantiene tutti i puntatori ai blocchi dei file in un'unica struttura: il **blocco indice (index table)**.

- Vista logica:



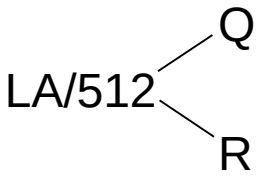
Esempio di allocazione indicizzata



Allocazione indicizzata

- Il Blocco indice occupa spazio.
- Supporta l'accesso diretto.
- Non c'è frammentazione esterna.

- Mapping da indirizzo logico a indirizzo fisico nel caso in cui il blocco indice sia contenuto in un solo blocco:

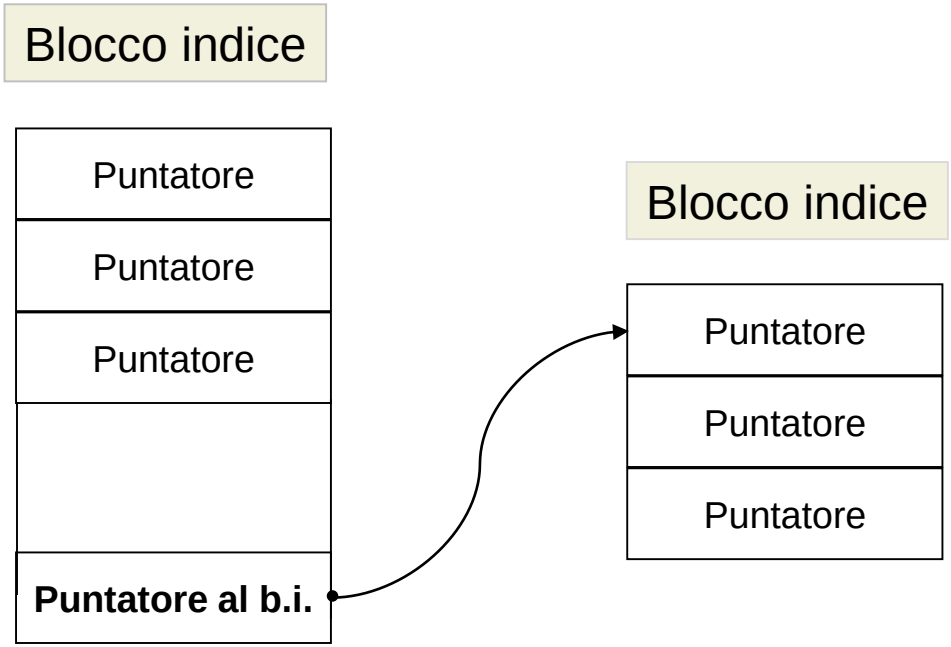


Q = spiazzamento nel blocco indice
R = spiazzamento nel blocco del file

- Se un blocco non è sufficiente a contenere il blocco indice di un file, per i blocchi indice si possono utilizzare:
  - **schema concatenato**
  - **schema multilivello**
  - **schema combinato**

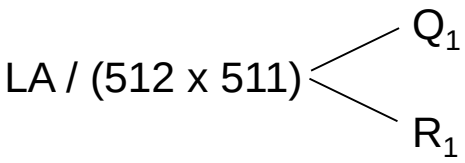
Allocazione indicizzata: schema concatenato

- L'ultima parte di un blocco indice contiene un puntatore ad un altro blocco indice (se il file è molto grande).

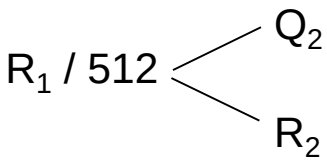


Allocazione indicizzata: schema concatenato

- Mapping:

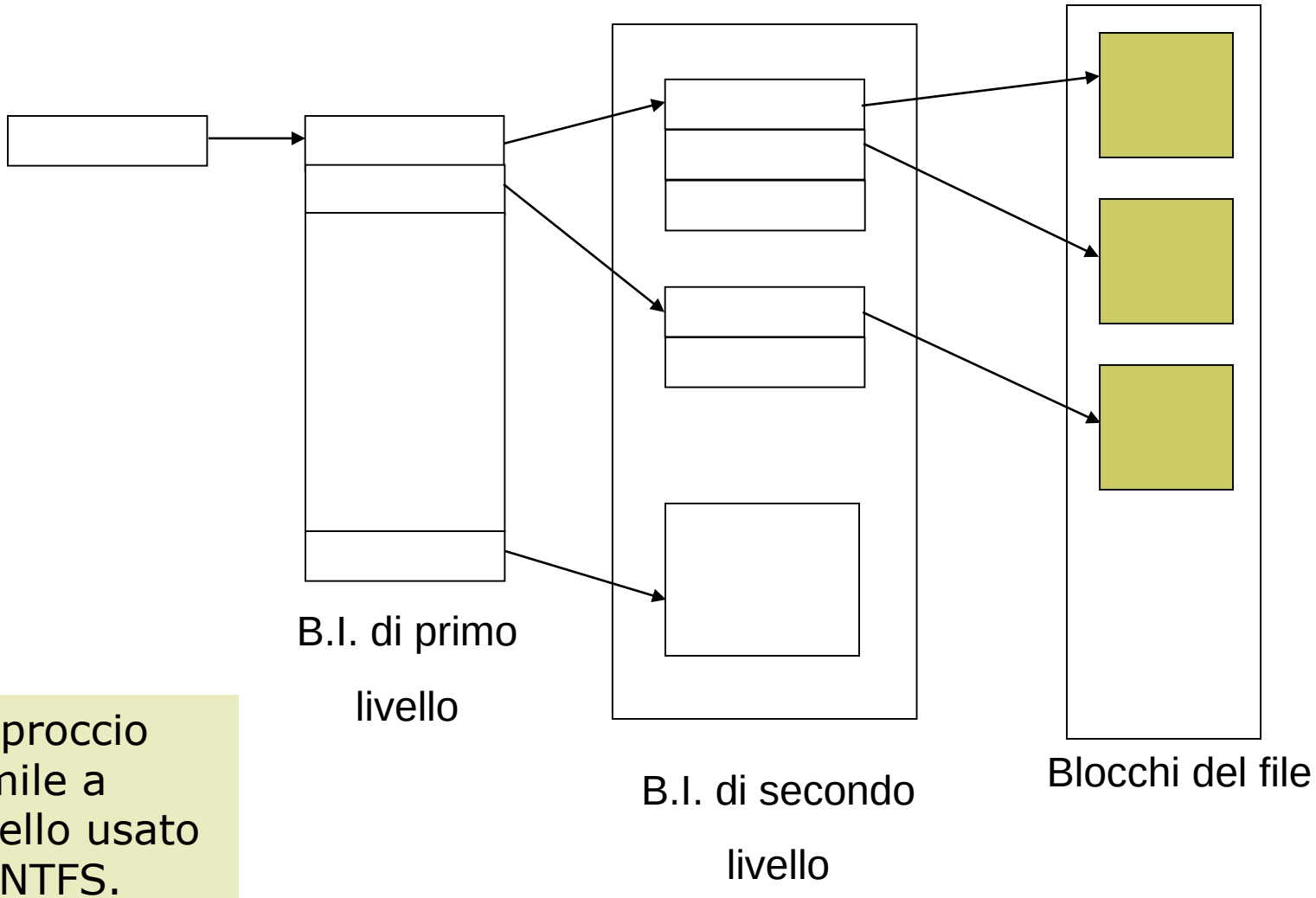


Q_1 = blocco della index table
R_1 è usato come segue:



Q_2 = spiazzamento nel blocco della index table
R_2 = spiazzamento nel blocco del file

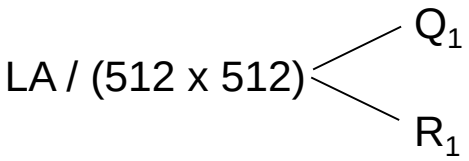
Allocazione indicizzata: schema a 2 livelli



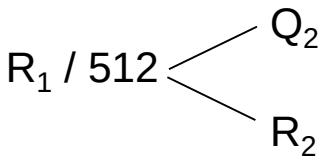
Approccio simile a quello usato in NTFS.

Allocazione indicizzata: schema a 2 livelli

- Mapping (dimensione massima del file 512^3)

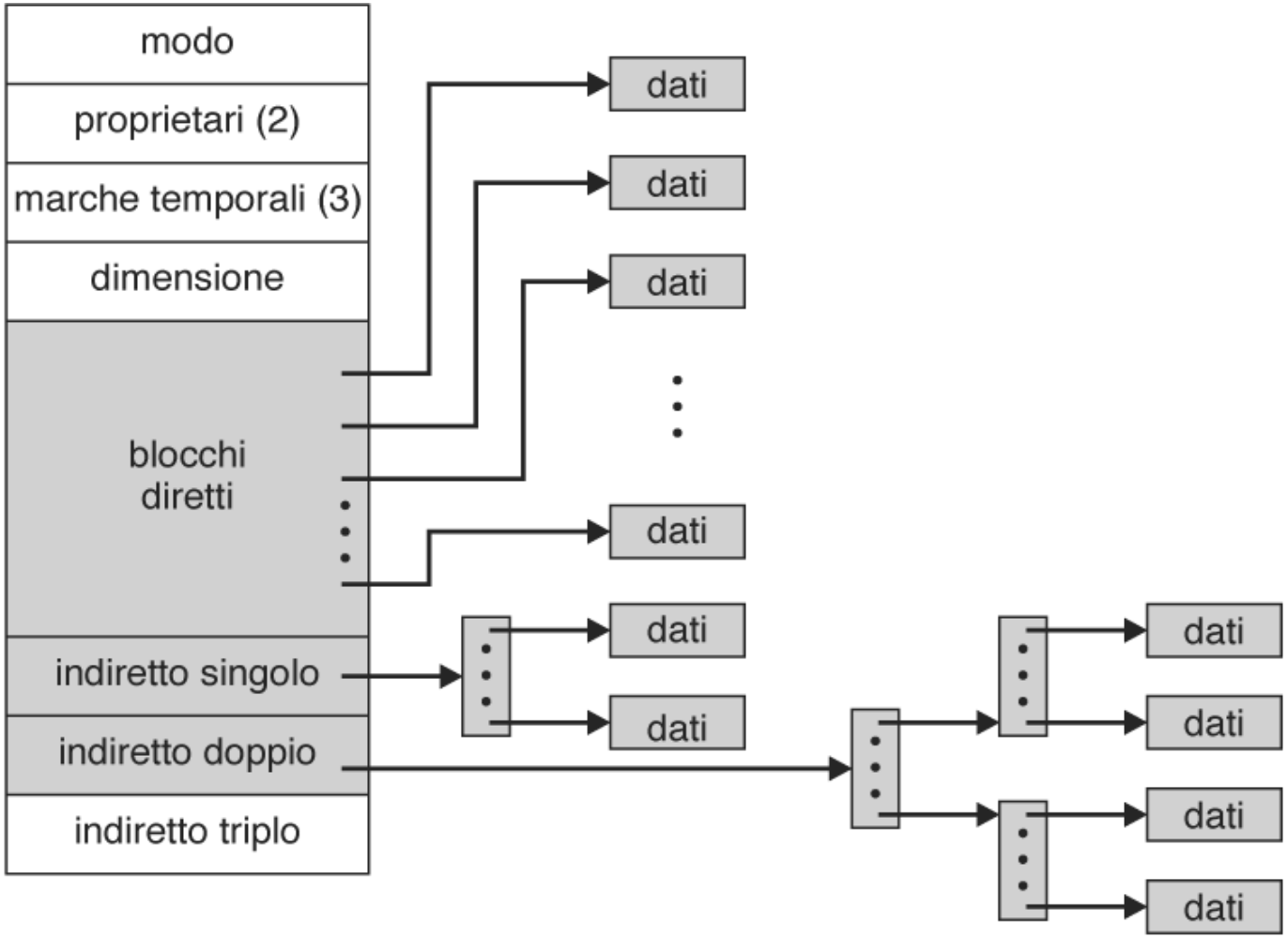


Q_1 = spiazzamento nell'indice di primo livello
R_1 è usato come segue:



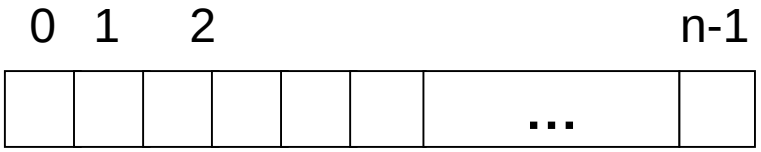
Q_2 = spiazzamento nel blocco della index table (di secondo livello)
R_2 = spiazzamento nel blocco del file

Schema combinato: *Inode* di UNIX (4KB per blocco)



Gestione dei blocchi liberi

- Per memorizzare i blocchi liberi si usa il vettore di bit (***n*** bit per **n** blocchi)



bit[*i*] =

1 ⇒ blocco[*i*] libero

0 ⇒ blocco[*i*] occupato

Calcolo del numero del primo blocco libero usando le word:

(numero di bit per word) * (numero di word con valore 0) + offset del primo bit 1

00000**1**01010101111110110000001010000000010110101011

00000000000000001**1**01100000010100000001010101011

Gestione dei blocchi liberi

- La bit map richiede uno spazio che potrebbe non essere tutto in memoria. Esempio:

  dim. blocco = 2^{12} byte

  dim. disco = 2^{30} byte (1 gigabyte)

  $n = 2^{30}/2^{12} = 2^{18}$ bit (o 32K byte)

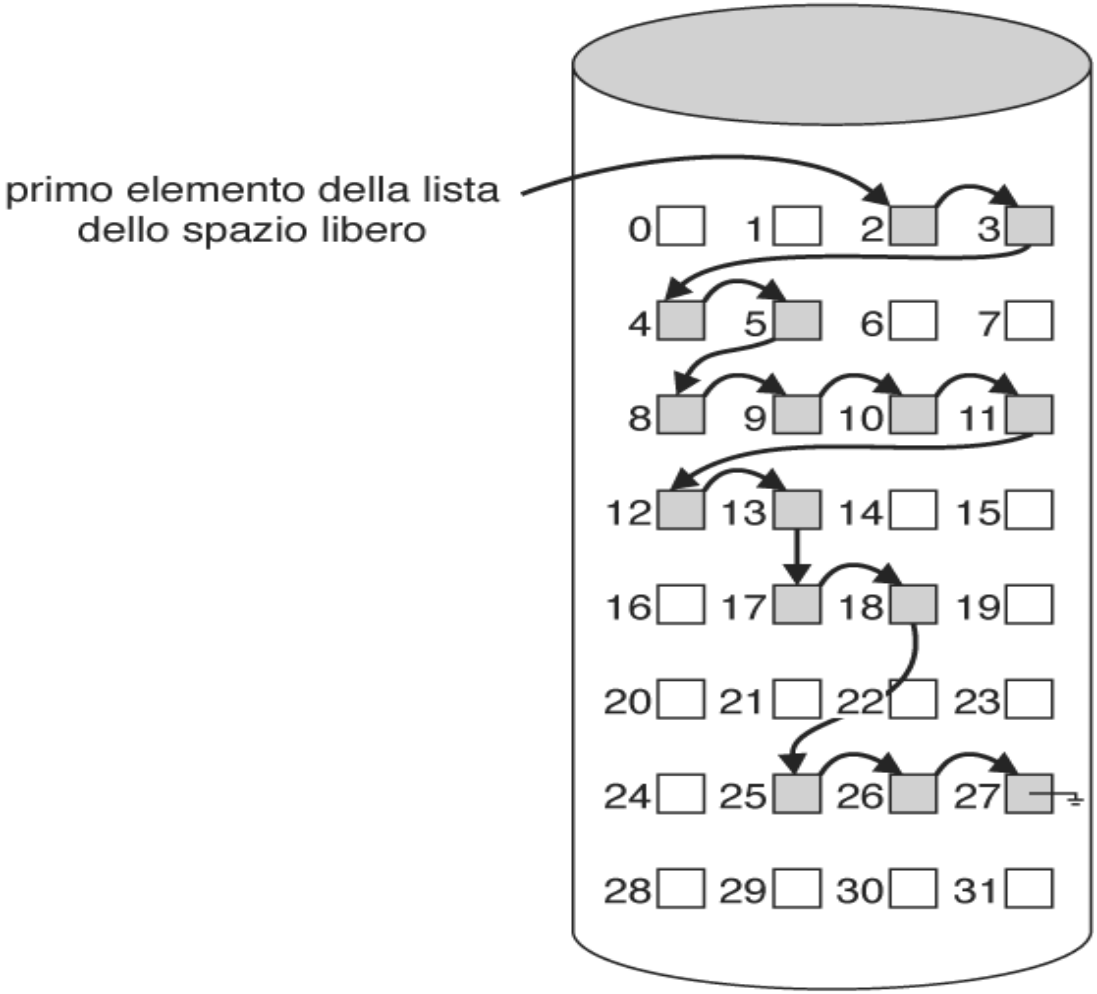
- Soluzione alternativa

  Lista concatenata (lista blocchi liberi)

  - Bassa efficienza nella ricerca
  - Non c'è spreco di spazio

- **Raggruppamento di blocchi** per migliorare le prestazioni. Si memorizza l'indirizzo di *n* blocchi liberi nel primo blocco libero e nell'ultimo blocco l'indirizzo dei prossimi *n* blocchi liberi.

Lista concatenata dei blocchi liberi



File System Distribuiti

- Un **file system distribuito** è un file system residente su computer differenti che offre una vista integrata dei dati memorizzati sui diversi dischi remoti.

- Esempi di file system distribuiti:
  - NFS
  - AFS
  - Coda
  - Plan9
  - xFS

Network File System (NFS)

- Originariamente sviluppato alla Sun Microsystems per le sue workstation con sistema operativo UNIX.
- E’ un modello per integrare file system differenti.
- Basato sull’idea che ogni file server fornisce una vista unificata del suo file system locale.
- NFS può essere usato su gruppi eterogenei di computer.

Architettura di NFS