File System

Concetti principali

Sommario

- Concetto di File e directory
- Metodi di accesso
- Struttura del File system

File system

- Per la maggior parte degli utenti il **file system** è l'aspetto più visibile del SO.
- Il file system rappresenta una astrazione del modo con cui i dati sono allocati e organizzati su un dispositivo memoria di massa.
- Attraverso il file system, Il SO offre una visione **logica** uniforme della memorizzazione delle informazioni sui diversi supporti (dischi, nastri, CD, ecc.).
- Elemento di base nella gestione a livello logico della memoria di massa è il **file**.

File

- Un **file** è un insieme di informazioni correlate e registrate nella memoria secondaria con un nome.
- Il file è la più piccola unità di memoria secondaria assegnabile all'utente che può scrivere sulla memoria secondaria solo registrando un file.
- Il file può avere una sua struttura interna, a seconda del formato o del tipo. Per esempio un file eseguibile è una sequenza di sezioni di codice che possono essere caricate ed eseguite.

Attributi

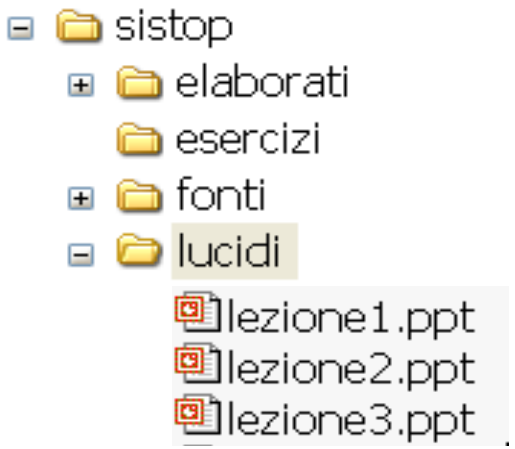
- Principali attributi del file sono:
  - **Nome:** è l'unica informazione mantenuta in un formato simbolico leggibile da una persona.
  - **Tipo**: usato nei sistemi che supportano più tipi diversi di file.
  - **Locazione**: puntatore al dispositivo e alla locazione di quel file sul dispositivo.
  - **Dimensione**.

Attributi

- Altri importanti attributi del file sono:
  - **Protezione**: informazioni di controllo che specificano chi può leggere, scrivere o eseguire il file.
  - **Ora, data e identificazione dell'utente**:informazioni usate per funzioni di protezione dei dati e di monitoring. Possono essere riferite a:
    - Creazione
    - Ultimo utilizzo
    - Ultima lettura

Directory

- Il file system può essere molto ampio (contenere molte migliaia di file) e tipicamente è organizzato in una struttura gerarchica che mantiene gli elementi terminali (i file) all'interno di contenitori detti **directory**.



File in uso

- Il sistema mantiene in memoria l'elenco dei file in uso in una apposita tabella detta **tabella dei file aperti**
- Quando viene iniziata una operazione sul file, il file viene inserito nella tabella in modo che ogni successiva operazione non comporti fasi di ricerca

Informazioni sui file aperti

- A ciascun file aperto sono associate diverse informazioni:
  - **Puntatore alla posizione corrente nel file**: per il supporto a read e write. Se più processi operano sul file esistono più puntatori alla posizione corrente.
  - **Contatore delle aperture**: se più processi possono aprire il file, il SO deve tenere conto delle aperture avvenute e quando sono tutte chiuse, rimuovere il file dalla tabella dei file aperti.
  - **Posizione su disco del file**: quando il file è aperto viene memorizzata la sua posizione sul dispositivo per evitare di doverlo cercare nuovamente ad ogni operazione.

Struttura dei file

- Tipicamente il SO non si occupa di gestire la **struttura interna** associata ai diversi tipi di file, ma lascia che siano le applicazioni a definirla e utilizzarla.
- Per il SO i file sono sostanzialmente insiemi ordinati di byte e le convenzioni che danno un significato specifico ai byte (cioè il formato dei file) sono gestiti a livello applicazione.
- I vantaggi nel delegare la gestione dei formati alle applicazioni sono diversi:
  - Dimensioni del codice del SO: per gestire tutti i formati il sistema dovrebbe essere enorme. Lasciando il controllo alle applicazioni si limita la dimensione del SO.
  - Flessibilità nella definizione di nuove strutture: quando occorre gestire nuovi formati non è necessario modificare il file system e il SO ma è sufficiente dotarsi delle applicazioni appropriate.
- Tutti i SO supportano almeno il formato dei file eseguibili, per poterli caricare in memoria ed eseguire.

Struttura interna del file

- Si noti che :
  - lo spazio fisico è allocato per blocchi, è probabile che l'ultimo blocco di un file sia usato parzialmente, ovvero che una parte dell'ultimo blocco di un file vada sprecata.
  - Maggiore è la dimensione del blocco, maggiore è la frammentazione interna legata a questo fenomeno.

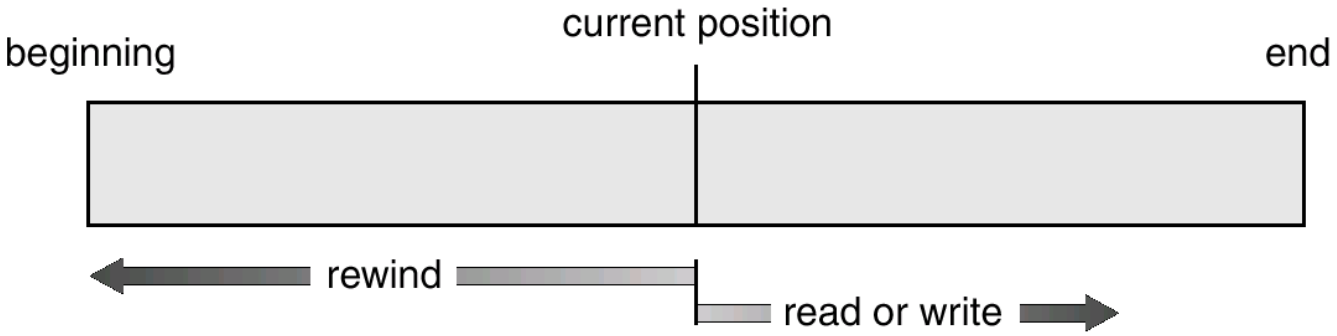
Metodi di accesso

- Il SO può fornire diversi metodi di accesso ai file:
  - **Sequenziale**: le informazioni del file vengono elaborate in ordine, un record dopo l'altro.
    - nastri
  - **Diretto**: il file è costituito da un insieme ordinato di blocchi a cui si accede direttamente.
    - dischi
  - **Attraverso indice**: al file vero e proprio è associato un file indice con lo scopo di velocizzare le ricerche.
    - Database

Accesso sequenziale

- Le informazioni del file vengono elaborate in ordine, un record dopo l'altro. E' il più utilizzato (ES: compilatori).
- Operazioni:
  - **Lettura**: **read**, legge la prossima porzione del file e avanza il puntatore di posizione.
  - **Scrittura**: **write**, scrive i nuovi dati in coda al file e avanza l'end of file.
  - **Reset**: riposiziona il puntatore di posizione a inizio file.

Accesso sequenziale



Accesso diretto

- Il modello i riferimento per l'accesso sequenziale sono i dispositivi come il nastro che consentono solo accessi in sequenza.
- I dispositivi ad accesso casuale (nel senso di non sequenziale) sono modello per un altro metodo di accesso al file: **l'accesso diretto** (o **accesso relativo**) in cui il file è costituito da record logici di lunghezza fissa e viene considerato come un insieme ordinato di blocchi (record) numerati che possono essere acceduti in modo arbitrario.
- Il **numero di blocco** è tipicamente assegnato in modo **relativo** (ciascun file inizia con il suo blocco 0).

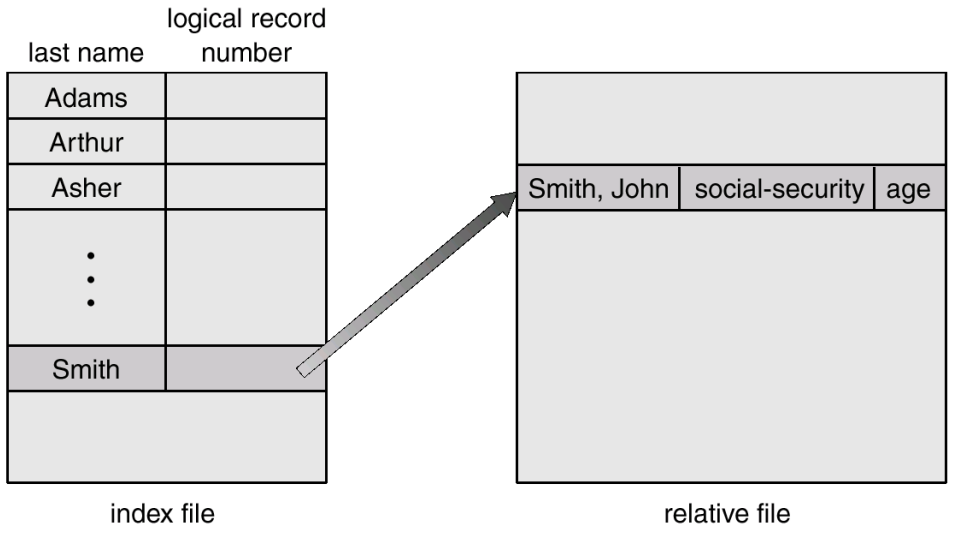
Accesso diretto

- Le istruzioni di lettura e scrittura devono quindi specificare su quale blocco devono essere eseguite:
  - **Lettura**: **read(n)**, legge il blocco con posizione relativa n.
  - **Scrittura: write(n)** scrive il blocco con posizione relativa n.
  - **Riposizionamento**: **seek(n)**, si posiziona sul blocco
- L'accesso sequenziale è facilmente realizzabile se si ha a disposizione l'accesso diretto (cp means current position):

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp*+1;
write next	*write cp*; *cp* = *cp*+1;

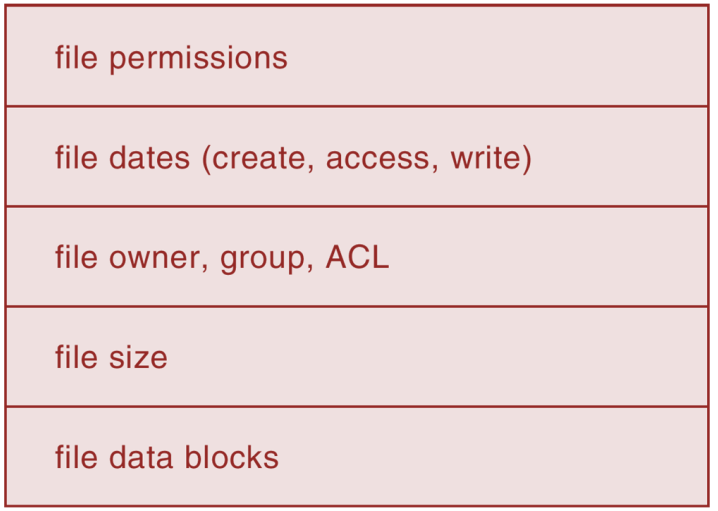
Indici

- Un altro metodo di accesso, che offre supporto alla ricerca veloce di occorrenze in un file, è quello che prevede la costruzione di un **indice**.
- L'indice contiene una o più chiavi di ricerca a cui sono associati i puntatori ai diversi blocchi del file.
- Il file contenente l'indice viene quindi associato ad un file di contenuti.



Strutture del file system: File Control Block

- Per il sistema operativo i file vengono memorizzati in un opportuno descrittore, detto File Control Block che contiene, tra l'altro:
  - Nome
  - Proprietario
  - Dimensioni
  - Permessi di accesso
  - Posizioni dei blocchi



- Il descrittore occupa un blocco su disco.
- Il descrittore si chiama inode nei file system unix.
- Altri blocchi sono utilizzati come blocchi dei dati.

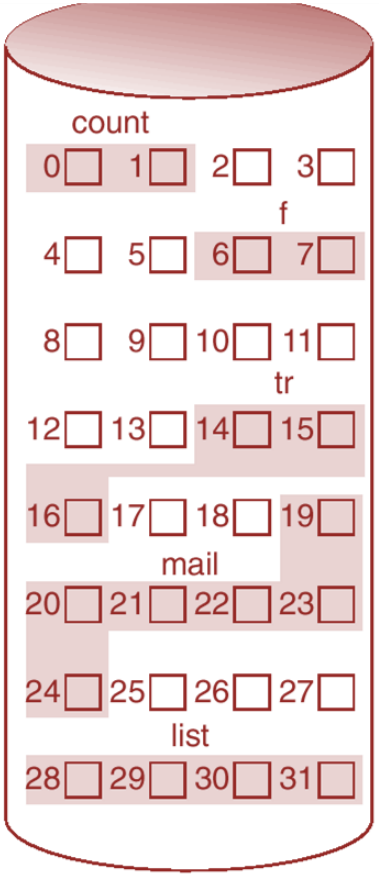
Allocazione dei blocchi

- Un elemento che incide fortemente sulle prestazioni è il **metodo di allocazione dei blocchi**, che specifica come i blocchi del disco sono allocati ai diversi file.
- Ci sono sostanzialmente tre metodologie per allocare i blocchi:
  - Contiguous allocation (allocazione **contigua**).
  - Linked allocation (allocazione **concatenata**).
  - Indexed allocation (allocazione **indicizzata**).

Allocazione contigua

- Nell'**allocazione contigua** ogni file deve occupare un insieme di blocchi contigui del disco.
  - Gli indirizzi del disco definiscono un ordinamento lineare dei blocchi.
  - L'accesso sequenziale al blocco **b+1** seguente all'accesso al blocco **b** non richiede spostamento della testina.
  - L'accesso diretto è fatto semplicemente calcolando la posizione assoluta in base a quella relativa.
  - Il file è definito dalla posizione iniziale e dalla lunghezza.
  - L'accesso è molto semplice.

Allocazione contigua



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Allocazione contigua

- La difficoltà sta nel reperire una porzione di disco sufficientemente grande da contenere tutto il file: il sistema di gestione dello spazio libero deve risolvere questo problema e alcune soluzioni sono lente.
- Il problema è simile a quello dell'allocazione della memoria e in questo caso le strategie più utilizzate sono **first fit** e **best fit**.
- Il sistema soffre di **frammentazione esterna** che viene risolta con routine di **deframmentazione.**

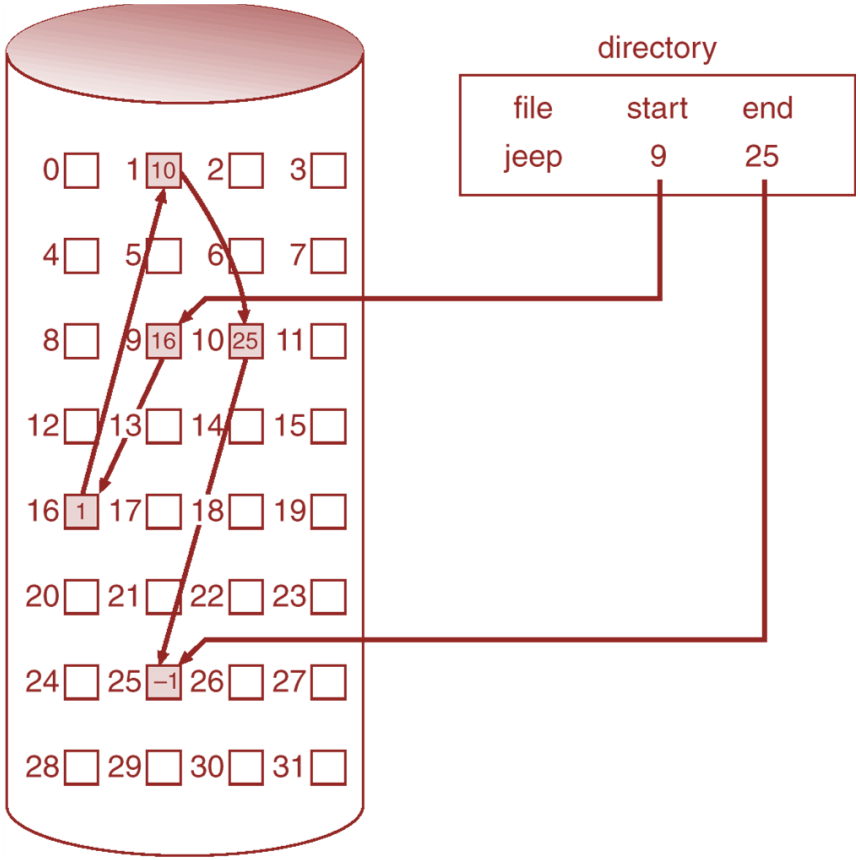
Allocazione contigua

- Se lo spazio allocato è quasi uguale alla dimensione del file il file è difficilmente **estendibile**, poiché occorre **copiare** il contenuto del file in una nuova allocazione più ampia prima di consentire l'estensione.
- Se per evitare la copia si alloca uno spazio più grande della dimensione attuale del file (**preallocazione**), si genera **frammentazione interna**.
- Si possono usare meccanismi per compattare lo spazio, recuperando quello perso in frammentazione esterna mediante copie di spostamento (ma l'operazione è lenta e non risolve la frammentazione interna).

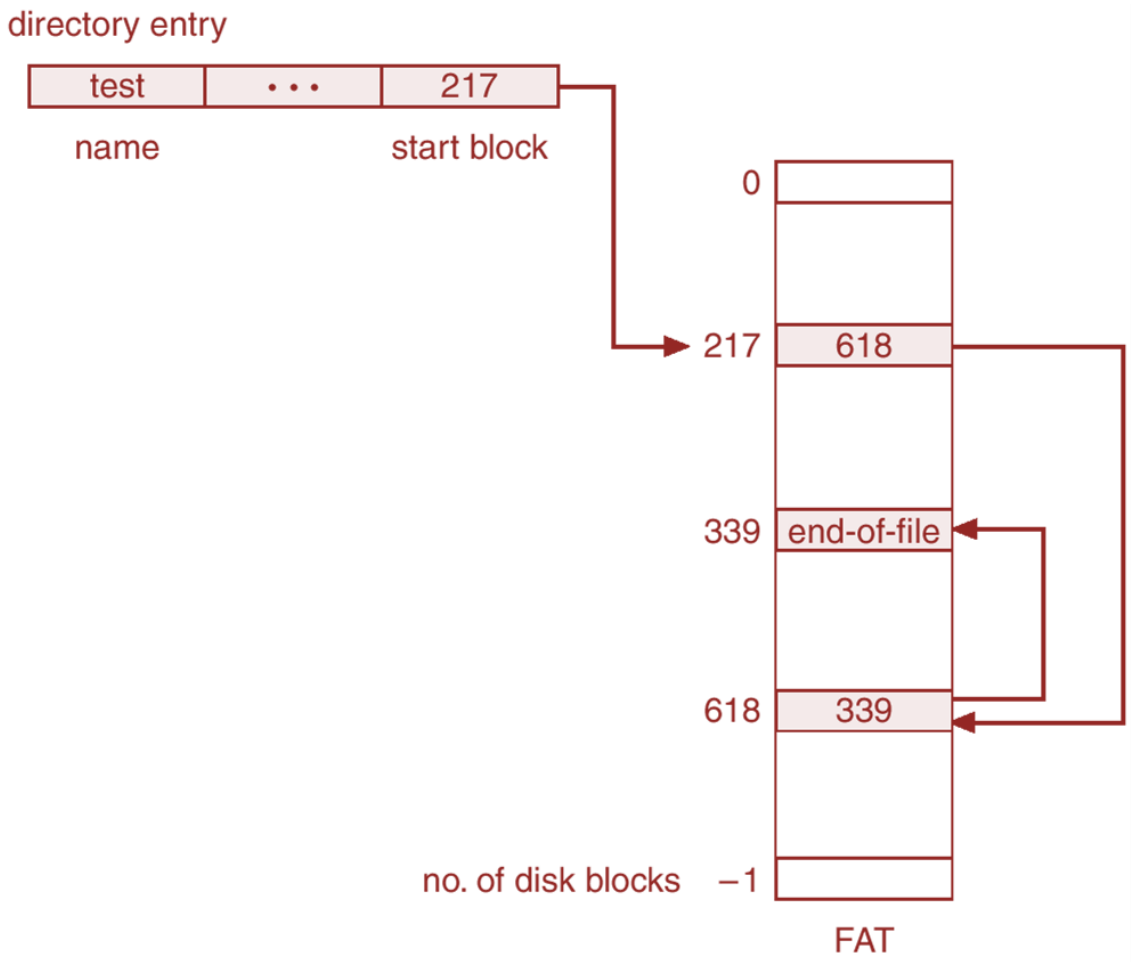
Allocazione concatenata

- Per risolvere i problemi di frammentazione introdotti dall'allocazione contigua si può utilizzare l'**allocazione concatenata**.
- Ogni file è costituito da una lista concatenata di blocchi del disco che possono essere distribuiti in qualunque parte del disco stesso.
- La directory contiene un puntatore al primo blocco del file.

Allocazione concatenata



File-Allocation Table



Allocazione concatenata

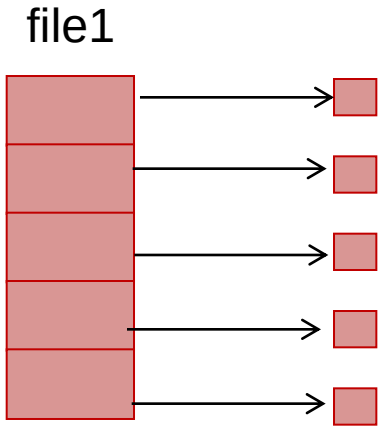
- Non esiste **frammentazione esterna** perché ogni blocco viene allocato singolarmente (quindi tutti i blocchi sono candidati ad essere allocati).
- Non esiste **frammentazione interna** perché non è necessario pre-allocare il file (che cresce al bisogno).
- In questo caso l'allocazione è semplificata a spese dell'accesso.
- **Problemi**:
  - È relativamente efficace nell'accesso sequenziale, perché comporta lo spostamento delle testine al nuovo blocco.
  - È inefficace nell'accesso diretto, perché occorre scorrere il file per trovare la locazione i-esima.
  - Usa molto spazio per i puntatori, poiché ogni blocco c'è un puntatore al successivo.

Clustering

- Per risolvere quest'ultimo problema solitamente non si allocano blocchi ma gruppi (**cluster**) di blocchi (ad esempio 4):
  - **Vantaggi**: meno puntatori, meno movimenti della testina, più semplice la gestione dei blocchi liberi.
  - **Svantaggi**: frammentazione interna (parte del blocco può rimanere inutilizzata).

Allocazione Indicizzata

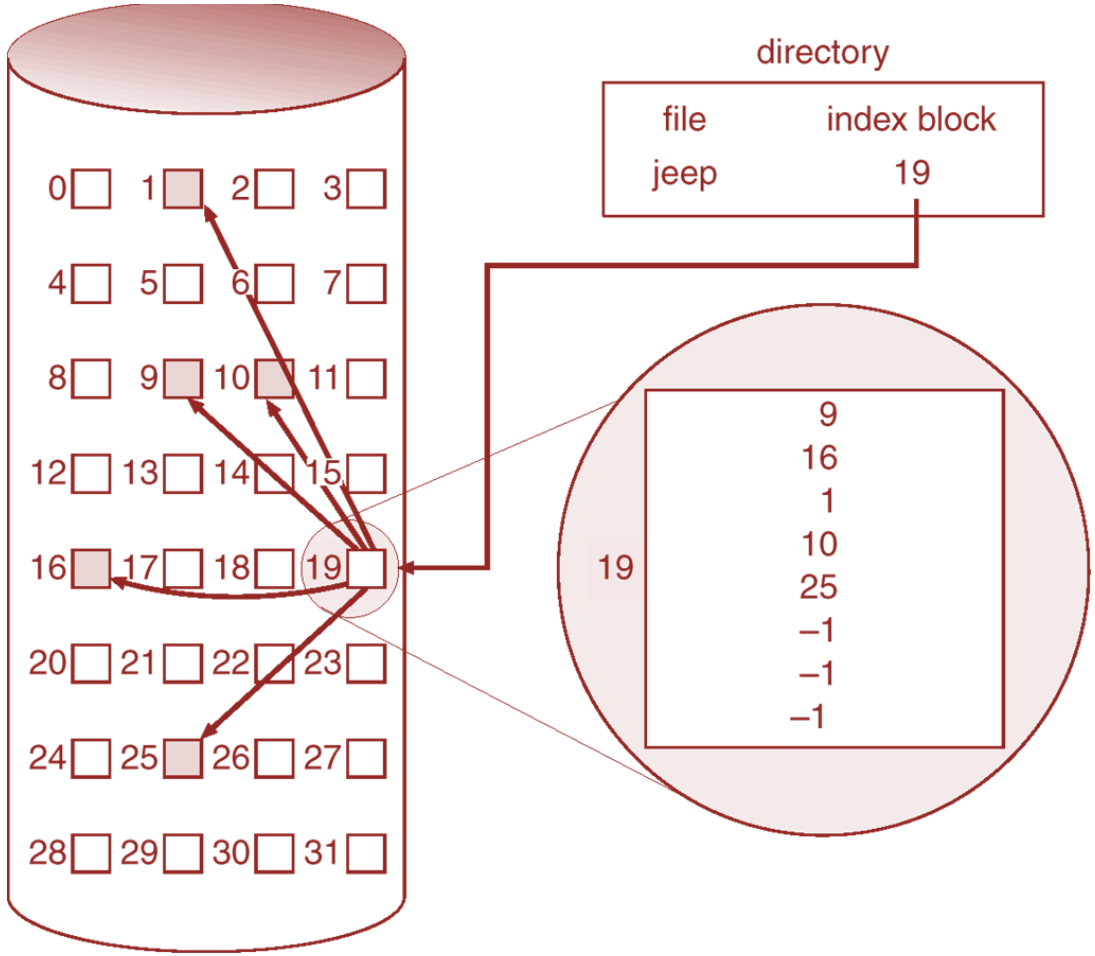
- La maggioranza dei problemi riscontrati con l'allocazione concatenata può essere risolto inserendo tutti i puntatori ai blocchi in una apposita tabella detta blocco indice (**index block**).
- Questo approccio è detto **allocazione indicizzata**.



Allocazione indicizzata

- Ogni file ha il proprio blocco indice in cui l''i-esimo elemento (del blocco indice) punta all'i-esimo elemento del file.
- Supporta l'eccesso diretto **senza frammentazione**.
- Comporta un certo **spreco di spazio** per il blocco indice che deve essere della giusta dimensione per consentire di aumentare le dimensioni del file e contemporaneamente non sprecare spazio.

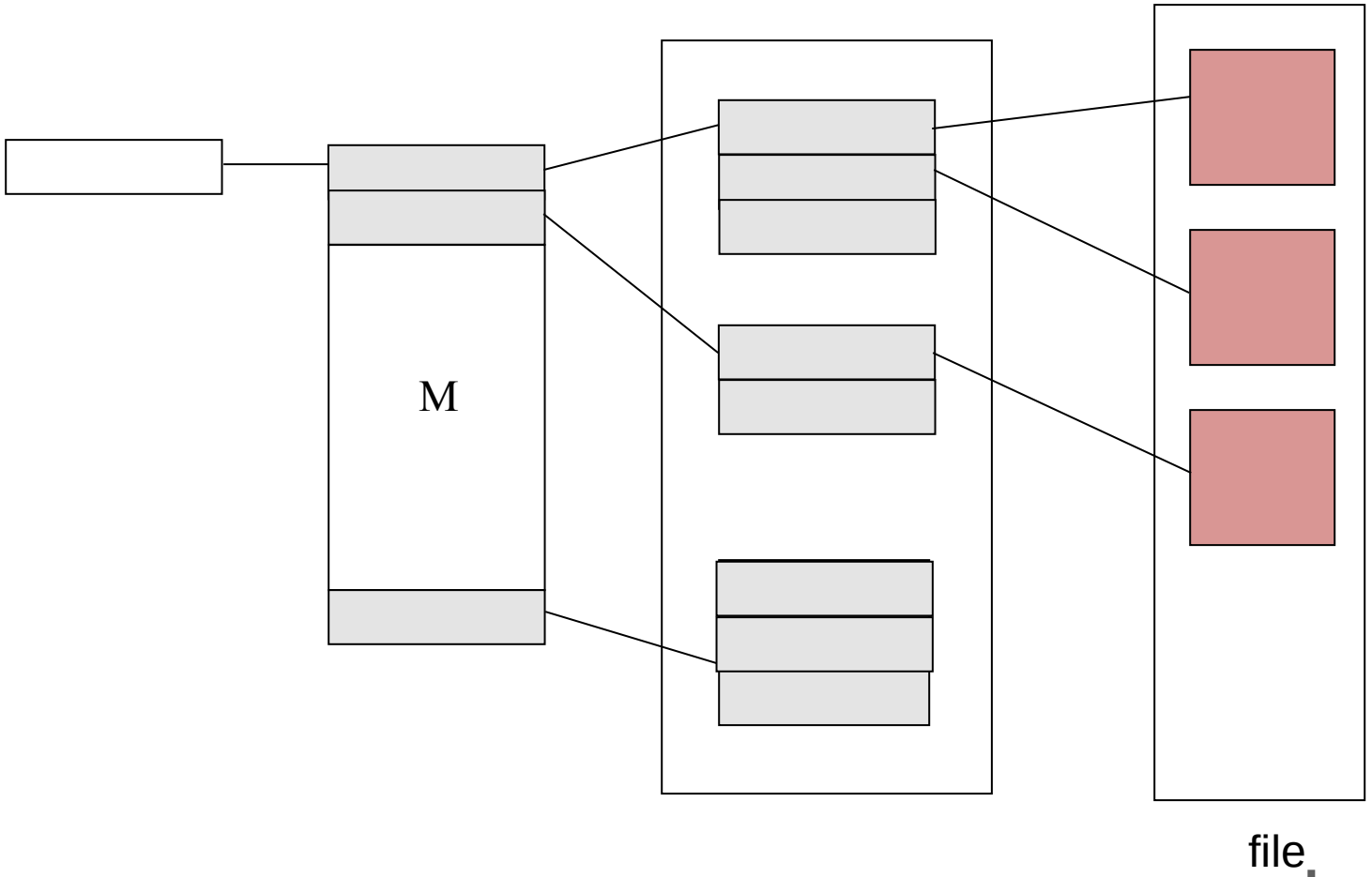
Allocazione Indicizzata



Allocazione indicizzata

- La dimensione e la **memorizzazione** del blocco indice è critica:
  - **Schema concatenato**: il blocco indice occupa esattamente un blocco. Se non è sufficiente un blocco l'ultimo puntatore del blocco indice punta ad un altro blocco indice.
  - **Indice multilivello**: il blocco indice di primo livello punta ad altri blocchi indice di secondo livello e così via
  - **Schema combinato**.

Multilivello



Allocazione dei Blocchi
Schema combinato

- Usato negli **inode** di alcune versioni di Unix.
- Una parte dei puntatori del blocco indice puntano direttamente ai blocchi del file (**blocchi diretti**).
- Una parte punta invece a indici multilivello (**blocchi indiretti singoli, doppi o tripli**)
- Se il file è piccolo si usano solo i blocchi diretti, più è grande e più indirezioni vengono coinvolte.

Allocazione dei Blocchi, Schema combinato

inode