Struttura del File System

Per l'anno accademico 2015-16, studiare solo le slide:

69, 70	struttura del file system,
73, 74	modulo di organizzazione dei file e file system logico
79	file control block (inode)
82	file descriptor per file aperti
88, 89	virtual file system
92, 111, 112	allocazione blocchi, schema conbinato: inode

Sommario

- Struttura del file system
- File Control Block
- Allocazione
- Gestione dello spazio libero
- Implementazione delle directory
- Prestazione
- Ripristino

Struttura del file system

- La memoria secondaria è costituita dai dischi, su cui sono memorizzati i file.
- I trasferimenti tra memoria centrale e dischi si effettuano per blocchi, composti da uno o più settori.
- Il sistema operativo può far uso di uno o più file system, ciascuno composto da più livelli che mappano la logica del file system sulla memoria secondari vera e propria.

Strutture dati (o metadati)

- In un file system si usano molte strutture dati, che variano nei diversi file system/sistemi operativi ma rispondono a principi generali.
- Queste strutture sono mantenute in parte su disco e in parte in memoria:
  - su **disco**, prevalentemente per ragioni di dimensione;
  - In **memoria**, prevalentemente per ragioni di velocità.

Strutture su disco

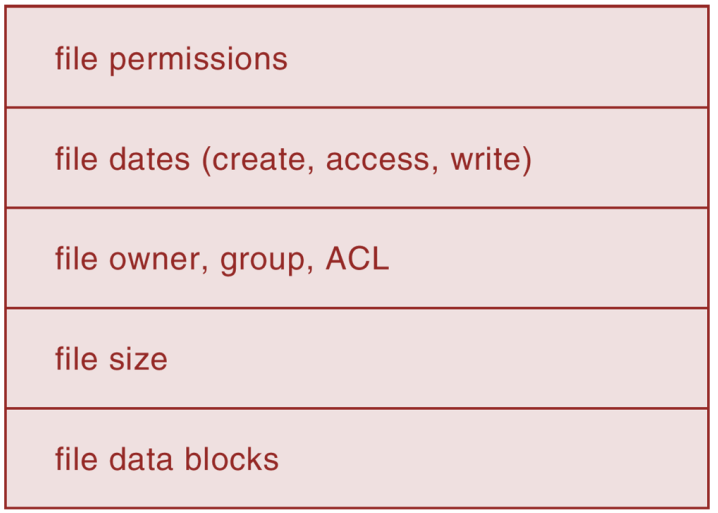
- Tra le strutture memorizzate su disco ci sono:
  - Il blocco di controllo dell'avviamento (**boot control block**), contenente le informazioni necessarie all'avviamento del sistema operativo;
  - I blocchi di controllo dei volumi (**volume control block**) contenenti i dettagli relativi a ciascun volume;
  - Le **strutture delle directory** utilizzate per memorizzare i file;
  - I blocchi di controlli dei file (**file control block**) che contengono le informazioni sui file.

Strutture in memoria

- Tra le strutture memorizzate su RAM:
  - La tabella di montaggio che contiene le informazioni riguardanti i volumi (partizioni) montati
  - La struttura delle directory relativa alle directory accedute recentemente dai processi
  - La tabella generale dei file aperti, contenente copia del FCB dei file aperti
  - I buffer per i blocchi dei file, durante la lettura e scrittura

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 si chiama inode nei file system unix**

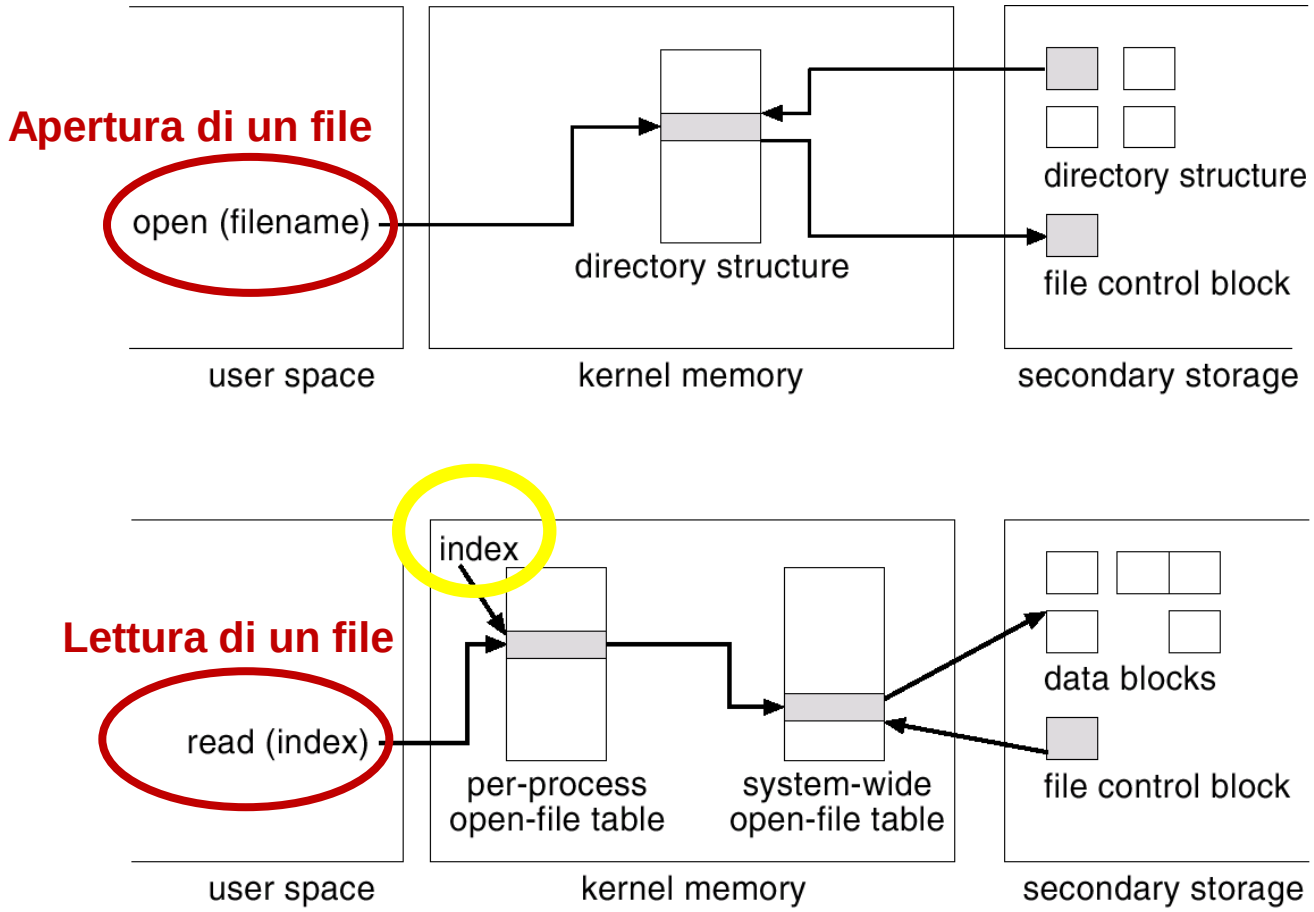
FCB: creazione

- Le applicazioni per creare un nuovo file effettuano una chiamata al file system logico (system call) che alloca un nuovo FCB.
- Il sistema carica quindi la directory appropriata in memoria centrale, la aggiorna con il nuovo FCB e la risalva su disco.
- Alcuni SO (incluso Unix e Linux) trattano le directory esattamente come i file e li distinguono con un campo apposito.
- Una volta creato il file per essere letto o scritto deve essere **aperto** (attraverso la system call **open**)

FCB: apertura

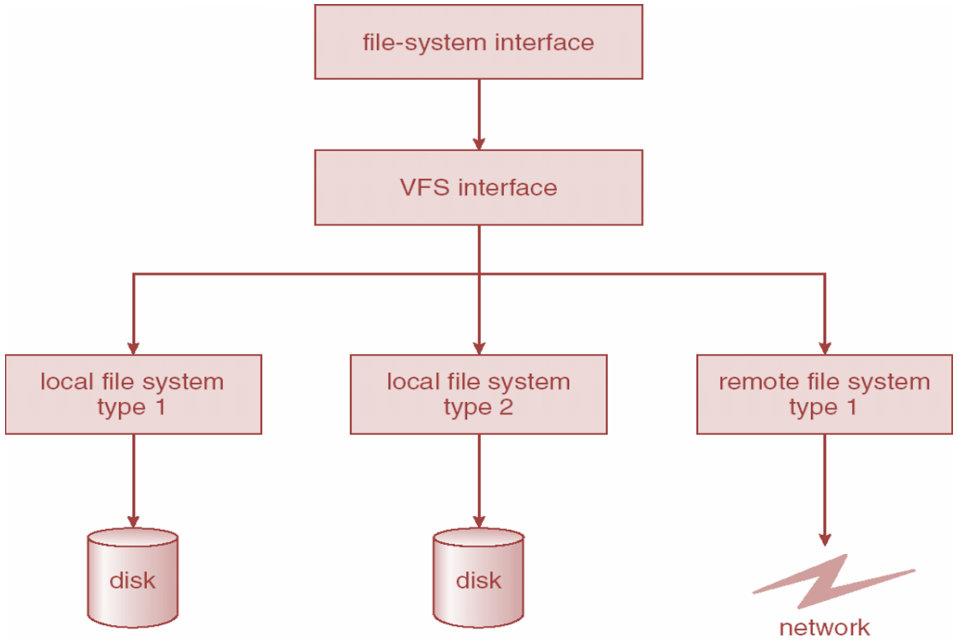
- La open:
  - Controlla se il file sia già in uso da parte di altri processi
  - Per farlo lo cerca nella tabella GENERALE dei file aperti e nel caso lo trovi, la aggiorna riferendo anche al nuovo processo
  - Se non lo trova, lo cerca nella directory e aggiunge l'opportuno elemento alla tabella GENERALE dei file aperti copiando in tabella anche il FCB originale
  - Aggiorna la tabella dei file aperti del PROCESSO, in cui copia il puntatore al FCB nella tabella GENERALE. In questa tabella locale al PROCESSO è mantenuto il puntatore alla posizione di lettura/scrittura

File Control Block: **file descriptor**



Virtual File System

- Per gestire più file system i sistemi operativi devono introdurre livelli d'astrazione che rendano efficiente la scrittura delle applicazioni
- In particolare nei sistemi Unix-like si utilizza un approccio object-oriented detto **Virtual File Systems** (VFS)



VFS

- Il Virtual File System:
  - Permette di utilizzare la stessa interfaccia alla system call (API) nei diversi file systems
  - Consente al programmatore di usare operazioni generiche e indipendenti dal file system, indipendentemente dai dettagli implementativi
  - Recapita la chiamata al file system specifico.
  - L'API in realtà è una interfaccia al Virtual File System piuttosto che ad uno specifico file system

Prestazioni

- Gli algoritmi di allocazione e gli algoritmi di gestione delle directory hanno un notevole impatto sulle prestazioni del file system
- Vedremo quindi diverse alternative per:
  - L'allocazione dei blocchi:
    - Ci sono diverse soluzioni che vediamo tra un po', e
  - La gestione delle directory:
    - Lista lineare
    - Tabella di hash

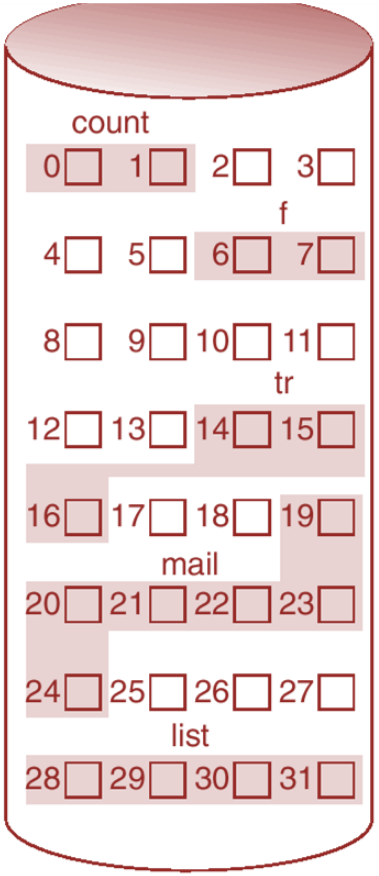
Allocazione dei blocchi

- L'altro 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.**

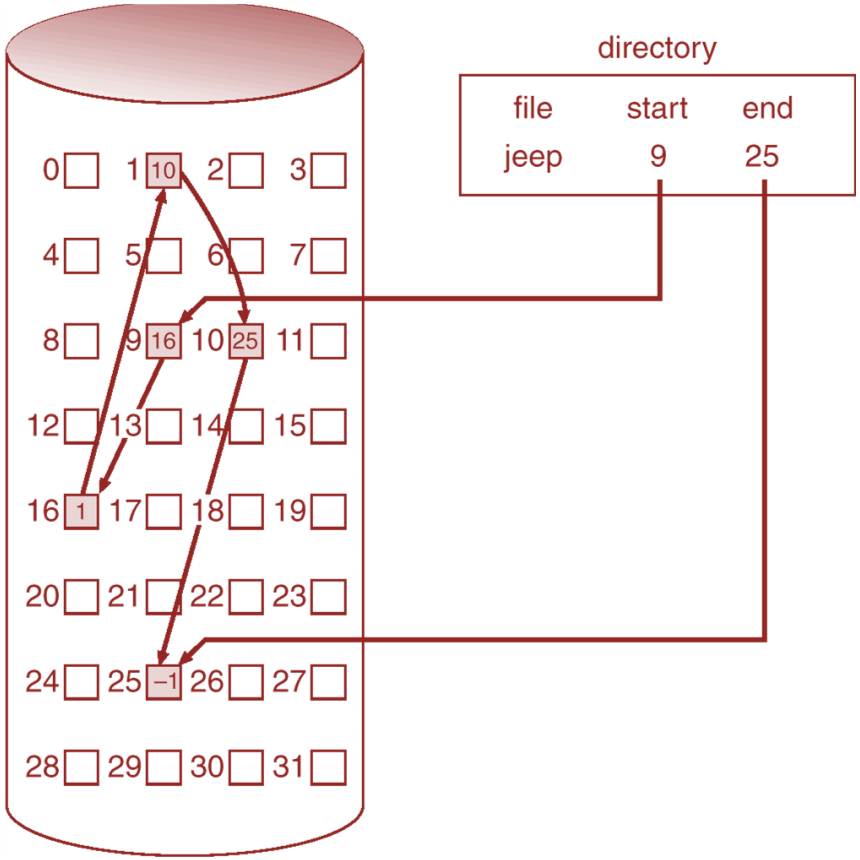
Allocazione contigua

- 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).

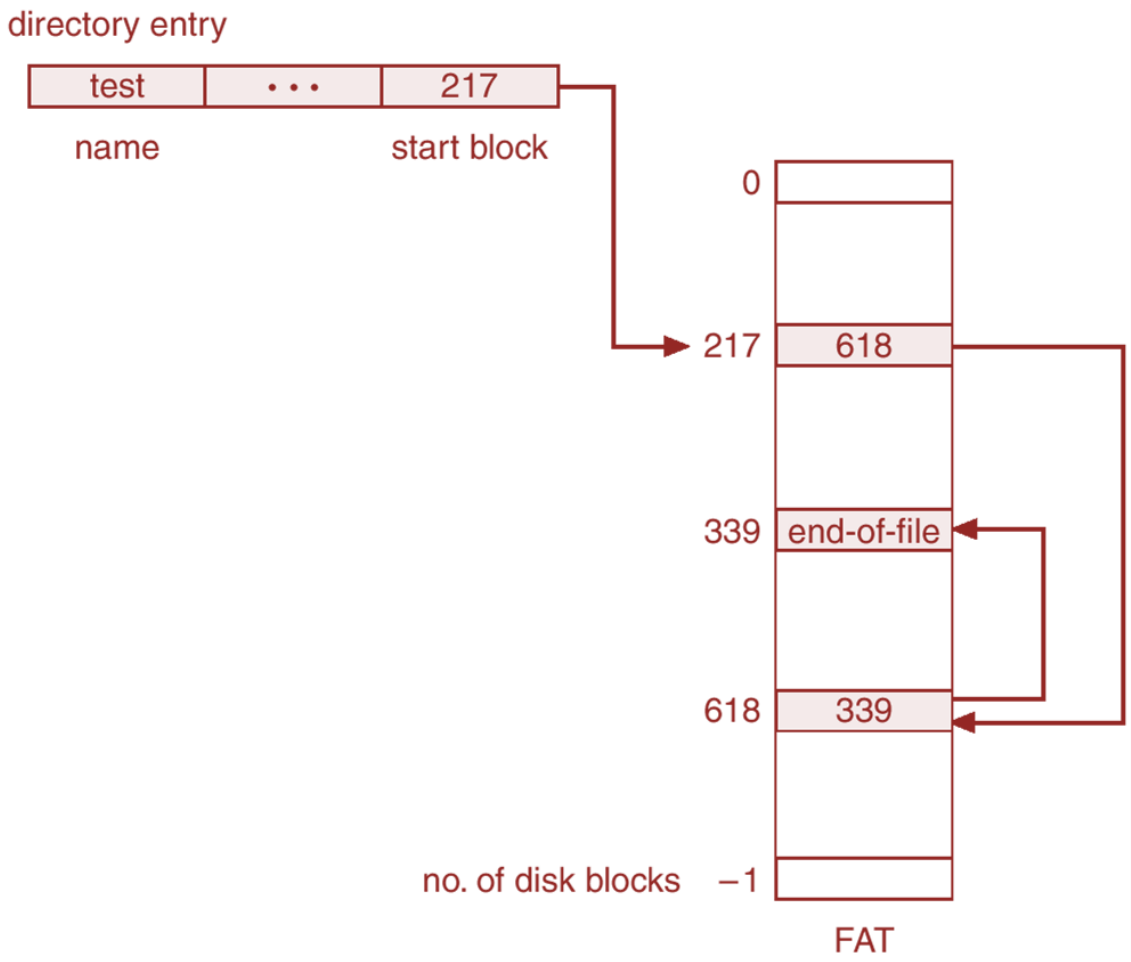
Allocazione concatenata

- 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

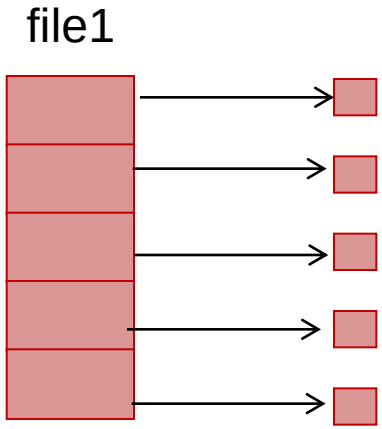


Allocazione concatenata

- 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.

Allocazione Indicizzata

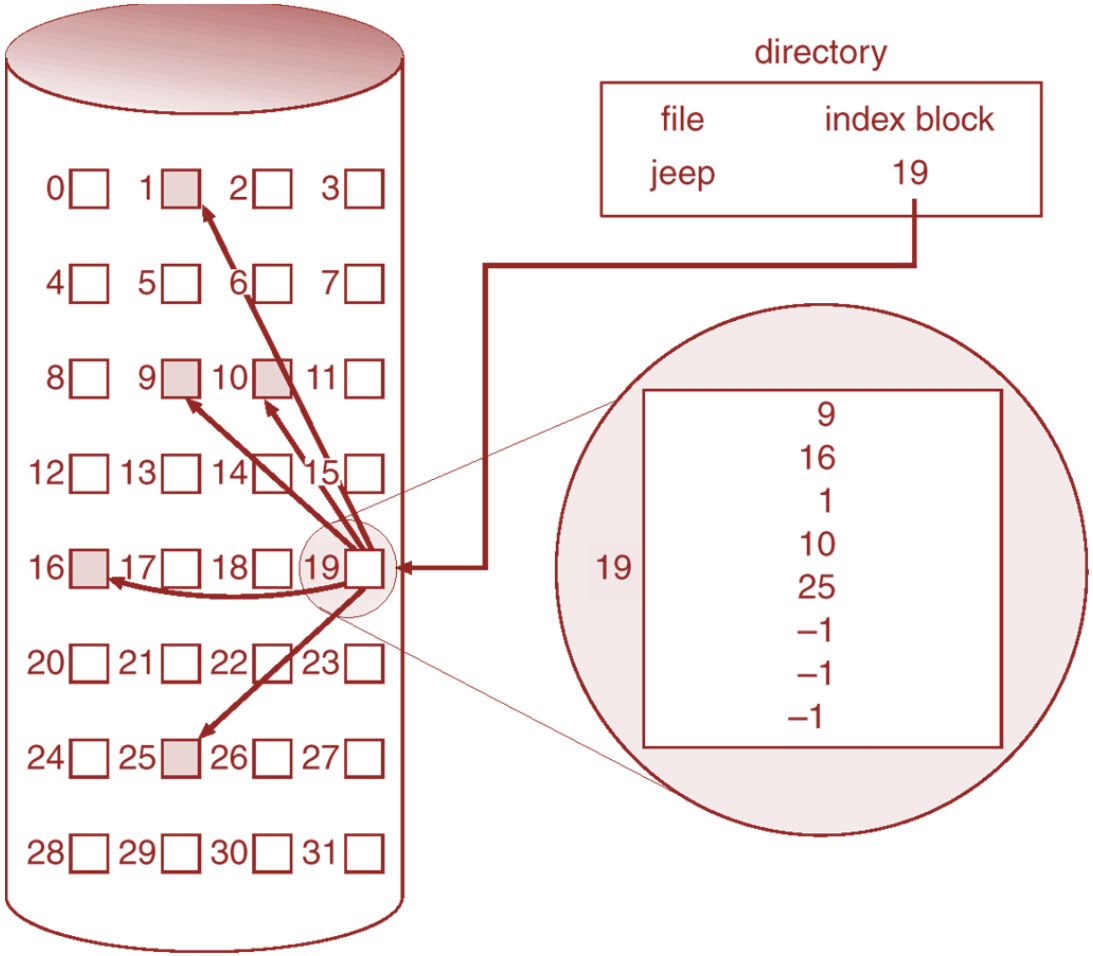
- La maggioranza dei problemi riscontrati con l'allocazione concatenata può essere risolto inserendo tutti i puntatori ai blocchi in una apposita tabella detta blocco indice (**index block**).
- Questo approccio è detto **allocazione indicizzata**.



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.

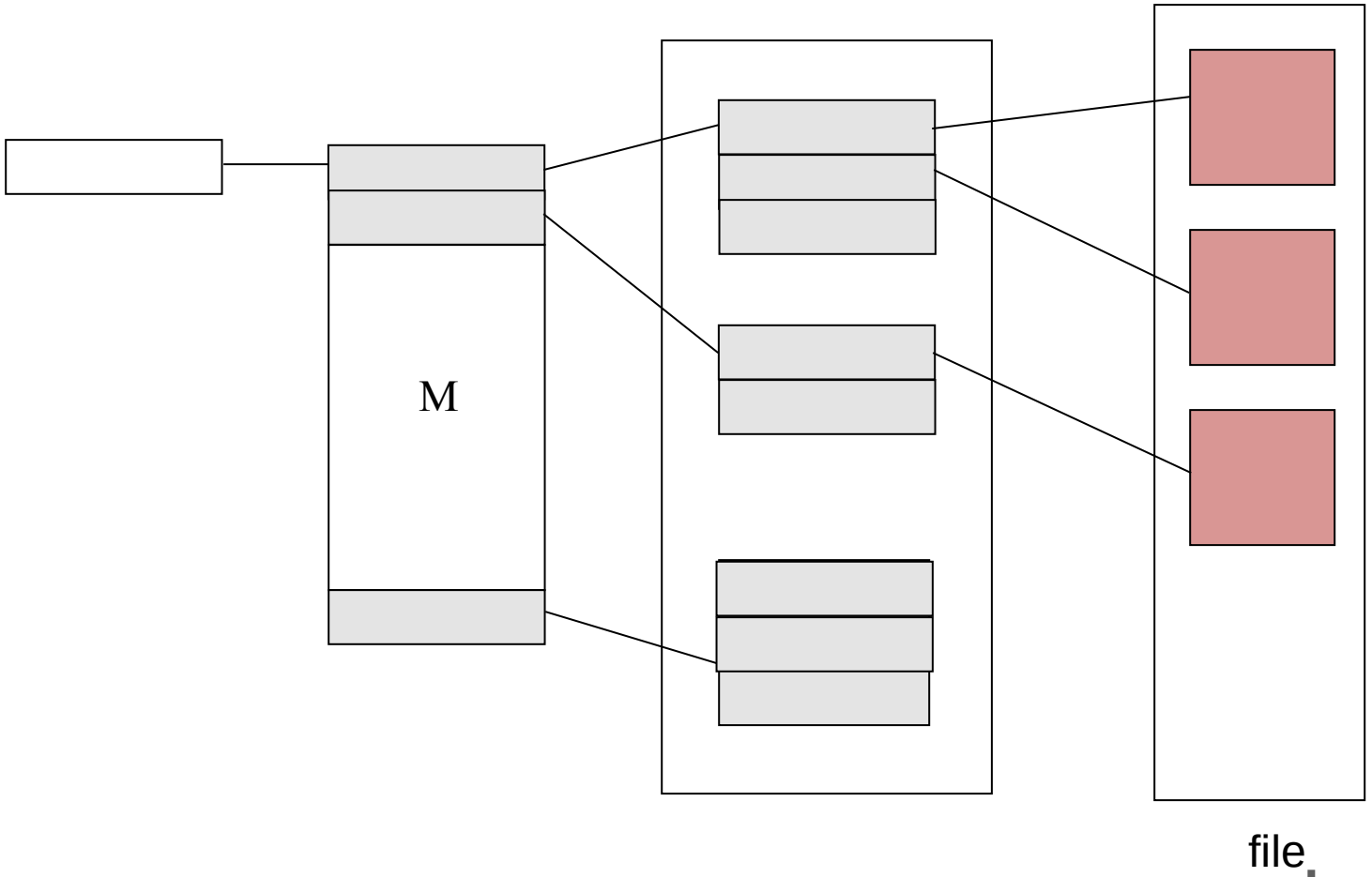
Allocazione Indicizzata



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