Politecnico
di Milano

Il File System

Architettura degli elaboratori

Docente: Ouejdane Mejri
mejri@elet.polimi.it

Sommario

- File
  - Attributi
  - Operazioni
  - Struttura
- Organizzazione
  - Directory
  - Protezione

File

- Il concetto di *file* offre una visone omogenea delle informazioni memorizzate
- La visione non dipende dal tipo di dispositivo fisico su cui le informazioni vengono memorizzate
- Un file è costituito da:
  - Un insieme di informazioni omogenee
  - Un nome simbolico
  - Un insieme di attributi
- Un file può contenere:
  - Dati
  - Programmi
  - Riferimenti

Attributi

- Ad un file sono associati alcuni attributi che ne descrivono alcune caratteristiche
  - *Nome*
    - E' un nome simbolico con cui ci si riferisce ad esso
  - *Tipo*
    - Definisce il tipo dei dati contenuti
    - A volte il tipo viene definito attraverso una estensione del nome
  - *Locazione*
    - E' un puntatore alla posizione fisica sul dispositivo
  - *Dimensione*
    - Dimensione dei dati espressa in bytes o blocchi

Attributi

- *Protezione*
  - Definisce le politiche di gestione degli accessi
- *Ora e Data*
  - Indicano il momento della creazione, dell'ultima modifica o dell'ultimo accesso
- *Proprietario*
  - Indica il nome dell'utente che ha creato il file

Operazioni

- Sui file possono essere compiute diverse operazioni
- Le operazioni vengono svolte attraverso delle richieste di servizi al sistema operativo
- Le operazioni più comuni sono elencate nel seguito

- *Creazione*
  - Viene aggiunto un nuovo file al file system
  - Le operazioni richieste sono:
    - Allocazione
    - Creazione del nuovo descrittore del file
    - Aggiunta del descrittore al file system

Operazioni

- *Scrittura*
  - Aggiunge dati ad un file già creato
  - Per scrivere dati su un file è necessario fornire
    - Il nome del file
    - I dati da scrivere
  - Il SO mantiene un puntatore alla posizione corrente
- *Lettura*
  - Preleva dati da un file già creato
  - Per leggere dati da un file è necessario fornire
    - Il nome del file
    - Un puntatore ad zona di memoria destinazione dei dati
  - Il SO mantiene un puntatore alla posizione corrente

Operazioni

- *Riposizionamento*
  - Sposta la posizione dei puntatori di lettura/scrittura
  - Le operazioni permesse dipendono dal tipo di accesso
  - Spesso viene mantenuto dal file system un solo puntatore valido per la lettura e per la scrittura
- *Cancellazione*
  - Elimina un file
  - Per eliminare un file è necessario specificarne il nome
  - Le operazioni necessarie sono:
    - Deallocazione dello spazio sul dispositivo fisico
    - Aggiunta alla lista dello spazio disponibile sul disco
    - Rimozione del descrittore del file dal file system

Operazioni

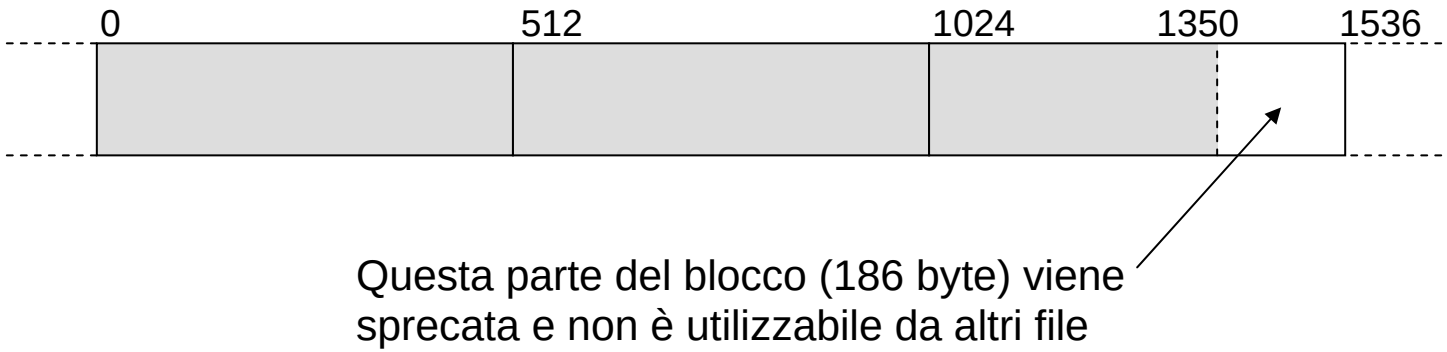
- Tutte le operazioni richiedono l'accesso ad un file
- Il file system deve cercare il file sul dispositivo
  - Per rendere più efficiente la ricerca, il file system mantiene una tabella dei file in uso
  - I file in uso si dicono *aperti*
- Servizi forniti dal sistema operativo
  - *Open*
    - Sulla base del nome individua la posizione del file
    - Copia il descrittore del file nella tabella dei file in uso
  - *Close*
    - Sulla base del nome individua il descrittore del file
    - Elimina il descrittore dalla tabella dei file in uso

Struttura

- Un file system ha
  - *Struttura logica*
    - I dati sono organizzati in unità logiche di lunghezza fissa ma arbitraria dette *blocchi logici (o record)*
      -> Nel sistema UNIX un record è un byte
  - *Struttura fisica*
    - I dati sono organizzati in unità fisiche di lunghezza fissa e dipendente dal dispositivo dette *blocchi fisici*
    - Dimensioni tipiche dei blocchi di unità a disco rigido variano da 32 a 4096 bytes, tipicamente 512
- La struttura logica e fisica sono differenti
  - I dati vengono impaccati prima di essere memorizzati in modo da sfruttare al meglio il dispositivo

Struttura: Esempio

- Tale differenza, unitamente alla dimensione fissata dei blocchi provoca uno spreco di spazio
- Si consideri un file lungo 1350 byte ed un disco con blocchi da 512 bytes



- Questo fenomeno è detto *frammentazione interna*

Struttura: Accesso

- *Accesso sequenziale*
  - I dati sono letti/scritti in sequenza
  - Le operazioni disponibili per tali file sono
    - Lettura e scrittura
    - Posizionamento all'inizio o alla fine del file
    - Posizionamento sul record precedente o successivo
- *Accesso diretto*
  - I dati vengono letti e scritti in una qualsiasi posizione
  - La posizione deve essere specificata in termini di blocco logico, relativamente all'inizio del file
  - I blocchi logici devono avere dimensione fissa per consentire il calcolo della posizione effettiva dei dati

File system

- Le memorie di massa contengono milioni di file
- Necessità di condividere uno o più file
- Tale mole di dati necessita una strutturazione
- Un file system è organizzato in
  - *Partizioni*
    - Contengono insiemi di file correlati
  - *Directory*
    - Una partizione è suddivisa in directory
    - Contengono informazioni sui file e fungono da indice
  - *File*
    - Contengono effettivamente i dati o i programmi

Implementazione del file system

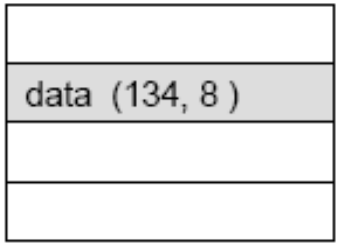
- Come vengono memorizzati i file e le directory
- Come gestire lo spazio su disco
- Quali strutture dati e operazioni sono necessarie al sistema operativo per una gestione efficiente e affidabile del file system

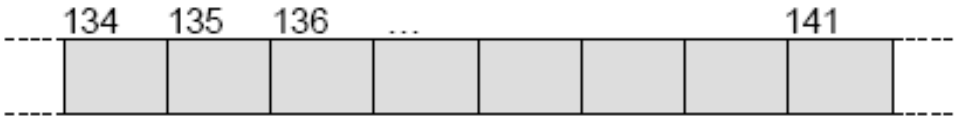
Implementazione dei file

- Necessario mantenere traccia della corrispondenza tra blocchi dei dischi e file
- Esistono diversi metodi:
  - Allocazione contigua
  - Allocazione concatenata
  - Allocazione indicizzata

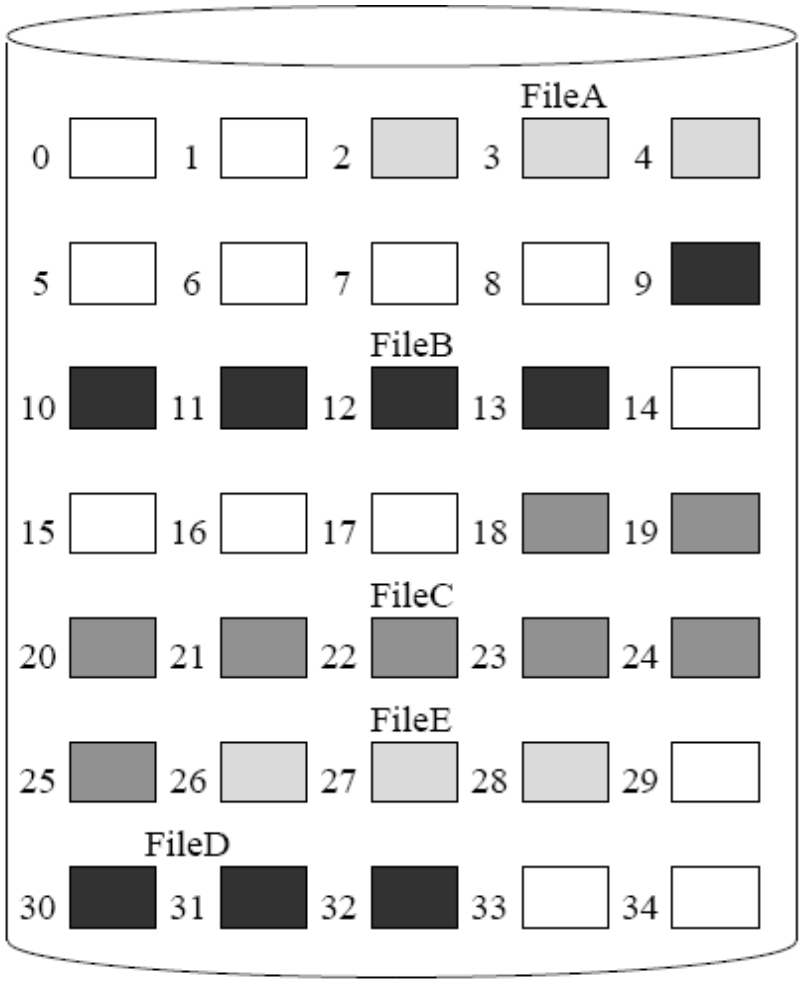
(a) Allocazione contigua

- I blocchi di un file sono *adiacenti*
- Un file di *n* blocchi è memorizzato nelle posizioni adiacenti *b*, *b+1*, ...,*b+n-2*, *b+n-1*
- Un descrittore deve indicare solo la coppia *(b,n)*





(a) Allocazione contigua



File Allocation Table

Nome file	Blocco in.	Lungh
FileA	2	3
FileB	9	5
FileC	18	8
FileD	30	2
FileE	26	3

(a) Allocazione contigua

- I tempi di accesso sono brevi in quanto l'accesso:
  - A due blocchi successivi *b* e *b+1* non richiede lo spostamento della testina in quanto sono sulla stessa traccia
  - Al blocco logico *i*-esimo richiede l'accesso diretto al blocco fisico *b+i*
- Problema: allocazione di spazio per un nuovo file

(a) Allocazione contigua

- Dato un file di *m* blocchi è necessario individuare una porzione di disco costituita da almeno *m* blocchi contigui
- Tre politiche:
  - *First-Fit:* La prima zona, di almeno *m* blocchi, viene usata
  - *Best-Fit:* La zona più piccola, di almeno *m* blocchi, viene usata
  - *Worst-Fit:* La zona più grande, di almeno *m* blocchi, viene usata
- Le tecniche migliori sono le prime due, in particolare il metodo *first-fit* risulta più veloce

(a) Allocazione contigua

Problema

- L'allocazione contigua crea, nel tempo, zone inutilizzate di piccole dimensioni
- Tali zone hanno una bassa probabilità di contenere un file
- Questo fenomeno viene detto *frammentazione esterna*

(a) Allocazione contigua

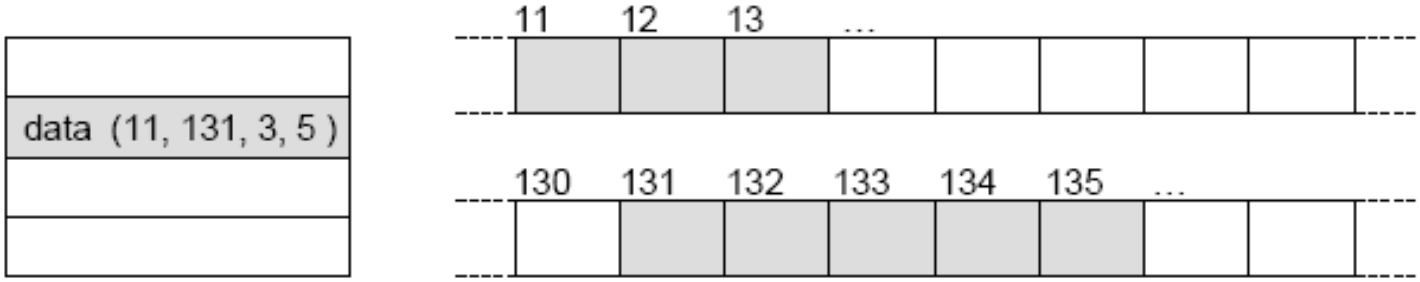
- Prima soluzione: *compattazione dei dischi*
  - I file su un disco vengono letti e memorizzati temporaneamente altrove (ad esempio su un secondo disco)
  - Il disco originale viene completamente cancellato
  - I file vengono riscritti in sequenza, eliminando così gli spazi vuoti
- Questa operazione è molto costosa in termini di tempo e deve essere compiuta con una certa frequenza

(a) Allocazione contigua

- Soluzione migliore: memorizzazione di un file in due zone differenti, ognuna formata da blocchi contigui
- La zona aggiuntiva prende il nome di *extent*
- Sono ancora necessari algoritmi per la ricerca di spazio disponibile

(a) Allocazione contigua

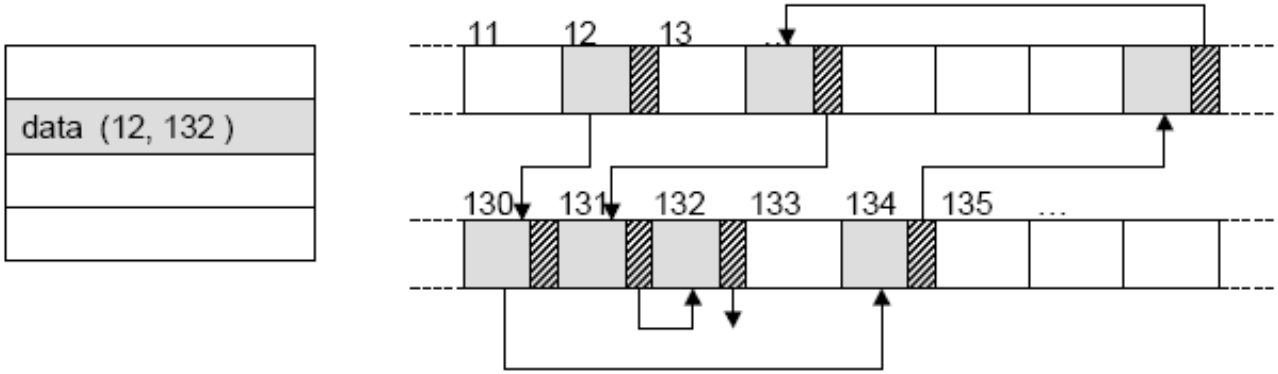
- Un descrittore di file indica *(b, e, n1, n2):*
  - *b:* base della sezione principale
  - *e:* base dell'extent
  - *n1:* dimensioni della sezione principale
  - *n2:* dimensione dell'extent



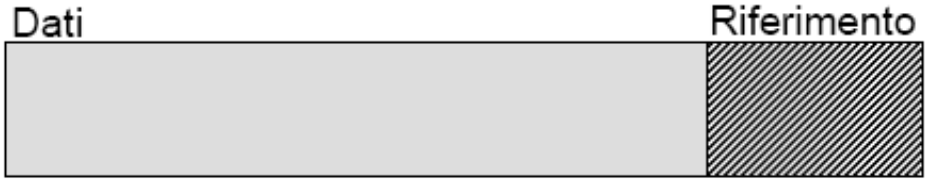
(b) Allocazione concatenata

- L'idea dell'uso di un extent può essere estesa ad un numero maggiore di estensioni
- In questo modo un file è costituito da una *sequenza di blocchi* in cui:
  - Il descrittore contiene il riferimento al primo ed all'ultimo blocco
  - Ogni blocco contiene un riferimento al blocco successivo

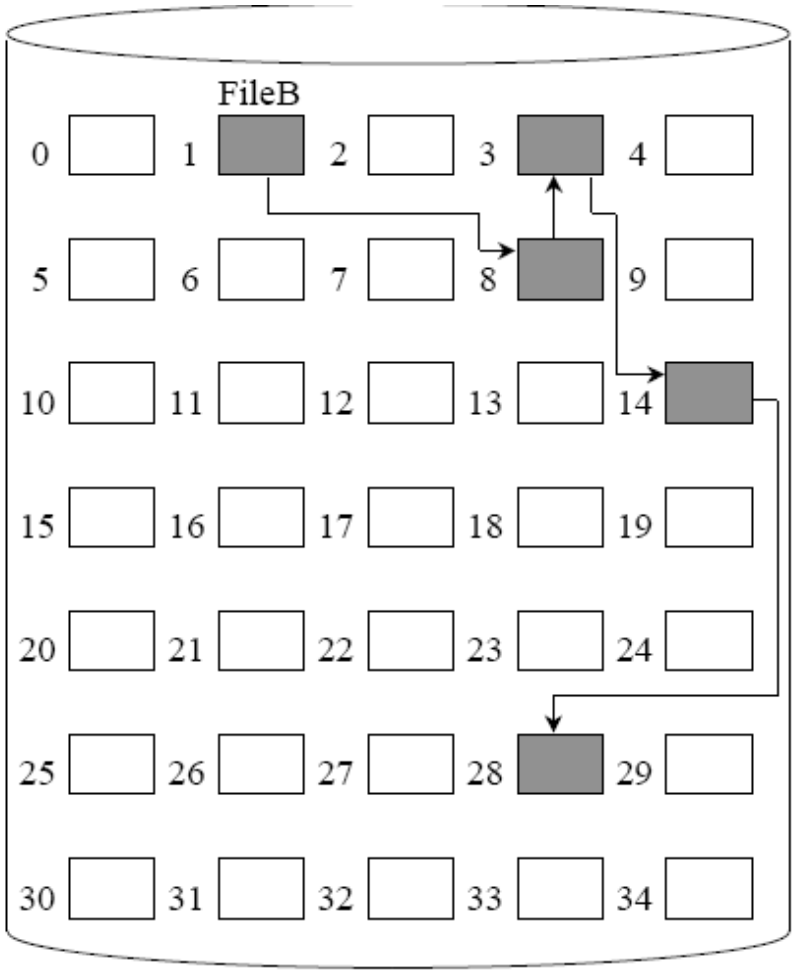
(b) Allocazione concatenata



Struttura di un blocco



(b) Allocazione concatenata



File Allocation Table

Nome File	Blocco in.	Lungh
...	...	...
FileB	1	5
...	...	...

(b) Allocazione concatenata

- Questa soluzione risolve tutti i problemi tipici della allocazione contigua, in particolare:
  - Non si ha frammentazione esterna
- Una parte di ogni blocco è dedicata a contenere un riferimento al blocco successivo
- La memorizzazione dei riferimenti riduce lo spazio disponibile per i dati
- Con blocchi di 512 byte e riferimenti di 4 byte si ha uno spreco di spazio pari allo 0.78% del disco

(b) Allocazione concatenata

- Svantaggi
  - Per l'accesso casuale è comunque necessario scorrere il file dall'inizio fino al blocco desiderato
  - L'accesso ad un file è meno efficiente in quanto può comportare molti riposizionamenti della testina
- Soluzione: raggruppare più blocchi in un *cluster* e prevedere l'accesso a tali gruppi piuttosto che ai singoli blocchi:
  - Si ha un miglioramento delle prestazioni per via del numero minore di riposizionamenti della testina
  - Si ha una riduzione dello spazio utilizzato per i riferimenti
  - Si ha una maggiore frammentazione interna

(b) Allocazione concatenata

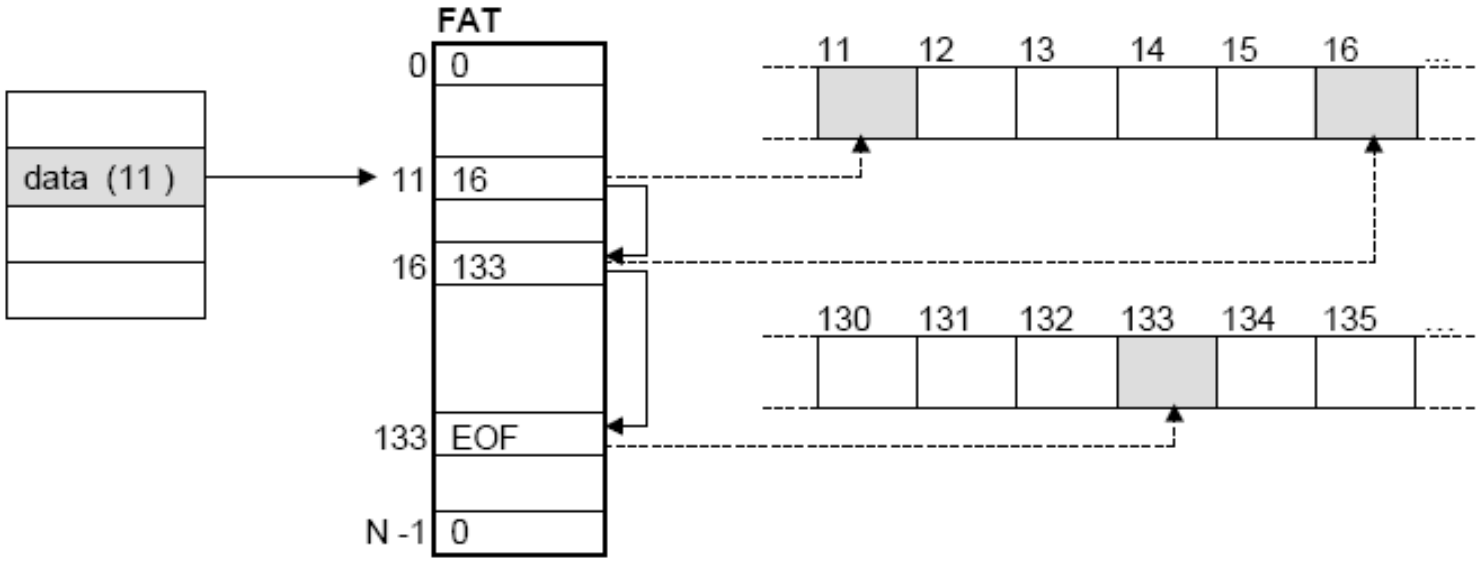
- Le liste di blocchi sono *fragili* in quanto la perdita di un solo riferimento può rendere inaccessibili grandi quantità di dati
- Una soluzione, adottata per esempio da MS-DOS e OS/2, consiste nella costruzione di un indice detto *File Allocation Table* (*FAT*) per ogni partizione

La FAT: File Allocation Table

- La *FAT*:
  - Contiene tanti elementi quanti sono i blocchi del disco
  - Un blocco disponibile è indicato da uno 0 nella tabella
  - L'ultimo blocco di un file è indicato da uno speciale valore *EOF*
  - Ogni elemento della tabella contiene l'indice dell'elemento della FAT che contiene il blocco successivo
- Questo metodo comporta mediamente un tempo di accesso ai file maggiore dell'allocazione concatenata

(b) Allocazione concatenata

- Un descrittore di file deve semplicemente indicare il numero del primo blocco del file
- Le informazioni sulla posizione dei blocchi successivi sono contenute nella FAT



(c) Allocazione indicizzata

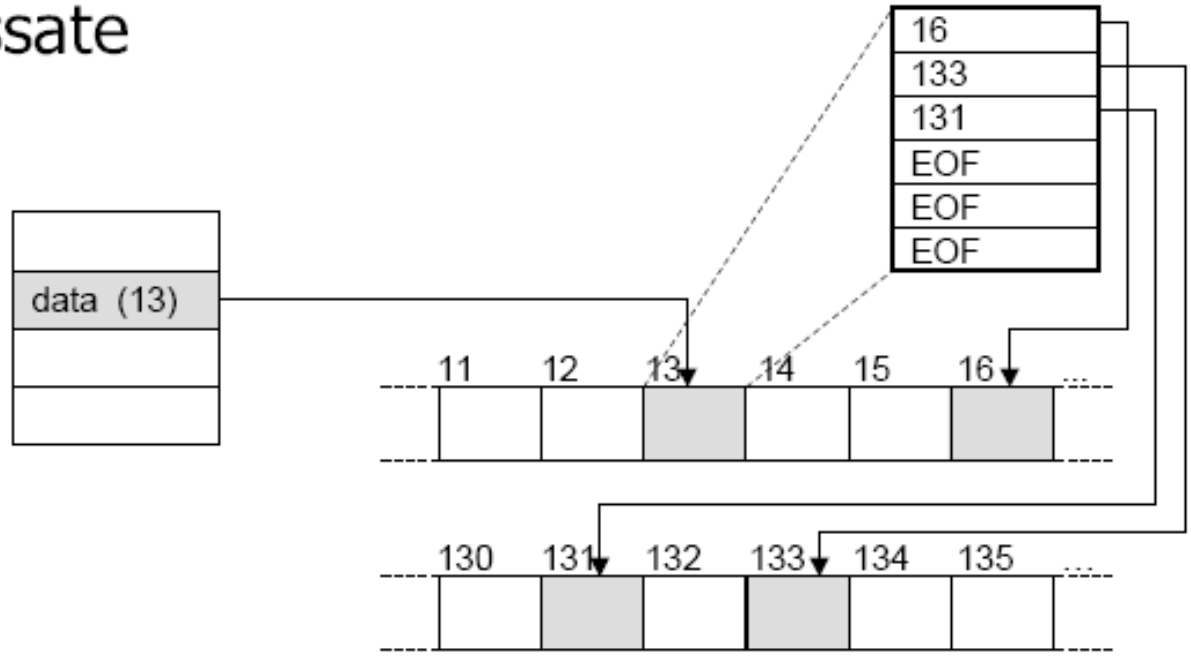
- Risolve i problemi di scarsa efficienza della allocazione concatenata raggruppando tutti i riferimenti
- I riferimenti ai blocchi di un file vengono memorizzati in un unico *blocco indice*
- Un *blocco indice* è quindi:
  - Un vettore di riferimenti ai blocchi del file
  - L'*i*-esimo elemento del vettore si riferisce all'*i*-esimo blocco del file
- Il descrittore del file contiene il riferimento al blocco indice

(c) Allocazione indicizzata

- In questo modo si ottiene:
  - Eliminazione della frammentazione esterna
  - Accesso casuale ai blocchi molto efficiente
- Lo spazio richiesto per i riferimenti è maggiore che nel caso di allocazione concatenata

(c) Allocazione indicizzata

- Gli elementi del blocco indice che non si riferiscono a nessun blocco hanno il valore EOF
- Se un file è costituito da pochi blocchi si ha un sottoutilizzo del blocco indice che ha dimensioni fissate



(c) Allocazione indicizzata

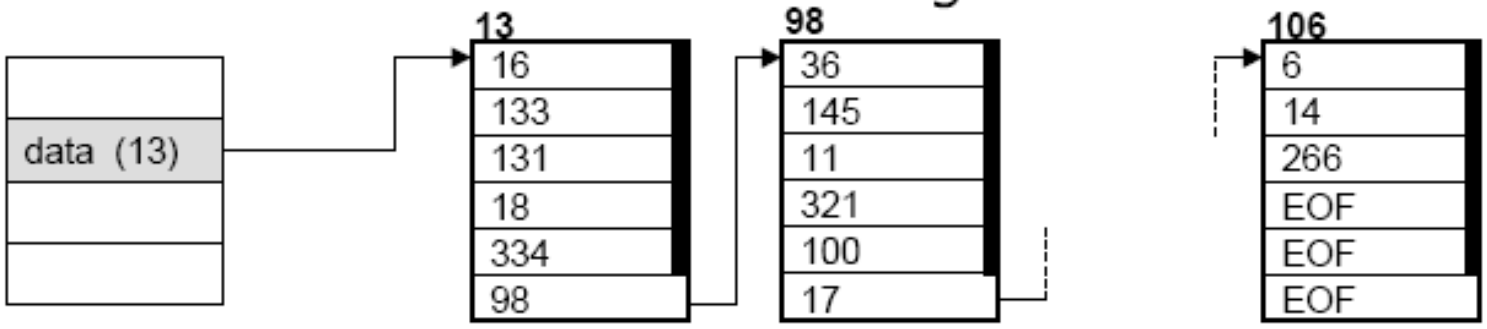
- E' importante scegliere una dimensione opportuna per il blocco indice:
  - Se troppo grande, molto spazio viene sprecato
  - Se troppo piccolo, non consente di trattare file di grandi dimensioni
- In genere il blocco indice coincide con un blocco fisico

(c) Allocazione indicizzata

- Esistono tre schemi possibili per risolvere i problemi legati alla gestione di file di diverse dimensioni:
  - Schema concatenato
  - Schema multilivello
  - Schema combinato
- Tutti questi metodi sono utilizzati in sistemi operativi e file system reali

(c) Allocazione indicizzata

- *Schema concatenato*
  - Il blocco indice contiene i riferimenti ai blocchi del file
  - L'ultimo elemento del blocco indice contiene un riferimento ad un secondo blocco indice, se il file è di grandi dimensioni
  - L'ultimo elemento del blocco indice contiene EOF se tutti i blocchi del file sono già stati referenziati

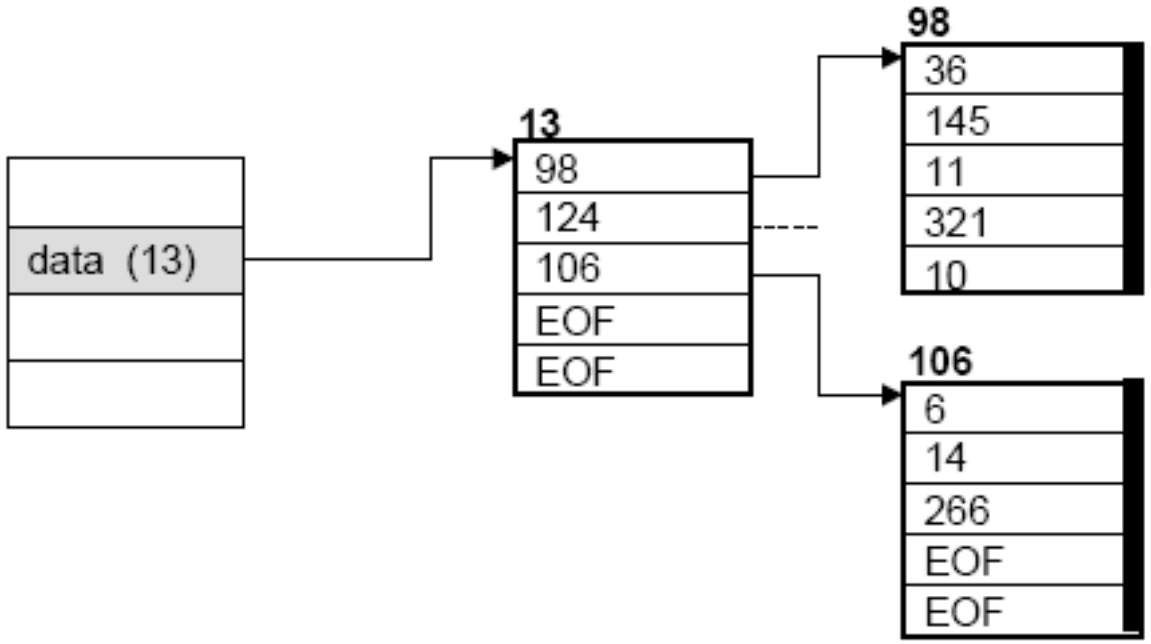


(c) Allocazione indicizzata

- *Schema multilivello:*
  - Un blocco indice di *primo livello* contiene i riferimenti ai blocchi indice di *secondo livello*
  - Un blocco di *secondo livello* contiene i riferimenti ai blocchi di *dati*
  - Con blocchi di 4.096 byte e riferimenti a 4 byte si possono indirizzare file con $(4.096/4)^2 = 1.048.576$ blocchi ovvero 4GB

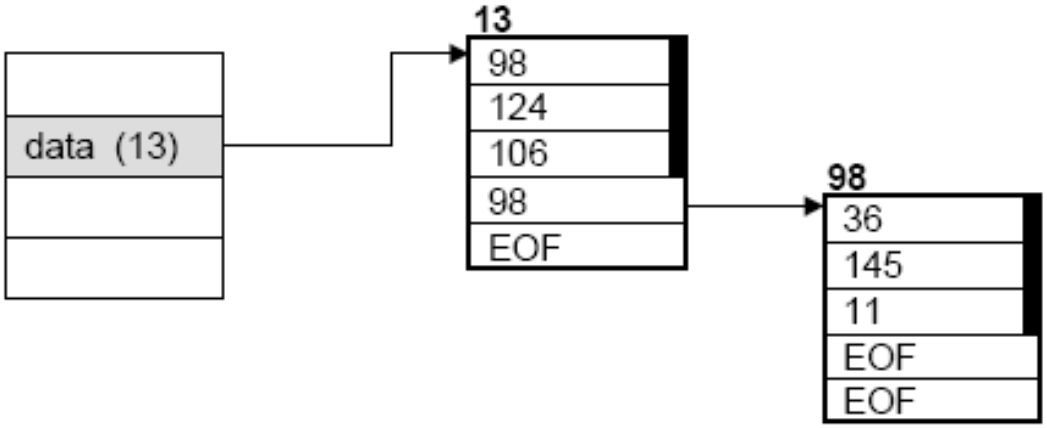
(c) Allocazione indicizzata

Schema Multi-livello



(c) Allocazione indicizzata

- *Schema combinato:*
  - La parte iniziale di un blocco indice contiene riferimenti a blocchi di dati del file
  - Gli ultimi elementi del blocco indice contengono riferimenti ad altri blocchi indice
  - Se il file ha dimensioni ridotte non viene utilizzato il secondo livello



Allocazione: UNIX i-node

- Il sistema operativo UNIX utilizza lo schema di allocazione indicizzata combinata
- Realizzato mediante una tabella chiamata *i-node (index-node)* che contiene la lista degli attributi e degli indirizzi dei blocchi di disco a cui sono associati i blocchi del file
- Primi indirizzi dei blocchi di disco sono memorizzati nell'i-node stesso

Implementazione di file e Directory in UNIX

- Ogni file o directory ha associato un i-node
- I-node: 64 byte di informazioni
- I-node dei file in un disco sono memorizzati in sequenza numerica all'inizio del disco o di ogni cilindro
- Dato il numero dell'i-node UNIX può localizzare la sua posizione calcolandone l'indirizzo su disco

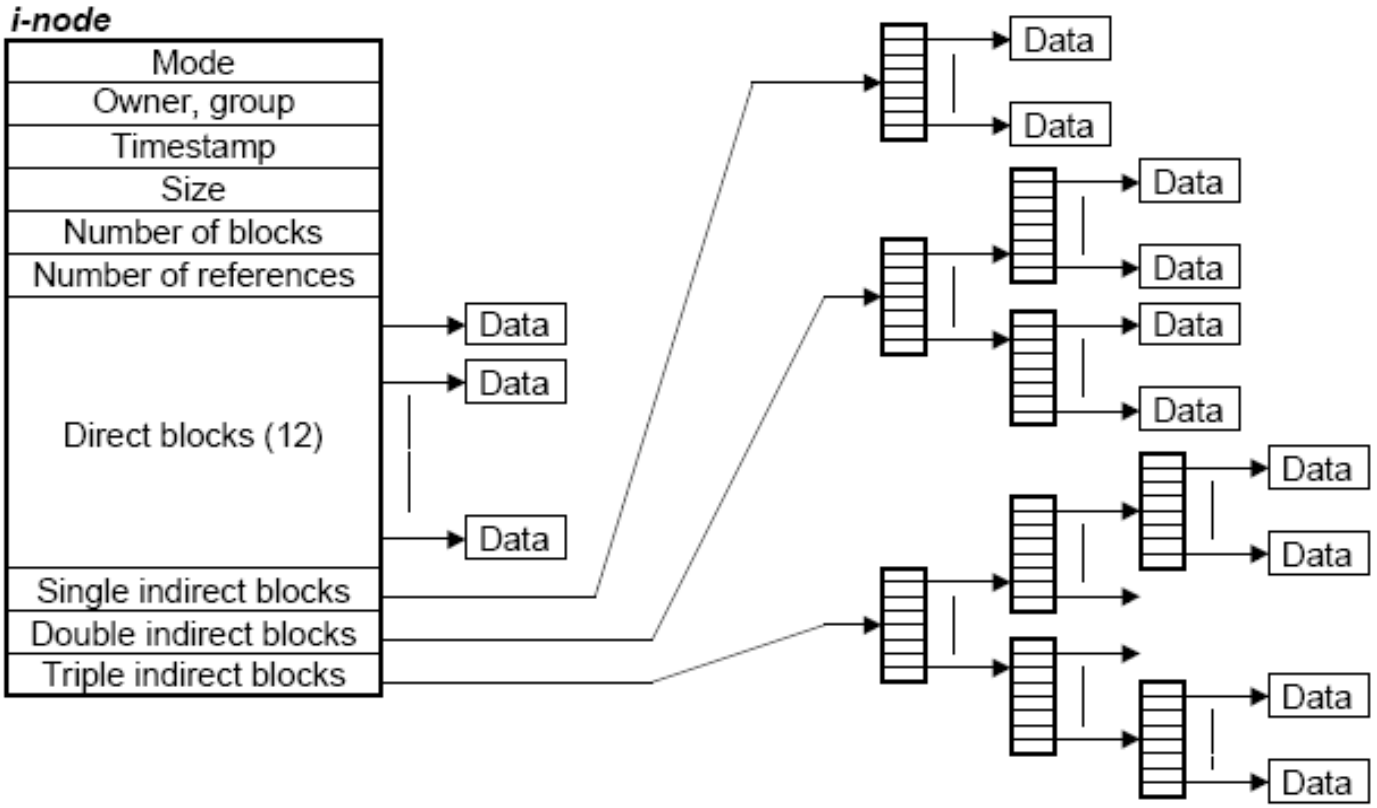
Informazioni nell' i-node

- Tipo di file, bit di protezione (9 rwx) e altri bit
- Numero di link al file
- Proprietario e gruppo del proprietario
- Lunghezza del file in byte
- 13 indirizzi su disco (blocchi su disco contenenti il file)
- Data e ora in cui il file è stato letto e scritto per l'ultima volta
- Data e ora dell'ultima modifica dell'i-node

UNIX i-node

- Per file di dimensioni maggiori l'i-node contiene l'indirizzo di un blocco del disco chiamato single indirect block che a sua volta contiene altri indirizzi di blocchi del disco
- Se questo non è sufficiente l'i-node mette a disposizione un altro indirizzo, chiamato double indirect block che contiene l'indirizzo di un blocco che a sua volta contiene una lista di single indirect block
- Ognuno di questi blocchi punta a sua volta a qualche centinaia (128 di solito) di single indirect block
- Esiste anche un triple indirect block nel caso la doppia indicizzazione non sia sufficiente

FILE SYSTEM UNIX



Gestione dello spazio libero

- All'atto della creazione e scrittura di un nuovo file è necessario individuare sul disco il primo blocco libero
- Una soluzione consiste nell'uso un *vettore di bit* in cui:
  - La posizione del bit indica il numero del blocco
  - Se il bit vale 0 il blocco è già utilizzato
  - Se il bit vale 1 il blocco è disponibile

Allocazione degli spazi liberi

- Individuazione del primo blocco libero :
  - Si scorrono le parole (di *b* bit) fino alla prima diversa da zero
  - Sia *k* il numero di parole uguali a zero
  - Si trova l'offset *d* del primo bit con valore 1
- L'indirizzo *n* del blocco è dato da:

$$n = k \times b + d$$

Gestione dello spazio libero

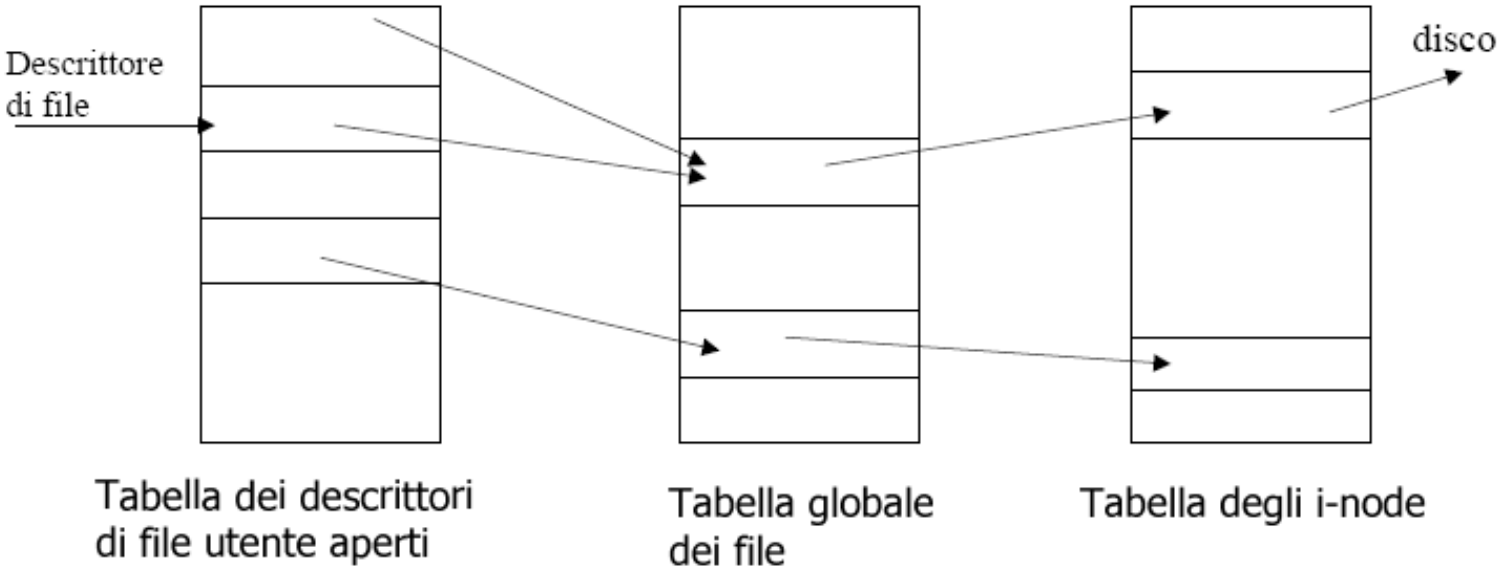
- Soluzione alternativa: utilizzo di una *lista concatenata*, simile a quella utilizzata per i file
- In questo schema:
  - La posizione del primo blocco disponibile è memorizzata in una zona riservata del disco
  - Ogni blocco disponibile contiene un riferimento al blocco disponibile successivo
- Questa soluzione è migliore della precedente per dischi di grandi dimensioni

Realizzazione delle directory

- Soluzione più semplice: realizzazione attraverso una *lista lineare:*
- Ogni elemento contiene
  - Il nome del file
  - Il descrittore del file
- Tutte le operazioni sui file (eliminazione, creazione, rinomina, ...) richiedono una ricerca del nome
- La ricerca è di semplice realizzazione ma è scarsamente efficiente in quanto è lineare nel numero degli elementi
- Una lista ordinata lessicograficamente:
  - Consente una ricerca binaria (logaritmica)
  - Comporta problemi aggiuntivi per mantenere la lista in ordine

Strutture dati del file system

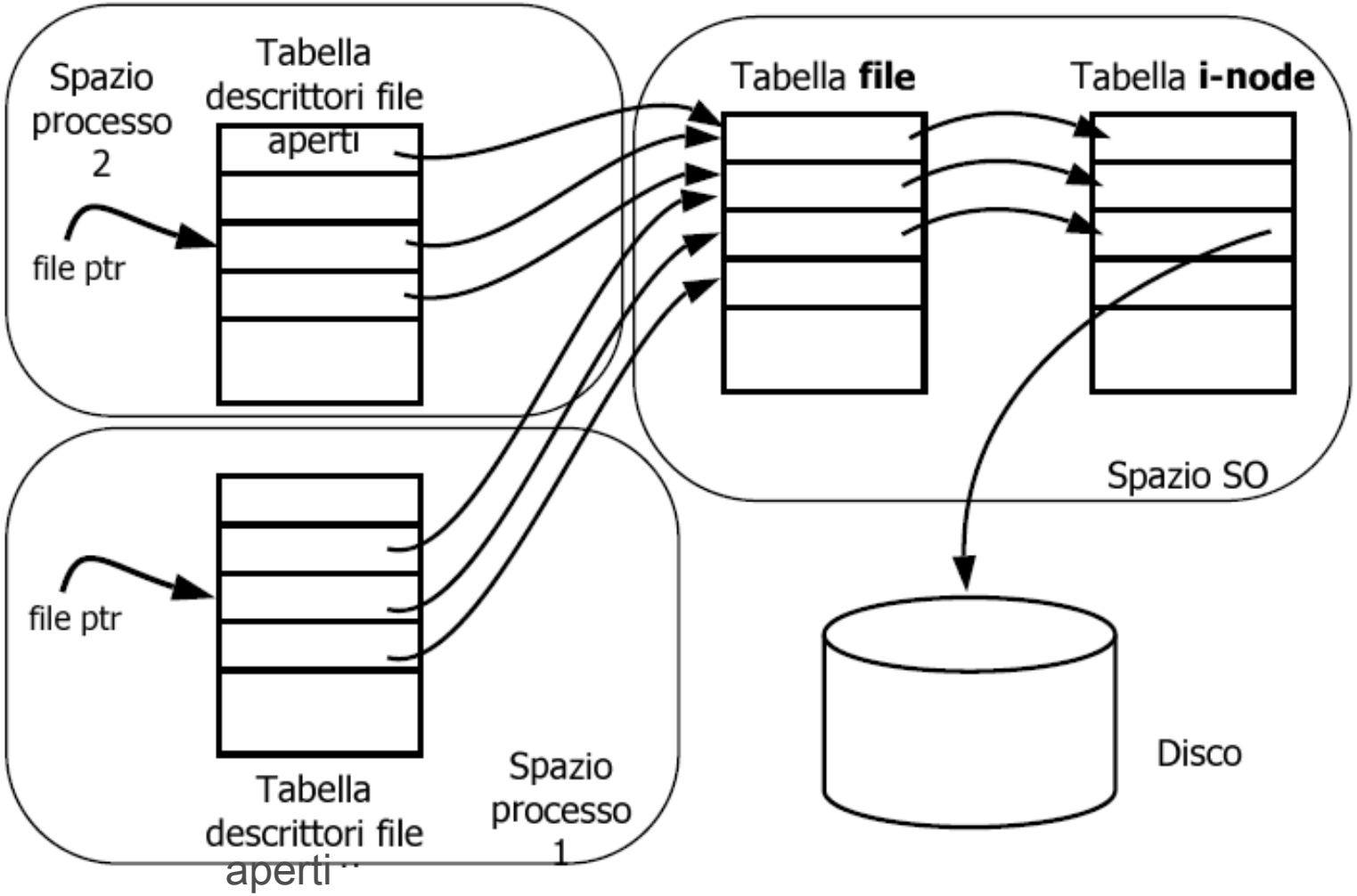
- UNIX contiene tre tabelle per la gestione dei file



Strutture dati del file system

- Tabella dei descrittori di file utente:
  - tabella associata ad ogni processo utente contenente una riga per ogni file aperto dal processo con l'indirizzo della riga della tabella globale dei file aperti relativa al file
- Tabella globale dei file aperti:
  - tabella del sistema operativo che contiene una riga per ogni file aperto nel sistema
  - Ogni riga contiene l'indirizzo del corrispondente i-node nella tabella degli i-node, l'indicatore della posizione corrente del file e il contatore al numero di riferimenti da parte dei processi a questo file
- Tabella degli i-node:
  - Le righe contengono la copia in memoria degli i-node del volume per maggiore efficienza nei riferimenti

Strutture dati del file system



Struttura del volume

Blocco di boot	Super block	Lista i-node	Blocchi dati

- Blocco 0 o blocco di boostrap: contenuto tipicamente nel primo settore contiene il codice di inizializzazione del sistema operativo
- Superblock: descrive lo stato del file system (es. dimensioni, spazio libero...)
- Lista degli i-node: dimensione definita in fase di configurazione del sistema operativo, accessibile dalla tabella degli i-node

Protezione

- I dati di un file system necessitano di protezione
  - Protezione da *danni fisici*
    - Malfunzionamenti dei dispositivi
    - Danni meccanici e/o elettrici
    - Soluzione: *backup* e *mirroring*
  - Protezione da *accessi impropri*
    - Riservatezza
    - Modifica o eliminazione accidentale di dati
    - Soluzione: definizione di una *politica di accesso*
- Con il termine *protezione* ci si riferisce alla
  - *Definizione* di una politica di accesso
  - *Implementazione* di una politica di accesso

Protezione

- Alcune banali politiche di accesso
  - Ogni utente accede solo ai propri file
    - Scelta limitante, ad esempio per i gruppi di lavoro
  - Ogni utente accede a tutti i file
    - È assente una politica di accesso
- La soluzione consiste nell'*accesso controllato*
  - Si definiscono regole di accesso ai file sulla base di
    - *Identità* e *gruppo* di lavoro dell'utente
    - *Proprietà* dei file
  - Tali regole dipendono dal *tipo di operazione* richiesta
    - Lettura
    - Scrittura, eliminazione o aggiunta
    - Esecuzione o lista

Protezione: Liste di accesso

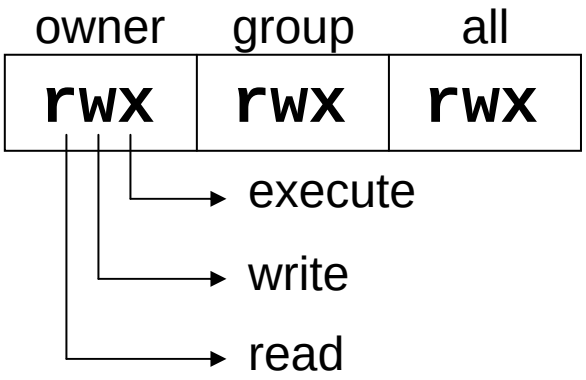
- L'accesso e le operazioni consentite dipendono dalla identità dell'utente
- Ad ogni file è associata una *lista di accesso* o *ACL*
  - Indica quali operazioni sono consentite a quali utenti
  - Alla richiesta di una operazione il SO controlla la lista di accesso per verificare se il richiedente:
    - È contemplato
    - Ha il permesso di compiere quel tipo di operazione
- Questa soluzione presenta alcuni svantaggi:
  - Le ACL possono avere dimensioni notevoli
  - Le ACL devono essere create e gestite per ogni file
  - Il tempo di accesso ad un file viene prolungato

Protezione: UNIX

- UNIX adotta una soluzione semplificata
- Gli *utenti* sono identificati in base a
  - *username* Identificativo dell'utente
  - *group* Identificativo di gruppo
- Gli utenti sono raggruppati in tre *classi*
  - *owner* Il proprietario del file
  - *group* I membri del gruppo del proprietario del file
  - *all* Tutti gli utenti
- Le *operazioni* sono raggruppate in tre classi
  - *read* Lettura, copia
  - *write* Scrittura, modifica, eliminazione
  - *execute* Esecuzione

Protezione: UNIX

- Ad ogni file sono associati
  - Owner
  - Group
  - Mode
- Il mode è formato da tre gruppi di bit
  - Ogni gruppo si riferisce ad una classe di utenti
  - Ogni bit del gruppo si riferisce ad una operazione



rwx	rwx	rwx
111	101	101
7	5	5