Realizzazione del file system

Pietro Braione

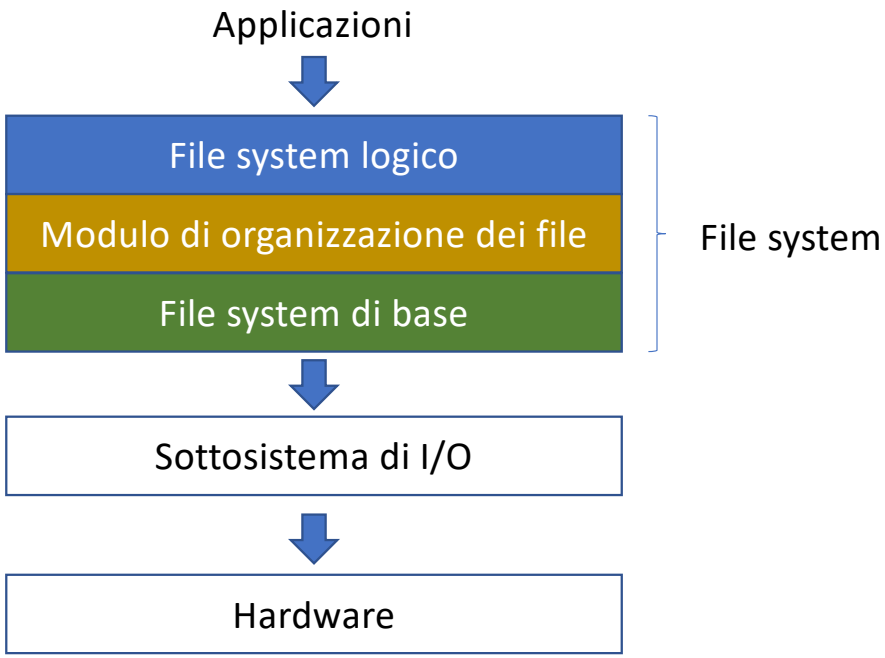
Reti e Sistemi Operativi – Anno accademico 2021-2022

Obiettivi

- Descrivere i dettagli realizzativi di file system locali e strutture di directory
- Esaminare l'allocazione dei blocchi e pro e contro degli algoritmi per la gestione dello spazio libero
- Esplorare i problemi relativi all'efficienza e alle prestazioni dei file system

Organizzazione stratificata del file system

- File system logico:
  - Gestisce i metadati (directory e file control block)
  - Responsabile di protezione e sicurezza
- Modulo di organizzazione dei file:
  - Traduce gli indirizzi dei blocchi logici in indirizzi dei blocchi fisici
  - Gestisce lo spazio libero
- File system di base:
  - Legge e scrive blocchi fisici dalle unità disco
  - Mantiene buffer e cache per i dati e metadati (directory)



Organizzazione stratificata: Vantaggi e svantaggi

- Vantaggi: minima duplicazione di codice all'aumentare del numero di file system
- (Notare che oggi il numero di file system utilizzati oggi è molto alto: ISO 9660 per CD-ROM e DVD, ext3 ed ext4 per Linux, FAT, FAT32, exFAT e NTFS per Windows 10, APFS e HFS+ per macOS, più ZFS, GoogleFS…)
- Svantaggi: all'aumentare del numero di strati diminuiscono le prestazioni

Strutture dati (1)

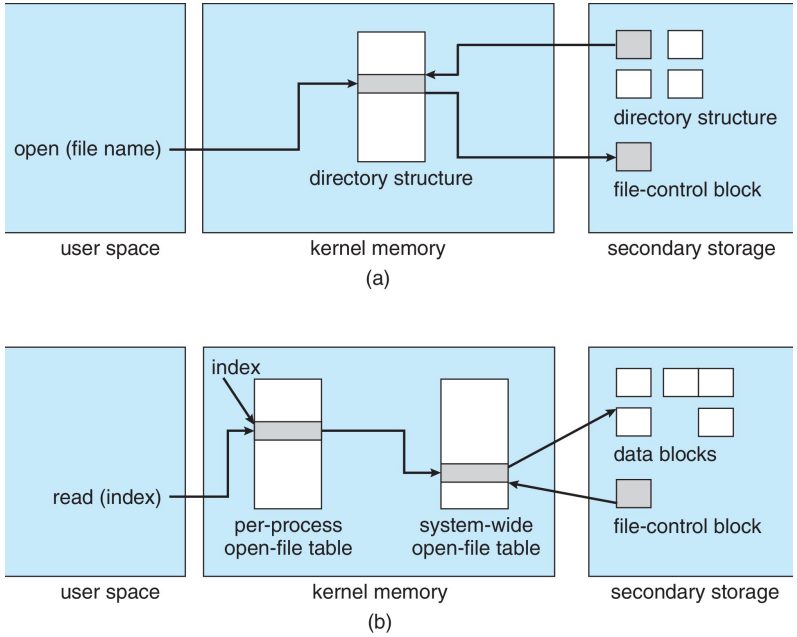
- Per realizzare le principali operazioni del file system servono diverse strutture dati, sia sul disco sia nel sistema operativo stesso
- Su disco:
  - **Boot control block**: contiene informazioni necessarie al sistema per effettuare l'avvio del sistema operativo (necessario solo sul disco dal quale il sistema effettua il boot)
  - **Volume control block**: contiene informazioni relative ad un certo volume, o partizione, del disco, come ad esempio il numero e la dimensione dei blocchi, il puntatore ai blocchi liberi e ai FCB liberi
  - **Struttura della directory**: (una per file system) informazioni relative alla organizzazione dei files
  - **File control block** (**FCB**): (uno per file) informazioni relative ad un singolo file

Strutture dati (2)

- In memoria:
  - **Tabella di montaggio**: contiene informazioni relative a ciascun volume montato
  - **Cache della struttura della directory**: informazioni relative alle directory utilizzate più di recente dai processi
  - **Tabella di sistema dei file aperti**: FCB dei file aperti dai processi
  - **Tabella dei file aperti per ciascun processo**: ogni processo ha una propria tabella dei file aperti, contenente puntatori alla tabella di sistema
  - **Buffer** di lettura/scrittura

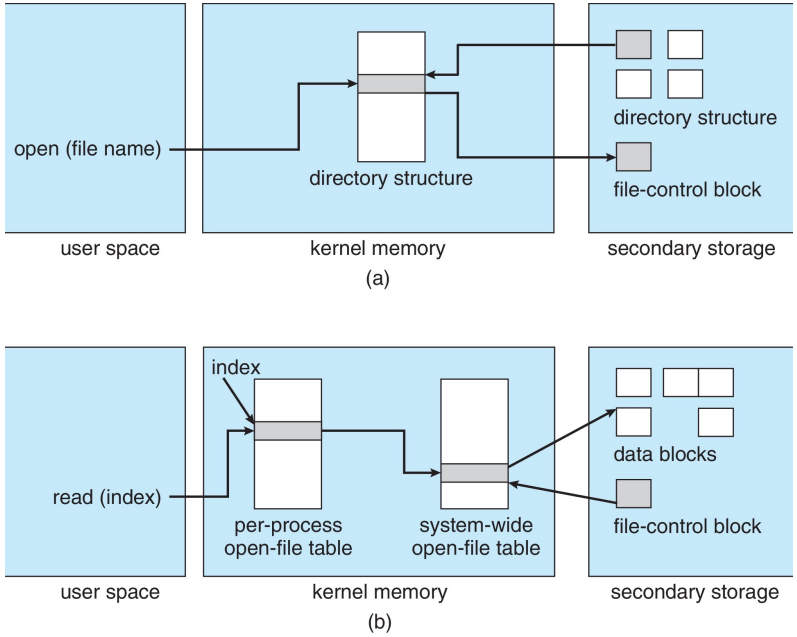
Realizzazione operazioni del file system (1)

- Quando un processo invoca `open()` il sistema operativo controlla la tabella di sistema dei file aperti
- Se esiste una entry per il file, aggiorna la tabella del processo per puntare alla corrispondente riga della tabella di sistema
- Altrimenti, legge la struttura della directory per recuperare da disco il FCB del file da aprire, e la aggiunge alla tabella di sistema
- La `open()` ritorna un indice alla tabella dei file aperti del processo, che viene usato nelle successive operazioni



Realizzazione operazioni del file system (2)

- La `close()` elimina l'elemento nella tabella dei file aperti del processo e decrementa il contatore nella tabella di sistema
- Quando questo raggiunge zero, i metadati vengono scritti nella struttura della directory su disco e l'elemento nella tabella di sistema viene cancellato
- Il caching viene usato, spesso aggressivamente, per ridurre il più possibile le operazioni su disco



Realizzazione delle directory

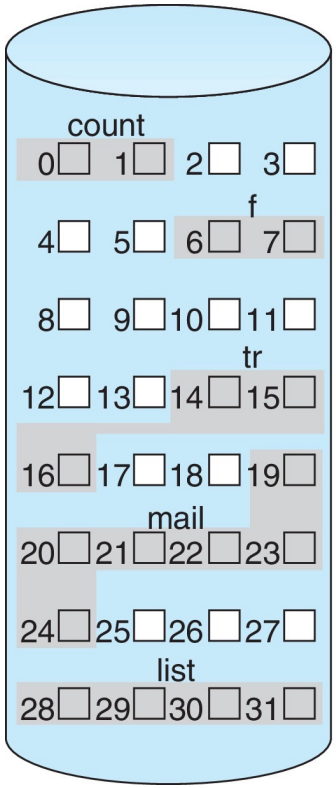
- Lista lineare:
  - Lista di nomi di file, con puntatori ai blocchi dei dati
  - Onerosa in termini di tempo: verificare l'esistenza di un file richiede nel caso pessimo la scansione di tutta la directory
  - Ottimizzazioni: caching, ordinamento, strutture dati bilanciate
- Tabella hash:
  - Funzione hash per calcolare l'indice nella lista a partire dal nome del file
  - Problemi: dipendenza funzione hash da dimensione lista; collisioni
  - Soluzione: tabella di dimensione fissa, gestione collisioni con liste

Metodi di allocazione

- Sono i metodi che i file system usano per assegnare i blocchi liberi ai file
- Tre diversi metodi:
  - Allocazione contigua
  - Allocazione concatenata
  - Allocazione indicizzata

Allocazione contigua

- Ad ogni file è assegnato un insieme contiguo di blocchi
- Vantaggi:
  - Di solito performance ottima
  - Semplice: è sufficiente memorizzare il blocco di partenza (start) e il numero di blocchi (length)
- Svantaggi
  - Occorre sapere in anticipo la dimensione del file
  - Occorre trovare sufficiente spazio contiguo per il file
  - Frammentazione esterna



directory

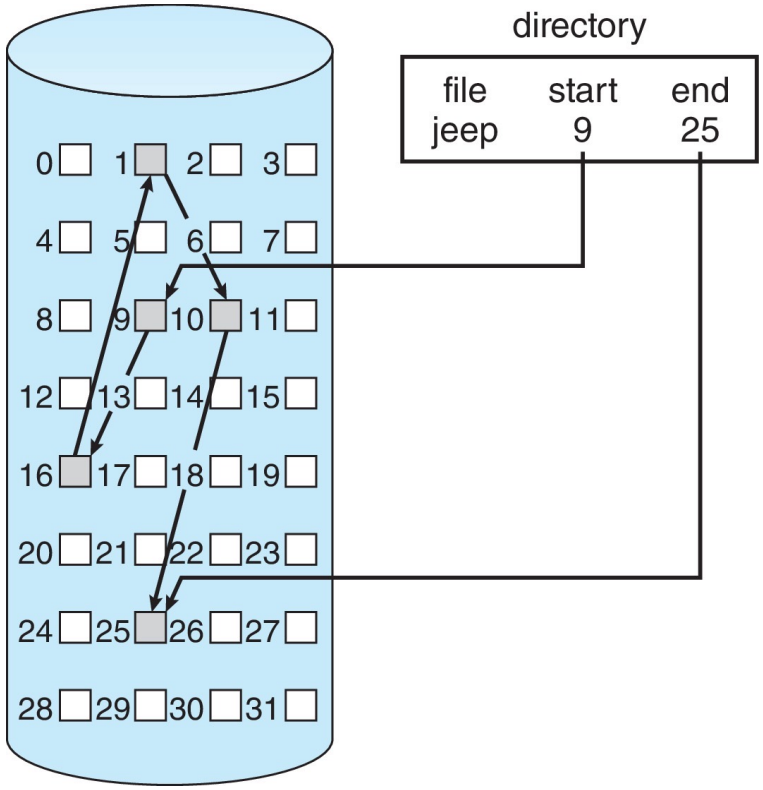
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Variante: allocazione basata su extent

- Un extent è una porzione contigua di un file
- L’allocazione basata su extent alloca ad un file uno o più extent, nel caso in cui il file debba crescere in dimensioni
- Esempio: file system Veritas

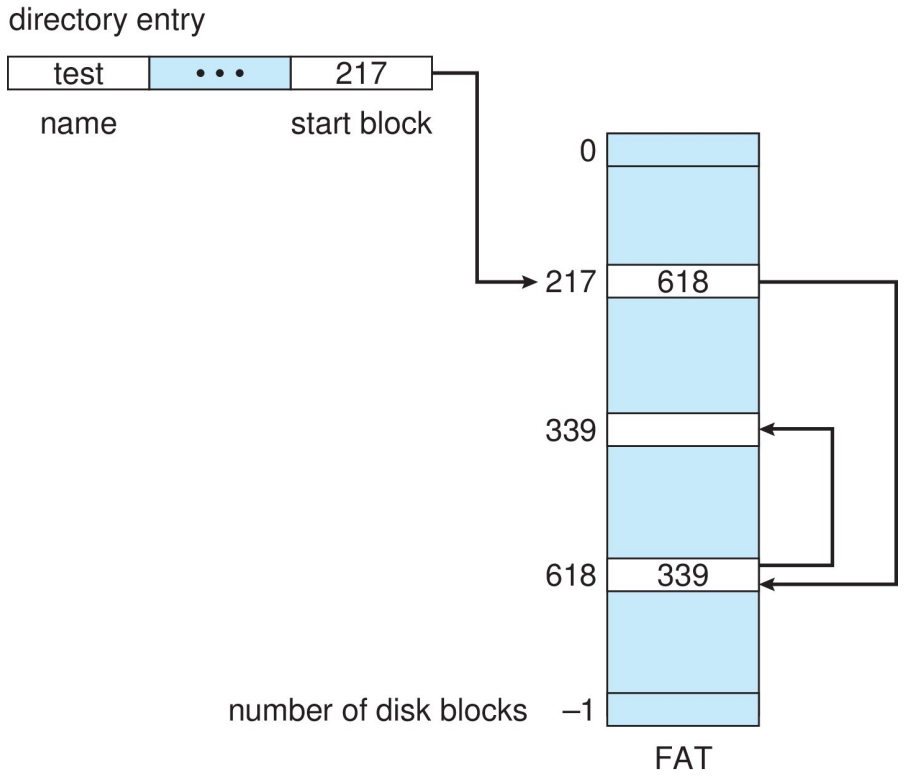
Allocazione concatenata

- Ogni file è composto da una lista concatenata di blocchi
  - La directory contiene un puntatore al primo blocco (e all'ultimo) del file
  - Ogni blocco contiene un puntatore al successivo
- Vantaggi:
  - Non è necessario specificare la dimensione del file in creazione
  - Il file può crescere finché c'è spazio a disposizione nel volume
  - Nessuna frammentazione esterna
- Svantaggi:
  - Accesso diretto inefficiente
  - Lo spazio per memorizzare il puntatore al blocco successivo è sottratto ai dati utente
  - Possibile ottimizzazione: operare su cluster (gruppi di blocchi)
  - Altro svantaggio: affidabilità



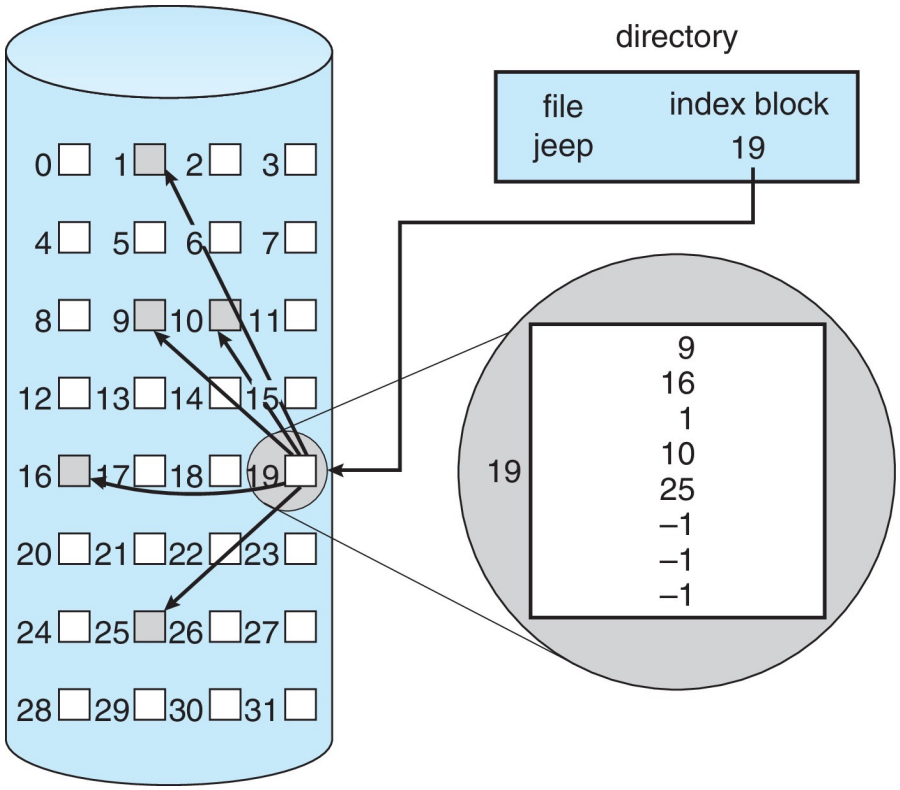
Variante: File Allocation Table (FAT)

- Simile all'allocazione concatenata, ma i puntatori ai blocchi sono tutti raccolti in una tabella (la file allocation table) in una sezione del volume
- Necessità di caching della FAT per evitare continui spostamenti della testina dalla FAT ai blocchi del file
- Migliore performance di accesso diretto (è possibile recuperare i puntatori ai blocchi in una sola operazione)
- Replica della FAT per migliore affidabilità



Allocazione indicizzata

- Ogni file ha un **blocco indice**, strutturato come un array di indirizzi di blocchi del disco
- La directory contiene il puntatore al blocco indice
- Vantaggi:
  - Nessuna frammentazione esterna
  - Nessun bisogno di dichiarare in anticipo la dimensione del file
  - Efficiente accesso diretto (è possibile recuperare i puntatori ai blocchi con una sola operazione)
- Svantaggio: overhead blocco indice con file di dimensioni ridotte



Allocazione indicizzata: schemi per blocco indice

- Schema concatenato: l'ultima parola del blocco indice punta ad un successivo blocco indice
- Indice a più livelli: i puntatori del blocco indice puntano a blocchi indice di secondo livello, e così via fino al numero di livelli desiderati
- Schema combinato: ad esempio, gli inode dei filesystem dei sistemi operativi UNIX-like hanno un certo numero di puntatori diretti, e gli ultimi tre puntatori sono ad una, due, tre indirezioni rispettivamente

Schemi di allocazione: prestazioni

- Il miglior schema di allocazione dipende dalla modalità di accesso
- L'allocazione contigua è perfetta per l'accesso sequenziale e diretto
- L'allocazione concatenata va bene per l'accesso sequenziale, ma non per quello diretto
- Per l'allocazione indicizzata il discorso è più complesso:
  - Se il blocco indice è in memoria, si può accedere direttamente al blocco da leggere
  - Altrimenti occorre caricare anche il blocco indice, il che può essere più o meno complesso a seconda della struttura dell'indice, della dimensione del file, e della posizione del blocco del file da leggere
- Per i dispositivi NVM, dove non ci sono testine e movimenti fisici, occorrono algoritmi ed ottimizzazioni diverse

Gestione dello spazio libero

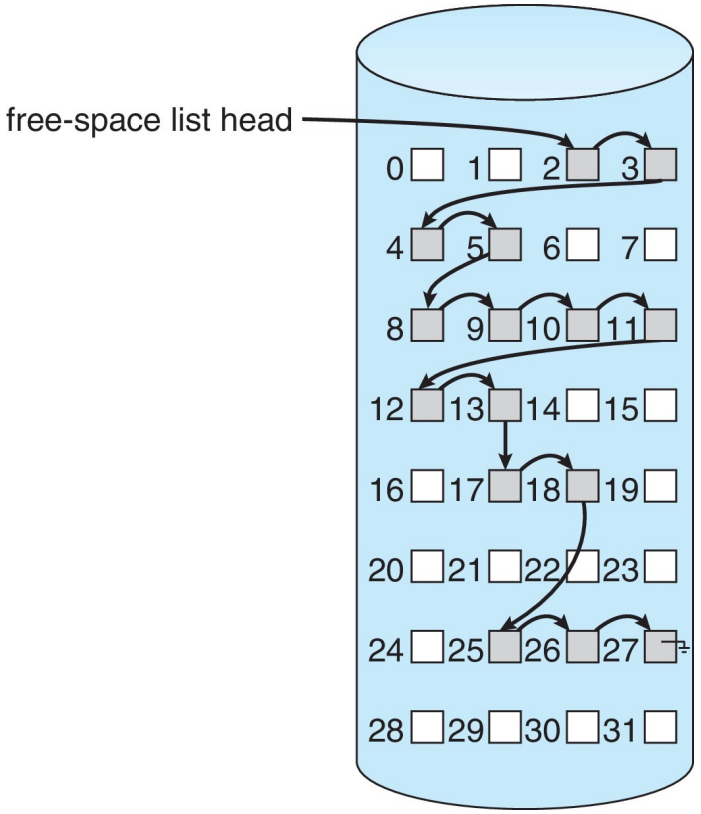
- Il file system mantiene un elenco dei blocchi liberi, la **lista dello spazio libero**
- In realtà non è necessariamente organizzata come una lista, ma può essere realizzata come:
  - Vettori di bit
  - Liste concatenate
  - Space maps (non le vedremo)

Spazio libero: Vettori di bit

- Ogni blocco è rappresentato da un bit, impostato a 1/0 se è libero/occupato
- Vantaggi:
  - Metodo semplice
  - Permette di trovare facilmente blocchi liberi contigui
  - Le CPU hanno istruzioni che permettono di individuare l'offset del primo bit impostato a 1 in una determinata parola
- Svantaggi:
  - L'efficienza si ha solo se tutto il vettore di bit è in memoria, ma per dischi grandi questo non è possibile
  - Il problema può essere mitigato con il clustering, ma le dimensioni dei dischi sono in continua crescita

Spazio libero: liste concatenate

- Approccio simile all'allocazione concatenata
- Svantaggio: potrebbe non essere semplice recuperare spazio contiguo



Varianti liste concatenate

- **Raggruppamento**: memorizzare in ogni blocco libero gli indirizzi di n altri blocchi liberi
- **Conteggio**: memorizzare in un blocco libero l'indirizzo del primo blocco libero + numero di successivi blocchi contigui liberi (nel caso ve ne siano)