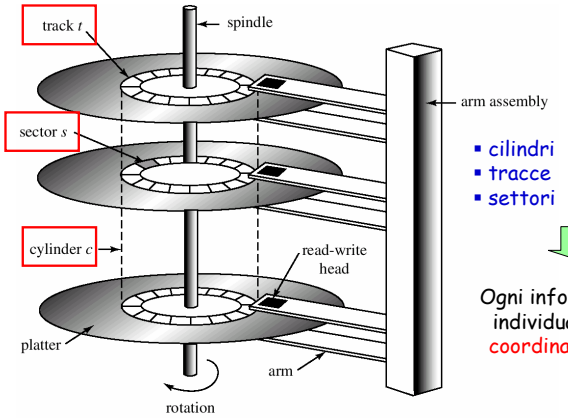
Memoria secondaria e file system

- Memoria secondaria
  - Struttura del disco
  - Scheduling del disco
  - Gestione dell'unità a disco
  - Gestione dello spazio di swap
  - La struttura RAID
- File system
  - File e directory
  - Struttura logica del file system
  - Metodi di allocazione
  - Gestione dello spazio libero

Organizzazione fisica dei dischi



- cilindri
- tracce
- settori

Ogni informazione e' individuata da tre coordinate (c, t, s)

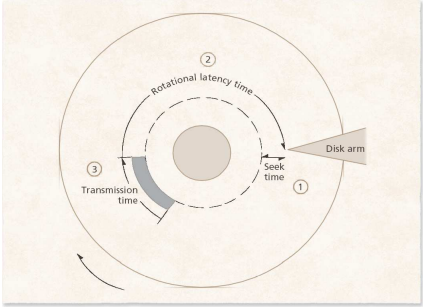
Organizzazione logica dei dischi

- A causa della lentezza dei tempi di accesso sarebbe impensabile effettuare un accesso al disco e leggere un solo byte
- I dischi vengono indirizzati come vettori monodimensionali di blocchi logici, dove il *blocco logico* rappresenta la minima unità di trasferimento (dim. tipica 512 byte).
- L'array di blocchi logici viene mappato sequenzialmente nei settori del disco:
  - Il settore 0 è il primo settore della prima traccia del cilindro più esterno.
  - La corrispondenza prosegue ordinatamente lungo la prima traccia, quindi lungo le rimanenti tracce del primo cilindro, e così via, dall'esterno verso l'interno.

Caratteristiche dei dischi

- Il tempo di accesso al disco si può scindere in :
  - Tempo di ricerca (seek time) — è il tempo impiegato per spostare la testina sulla traccia che contiene il settore desiderato.
  - Latenza di rotazione (rotational latency) — è il tempo necessario perché il disco ruoti fino a portare il settore desiderato sotto alla testina.
  - Tempo di trasferimento (trasmission time) — e' il tempo necessario per leggere/scrivere i 512 byte del blocco logico
- La larghezza di banda del disco è il numero totale di byte trasferiti, diviso per il tempo trascorso fra la prima richiesta del servizio e il completamento dell'ultimo trasferimento

Caratteristiche dei dischi



Modello	seek time (ms)	rotational latency (ms)
Maxtor DiamondMax Plus 9 (High-end desktop)	9.3	4.2
WD Caviar (High-end desktop)	8.9	4.2
Toshiba MK8025GAS (Laptop)	12.0	7.14
WD Raptor (Enterprise)	5.2	2.99
Cheetah 15K.3 (Enterprise)	3.6	2.0

Scheduling del disco

- Il SO è responsabile dell'uso efficiente dell'hardware. Per i dischi ciò significa garantire tempi di accesso contenuti e ampiezze di banda maggiori.
- Una richiesta di accesso al disco può venire soddisfatta immediatamente se unità a disco e controller sono disponibili; altrimenti la richiesta deve essere aggiunta alla coda delle richieste inevase per quell'unità.
- Il SO ha l'opportunità di scegliere quale delle richieste inevase servire per prima: uso di un algoritmo di scheduling

Scheduling del disco

- Il tempo di trasferimento dipende dal supporto hardware.
- L'ordine con cui vengono esaudite le richieste determinano il

tempo media di ricerca (quale traccia accedere per prima?)

Latenza media di rotazione (quale settore della traccia accedere per prima?)

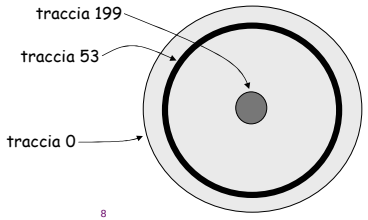
OBIETTIVO:
Massimo numero di richieste nell'unita' di tempo
Oppure
Minimo tempo medio di accesso

Ottimizzazione del seek time

- Gli algoritmi di scheduling del disco verranno testati sulla coda di richieste per le tracce (0–199):

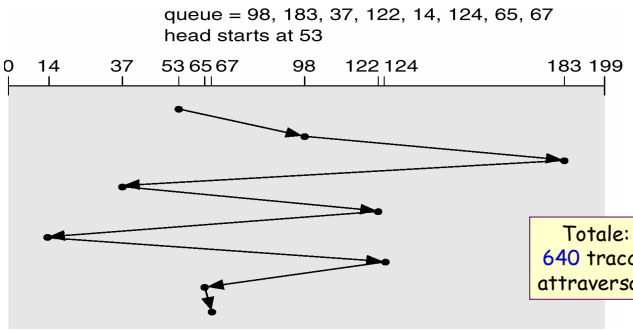
98, 183, 37, 122, 14, 124, 65, 67

La testina dell'unità a disco è inizialmente posizionata sulla traccia 53.



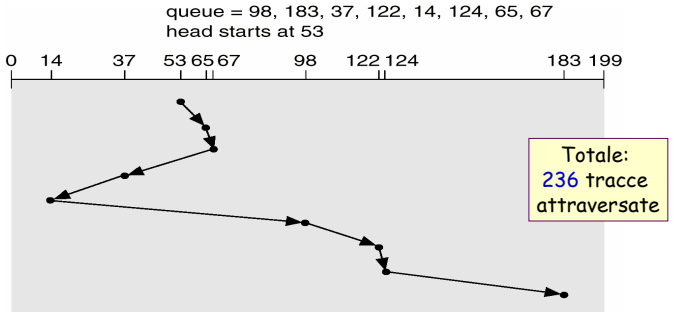
Scheduling First Come First Served (FCFS)

- Soddisfa le richieste secondo l'ordine di arrivo



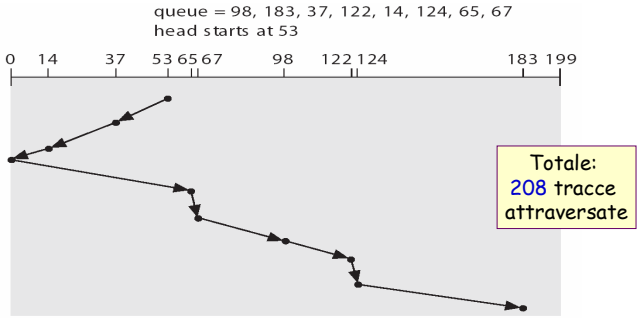
Scheduling Shortest Seek Time First (SSTF)

- Seleziona la richiesta con il minor tempo di seek a partire dalla posizione corrente della testina.
- Ordine di richieste molto localizzato
- Rischio di attesa indefinita delle richieste sulle tracce esterne e interne



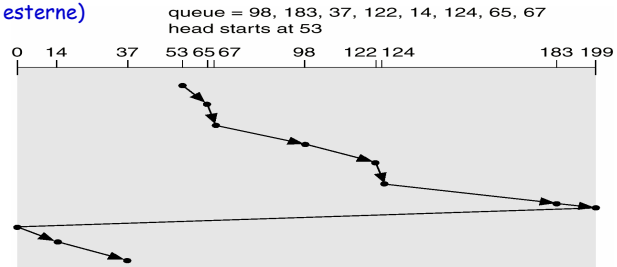
Scheduling SCAN

- Il braccio della testina si muove da un estremo all'altro del disco, servendo sequenzialmente le richieste;
- giunto ad un estremo inverte la direzione di marcia e, conseguentemente, l'ordine di servizio. È chiamato anche algoritmo dell'ascensore.



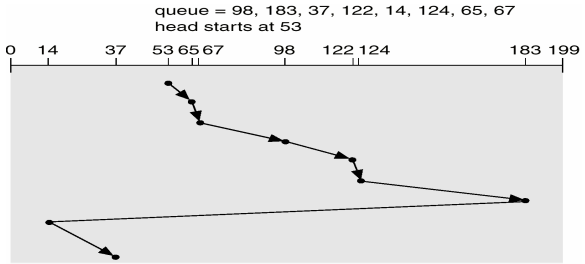
Scheduling Circular SCAN (CSCAN)

- Garantisce un tempo di attesa più uniforme rispetto a SCAN.
- La testina si muove da un estremo all'altro del disco servendo sequenzialmente le richieste. Quando raggiunge l'ultima traccia ritorna immediatamente all'inizio del disco, senza servire richieste durante il viaggio di ritorno.
- Considera le tracce come una lista circolare, con l'ultima traccia adiacente alla prima (minore discriminazione delle tracce interne e esterne)



Scheduling Circular LOOK (CLOOK)

- Versione ottimizzata (e normalmente implementata) di C-SCAN.
- Il braccio serve l'ultima richiesta in una direzione e poi inverte la direzione senza arrivare al termine del disco.



Scelta di un algoritmo di scheduling

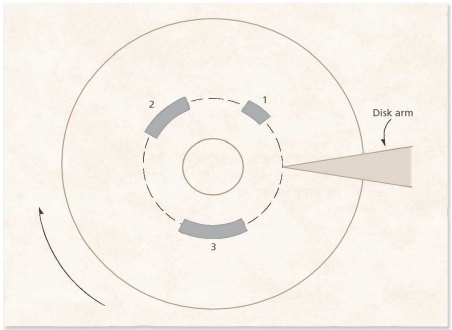
- SSTF è comune ed ha un comportamento "*naturale*".
- LOOK e C-LOOK hanno migliori prestazioni per sistemi con un grosso carico di lavoro per il disco (minor probabilità di blocco indefinito).
- Le prestazioni dipendono dal numero e dal tipo di richieste.
- Le richieste di servizio al disco possono essere influenzate dal metodo di allocazione dei file.
- L'algoritmo di scheduling del disco dovrebbe rappresentare un modulo separato del SO, che può essere rimpiazzato da un algoritmo diverso qualora mutassero le caratteristiche del sistema di calcolo.
- Sia SSTF che LOOK sono scelte ragionevoli per un algoritmo di tipo generale.

Ottimizzazione della latenza di rotazione

- Anni fa il tempo di ricerca dominava il tempo di accesso
- Oggi il tempo di ricerca e la latenza di rotazione hanno lo stesso ordine di grandezza
- Recentemente si e' cercato di migliorare le prestazione del disco riducendo la latenza di rotazione

Shortest-latency-time-first scheduling (SLTF)

- SLTF serve per prima le richieste con la piu' piccola latenza di rotazione indipendentemente dall'ordine di arrivo
- Facile da realizzare
- Ottiene prestazioni quasi ottimali

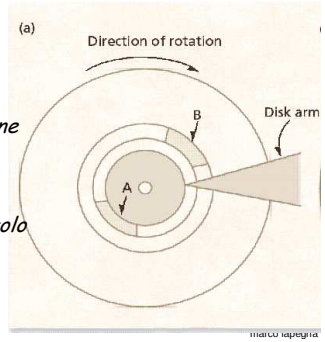


Shortest-positioning-time-first (SPTF)

- Tempo di posizionamento = tempo di ricerca + latenza di rotazione
- SPTF esaudisce per prima le richieste con il piu' piccolo tempo di posizionamento
- Buone prestazioni, ma rischio di attesa indefinita

A stessa traccia ma distante $\frac{1}{2}$ rotazione
B traccia adiacente e vicina

Se il seek time e' sufficientemente piccolo
viene servita prima B e poi A

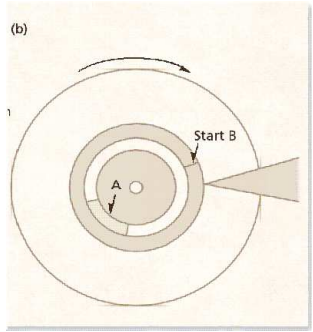


Shortest-access-time-first scheduling (SATF)

- Tempo di accesso = tempo di posizionamento + tempo di trasmissione
- SATF esaudisce per prima le richieste con il piu' piccolo tempo di accesso
- Maggiore througput ma rischio di attesa indefinita per richieste grandi

A lontana ma piccola
B vicina ma grande

viene servita prima A e poi B



Strutture RAID

La velocita' di trasferimento dei dischi non cresce allo stesso modo della velocita' doperativa della CPU

Peggioramento del througput

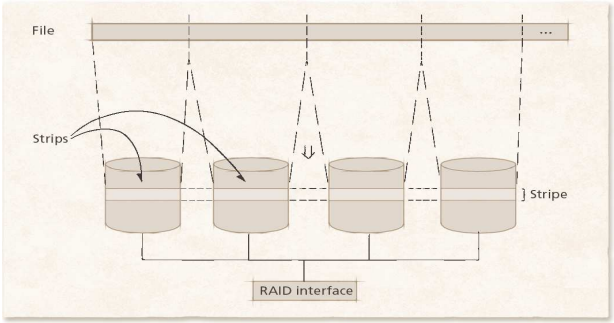
IDEA

Utilizzare piu' dischi da utilizzare "in parallelo"

RAID, *Redundant Array of Independent Disks*

Caratteristiche dei RAID

- I dischi sono visti come un'unica unita' di memorizzazione
- I file sono divisi in strisce (strips) distribuite sui vari dischi
- I pezzi della striscia relativi allo stesso file hanno la stessa locazione in ogni disco



Caratteristiche dei RAID

- La gestione dei RAID e' un'operazione dispendiosa che puo' essere causa di rallentamento della CPU e del S.O.

⇓

Hardware specializzato dedicato allo scopo (RAID controller)

- Un maggior numero di dischi aumenta la probabilita' di un guasto con conseguente perdita di dati

⇓

Ridondanza delle informazioni memorizzate

Quindi

Motivazioni dei RAID

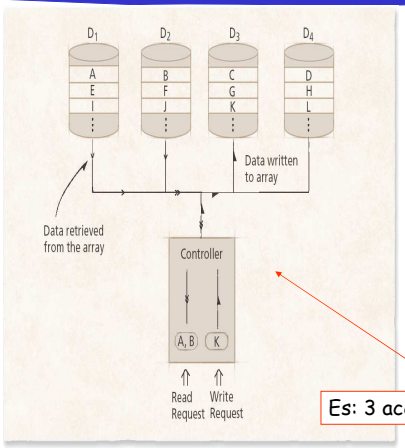
Efficienza (maggiore velocita' di accesso ai dischi)

Affidabilita' (maggiore sicurezza di non perdere dati)

⇓

Esistono molteplici schemi di gestione dei RAID chiamati Livelli RAID

Livello 0 (striping)



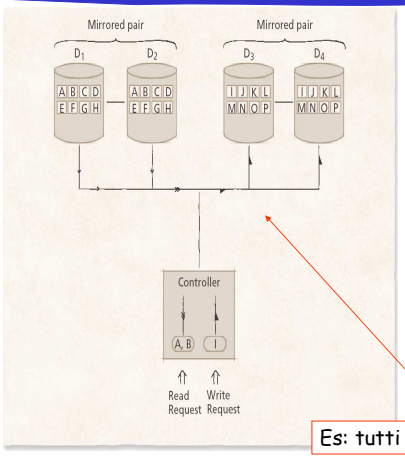
- Piu' semplice schema RAID
- striping senza ridondanza
- elevata efficienza
- rischio di perdere i dati

⇓

indicato quando le esigenze di performance superano quelle dell'affidabilita'

Es: 3 accessi contemporanei

Livello 1 (mirroring)



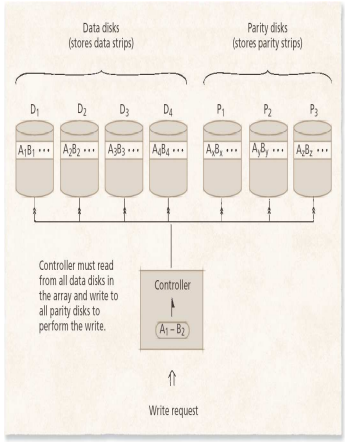
- Ogni disco ha un suo disco di copia
- ridondanza senza striping
- Buona performance in lettura, meno in scrittura
- Alto overhead di storage

⇓

Adatto quando le esigenze di affidabilita' sono importanti

Es: tutti i dati sono duplicati

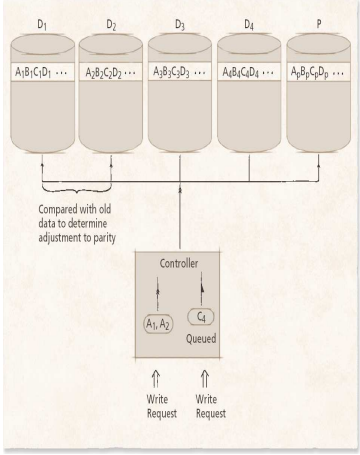
Livello 2 (Hamming error correcting code)



- Ridondanza e striping
- Dischi per i dati e dischi per "bit di parita' " per correggere errori
- Overhead di storage (ma meno del livello 1) e performance

Poco utilizzato

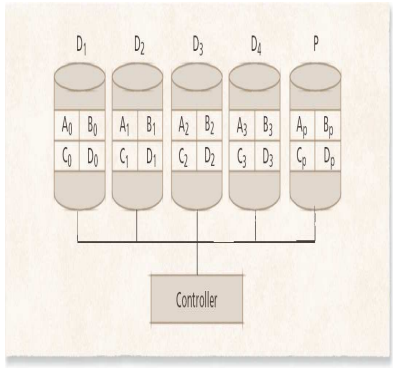
Livello 3 (XOR error correcting code)



- Ridondanza e striping
- simile al livello 2
- correzione di errori con diverso meccanismo
- 1 solo disco per i bit di parita'
- scarsa efficienza (bottleneck del disco di parita')

Poco utilizzato

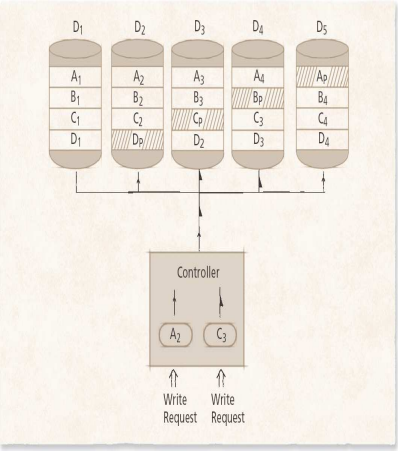
Livello 4 (block XOR eec)



- Ridondanza e striping
- versione a blocchi del livello 3

Poco utilizzato

Livello 5 (block distributed XOR eec)



- I blocchi di parita' sono distribuiti sui dischi dei dati
- Rimozione del bottleneck
- Piu' veloce dei livelli 2, 3 e 4
- Difficile da implementare

Miglior compromesso tra efficienza e affidabilita'

Soluzione piu' utilizzata per i sistemi general purpose

File

- Il maggior problema nella gestione della memoria secondaria e' la necessita' di fare riferimenti alla organizzazione fisica dei dispositivi.
- Esempio:
  - un programma in linguaggio macchina puo' risiedere nei blocchi 45, 51, 68, 73 e 99 di un disco
  - Una modifica al codice puo' produrre una diversa memorizzazione
  - Necessita' di fare riferimento ai blocchi fisici anche se l'entita' logica (il programma) e' sempre la stesso

⇩

FILE

Una **raccolta di dati associata ad un nome**, residente in memoria secondaria e manipolabile come una unica entita'

File

Dal punto di vista dell'utente, il file e' la piu' piccola unita' di memoria secondaria utilizzabile

⇩

Tutti i dati presenti nella memoria secondaria devono essere organizzati in file

Tipi di file:

- testo (sequenza di caratteri)
- sorgente (sequenza di procedure e funzioni)
- eseguibile (sequenza di moduli in linguaggio macchina che il loader puo' caricare in memoria)
- ...

Il file system

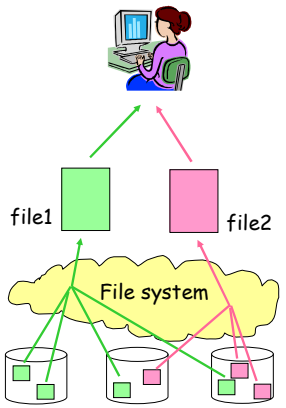
- Un sistema operativo puo' gestire
  - milioni di file,
  - di differenti utenti
  - di tipi differente

⇩

Necessita' di organizzare logicamente i file

L'organizzazione logica dei file di un sistema operativo e' chiamata FILE SYSTEM

Il file system:



- Organizza i file e gestisce l'accesso ai dati
- Responsabile della integrita' dei file, della gestione dei metodi di accesso (lettura/scrittura) e della gestione della memoria secondaria

Visione dei file indipendente dai dispositivi fisici !!!

Dispositivi fisici

Organizzazione del file system

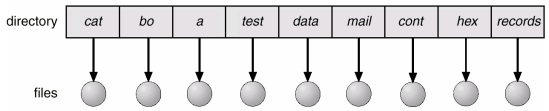
L'organizzazione del file system deve garantire…

- **Efficienza** — capacità di reperire file rapidamente.
- **Nominazione** — conveniente per gli utenti.
  - Due utenti possono utilizzare nomi uguali per file diversi.
  - Lo stesso file può avere diversi nomi.
- **Grouping** — Raggruppamento dei file sulla base di proprietà logiche, (ad esempio, tutti i programmi Java, tutti i giochi, etc.).

Directory

- Lo strumento piu' comune usato per realizzare un file system e' la directory *(elenco o cartella)*
- Una directory e' un file che contiene nomi e locazioni di file presenti nel file system (non possono contenere dati utente)
- Un elemento di una directory conserva informazioni come:
  - Nome file
  - Posizione
  - grandezza
  - Tipo
  - Privilegi di accesso
  - Tempo di creazione e modifica

File system monolivello (file system piatto)



- gli elementi di una directory sono solamente file
- Una directory unica per tutti i file.
- Problemi di nominazione: occorre scegliere un nome diverso per ogni file.
- Problemi di performance: ricerca lineare nella directory
- Nessun raggruppamento logico.

- raramente implementato

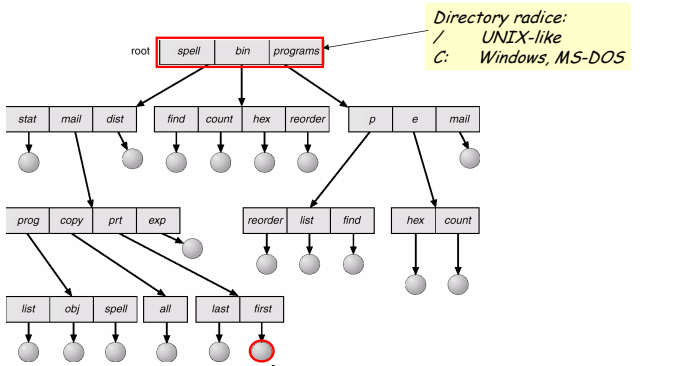
File system gerarchico (ad albero)

- Gli elementi di una directory possono essere file o directory
- Una directory principale (root directory, master file directory) indica la radice dell'albero dalla quale partono le directory di secondo livello
- L'unicita' dei nomi di file e' realizzata mediante il "pathname assoluto" costituito dai nomi delle directory incontrate dalla radice fino al file

- I nomi dei file possono essere unici solo all'interno della stessa directory (pathname relativo) ➔ nominazione
- Ricerca solo nella directory corrente ➔ efficienza
- Possibilita' di raggruppamenti logici ➔ grouping

- Implementato nella quasi totalita' dei s.o.

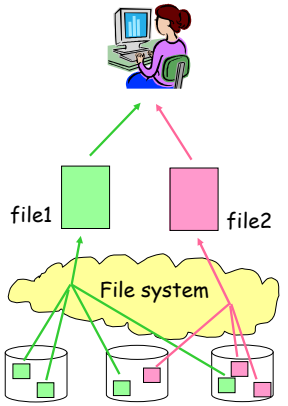
File system gerarchico (ad albero)



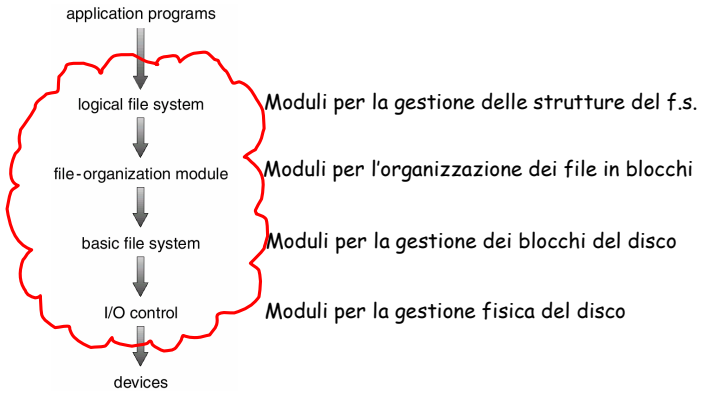
Problema

Un problema fondamentale nella realizzazione di un f.s. e' la definizione di algoritmi e strutture dati per far corrispondere il file system logico (file, directory,...) al file system fisico (dispositivi fisici della memoria secondaria)

Organizzazione a livelli

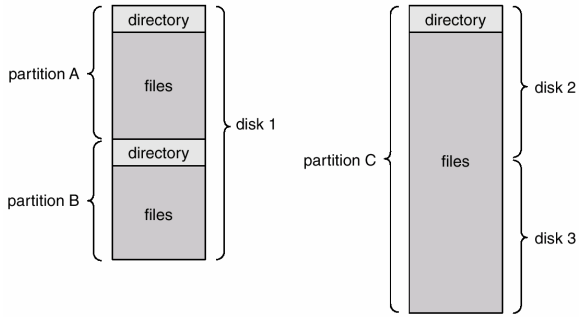


Struttura del file system



Struttura del disco

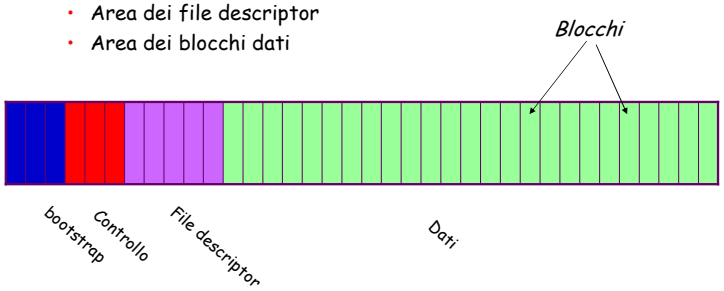
- Il disco e' suddiviso in partizioni (minidischi, volumi, ...)
- Ogni file system e' contenuto in un volume e un volume puo' contenere un solo file system



Corrispondenza 1-1 tra file system e partizione

Organizzazione di una partizione

- Dipende dal s.o. e dal file system
- In generale una partizione e' divisa in 4 aree distinte
  - Area di bootstrap
  - Area di controllo
  - Area dei file descriptor
  - Area dei blocchi dati



Organizzazione di una partizione

- Area di bootstrap
  - Contiene il programma per l'inizializzazione del sistema
  - Di solito e' il primo blocco di una partizione
  - Per uniformita' ce n'e' una in ogni partizione
  - Diversi nomi
    - Boot block (Unix file system)
    - Partition boot sector (NTFS)
- Area dei file descriptor
  - Contiene le strutture dati e gli indirizzi dei blocchi dei dati dei file
  - Diversi nomi
    - i-list (UFS)

Organizzazione di una partizione

- Area di controllo
  - Contiene informazioni su tutta la partizione
    - Dimensione della partizione
    - Numero e posizione dei blocchi liberi
    - Dimensione della i-list (area dei file descriptor)
  - Diversi nomi
    - Superblocco (UFS)
    - Master file table (NTFS)
- Area dei dati
  - Contiene i blocchi dei dati dei file e i blocchi liberi

Area dei blocchi dei dati

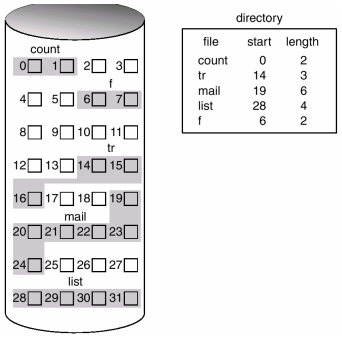
- Problema: come allocare lo spazio disco ai file in modo da avere spreco minimo di memoria e rapidità di accesso?
- Il metodo di allocazione dello spazio su disco descrive come i blocchi fisici del disco vengono allocati ai file:
  - Allocazione contigua
  - Allocazione concatenata
  - Allocazione indicizzata

Allocazione contigua

- Ciascun file occupa un insieme di blocchi contigui sul disco.
- Per reperire il file occorrono solo la locazione iniziale (# blocco iniziale) e la lunghezza (numero di blocchi).

Principali problemi

- i file non possono crescere
- frammentazione



directory

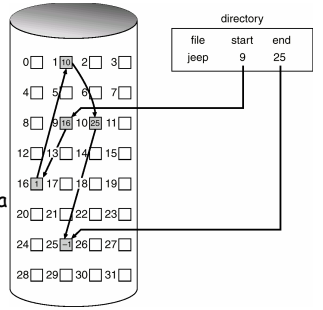
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Allocazione concatenata

- Ciascun file è una lista concatenata di blocchi su disco:
- i blocchi possono essere sparsi ovunque nel disco.
- La directory contiene l'indirizzo del blocco iniziale.
- Ogni blocco possiede un'area puntatore

Principali problemi

- per trovare un blocco e' necessario scorrere tutta la lista sul disco con numerosi posizionamenti della testina
- Perdita di efficienza



directory

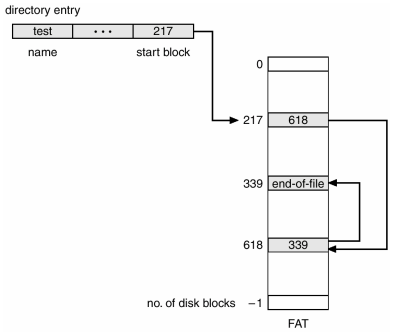
file	start	end
jeep	9	25

Allocazione tabellare

- Rappresenta una possibile soluzione al problema dello scorrimento della lista sul disco
- Mantenere una tabella contenuta in un unico blocco che contiene la lista dei puntatori ai blocchi del disco
- Per trovare un blocco si scorre la lista nella tabella e non sul disco

File Allocation Table
FAT

Implementata nei f.s. di
- MS-DOS
- OS/2



FAT (Microsoft)

- Prima versione della FAT dell'MS-DOS 1.0 (FAT12) usava 12 bit per l'indirizzamento dei blocchi
- 2^{12} = 4096 blocchi

Dim. del disco	Dim. dei blocchi
4 MB	1K
16 MB	4K
64 MB	16K

- Successive versioni a 16 e 32 bit

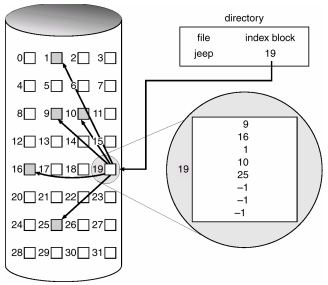
Blocchi grandi
=
frammentazione

Allocazione indicizzata

- Colleziona tutti i puntatori in un unico *blocco indice*.
- La directory contiene l'indirizzo del blocco indice
- Richiede una tabella indice.
- Permette l'accesso dinamico senza frammentazione esterna;

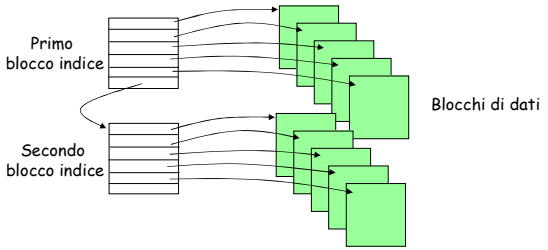
Principali problemi

- overhead per l'accesso al blocco indice (comunque inferiore all'allocazione concatenata)



File di grandi dimensioni

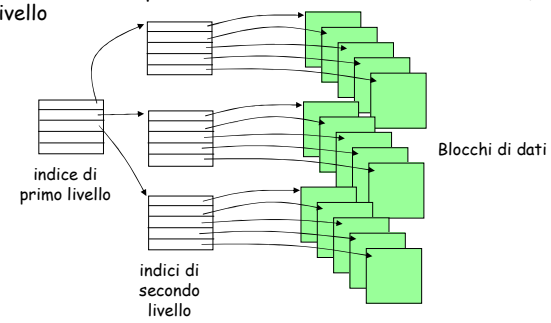
- Nell'esempio precedente se il file ha dimensione maggiore di 256Kbyte e' possibile concatenare i blocchi indice



Allocazione indicizzata – schema concatenata

Una alternativa

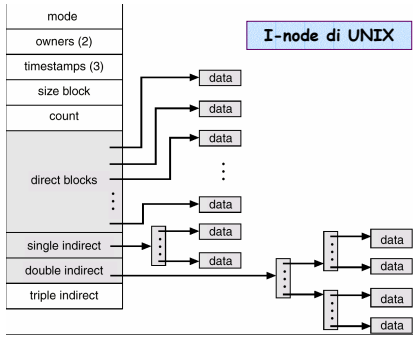
- Il blocco indice punta ad altri blocchi indice di secondo (e terzo) livello



Allocazione indicizzata – schema multilivello
(e multiaccesso !!)

Schema combinato: Unix

- Nel Unix File System, l'area dei descrittori (chiamata I-list: index list) contiene una tabella di descrittori chiamati i-node (index node)
- Ogni i-node e' accessibile attraverso l'indice della tabella (i-number)
- Un i-node contiene gli attributi di un file



- Indirizzi dei blocchi
  - 12 diretti
  - 1 indiretto singolo
  - 1 indiretto doppio
  - 1 indiretto triplo

Gestione dello spazio libero

- Poiche' i file sono entita' estremamente dinamiche (aumentano e diminuiscono in numero e in dimensione) e' necessario assicurarsi una gestione efficiente dello spazio libero
- L'area di controllo (superblocco) di una partizione contiene le informazioni sulla posizione dei blocchi liberi
- Tre possibili gestioni:
  - Bitmap
  - Lista dello spazio libero
  - raggruppamento

Bitmap

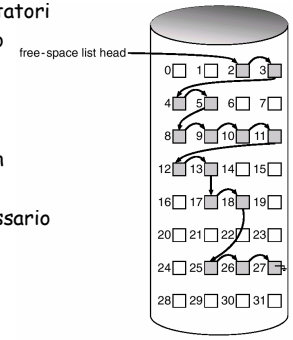
- Vettore di bit (n blocchi)

$$bit[i] = \begin{cases} 0 \Rightarrow \text{blocco}[i] \text{ occupato} \\ 1 \Rightarrow \text{blocco}[i] \text{ libero} \end{cases}$$

- Vantaggi: Buone prestazioni se il vettore è conservato in memoria centrale.
- Svantaggi: per trovare un blocco libero puo' essere necessario esaminare gran parte della bitmap

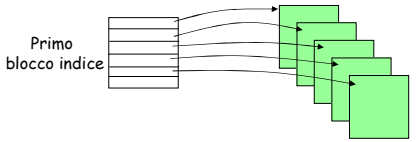
Lista concatenata

- Lista di blocchi liberi legati da puntatori
- Puntatori alla testa e alla coda sono memorizzati nel superblocco
- Vantaggio: 1 accesso per trovare un blocco libero
- Svantaggio: inefficiente se e' necessario trovare molti blocchi



Raggruppamento

- Raggruppamento: memorizzazione degli indirizzi di n blocchi liberi sul primo di tali blocchi. Sull'ultimo sono indirizzati altri blocchi, etc.

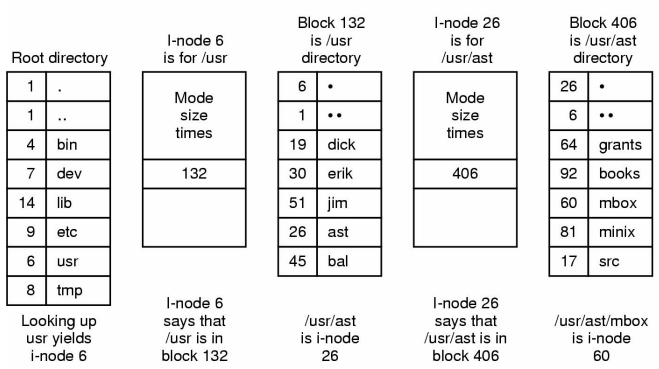


- Vantaggio: rapida ricerca di blocchi liberi
- Svantaggio: gestione inefficiente se il numero di blocchi e' grande

Esempio di implementazione di file system

UNIX V7

Qual'e' l'i-node del file /usr/ast/mbox?



File system annotati

- Gli algoritmi di ripristino basati sulla registrazione delle modifiche possono essere applicati con successo al problema della coerenza dei file system.
- I file system annotati (*journaling file system*) annotano tutte le modifiche ai metadati in un *giornale delle modifiche.*
  - Quando le modifiche sono state annotate, le operazioni si considerano portate a termine e le chiamate di sistema ritornano.
  - Completate le modifiche alle strutture dati del file system, la transazione è completata e le annotazioni possono essere rimosse dal giornale.
- Tali tecniche hanno un impatto anche sulle prestazioni del sistema, in quanto velocizzano gli aggiornamenti percepiti dalle applicazioni.
  - Le modifiche avvengono con accessi sequenziali, invece che con accessi diretti e sincroni alle strutture dati del file system.