memoria secondaria

capitolo 12 del libro (VII ed.)

wikipedia:
GRUB, LILO, NTLDR, RAID

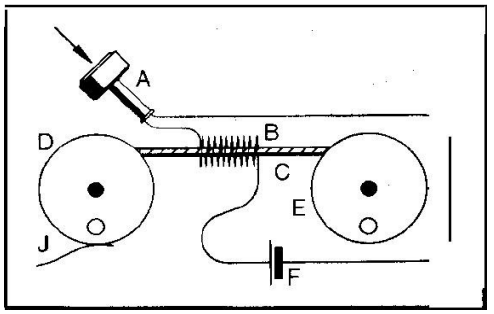
Introduzione

- dopo aver visto il FS dal punto di vista logico e dal punto di vista delle strutture di SO necessarie per la sua gestione, vediamo la struttura dei supporti di memorizzazione su cui il FS risiede
- Vari tipi di supporti
  - memoria secondaria: dischi magnetici
  - memoria terziaria: dischi rimovibili (ottici o magnetici), flash memory (eventualmente accessibile tramite supporto USB), nastri magnetici, ...

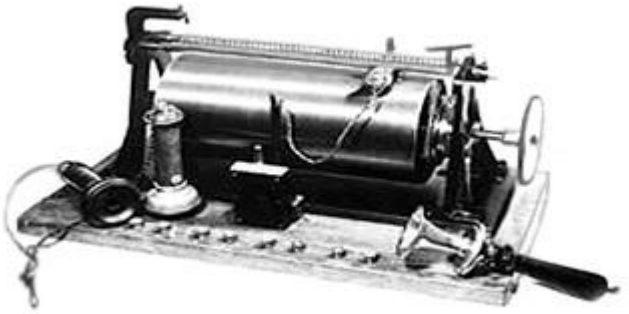
Un po' di storia

- La memoria su supporto magnetico viene proposta nel 1888, ad opera di Oberlin Smith, come supporto per la registrazione della voce
- solo nel 1898, Valdemar Poulsen brevetta il primo registratore magnetico e lo chiama *telegrafono*
- nel 1928 Fritz Pfeumler progetta il primo registratore su nastro magnetico
- In origine tutte le registrazioni su supporto magnetico sono di tipo analogico, cioè invece di registrare bit venivano registrati valori compresi in un intervallo continuo
- Oggi la memoria secondaria degli elaboratori è costituita da uno o più dischi magnetici su cui i dati sono registrati in formato digitale

Oberlin Smith



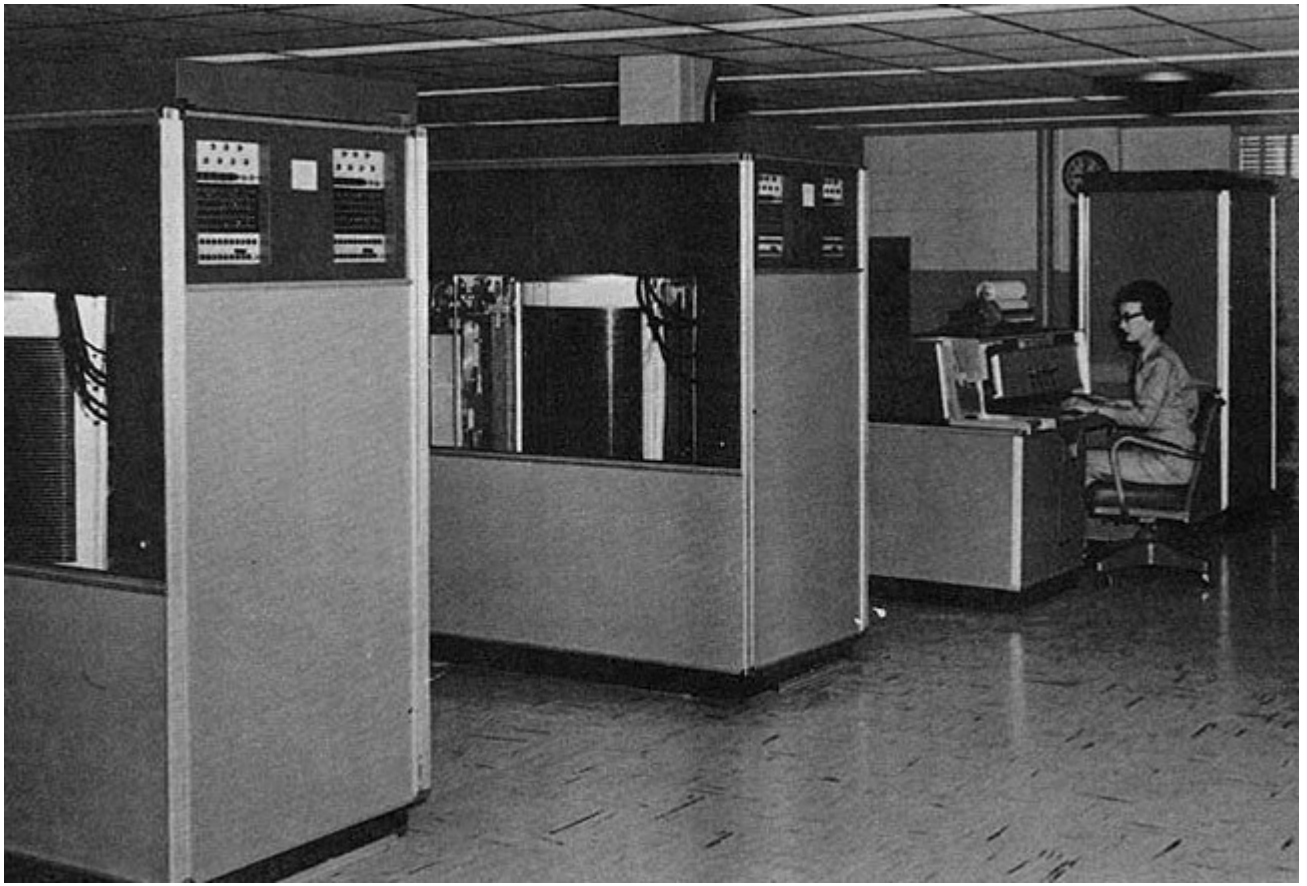
il suo progetto



il telegrafono di Poulsen

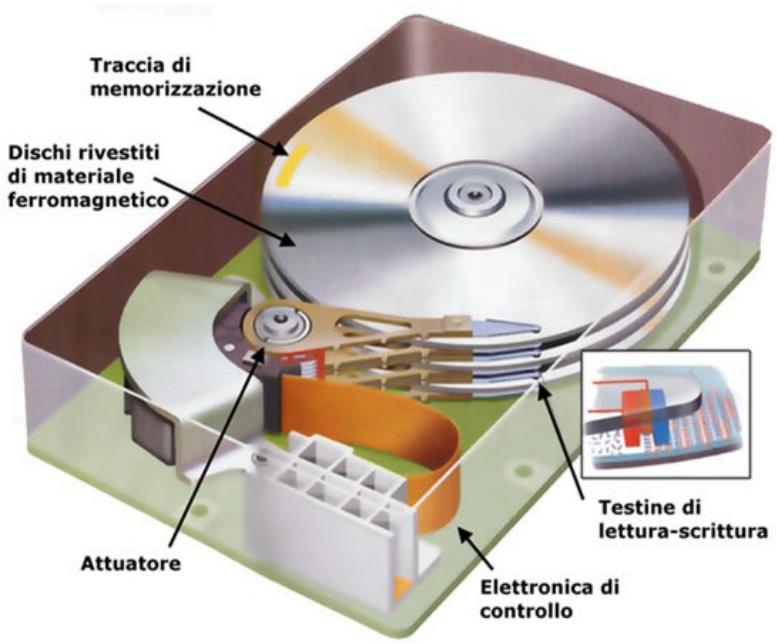
Un po' di storia

- Disco magnetico come memoria secondaria, sviluppato da IBM nel 1956
- Nome "hard disk" usato a partire dagli anni '80 in contrapposizione a Floppy Disk

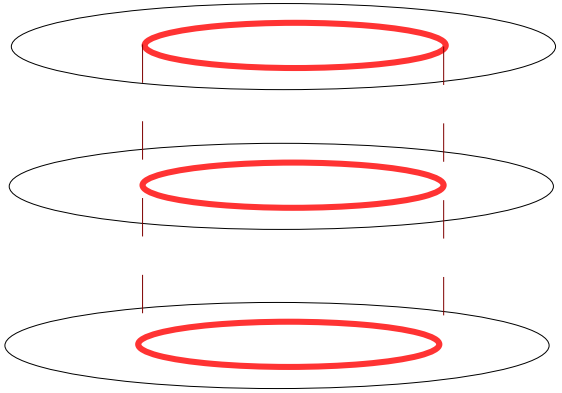


Struttura di un disco

- Un disco è costituito da un insieme di piatti le cui due superfici sono ricoperte di materiale magnetico
- Ogni disco ha una coppia di testine di lettura/scrittura che sono sospese sul relativo piatto (non lo toccano ma l distanza è minima, dell'ordine dei micron) e operano ciascuna su una superficie. Se una testina plana sul disco lo ara e occorrerà sostituirlo.



Ogni superficie è suddivisa in un insieme di tracce circolari, divise in porzioni dette settori. L'insieme di tracce corrispondenti appartenenti ai diversi piatti è detto cilindro



Principio CHS

- CHS = Cylinder, Head, Sector
- I dati organizzati secondo questo principio hanno come indirizzo fisico un'ennupla contenente i seguenti id:
    - Id piatto
    - Id traccia
    - Id cilindro
    - Id settore
    - Id blocco
    - Id testina R/W
    - Id cluster

Funzionamento

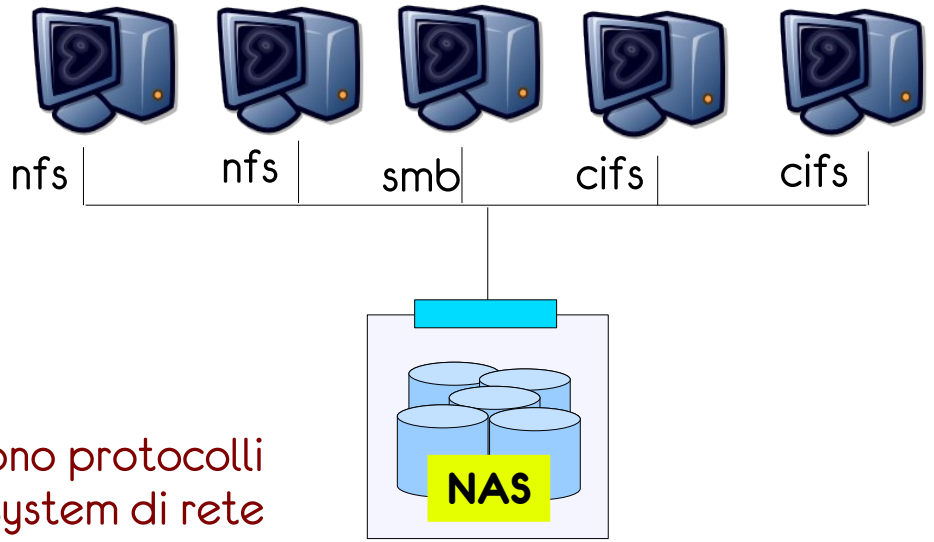
- Quando è in funzione un disco ruota a una velocità oggi compresa fra i 4000 e i 15000 giri al secondo
- Tutte le operazioni che abbiamo menzionato (lettura, scrittura, ricerca di un blocco) richiedono la ricerca di qualche settore su una delle superfici di qualche piatto del disco
- Il tempo necessario, detto tempo di posizionamento, comprende:
  - tempo di seek: richiesto per posizionare il braccio sul cilindro giusto
  - latenza di rotazione: richiesto affinché il settore giusto si posizioni, a seguito della rotazione, sotto alla testina del braccio
- a ciò si aggiunge il tempo di trasferimento da RAM a disco o viceversa

Connessione al calcolatore

- Un disco è connesso direttamente a un calcolatore attraverso un bus di I/O
- Esistono diverse tecnologie di connessione, es:
  - EIDE
  - ATA/SATA (già menzionate)
  - SCSI (già menzionata)
  - USB
- Il trasferimento dei dati è gestito da una coppia di controllori:
  - un adattatore, posto sul lato computer
  - un controllore del disco, incorporato nel disco stesso
- un comando di I/O viene inserito in RAM in una pagina mappata sull'adattatore, viene quindi trasferito all'adattatore, che lo passa al controller del disco, che esegue il comando utilizzando il buffer cache

Connessione via rete

- i dischi possono essere fisicamente separati dagli elabroatori e connessi alle postazioni di lavoro tramite una rete
- si parla di NAS, network-attached storage, accessibili tramite chiamate di procedura remota (RPC)
- la memoria secondaria in questi casi è spesso organizzata come una batteria di dischi RAID (es. in laboratorio)



nfs, cifs, smb sono protocolli per file system di rete

Struttura e formattazione

- Dal punto di vista logico un disco è un array di blocchi di pari dimensioni, tipicamente 512 byte
- L'array è mappato sul disco a partire dal primo settore della prima traccia del primo cilindro (quello più esterno). Completato un cilindro ci si sposta internamente, al successivo
- Appena prodotto però un disco non ha questa struttura, che viene imposta con un'operazione detta formattazione fisica, solitamente operata dal costruttore
- L'acquirente/utilizzatore può operare un altro tipo di formattazione detta formattazione logica, che consiste nella creazione di un file system

formattazione fisica

- la formattazione fisica crea una struttura dati speciale per ogni settore, contenente un'intestazione, un'area dati e una coda.
- intestazione e coda sono usati dal controller del disco, contengono fra le altre cose:
  - numero del settore
  - un codice per la correzione degli errori: tramite questo codice il controller può verificare se il settore è integro e, se solo pochi bit risultano alterati, li può identificare e correggere
- la dimensione dei settori è tipicamente di 512 byte

formattazione logica

- Quando un utente installa un SO effettua un altro tipo di formattazione, legato al partizionamento dei dischi e alla scelta dei file system da installare
- Si parla di formattazione logica
- La formattazione logica crea sul disco stesso le strutture dati per la gestione di quel tipo di file system (es. tabella FAT) impostandone i valori iniziali
- Se si hanno più partizioni si installeranno più file system, con la conseguente scrittura delle relative strutture dati, file system per file system

Scheduling del disco

- Il SO è il gestore di tutte le risorse fisiche dell'elaboratore, dischi compresi
- Gestire bene un disco significa (1) minimizzare i tempi di posizionamento (2) massimizzare l'ampiezza di banda:
  - tempo di seek (o di ricerca)
  - tempo di latenza (latenza di rotazione)
  - ampiezza di banda: numero di byte trasferiti fratto il tempo intercorso fra la prima richiesta e il termine dell'ultimo trasferimento
- tutte questi parametri sono migliorabili adottando opportune strategie

Scheduling del disco

- richiesta: è una system call effettuata dal SO
- le richieste richiedono tempo per essere evase
- di solito ogni disco ha una coda di richieste pendenti piuttosto lunga
- lo scheduling del disco è un'implementazione di un criterio di selezione tramite il quale si decide quale richiesta pendente verrà eseguita come successiva
- la scelta è fatta tenendo conto dei dati che costituiscono la richiesta stessa:
  - operazione di lettura o di scrittura
  - indirizzo su disco
  - indirizzo in RAM
  - quantità di byte da trasferire

Politiche di scheduling

- FCFS: first come first served
- SSTF: shortest seek time first
- SCAN: algoritmo per scansione o dell'ascensore
- C-SCAN: scansione circolare
- LOOK/C-LOOK

First come First Served

- le richieste sono processate secondo l'ordine di arrivo
- è una strategia equa ma non particolarmente veloce
- supponiamo di avere questa sequenza di richieste, i vari numeri rappresentano i cilindri contenenti le tracce di interesse:

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

- supponiamo che inizialmente la testina sia posizionata sul cilindro 53, complessivamente verrà percorsa una distanza, misurata in cilindri attraversati, pari a

  (98-53)+(183-98)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65) = 640

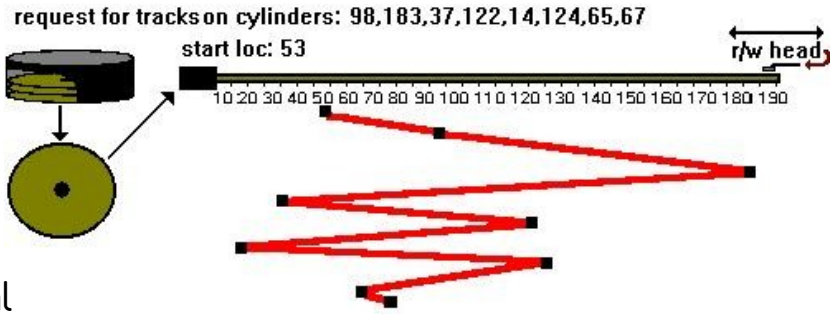


immagine tratta da
http://www.cs.wayne.edu/~tom/guide/os.html

Shortest seek time first

- per migliorare le prestazioni del disco è possibile adottare politiche per privilegiano le richieste inerenti cilindri vicini alla posizione corrente della testina
- shortest-seek time first privilegia appunto la richiesta che minimizza il tempo di seek
- nel nostro esempio la sequenza

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

- verrebbe gestita in quest'ordine (sempre partendo da 53):
- 65, 67, 37, 14, 98, 122, 124, 183
- la distanza coperta risulta: (65-53) + (67-65) + (67-38) + ... + (183-124) = 236

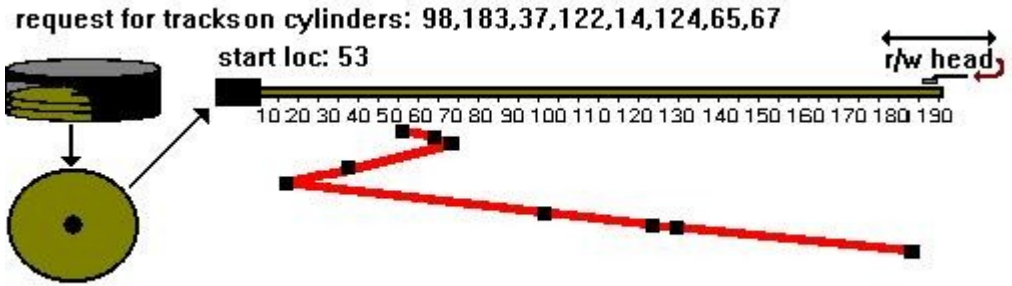


immagine tratta da http://www.cs.wayne.edu/~tom/guide/os.html

SSTF

- SSTF ha gli stessi difetti di tutte le tecniche a priorità: possibilità di generare starvation, cioè produrre attese indefinite per qualche richiesta
- Si osservi che il risultato ottenuto, pur essendo migliore di FCFS, non è ottimale infatti incrementando il "costo a breve termine" è possibile ridurre ulteriormente quello "a lungo termine"
- **Esempio**: se mi sposto da 53 a 37 e poi a 14 prima di invertire la direzione e attraversare 65, 67, 98, 122, 124 e 198 percorro complessivamente 208 cilindri!!
- 37 è più lontano da 53 di 65, quindi a breve termine la scelta sembra peggiore, invece sul lungo termine questa strategia risulta molto migliore

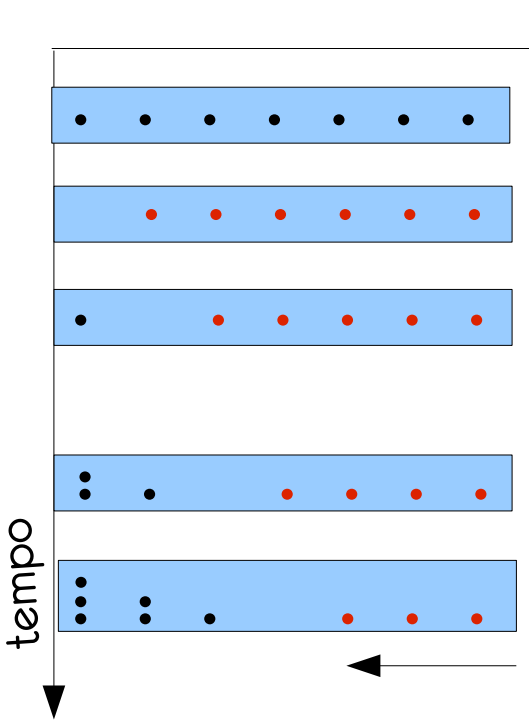
SCAN

- anche detto algoritmo dell'ascensore:
  - il braccio parte da un estremo del disco e lo percorre tutto servendo le richieste man mano che attraversa i cilindri richiesti
  - arrivato all'altro estremo torna indietro ripetendo la procedura in senso opposto
- consideriamo il solito esempio e supponiamo che la testina si stia muovendo verso il cilindro 0 (esterno), verranno servite nell'ordine:

  37, 14, 65, 67, 98, 122, 124, 183
- per un totale di 208 cilindri attraversati
- se però la direzione fosse stata opposta avremmo ottenuto 65, 67, 98, 122, 124, 183, 37, 14 per un totale di 299 cilindri attraversati (!!)

SCAN: commenti

- questo algoritmo presenta un problema
- se le richieste riguardanti i vari cilindri giungono secondo una distribuzione uniforme si avrà un accumulo di richieste non soddisfatte verso l'estremo del disco opposto a quello a cui si trova o si sta muovendo la testina
- si ha qualcosa del genere:



T0: ho una richiesta per ogni cilindro del disco

T1: consumo la richiesta per il primo cilindro e avanzo di uno

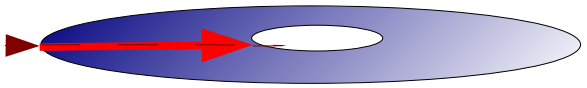
T2: consumo la richiesta per il II cilindro, intanto mi giunge una richiesta per il primo, non la servo perché procedo nell'altro senso. Non visualizzo le richieste per i cilindri successivi perché saranno presto servite

T3: consumo le richieste per il III cilindro, intanto mi giungono nuove richieste

T4: all'estremo sinistro le richieste accumulate diventano sempre più vecchie, quando tornerò indietro servirò richieste accumulatesi di recente

C-SCAN

- C-SCAN (circular scan) cerca di ovviare all'inconveniente appena descritto
- si comporta come SCAN percorrendo il disco in una direzione e servendo via via le varie richieste
- quando raggiunge l'estremo opposto la testina viene riportata direttamente alla posizione di inizio, senza servire alcuna richiesta nel viaggio di ritorno, e si ricomincia
- è un algoritmo circolare perché tratta il primo e l'ultimo cilindro come se fossero adiacenti



- Il vantaggio rispetto a prima è che i tempi di attesa delle richieste risultano più uniformi
- verranno servite nell'ordine: 65, 67, 98, 122, 124, 183, 199 (fine piatto), 0 (inizio piatto), 14, 37
- totale: 2+21+24+2+59+[16]+[199]+14+23= 145+[215] = 360 di cui 215 senza letture quindi attraversati velocemente

Look

- LOOK e C-LOOK sono le due varianti SCAN e C-SCAN effettivamente usate
- la differenza è che il braccio procede in una direzione finché ci sono richieste per cilindri che si raggiungono spostandosi in quella direzione
- non necessariamente si raggiunge l'estremo opposto
- per esempio C-LOOK si comporta in questo modo: 65, 67, 98, 122, 124, 183, 14, 37
- da 183 non si va a fine piatto e quando si salta indietro si salta al primo cilindro richiesto (14)
- in totale avremo 131+[169] = 300

Scelta dell'algoritmo di scheduling

- fino ad ora abbiamo considerato solo il tempo di seek però il tempo di posizionamento su di un blocco dipedne anche dalla latenza di rotazione, che può essere altrettanto lunga
- l'associazione fra blocchi e settori di solito non è nota al SO quindi non si possono attuare strategie particolari
- in genere se la coda delle richieste è piuttosto scarica (es. PC di casa) il sovraccarico computazionale dato dall'ordinamento della coda delle richieste non è giustificato, quindi un FCFS va benissimo
- se invece la coda è lunga (es. server) SSTF e LOOK sono una scelta migliore
- un impatto importante sui tempi di accesso ai blocchi dati dei file è comunque dato dalla struttura delle directory e dalla tecnica di allocazione dei blocchi disco. Il tempo di reperimento dei dati sarà tanto maggiore quanti più salti occorre fare nel disco (es. allocazione concatenata)