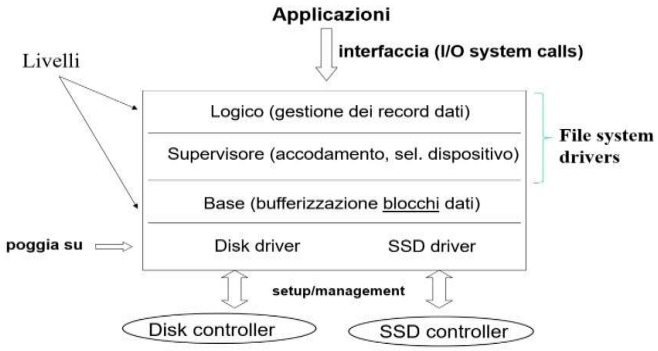
Virtual file system 2 File

martedì 28 ottobre 2025 17:27

Focalizziamo ora su una parte del VFS ovvero il **file system** : ovvero entità dove la minima unità informativa archiviabile è il **file (unico archivio di info**). Similmente alla minima unità informativa archiviabile vi è quella minima unità informativa accessibile : **record : quindi un file è un insieme di record** . Le applicazioni lavorano solo sui record. Quindi un file system associa ad ogni file dei metadati (nome, protezione ed altro). in generale un file system ha questa architettura :



Quindi vediamo le operazioni base che possiamo eseguire (esposte come system call) :

Creazione
- allocazione di un "record di sistema" (RS) per il mantenimento di informazioni relative al file (e.g. attributi) durante il suo tempo di vita

Scrittura/Lettura (di record)
- aggiornamento di un indice (puntatore) di scrittura/lettura valido per sessione

Apertura (su file esistenti)
- inizializzazione dell'indice di scrittura/lettura per la sessione corrente

Chiusura
- rilascio dell'indice di scrittura/lettura

Riposizionamento
- aggiornameno dell'indice di scrittura/lettura

Eliminazione
- deallocazione di RS e rilascio di memoria (blocchi dati) sul dispositivo

Troncamento
- rilascio di memoria (blocchi dati) sul dispositivo

Torniamo ora all'indice di lettura/scrittura (mantenuto nella sessione) :

- L'indice di scrittura/lettura **NON** fa parte di RS (accessi concorrenti su punti del file scorrelati)

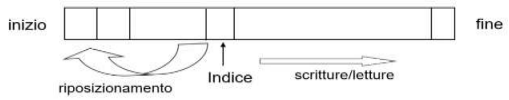
- L'indice di scrittura/lettura **PUO'** essere condiviso da piu' processi
  - ✓ quindi **NON** fa parte della singola immagine di processo mantenuta dal sistema operativo
  - ✓ fa tipicamente parte dell'immagine di sessione

- Le modalita' di aggiornamento dell'indice di scrittura/lettura **in riposizionamento** dipendono dai metodi di accesso ai record di un file supportati dallo specifico file system

Vediamo ora i metodi di accesso ai file : metodo che permette realmente di lavorare sui record del file :

1. **Sequenziale**
   - i records vengono acceduti sequenzialmente
   - l'indice di scrittura/lettura e' incrementato di una unita' per ogni record acceduto
   - il riposizionamento dell'indice puo' avvenire solo all'inizio del file

   a.

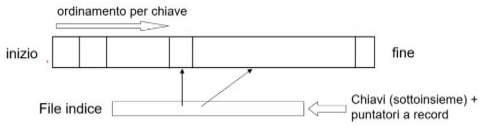


Tipico di:
- **File sequenziali**, caratterizzati da record di taglia e struttura fissa
- **File a mucchio**, caratterizzati da record di taglia e struttura variabile (ogni record mantiene informazioni esplicite su taglia e struttura)

   b. Con questo metodo il riposizionamento dell'indice non è arbitrario : se in avanti devo scorrere tutti i record, mentre se indietro ripunterà al primo record

c. **Indicizzato**

- **File sequenziali indicizzati**, caratterizzati da record di taglia e struttura fissa, ordinati in base ad un campo chiave



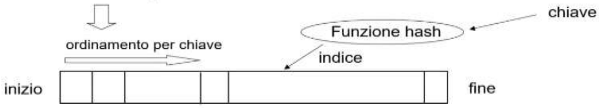
i.
- esiste un file sequenziale di indici associato a ciascun file di dati
- i record sono ordinati per "chiave"
- tramite il file di indici ci si puo' posizionare in punti specifici del file di dati (ovvero in punti con valori specifici del campo chiave)
- i records vengono acceduti sequenzialmente una volta posizionati sui punti stabiliti
- l'indice di scrittura/lettura e' manipolato (incrementato) di conseguenza
- il riposizionamento dell'indice puo' avvenire solo all'inizio del file

ii. Anche qui il riposizionamento non avviene in modo arbitrario ma solo all'inizio, ma usando il file degli indici ci spostiamo subito al dato.

2. **Diretto**

- il riposizionamento dell'indice puo' avvenire in un qualsiasi punto del file
- si puo' accedere direttamente all'i-esimo record (senza necessariamente accedere ai precedenti)
- dopo un accesso all'iesimo record, l'indice di scrittura/lettura assume il valore i+1

a. **Tipico di:**
- **File diretti**, caratterizzati da record di taglia e struttura fissa
- **File hash**, caratterizzati da record di taglia e struttura fissa con ordinamento per chiave



b. Utilizzato da tutti i moderni file system.

Vediamo ora l'ultima cosa , ovvero come sono fatte le directory:

- La directory e' un file ``speciale``
- Essa contiene informazioni per poter accedere a file veri e propri, contenenti record di dati
- Il modo con cui le informazioni vengono mantenute nelle directory (ovvero nei file associati alle directory) determinano la cosi' detta **struttura di directory**

Tipica struttura di directory
- nomi dei file contenuti nella directory
- informazioni di identificazione dei RS associati ai file

Tornando ai file : vediamo ora che collegamento c'è tra un file e le informazioni memorizzate in un dispositivo (memoria di massa , hard disk ecc ecc): ciascun file è allocato sul dispositivo come insieme di blocchi , i quali non sono necessariamente contigui. Ma come sono organizzati questi blocchi ? Di solito si usano 3 principali modi :

1. **Organizzazione fissa**
   a. I record del file sono di taglia fissa (non file a mucchio)
   b. Occhio alla frammentazione interna
2. **Organizzazione variabile con riporto**
   a. I record del file hanno taglia variabile e possono venir divisi tra più blocchi
3. **Organizzazione variabile senza riporto**
   a. I record del file hanno taglia variabile e possono venir divisi tra più blocchi
   b. Ma in questa configurazione non è possibile fare "il riporto": o scrivo/leggo tutto il blocco oppure il pezzetto che rimane lo devo mettere in un nuovo blocco.
   c. Si ha quindi **frammentazione**

Nota : per usare i blocchi , dobbiamo conoscere il loro stato (se possibile utilizzarlo o già utilizzato) . Di solito per vedere se un blocco è disponibile e/o ha posti liberi si usano due tecniche o lista libera o bit map. Vediamo la bit map : mappa di bit con bit associato a ciascun blocco del dispositivo (se 1 non libero altrimenti si); mentre la seconda è una lista che mostra i blocchi possibili in **modo**

compatto usando i record di sistema.

Track	Sector	Number of sectors in hole
0	6	5
0	6	6
1	0	10
1	11	1
2	1	1
2	3	3
2	7	5
3	0	3
3	9	3
4	3	8

(a)

Track	Sector 0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	1	0
2	1	0	1	0	0	1	1	0	0	0	0	0
3	0	0	0	1	1	0	0	1	1	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	1

(b)

(a) Lista Libera
(b) Bit Map