12 - File System

sabato 14 maggio 2022 17:04

•File System: insieme di strutture che permette la conservazione, la gestione e l'accesso di dati nella memoria secondaria

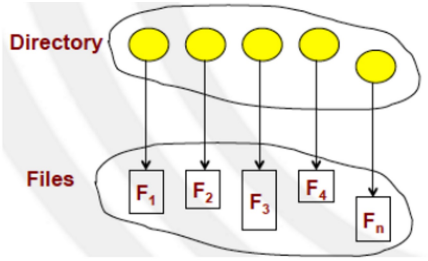
INTERFACCIA DEL FILE SYSTEM

FILE

•File: insieme di dati digitali correlati da un nome
•Il file dispone di diversi attributi che vengono memorizzati su disco nella directory di appartenenza:
-**nome**
-**tipo**
-**posizione: puntatore**
-**dimensione**
-**protezione**
-**tempo, data, identificazione:** quando è stato creato, ultimo accesso, ecc.
•È possibile eseguire le seguenti operazioni con i file:
-**creazione:** si cerca spazio su disco e si aggiunge un nuovo elemento alla directory
-**scrittura:** tramite una system call che specifica nome file e dati da scrivere (avviene in memoria)
-**lettura:** tramite una system call che specifica nome file e dove mettere i dati letti in memoria (avviene in memoria)
-**riposizionamento all'interno del file:** spostare il puntatore di lettura file
-**cancellazione:** libera spazio associato al file
-**troncamento:** mantiene gli attributi ma cancella il contenuto del file
-**apertura:** cerca il file nella directory su disco, lo copia in memoria e inserisce un riferimento nella tabella dei file aperti
-**chiusura:** copia il file in memoria su disco

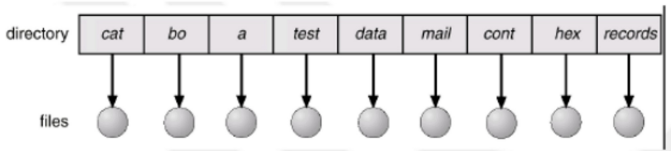
DIRECTORY

•Directory: sono a sua volta dei file che contengono dei nodi che puntano ai file
•Contengono gli attributi di ogni singolo file presente in quella cartella
•È possibile eseguire le seguenti operazioni con le directory:
-**aggiungere file**
-**cancellare file**
-**visualizzare il contenuto di una directory**
-**rinominare file**
-**ricercare file** (anche attraverso dei filtri)
-**attraversare il file system**



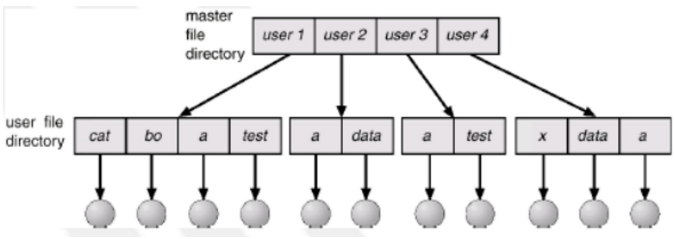
DIRECTORY A UN LIVELLO

•Unica directory per tutti gli utenti
•Problemi:
-serve un nome diverso per ogni file
-usabile da solo 1 utente



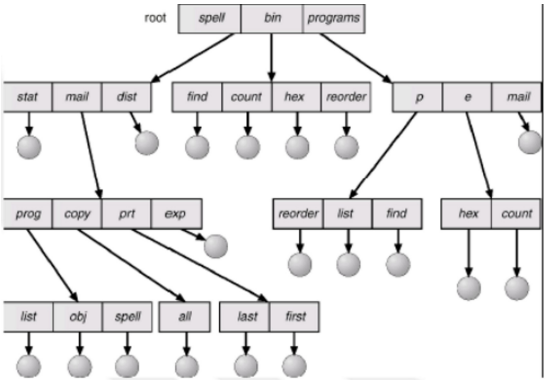
DIRECTORY A 2 LIVELLI

•Directory separata per ogni utente
•Più utenti possono avere file con lo stesso nome



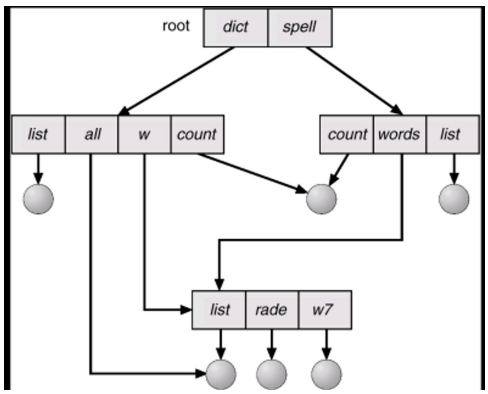
DIRECTORY AD ALBERO

- Ogni directory può avere altre cartelle al suo interno
- Facilita la ricerca dei file



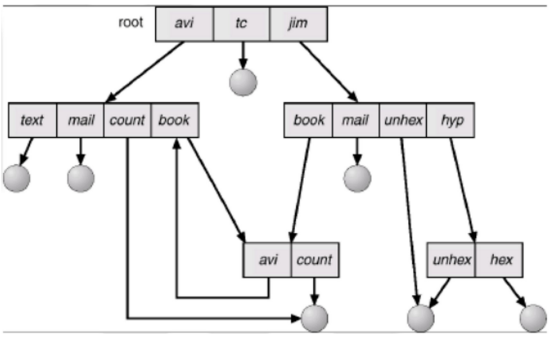
DIRECTORY A GRAFO ACICLICO

- Permette a più utenti di condividere directory e file
- La condivisione può avvenire tramite:
  - **link simbolico:** file contenente il percorso al file richiesto
  - **hard link:** contatore che mantiene il # di riferimenti



DIRECTORY A GRAFO GENERICO

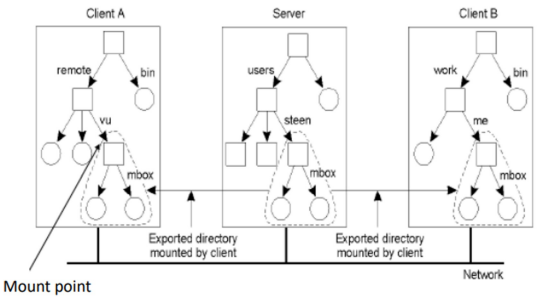
- Consente l'utilizzo di cicli
  - esempio immagine: avi che punta book
- Problema: i cicli possono portare a loop infiniti
  - si può risolvere con algoritmi di controllo (molto costosi)



MOUNT DI FILE SYSTEM

- Mount: attaccare un file system a una directory
  - esempio: quando inserisco una chiavetta USB al pc sto "mountando" il file system della chiavetta USB al pc

Mount di file system – esempio



PROTEZIONE

- Il possessore di un file può decidere:
  - cosa è possibile fare su un file
  - chi può farlo

IMPLEMENTAZIONE DEL FILE SYSTEM

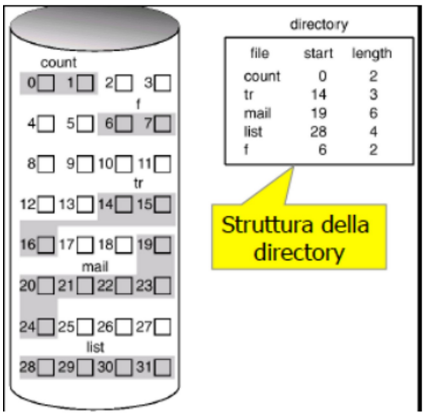
•Blocco: cluster, unità minima di settori

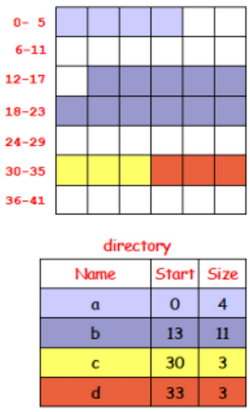
Per gestire un file system si usano diverse strutture dati:
•Strutture dati su disco:
-**blocco di boot:** informazioni necessarie per avviare l'OS
-**blocco di controllo delle partizioni:** dettagli sulle partizioni
-**strutture di directory:** descrivono l'organizzazione dei file
-**descrittori di file:** dettagli sui file e puntatori dei blocchi
•Strutture dati su memoria:
-**tabella delle partizioni:** informazioni sulle partizioni montate
-**strutture di directory:** copia in memoria delle directory a cui si è fatto accesso di recente
-**tabella globale dei file aperti:** tabella che tiene traccia dei descrittori dei file aperti al momento
-**tabella dei file aperti per ogni processo:** puntatore alla tabella precedente

•I blocchi su disco possono essere allocati in 3 modi:
-**allocazione contigua**
-**allocazione a lista concatenata (linked)**
-**allocazione indicizzata**

ALLOCAZIONE CONTIGUA

•Ogni file occupa un insieme di blocchi contigui (uno dopo l'altro) su disco
•**Vantaggi:**
-semplice da implementare
-non muovo la testina perché basta ruotare il disco
•**Svantaggi:**
-elevata frammentazione del disco
-estendere i file (farli aumentare di dimensione) può risultare problematico





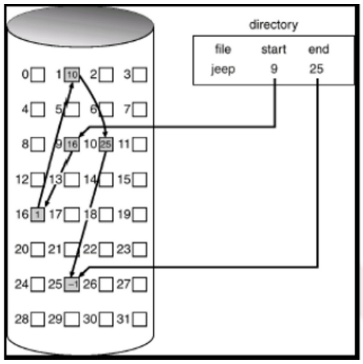
ALLOCAZIONE CONTIGUA - VARIANTE LINUX

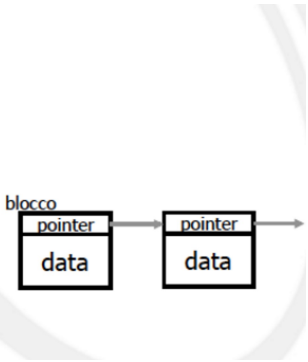
•Alcuni file system moderni usano una variante dell'allocazione contigua (es Linux ext2fs)
•Questa consiste nell'utilizzare extent al posto dei blocchi
•Extent: serie di blocchi contigui. Più extent formano un file
NOTA: i vari extent non sono contigui, ma connessi come in un'allocazione a lista (vedi sotto)

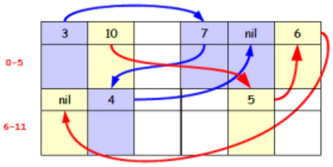
ALLOCAZIONE A LISTA

•Ogni file è una lista di blocchi
•I blocchi possono essere **ovunque** nel disco
-ogni blocco ha un puntatore che punta al successivo
•**Vantaggi:**
-creazione semplificata: basta cercare qualsiasi blocco libero
-estensione semplificata: stesso motivo di sopra
-niente frammentazione

•**Svantaggi:**
-ci vuole tempo a scorrere tutti i blocchi
-se si perde un puntatore, si perde l'intero file (si può risolvere con liste doppiamente concatenate)







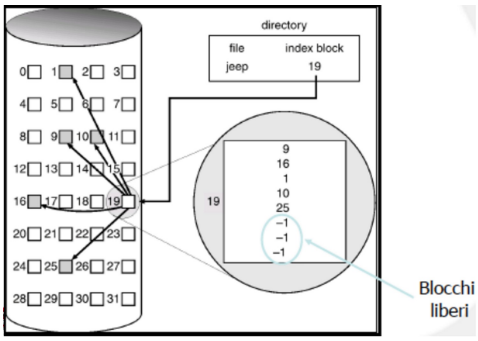
directory

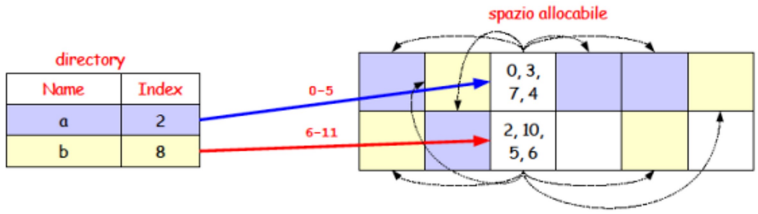
Name	Start	End
a	0	4
b	1	6

•Si può migliorare l'efficienza di questa allocazione tramite l'uso di una FAT (File Access Table)
•FAT: tabella che occupa i primi blocchi e che contiene una lista di tutti i blocchi allocati
-per ogni blocco si registra anche la posizione del suo blocco successivo
•Problema: le FAT occupano troppa memoria rendendo la scansione inefficiente

ALLOCAZIONE INDICIZZATA

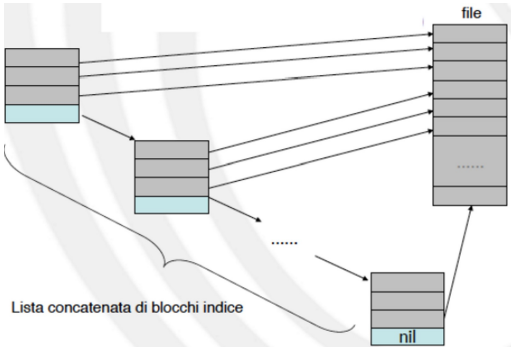
•Ogni file ha un blocco indice punta a tutti gli altri blocchi dello stesso file
•**Vantaggi:**
-niente frammentazione
-accesso casuale efficiente
•**Svantaggi:**
-overhead del blocco indice per la index table
-tende a sprecare più memoria, perché ogni file richiede l'allocazione del blocco indice
-la dimensione del blocco indice indice limita la dimensione massima del file (soluzioni sotto)





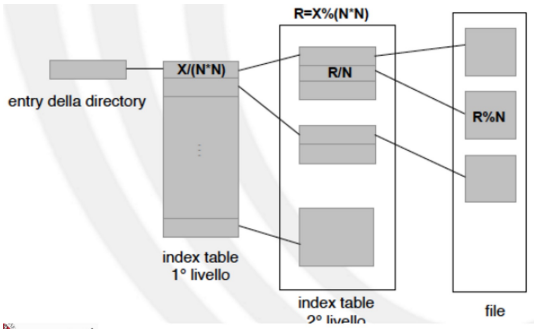
SOLUZIONE 1 - LINKED SCHEME (SCHEMA CONCATENATO)

•Si usano **più blocchi indice collegati tramite una linked list**
•Se un blocco non è sufficiente, se ne crea un altro riferibile dall'ultimo blocco indice



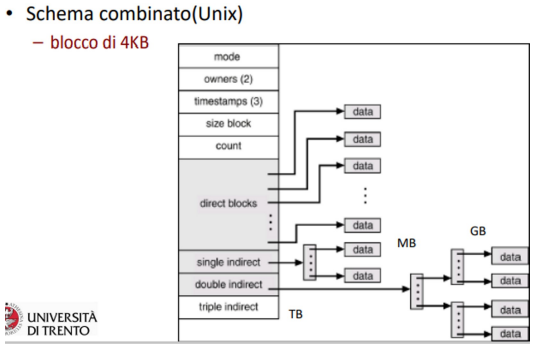
SOLUZIONE 2 - MULTILEVEL SCHEME (SCHEMA MULTILIVELLO)

- Si usa una **tabella esterna contenente tutti i blocchi indice di un file**
- Può essere a più livelli



SOLUZIONE 3 - COMBINED SCHEME (SCHEMA COMBINATO)

- Si usano gli **i-node, blocchi indice che puntano ad altri i-node**
  -nelle prime n-1 posizioni, puntano a blocchi di un file
  -nell'ultima posizione, puntano ad un altro i-node
- Possono puntare a più i-node alla volta (single, double triple indirect)



IMPLEMENTAZIONE DELLE DIRECTORY

- La directory contiene una lista di informazioni di ogni file/sottocartella che contiene

Può essere implementata in 2 modi:

- **Lista di puntatori:** la directory contiene una lista di puntatori al primo blocco di ogni file
  -utile per directory con pochi file
  -poco efficiente
- **Tabella hash:** struttura dati più pesante
  -utile per directory con tanti file
  -molto efficiente
  -possono esserci collisioni

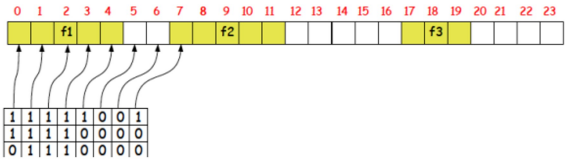
GESTIONE DELLO SPAZIO

•Bisogna tener traccia dei blocchi liberi
•Si può fare in 4 modi

MODO 1 - VETTORE DI BIT

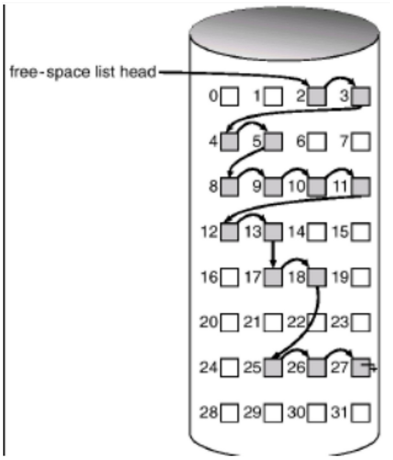
•Vettore di bit dove ogni posizione ci indica se un blocco è libero o occupato
-molto poco efficiente

- Vettore di bit, uno per blocco
  - Bit[i] = 0 ⇒ blocco i libero
  - Bit[i] = 1 ⇒ blocco i occupato
- Esempio: n blocchi



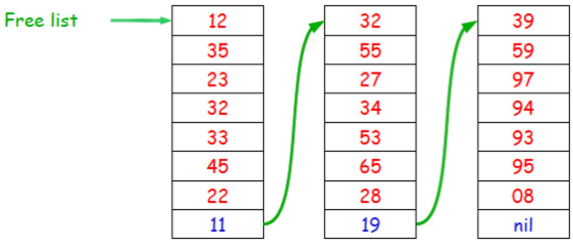
MODO 2 - LINKED LIST

•Ogni blocco libero punta a un altro blocco libero
-spazio contiguo non ottenibile



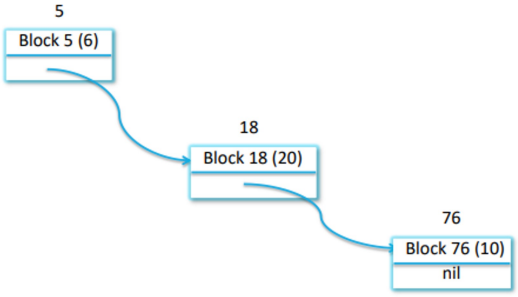
MODO 3 - RAGGRUPPAMENTO

•Uso dei blocchi come liste che elencano tutti i blocchi liberi
•L'ultima entry di questi blocchi punta a un altro blocco usato come lista



MODO 4 - CONTEGGIO

•Si usa una lista che contiene ad ogni entry un blocco libero e quanti blocchi liberi consecutivi ha
-si presuppone che i blocchi siano liberi in maniera contigua



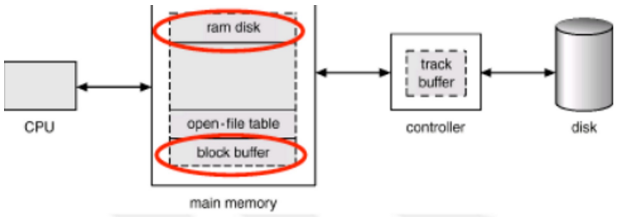
EFFICIENZA E PRESTAZIONI

EFFICIENZA

- I dischi funzionano a **collo di bottiglia**
- La loro efficienza dipende da:
  - l'algoritmo di allocazione dello spazio su disco
  - tipi di dati contenuti nella directory

PRESTAZIONI

- Per migliorare le prestazioni si usano:
  - **Dischi virtuali (RAM disk)**
  - **Cache del disco**



DISCHI VIRTUALI

- Una parte della RAM viene gestita come se fosse un disco
  - accetta tutte le operazioni standard dei dischi eseguendole in memoria
  - veloce
  - supporta solo file temporanei

CACHE DEL DISCO

- Una cache in memoria che memorizza alcuni blocchi per accedervi più velocemente
- Sfrutta il principio di località:
  - **spaziale:** memorizza i dati "vicini" a quelli usati
  - **temporale:** memorizza i dati che sono stati usati più volte