

DIRECTORY > 13.3

Le directory sono organizzate in una struttura ad albero.
Ogni directory contiene **logicamente** un insieme di nodi (file/dir),
i quali possono trovarsi in dischi fisici differenti.
È possibile condividere una directory tramite

COPIA DEL CONTENUTO: Richiede sync continuo. (-)

LINKING / ALIASING

Tecnica in cui si crea un collegamento ad un file/dir, per
facilità e/o condivisione.

- Può creare cicli (Int. su algo sic)
- Dangling Link: Puntatori sospesi (Elim a cascata)

MOUNTING / UNMOUNTING >

Si può vedere il file system ^{anche} come un insieme di mount point
contenuti a loro volta altri file system. Per collegarsi/
scollegarsi si usano le procedure di mount / unmount.

Il mount può essere automatizzato con l'ISO o directory.

SHARING FS >

I file system possono essere condivisi in maniera

- Locale: (Utenti stessa macchina)
Realizzabile con meccanismi di permessi (Protection)
- Globale: (Utenti su macchine diverse)
Usando protocolli di trasferimento
 - FTP
 - NTFS (Network FS Linux)
 - CFS | SAMBA : Windows

PROTECTION > 13.4

Il SO si occupa di proteggere i dati da danni fisici e usi
impropri. Per fare ciò implementa un meccanismo di
permessi per regolarne gli accessi:

- R. tipo, utenti: Users Group Others chmod rwx
R. tipi permessi: Read Write execute rwx

Il FS è stato creato per gestire la memorizzazione dei dati nel sistema. I dati salvati in mem. secondaria sono organizzati a blocchi.

LUGLI DEL FS 14.1

Il FS è implementato in vari strati

(1) I/O Control:

Driver e gestori di int per comun. fra mem. sec ↔ prim.

(2) Base FS:

Invio di istruzioni fisiche ai driver per la gestione degli elementi, cache ecc.

(3) M. Org. File:

Traduce ind. logici \leftrightarrow fisici e gestisce spazio libero

(4) Log. File System:

Gestisce i metadati del FS, salvandoli nel File Control Block.

\uparrow
su disco

Molti SO usano diversi file system al loro interno e consentono la realizzazione del file System in user space usando delle callback a livello SO (usate nei container)

COMPONENTI DISCHI 14.2.1

Ogni disco o partizione contiene delle aree speciali

- Boot Control Block: Contiene info per l'avvio dell'S.O
- Volume Control Block: Contiene info per l'accesso a volumi e dischi. (# Blocchi, # free blocks, ecc)

IN MEM PRIMARIA 14.2.2

In mem. primaria sono memorizzate le Mount Table contenenti dei mount points per montare i FS

Il Kernel mantiene in memoria ^{prim} delle strutture dati tabellari per compiere operazioni (lettura, scr.)

Ese: Req. di apertura (lettura)

1. Si restituisce un handle \rightarrow file desc



Figura 14.1 File system stratificato.

permessi per il file
data e ora di creazione, di ultimo accesso e di ultima scrittura
proprietario del file, gruppo, ACL
dimensione del file
blocchi di dati del file o puntatori a blocchi di dati del file

Figura 14.2 Tipica struttura di file (file-control-block).

↑

memoria

↓

memoria

2. Cerca nel disco il File Control Block e salva in memoria per poi restituirlo.

ES: Req. di (scrittura)

1. Aprire un file usando la tabella dei file aperti del processo o la tab globale (nel processo)

2. Persistere

IMP. DIRECTORY > 14.3

LINEAR LIST

Liste lineari di nomi con puntatori ai data block

- ricerche lineari $O(n) \rightarrow O(\log n)$ usando Heap / Btree

HASH TABLE: lista di concatenamento

ALLOCAZIONE DEL FILE > 14.4

L'allocazione dei file prevede l'allocazione dei blocchi fisici

CONTIGUOUS ALLOCATION 14.4.1

Ogni file cerca di allocare blocchi contigui. Necessita segnarsi eventuali interruzioni.

- frammentazione (esterna o interna)
- deframm. bloccante o se non lo è è **costosa**
- Dichiara di cui

EXTENDED BASED SYSTEM

Si allocano degli extend, ossia dei blocchi estendibili su più dischi per evitare la frammentazione.

LINKED ALLOCATION 14.4.2

Si usa una linked list di blocchi

- + No fram.
- Richiede spazio punt: cluster
- Accessi seq
- Nodo perso

FAT: File Allocation Table

Variante che indica ogni blocco e conserva una tabella con gli ind. iniziali di ogni elem.

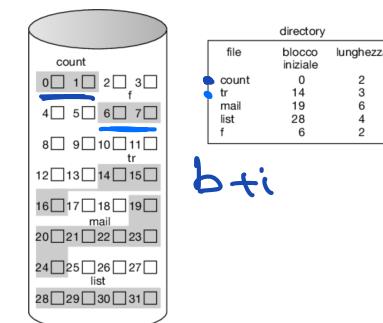
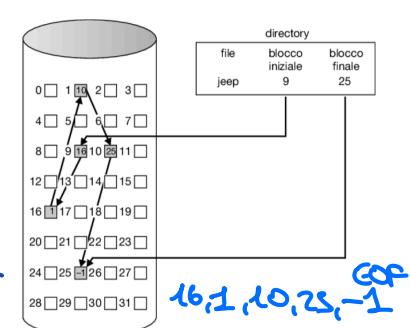


Figura 14.4 Allocazione contigua dello spazio dei dischi.



16,1,10,25,-1 GOF

Figura 14.5 Allocazione concatenata dello spazio dei dischi.

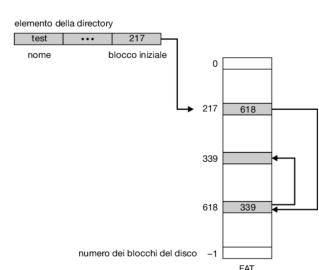


Figura 14.6 Tabella di allocazione del file.

INDEXED ALLOCATION 14.4.3

Si indicizzano i blocchi ed ogni file conserva un'array di indici dei propri blocchi

+ ind. efficiente - Index Block per file grandi

Per tenere l'index block piccolo si può:

- + Index Block concatenato
- = multi level
- = entrambi

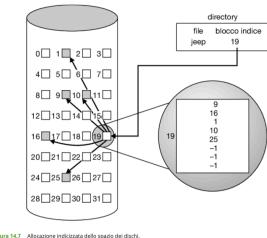


Figura 14.7 Adattamento indiretta dello spazio del disco.

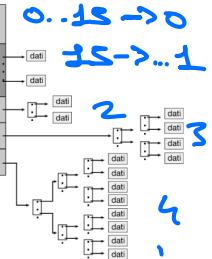


Figura 14.8 Indice di unico.

La metrica importante sono gli n. op. ls.

FREE SPACE MANAGEMENT > 14.5

L'S.O deve tenere traccia dei nodi liberi. Realizzabile con

- Bitmap: Consente di identificare nodi vicini (1 bit per blocco)
 - Pesa sullo spazio + facile avere contiguous
- Big Vector: Vettore di Bit in mem. primaria (eff.)
 - (14.5.1) + Otimizzata spazio

blocco

1=free

00111100111110001100000011100000...

(numero di bit per parola) × (numero di parole di valore 0) + offset del primo bit 1.

- Linked List:

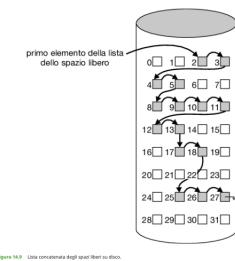


Figura 14.9 Lista concatenata degli spazi liberi su disco.

Ottimizzazioni possono riguardare:

- grouping di blocchi liberi (Linked List)
- conteggio dei blocchi liberi (contiguo)

EFFICIENZA FS > 14.6

L'ottimizzazione può agire su

- Tenere vicini dati e metadati: Per accesso fisico veloce.
- Caching: Si occupa memoria libera per bufferizzare il disco (Buffer Cache). Su richiesta alcune app. possono bypassare cache. Ottimizzabile con
 - Read Ahead: Leggo in anticipo sezioni vicine
 - free Behind: Rimozione della pagina su richiesta succ.
 - Page Cache: Su mem mapped I/O si tengono in cache per evitare accessi al disco

e (per gli I/O) syscalls.

Necessita di metodi di gestione quando piena.

RECOVERY > 14.7

Vengono salvate le transazioni (In un mini buffer perodicamente (Scavaged)) fino a quando non vengono consolidate le op. Le transazioni vengono salvate su disco e mem. ed eventualmente eliminate quando sono complete.