Il comando ***fork()*** duplica il processo chiamante creando un processo figlio uguale ma distinto
Che accade se questi include più thread?
• Vi sono 2 possibilità
– Tutti i thread del padre vengono clonati(Difficile gestire il loro accesso concorrente ai dati e alle risorse condivise con thread del padre)
– Solo un thread del padre viene clonato (Possibile sorgente di inconsistenza rispetto alle esigenze di cooperazione con le thread non clonate

---

Il ***file system*** è un servizio del SO progettato per:
- la persistenza dei dati
- possibilità di condivisione dei dati tra applicazioni distinte
- nessun limite di dimensione a priori

Esso si occupa dell'organizzazione,gestione,realizzazione e accesso di e ai file.

Sono cmq memorizzati su disco che può aver diverse partizioni (quindi piu FS).Ricordiamo che il settore 0 del disco contiene l'MBR che è inizializzato dal BIOS e che contiene la descrizione delle partizioni attive.*Dischi letti e scritti a blocchi.La struttura interna della partizione è specifica in base al FS utilizzato*

-Un ***file*** è un concetto logico realizzato tramite meccanismi di astrazione,per salvare informazioni su memoria secondaria e potendola ritrovare in seguito senza conoscerne ne la struttura logica ne fisica ne il funzionamento (all utente finale non interessa).
-Infatti all utente interessa solamente designare le proprie unità mediante nomi logici unici e distinti (l'utente vede e tratta solo nomi di file.

Le *caratteristiche* di un file sono:
-*Nome*
-*Attributi* (dimensione carattere,data creazione,data ultima mod,creatore e poss,permessi)
-*Struttura interna*(livello utente,struttura logica,struttura fisica)
-*Operazioni* ammesse

Come allocare un file su disco?

Allocazione contigua

Cerca di memorizzare i file su blocchi consecutivi.Ogni file è descritto dall indirizzo del suo primo blocco e dal numero di blocchi utilizzati.Consente sia accesso sequenziale che diretto.Un file può essere letto\scritto con un solo accesso (ideale per cd\dvd).Però ogni modifica di file comporta il rischio di frammentazione esterna (ricompattazione periodica MOLTO costosa

Allocazione a lista concatenata

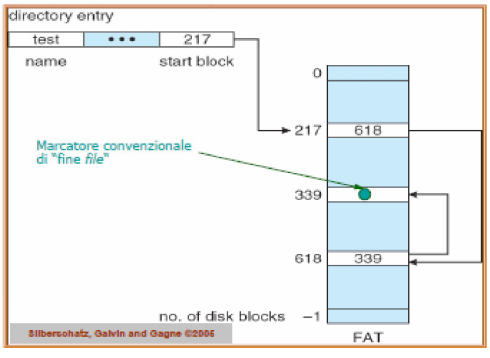
File come una lista concatenata di blocchi.Si ha un puntatore al primo blocco (certe file anche all ultimo).Ciascun blocco di file ha un puntatore al blocco successico (ciò sottrae spazio ai dati)
Un solo guasto corrompe l'intero file.Accesso sequenziale semplice ,mentre quello diretto piu oneroso.

Allocazione a lista indicizzata

I puntatori ai blocchi risiedono in strutture apposite (ciascun blocco contiene SOLO dati)
Il file è quindi descritto dall insieme dei suoi puntatori
Non causa frammentazione esterna,consente accesso sequenziale e diretto
2 Strategie di organizzazione

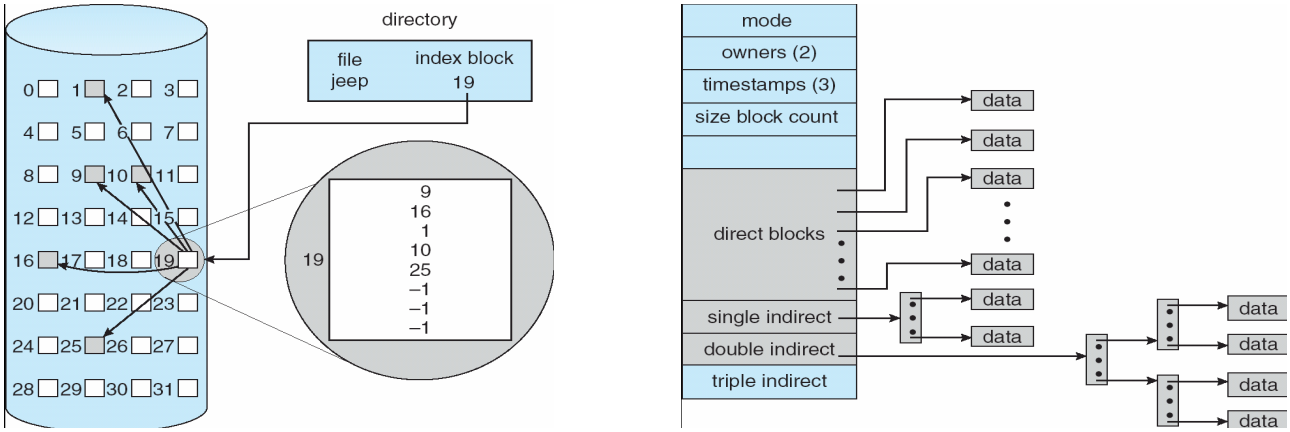
- *Tabulare (FAT,File Allocation Table)*

Tabella ordinata di puntatori
un puntatore per ogni blocco (cluster) del disco (la tabella cresce all aumentare del disco)
La porzione di FAT relativa ai file in uso deve sempre risiedere in RAM



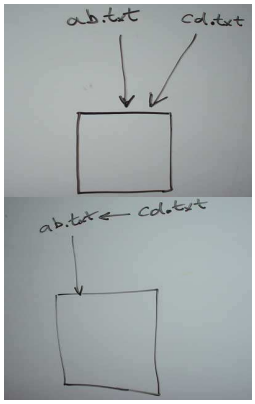
- *Indicizzata (i-node)*

Un I-node per ogni file con gli attributi del file e i puntatori e i suoi blocchi
In RAM una tabella di I-node per i soli file in uso
A seconda della dimensione del file si può sfruttare gli i-node in modi differenti (livelli di indirezione.



Hard Link: puntatore diretto al descrittore (i-node) del file originale
Più possessori di descrittori dello stesso file condiviso.
Il file condiviso non può più essere distrutto fin quando esistano suoi descrittori remoti anche se il suo proprietario avesse inteso cancellarlo.

Symbolic Link:puntatore al puntatore all'inode del file originale
Esiste così 1 solo descrittore (i-node) del file originale.
L'accesso condiviso avviene tramite cammino sul FS.



Come gestire i blocchi liberi?
- *Bitmap* dove ogni bit indica lo stato del corrispondente blocco (0 libero 1 occupato)
- *Lista concatenata di blocchi* sfruttando i campi al puntatore successivo (scelta nell architettura FAT).E' l'equivalente di un file composto dai soli blocchi liberi.

Come gestire i blocchi danneggiati?
- *Soluzione HW:*creando e mantenendo in un settore del disco un elenco di blocchi danneggiati e dei loro sostituti
- *Soluzione SW*:ricorrendo a un falso file che occupi tutti i blocchi danneggiati

---

Directory (fornisce info su nome collocazione ed attributi)

-File e directory risiedono in aree logiche distinte

- A livello singolo:tutti i file sono elencati su un unica lista lineare (nomi dei file unici).Semplice realizz,facili da trovare,ma gestione onerosa all aumentare dei file
- A due livelli:una root Directory cotiene una user file directory per ciascun utente di sistema (l'utente registrato può vedere solo la propria user file directory).I file sono localizzati tramite percorso.Efficienza di ricerca,libertà di denominazione,ma non c'è libertà di raggruppamento
- Ad Albero:numero arbitrario di livelli a partire dal livello di root.Ogni directory può contenere file o altre directory di livello inferiore.Ogni utente ha la sua directory corrente che può cambiare con comandi di sistema.Ricerca abbastanza efficiente,libertà di denominazione,libertà di raggruppamento
- A grafo come albero ma lo stesso file può appartenere a più directory.Unix e GNU/linux utilizzano link tra il nome reale del file e la sua presenza virtuale

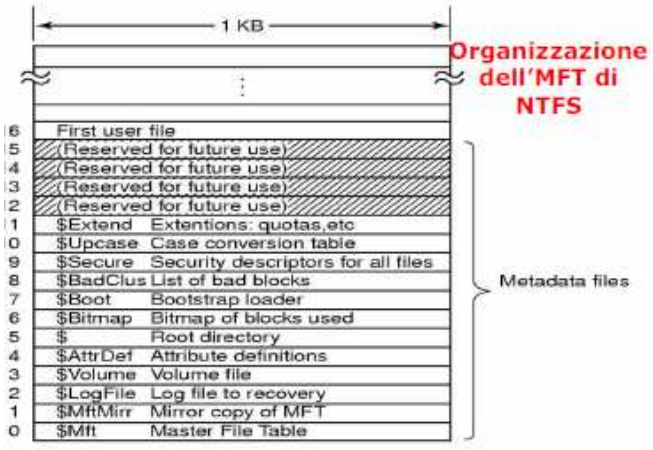
---

NTFS file system usato da Win NT XP e Vista,supporta l'intera gamma di FS Windows e anche ext2fs di GNU/Linux

- Nome di file fino a 255 caratteri in codifica Unicode (2\4B/carattere)
- Esso è una collezione di volumi logici (un volume logico può mappare su più partizioni e anche si più dischi).
- Per volume si intende una sequenza lineare di blocchi (cluster) di ampiezza fissa (tra 512B e 4KB).
- Blocco piccolo=ridotta frammentazione interna ma piu accessi a disco,blocchi grandi viceversa

La principale struttura dati del NTFS è la *MFT* (*Master File Table*)

- ce n'è una per volume
- fisicamente realizzata come un file (può essere salvata ovunque)
- logicamente strutturata come una sequenza lineare di ≤ 2 alla 48 record di ampiezza da 1 a 4KB (ciascun
- record descrive 1 file identificato da un indice ampio 48bit).Gli altri 16 bit servono per il contatore di riuso.
- SEMPRE i primi 16 record sono riservati per i *metadata* (record che descrivono l'organizzazione del volume.



Ciascun Record contiene un numero variabile di coppie del tipo <*descrittore di attributo,valore*> dove il primo campo specifica la struttura dell attributo (questo campo ha ampiezza 24B per attributi residenti,piu ampio per quelli non residenti).

- La strategia NTFS consente di rappresentare file di ampiezza virtualmente illimitata.
- *Il numero di record necessari per i dati di un singolo file dipende dalla sua contiguità piuttosto che dalla sua ampiezza.*

Se rappresentare un file costa piu di un record,NTFS usa una tecnica a continuazioni analoga a quella usata dagli I-node di UNIX e GNU/Linux.

Infatti il record base di MFT contiene un puntatore a N>=1 record secondari in MFT che descrivono la sequenza di blocchi del file.Se non vi fosse abbastanza spazio in MFT per i record secondari di un dato file la loro intera lista sarebbe posta in un altro file denotato da un record posto in MFT.