**File system**

Per memorizzare grandi quantità di informazioni, rendere la memoria permanente e dare la possibilità a più processi di accedere alle stesse informazioni è necessario registrare tali informazioni in dischi o altri supporti in unità dette file. Il sistema operativo consente la loro gestione mediante il file system. Oltre il contenuto dei file, vengono gestite informazioni complementari come la struttura, il nome, il tipo di accesso, la protezione ecc. i file consentono di ritrovare le informazioni che contengono mediante il proprio nome sul supporto in cui risiedono. Il nome del file si suddivide generalmente in nome.estensione entrambi con un numero variabile di caratteri e varia utilizzano del maiuscolo/minuscolo.

I file contengono le informazioni in te tipi di strutture:

1. **Sequenza non strutturata di byte**: consente la massima flessibilità al programma utente.
2. **Sequenza di record a lunghezza fissa**: ha perso popolarità con il tempo
3. **Albero di record con campo chiave**: è ancora in uso sui grossi mainframe usato per l’elaborazione di dati commerciali.

Vi sono quattro tipi diversi di file:

1. **File regolari**: contengono le informazioni dell’utente, un programma eseguibile ecc.
2. **Directory**: conservano la struttura del file system ed informazioni sui file regolari;
3. **File** **speciali** **a** **caratteri**: sono usati per modellare unità di input/output seriali come video, stampanti, reti, ecc.
4. **File** **speciali** **a** **blocchi**: sono usati per modellare unità di input/output a blocchi come i dischi

Le modalità di lettura delle informazioni in un file sono di due tipi:

* **Accesso sequenziale**
* **Accesso casuale**

Alcuni sistemi operativi consentono di mappare un file in memoria, ossia di caricarlo tutto o in parte, assegnandoli degli indirizzi virtuali. Questo permette ai processi di accedere in maniera trasparente ai file che possono essere quindi contenuti in memoria centrale per intero o per blocchi. Questa tecnica velocizza le operazioni compiute sul file, soprattutto quando si riferiscono a parti limitate e non all’intero file. Quando il processo che sta usando il file termina, quest’ultimo viene riscritto sul disco.

La virtualizzazione dell’accesso comporta alcuni problemi che vanno attentamente gestiti:

* In caso di apertura contemporanea del file da parte di più processi, alcune informazioni possono risultare inconsistenti
* In caso di caduta del sistema, alcune informazioni posso essere perse
* Sono necessarie tecniche per la rimozione delle pagine o dei blocchi dei file

Per tener traccia dei file, il file system fornisce un indice comunemente chiamato directory. Tipicamente un elemento della directory contiene:

* Il nome di un file ed eventualmente include anche gli attributi del file
* In alternativa il nome di un file in cui si trovano gli attributi e gli indirizzi del disco

Ogni directory può contenere:

* **File regolari**
* **Altre directory**

In questo modo a partire da una directory radice è possibile generare un albero di directory e subdirectory.

Quando il file system è organizzato come un albero di directory, è necessario un qualche modo per specificare i nomi dei file. Questo può essere fatto in due modi diversi:

1. **Path name assoluto**;
2. **Path name relativo**.

Il primo consiste nel cammino dalla directory radice al file. I componenti del cammino sono separati da opportuni simboli di separazione come “/”. Il secondo caso si usa congiuntamente al concetto di **directory lavoro**. Un utente può definire una directory come directory di lavoro corrente. In questo caso tutti i path name che non iniziano con la directory radice sono considerati relativi alla directory di lavoro.

Un fattore chiave nell’implementazione della memorizzazione dei file è tener traccia di quali blocchi del disco associare a ciascun file. Le modalità di allocazione dei blocchi e del relativo reperimento sono:

* **Allocazione contigua**
* **Allocazione a lista concatenata**
* **Allocazione a lista concatenata con indice**
* **Allocazione mediante uso di tabelle i-node**

**L’allocazione contigua memorizza** i file in blocchi di disco consecutivi. È semplice da implementare poiché tiene traccia del solo indirizzo di inizio del file memorizzando un solo numero, è molto efficiente perché permette di leggere l’intero file con una sola operazione.

Non è realizzabile senza che si conosca la dimensione massine del file, il disco risulta notevolmente frammentato e molto spazio viene sprecato.

**Nell’allocazione a lista concatenata** ogni blocco associato al file contiene i dati veri e propri e il numero del blocco successivo assegnato al file. Non implica spreco di spazio perché i blocchi vengono allocati dinamicamente e nell’elemento della directory viene memorizzato solo l’indirizzo del primo blocco. La lettura del file è semplice ma l’accesso diretto ai blocchi è estremamente lento, la quantità di dati nel blocco non corrisponde più ad una potenza di due.

**Nell’allocazione a lista concatenata** con indice si assegna ai file blocchi di disco non necessariamente contigui e mantiene l’elenco dei blocchi assegnati mediante una tabella in memoria. L’intero blocco è disponibile per i dati, l’accesso diretto ad un blocco è molto più rapido e l’elemento della directory contiene solo il numero del primo blocco. Lo svantaggio principale consiste nel fatto che ad un grande numero di blocchi nel disco corrisponde una tabella grande che occupa molto spazio in memoria centrale.

**Nell’allocazione mediante uso di tabelle i-node** si memorizzano gli attributi dei file, gli indirizzi dei primi dieci blocchi assegnati al file e tre campi che consentono l’indirizzamento dei blocchi successivi mediante un blocco a singola indirezione o un blocco a doppia ed a tripla indirezione. Le tabelle i-node sono in posizione definita all’interno del file system. Alcuni vantaggi sono che l’elemento della directory contiene solo il nome del file ed il numero di i-node, si consente di mantenere in memoria gli i-node dei file aperti sprecando poco spazio, consente di indirizzare un numero enorme di blocchi di disco, per i file di piccole dimensioni consente di reperire rapidamente l’indirizzo dei primi blocchi di disco.

Il file system, mediante il pathname fornito dall’utente localizza l’elemento della directory ed estrae le informazioni necessarie per trovare i blocchi di disco assegnati al file.

I file sono generalmente memorizzati su disco, pertanto la gestione dello spazio su disco è uno dei problemi fondamentali nell’implementazione del file system. I file vengono memorizzati in blocchi non necessariamente adiacenti, questo perché memorizzando un file come una sequenza contigua di byte si presenta l’ovvio problema che se il file aumenta di dimensione molto probabilmente dovrà essere spostato nel disco. Lo stesso vale per i segmenti in memoria, con la differenza che muovere un segmento di memoria è un’operazione molto più veloce rispetto allo spostare un file da una posizione del disco a un’altra.

Il primo problema da risolvere è la scelta della dimensione dei blocchi. La scelta di un blocco molto grande rende più veloce il tempo di lettura ma provoca maggior spreco di spazio. Mentre la scelta di un blocco molto piccolo causa numerosi accessi ai blocchi per leggere i dati ma consente una gestione più efficiente dello spazio. L’efficienza nel tempo e nello spazio sono intrinsecamente in conflitto nella scelta della dimensione delle unità di allocazione. Confrontando la velocità di trasferimento dei dati e l’efficienza dello spazio su disco si può vedere che una buona utilizzazione dello spazio corrisponde a una bassa percentuale di dati e viceversa.

Un altro problema successivo alla scelta della dimensione dei blocchi è quello della gestione dei blocchi liberi. I metodi che vengono usati frequentemente sono:

* **Lista concatenata di blocchi** che contengono gli indirizzi dei blocchi liberi nel disco
* **Mappa di bit:** un disco di n blocchi richiede una mappa di n bit. I blocchi liberi si rappresentano con un 1 sulla mappa, i blocchi allocati invece con uno 0.

Il metodo della lista concatenata è più efficiente se il disco è quasi pieno, poiché il numero di blocchi è molto limitato. Inoltre in queste condizioni, la mappa di bit potrebbe risultare spesso pena e richiedere continui accessi ad altre parti di mappa per assegnare blocchi liberi. Se c’è abbastanza memoria principale per contenere la mappa di bit, è generalmente preferibile quest’ultimo metodo.

Il file system può contenere dati che non possono essere persi poiché di difficile o costoso recupero. I dispositivi di memorizzazione presentano spesso blocchi inutilizzabili per via di difetti di fabbricazione, usura, urti violenti e per tanto non devono essere assegnati a nessun file. Vi sono due soluzioni a questo problema, un hardware e una software. Per quanto riguarda la soluzione hardware viene riservata una parte di disco per tener traccia dei blocchi rovinati e contiene dei blocchi di riserva, in fase di inizializzazione il controllore del dispositivo legge questi blocchi e li sostituisce con blocchi di riserva, memorizzandone l’avvenuta sostituzione. Questa operazione è del tutto trasparente al file system. Per quanto riguarda la soluzione software, l’utente o il file system tiene traccia dei blocchi rovinati assegnandoli tutti ad un file speciale da non utilizzare e nelle operazioni di backup si deve evitare la sua lettura.

Il metodo per ridurre le perdite di dati in caso di crash del file system o del disco è effettuare le copie di backup. Nella pianificazione delle operazioni di backup vanno valutati principalmente:

* La frequenza con cui effettuare le copie di backup
* I file che presentato la maggior necessità di essere copiati
* La tecnica migliore per effettuare il backup

Per quanto riguarda la **frequenza**, la scelta va pianificata in base alla quantità di lavoro svolta nel tempo e al costo di ripristino. La scelta dei file va effettuata in base al **contenuto** dei file stessi:

* I file critici potrebbero essere raggruppati in directory dedicate delle quali si effettua una copia con maggiore frequenza
* La duplicazione di alcuni file critici può essere opportuna se il quantitativo di memoria sprecato è giustificabile.

**La tecnica di backup** va pianificata in base al tempo disponibile e al costo sopportabile per i supporti. Le tecniche di backup possono essere:

* Backup su nastri dell’intero file system
* Backup incrementale (copia solo dei file nuovi o modificati dall’ultimo backup)
* Duplicazione incrociata dei dati su due dispositivi: metà di ogni disco contiene i propri dati e l’altra metà contiene i dati dell’altro disco

Nel corso della vita del file system può succedere che la sua consistenza venga a mancare a causa:

* Crash
* Programmi che non rispettano le regole di allocazione

La maggior parte dei sistemi operativi prevede un programma di servizio per la verifica della consistenza del file system e la correzione degli errori. Ad esempio vi sono controlli di consistenza che possono essere divisi in due tipi: sui blocchi e sui file. Nell’effettuare il controllo di consistenza si usa una tabella con due contatori per ogni blocco: il primo registra quante volte il blocco è presente in un file, il secondo quante volte il blocco è presente nella lista libera. Il test consiste nella scansione degli i-node, per individuare blocchi assegnati e la scansione della lista dei blocchi liberi. Al termine del check si possono verificare i seguenti casi:

* **Caso normale**: un blocco è presente una sola volta nella lista xor in un file
* **Blocco mancante**: un blocco non è presente nella lista libera e non è assegnato a nessun file, viene assegnato alla lista libera
* **Blocco duplicato nella lista libera**: il doppio riferimento viene cancellato
* **Blocco assegnato a due file:** è il caso più grave perché è sintomo di possibili dati alterati. Il contenuto del blocco è copiato in un altro blocco ed ognuno dei blocchi assegnato ad un file
* **Blocco presente** **nella lista libera ed assegnato ad un file**: viene eliminato dalla lista libera