**STRUTTURA DEL FILE SYSTEM**

Per fornire un efficiente e conveniente accesso al disco, il SO fa uso di uno o più file system che consentono di memorizzare, individuare e recuperare facilmente i dati.

Il file system è composto da più livelli creando una struttura stratificata. Ogni livello si serve delle funzioni del livello inferiore per poterne creare di nuove da impiegare nei livelli superiori.

Il livello più basso, il controllo dell'I/O, è costituito dai **driver dei dispositivi** e dai gestori dei segnali d'interruzione, si occupa del trasferimento delle informazioni tra memoria centrale e memoria secondaria.

Nel livello successivo incontriamo il **file system di base** che deve soltanto inviare dei generici comandi all'appropriato driver di dispositivo per leggere e scrivere blocchi fisici nel disco. Ogni blocco fisico si identifica col suo indirizzo numerico nel disco.

Nel livello successivo troviamo il **modulo di organizzazione dei file** che è a conoscenza dei file e dei loro blocchi logici, così come dei blocchi fisici dei dischi. Conoscendo il tipo di allocazione dei file usato e la locazione dei file, può tradure gli indirizzi dei blocchi logici, che il file system di base deve trasferire, negli indirizzi dei blocchi fisici. I blocchi logici di ciascun file sono numerati da 0 a n. Il modulo di organizzazione dei file comprende anche il gestore dello spazio libero, che registra i blocchi non assegnati e li mette a disposizione del modulo di organizzazione dei file quando sono richiesti.

Infine, nell'ultimo livello, c'è il **file system logico** che gestisce i metadati si tratta di tutte le strutture del file system, eccetto gli dati.

Il file system logico gestisce la struttura della directory per fornire al modulo di organizzazione dei file le informazioni di cui necessita. Mantiene le strutture de file tramite i **blocchi di controllo dei file** (FCB), contenenti informazioni sui file.

**METODI DI ALLOCAZIONE**

Esistono tre metodi di allocazione dello spazio in un disco: contigua, concatenata, indicizzata

**ALLOCAZIONE CONTIGUA**

Per usare l'allocazione contigua ogni file deve occupare un insieme di blocchi contigui nel disco. L'allocazione contigua dello spazio per un file è definita dall'indirizzo del primo blocco e dalla lunghezza, espressa in blocchi.

Nella directory va salvato l’indirizzo di del blocco di partenza e della lunghezza espressa in blocchi.

Accedere a un file il cui spazio è assegnato in modo contiguo è facile.

Quando si usa un accesso sequenziale, il file system memorizza l'indirizzo dell'ultimo blocco cui è stato fatto riferimento e se è necessario legge il blocco successivo. Nel caso di un accesso diretto al blocco 'i' di un file che comincia al blocco 'b' si può accedere immediatamente al blocco b+i. Quindi sia l'accesso sequenziale che quello diretto si possono gestire con l'allocazione contigua.

L'allocazione contigua presenta però alcuni problemi.

Uno riguarda l'individuazione dello spazio per un nuovo file. Il problema dell'allocazione contigua dello spazio dei dischi si può considerare un'applicazione particolare del problema dell'allocazione dinamica della memoria.

I criteri per scegliere un buco liberi più utilizzati sono il first-fit e il best-fit.

Questi algoritmi soffrono della frammentazione esterna

Il problema viene risolto con la tecnica della compattazione.

Una strategia per evitare di perdere grandi quanti di spazio a causa della frammentazione, consiste nel salvare il filesystem su di un altro disco, liberare il primo disco così da avere una grande quantità di spazio libero contiguo e infine ricopiare il filesystem nel primo disco.

Questo tipo di funzione può essere realizzata offline, ossia con il filesystem non montato durante questo periodo morto il normale funzionamento del sistema non è possibile. Questo tipo di soluzione è da evitare nei sistemi in attività.

Molti sistemi moderni sono in grado di eseguire la deframmentazione online, ossia durante il loro normale funzionamento, a prezzo però di una sostanziale perdita di prestazioni.

Un altro problema dell'allocazione contigua è la determinazione della quantità di spazio necessaria per un file.

Quando si crea un file, occorre trovare e allocare lo spazio di cui si necessita.

Esiste il problema di conoscere la dimensione del file da creare.

In alcuni casi questa dimensione si può stabilire in modo abbastanza semplice, ad esempio quando si copia un file esistente.

Non è facile stimare la dimensione di file che deve contenere dei dati emessi da un programma.

Se un file riceve poco spazio può essere impossibile estenderlo, soprattutto se si utilizza un criterio best-fit, lo spazio oltre le estremità del file può già essere in uso e quindi non è possibile ampliare il file in modo contiguo.

Esistono due possibilità:

* Il programma utente termina con un messaggio d'errore; l'utente deve allora allocare più spazio ed eseguire di nuovo il programma
* Trovare un buco più grande, copiare il contenuto del file nel nuovo spazio e rilasciare lo spazio precedente

Anche se si conosce in anticipo la dimensione di un file, l’allocazione preventiva può essere in ogni modo inefficiente, poiché se il file cresce lentamente nel tempo e noi abbiamo allocato uno spazio tale da contenere la sua dimensione, fin quando il file non sarà grande in modo da tale da occupare tutto lo spazio messo a suo disposizione avremo la frammentazione interna.

Molti sistemi adottano una allocazione concatenata modificata, ad ogni file si assegna una porzione di spazio e se questa non è sufficiente se ne aggiunge un'altra.

**ALLOCAZIONE CONCATENATA**

L'allocazione concatenata risolve tutti i problemi dell'allocazione contigua.

Ogni file è composto da una lista concatenata di blocchi del disco i quali possono essere sparsi in qualsiasi punto del disco stesso.

La directory contiene un puntatore al primo e all'ultimo blocco del file.

Ogni blocco contiene un puntatore al blocco successivo.

Per creare un nuovo file si crea un nuovo elemento nella directory.

Con l'allocazione concatenata ogni elemento della directory ha un puntatore al primo blocco del file.

Questo puntatore s'inizializza a **nil** (valore del puntatore a fine lista) a indicare un file vuoto, impostando anche la dimensione a 0.

Un'operazione di scrittura nel file determina la ricerca di un blocco libero attraverso il sistema di gestione dello spazio libero, la scrittura di tale blocco e la concatenazione di tale blocco alla fine del file.

Per leggere un file occorre semplicemente leggere i blocchi seguendo i puntatori da un blocco al altro.

Con l'allocazione concatenata non esiste frammentazione esterna e per soddisfare una richiesta si può utilizzare un qualsiasi blocco libero della lista.

Inoltre non è necessario dichiarare la dimensione di un file al momento della sua creazione, un file può continuare a crescere finché sono disponibili blocchi liberi, di conseguenza non serve utilizzare la compattazione.

L'allocazione concatenata presenta alcuni svantaggi:

* Può essere utilizzata in modo efficiente solo per i file ad accesso sequenziale
* I puntatori hanno bisogno di spazio, il quale potrebbe essere utilizzato per salvare informazioni. Ogni file deve memorizzare un po’ di spazio in più

La soluzione è di riunire un certo numero di blocchi contigui in **cluster** (gruppi di blocchi) e allocare cluster al posto di blocchi. In questo modo i puntatori usano una quantità di spazio di disco che si riduce in modo proporzionale al n di cluster. Questo metodo migliora la produttività del disco ma ha lo svantaggio di dar vita alla frammentazione interna.

Un altro problema dell'allocazione concatenata riguarda l’affidabilità: i file sono mantenuti insieme dai puntatori, nel caso in cui si perdesse o danneggiasse un puntatore potrebbe compromettere il collegamento alla lista dei blocchi liberi oppure a un altro file. La soluzione potrebbe essere quella di utilizzare delle liste doppiamente concatenate; oppure memorizzare il nome del file e il relativo numero di blocco in ogni blocco.

Una variante importante del metodo di allocazione concatenata consiste nell'uso della **tabella di allocazione dei file (FAT).**

Per contenere tale tabella si riserva una sezione del disco all'inizio di ciascun volume; la FAT ha un elemento per ogni blocco del disco ed è indicizzata dal numero dal numero di blocco; si usa essenzialmente come una lista concatenata.

L'elemento di directory contiene il numero del primo blocco del file.

L'elemento della tabella indicizzato da quel numero di blocco contiene a sua volta il numero del blocco successivo del file. Questa catena continua fino all'ultimo blocco, che ha come elemento della tabella un valore speciale di fine del file.

I blocchi inutilizzati sono indicati nella tabella da un valore 0.

L'allocazione di un nuovo blocco a un file implica semplicemente la localizzazione del primo elemento della tabella con valore 0 e la sostituzione del valore di fine del file precedente con l'indirizzo del nuovo blocco; lo 0 è quindi sostituito con il valore di fine del file.

Un vantaggio è dato dall'ottimizzazione del tempo d'accesso diretto, poiché la testina del disco può trovare la locazione di ogni blocco leggendo le informazioni contenute nella FAT.

**ALLOCAZIONE INDICIZZATA**

L'allocazione concatenata non è in grado di sostenere un efficiente accesso diretto, poiché i puntatori ai blocchi sono sparsi, per tutto il disco e si devono recuperare in ordine.

L'**allocazione indicizzata** risolve questo problema, raggruppando tutti i puntatori in una sola locazione: **blocco indice**.

Ogni file ha il proprio blocco indice: si tratta di un array di indirizzi di blocchi del disco. L'i-esimo elemento del blocco indice punta all'i-esimo blocco del file.

La directory contiene l'indirizzo del blocco indice.

Per individuare e leggere l'i-esimo blocco occorre usare il puntatore che si trova nell'i-esimo elemento del blocco indice, per poi localizzare e leggere il blocco desiderato.

Una volta creato il file, tutti i puntatori del blocco indice sono impostati a nil.

Quando si scrive l'i-esimo blocco per la prima volta, il gestore dei blocchi liberi fornisce un blocco; l'indirizzo di questo blocco viene inserito nell'i-esimo elemento del blocco indice.

Poiché ogni blocco libero del disco può soddisfare una richiesta di maggiore spazio, l'allocazione indicizzata consente l'accesso diretto senza soffrire di frammentazione esterna.

Siccome ogni file deve avare il proprio blocco indice quest’ultimo deve essere il più piccolo possibile, ma se il blocco indice è troppo piccolo non può contenere un numero di puntatori sufficiente per un file di grande dimensione.

Ci sono vari meccanismi che permettono di gestire questa situazione:

* **Schema concatenato**: Per permettere la presenza di lunghi file è possibile collegare tra loro parecchi blocchi indice
* **Indice a più livelli**: Consiste nell'impiego di un blocco indice di primo livello che punta a un insieme di blocchi indice di secondo livello che a loro volta puntano ai blocchi dei file.
* **Schema combinato**: Un'altra possibilità consiste nel tenere i primi 15 puntatori del blocco indice nell'**inode** del file.

I primi 12 di questi puntano a **blocchi diretti** (cioè contengono direttamente gli indirizzi dei blocchi che contengono i dati).

Gli altri 3 puntatori puntato a **blocchi indiretti.** Il primo puntatore di blocco indiretto è l'indirizzo di un **blocco indiretto singolo**; si tratta di un blocco indice che non contiene dati, ma indirizzi di blocchi che contengono dati. Quindi c'è un puntatore di **blocco indiretto doppio** contenente l'indirizzo di un blocco che a sua volta contiene gli indirizzi di blocchi contenenti puntatori agli effettivi blocchi di dati. L'ultimo puntatore contiene l'indirizzo di un **blocco indiretto triplo.**

Gli schemi d'allocazione indicizzata soffrono di alcuni dei problemi di prestazione dell'allocazione concatenata. I blocchi indice si possono caricare in memoria, ma i blocchi dei dati possono essere sparsi per un intero volume.