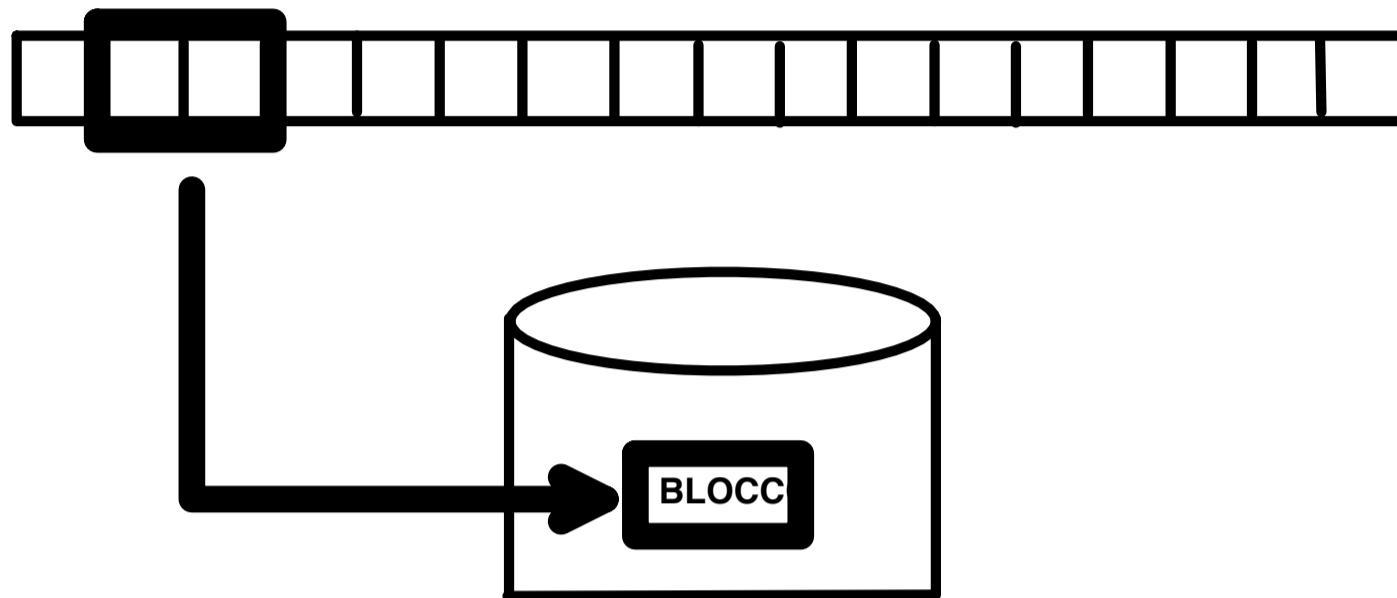


File vs blocchi di dispositivo (i)

- Ciascun file è allocato sul dispositivo di memoria di massa come un insieme di blocchi (chiamati anche blocchi logici) non necessariamente contigui

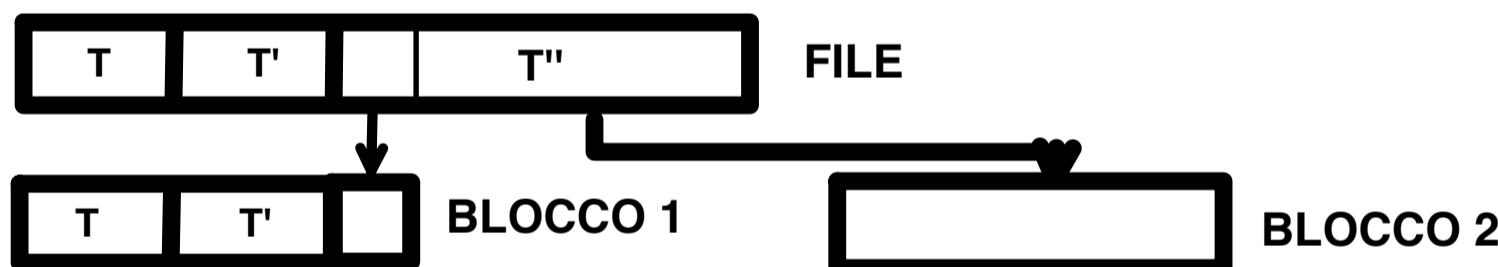
Supponiamo di avere un contenuto di un file costituito da record: come vanno a finire questi record all'interno di blocchi di una memoria di massa?

Un blocco può ospitare una quantità superiore all'unità dei record.



1. Organizzazione fissa: abbiamo dei file i cui record sono di taglia fissa - stiamo escludendo di fatto i file a mucchio - e questi record che appartengono al file vengono presi e, per quanti ce ne entrano all'interno di un blocco, vengono presi e mantenuti all'interno del blocco. Ma può essere che il blocco non è grande esattamente un multiplo di T , dove T è la dimensione di un singolo record, quindi per far entrare i blocchi viene chiamata in causa la "FRAMMENTAZIONE INTERNA", internamente ad un blocco abbiamo un frammento inutilizzabile.

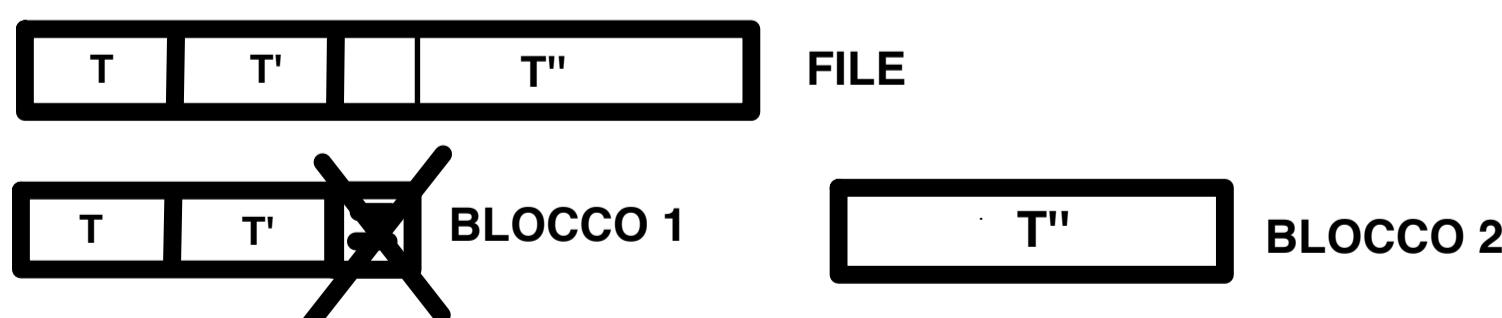
2. Organizzazione Variabile con Riporto: i record dei file che andiamo a collocare all'interno dei blocchi dei dispositivi, possono essere di taglia variabile ma abbiamo la possibilità di prendere uno di questi record e di registrarlo tra blocchi differenti.



Una parte rimanente del blocco accoglie una piccola parte del record T''.

Il resto del record T'' va tutto in un altro blocco del dispositivo

3. Organizzazione variabile senza riporto: è come il numero 2, però non abbiamo il riporto, il che implica dire che il riporto di T'' (una parte) all'interno del primo blocco NON è possibile, quindi se una parte del blocco non è sufficiente ad ospitare una parte del record, non lo riportiamo.



L'ultima parte del blocco 1 in questo caso rimane inutilizzata.

Per poter utilizzare questi blocchi nel dispositivo chiaramente dobbiamo conoscerne il loro stato.

Noi dobbiamo sapere se, per esempio, un blocco è correntemente NON utilizzato per mantenere record di un file (Utilizzabile) oppure invece è già utilizzato.

Questo ovviamente ci serve nel momento in cui andiamo a creare dei nuovi file all'interno del nostro file-System o andiamo ad "Ampliare" il contenuto di file già

esistenti: abbiamo bisogno di altri blocchi associati ad un dispositivo di memoria di massa per mantenere i nuovi record del nostro file.

Come facciamo a tenere traccia se i blocchi del dispositivo sono "Liberi" oppure "Occupati"?

- Il file system tiene conto degli spazi liberi sul dispositivo di memoria di massa tramite apposite strutture:

A) Lista Libera: tiene traccia di unità di allocazione libere

B) Bit Map: dedica un bit ad ogni unità di allocazione, per indicare se essa è libera o meno

Con la **bitMap** noi abbiamo effettivamente una mappa di bit con un bit associato a ciascun blocco del dispositivo, se questo bit è pari a 1 quel blocco è **NON libero** (pieno), ovviamente se il bit è pari a 0 abbiamo che il blocco è **libero**.

Track	Sector											
	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
2	1	0	1	0	0	0	1	0	0	0	0	0
3	0	0	0	1	1	1	1	1	1	0	0	0
4	1	1	1	0	0	0	0	0	0	0	0	1

(b)

Questa è una bitmap su un dispositivo che è un hard disk, in cui i blocchi sono organizzati in base alle tracce ed in base ai settori. Le tracce ci vanno ad indicare in che circolo del nostro disco a rotazione ci stiamo muovendo e il settore in che punto di quel circolo ci stiamo muovendo. Traccia 0, Settore 0, ha un blocco libero: può essere ovviamente prelevato per andare a scriverci dei dati associati ad uno specifico file.

Sulla quinta posizione della traccia zero della hard drive abbiamo un 1, ed è già utilizzato.

I dati che sono presenti all'interno di un HARD-DRIVE, ossia i blocchi che sono ancora liberi, possono essere riportati all'interno di una lista libera, all'interno in cui noi elenchiamo "quali sono le zone libere all'interno del dispositivo", ossia quelle che possono essere utilizzate per allocarci dei nuovi file.

A partire dallo zero settore sulla traccia zero, abbiamo 5 elementi contigui che sono liberi.

Track	Sector	Number of sectors in hole
0	0	5
0	6	6
1	0	10
1	11	1
2	1	1
2	3	3
2	7	5
3	0	3
3	9	3
4	3	8

- RS tiene traccia di quanti e quali blocchi sono allocati per un dato file

I metodi di accesso rappresentano la forma con cui le applicazioni possono operare sul contenuto dei files, quindi sui record.

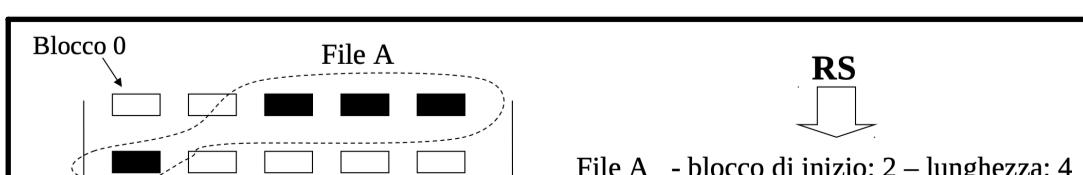
Mentre invece le tecniche di allocazione sono tecniche che, dato un RS che mi va ad indicare dove i dati sono presenti all'interno di un dispositivo di memoria di massa, secondo la regola che viene utilizzata dal file system.

Metodi di accesso e tecniche di allocazione, sono due cose differenti.

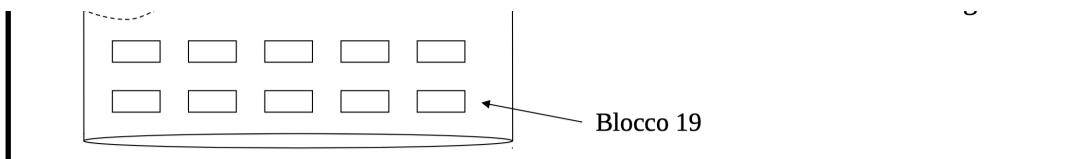
Allocazione di file contigua

L'allocazione è contigua quando noi abbiamo un file system che alloca l'insieme dei blocchi di un dispositivo, usati per mantenere i record di un certo file, secondo una regola contigua.

- un insieme di blocchi contigui è allocato per un file all'atto della creazione



File A - blocco di inizio: 2 – lunghezza: 4



- la taglia massima dipende dal numero di blocchi allocati

L'occupazione reale dei file all'interno del dispositivo è pari al numero di blocchi allocati. Anche se essi non contengono record del file.

- RS mantiene informazioni sul primo blocco e sul numero di blocchi

Ora supponiamo che, a partire da un certo blocco (7), tutti i blocchi successivi sono riservati per un altro file. Ora supponiamo di voler creare un file esattamente riservando 3 blocchi: lo zero, il primo e il sesto. Ma non sono contigui. Quindi questo dispositivo di memoria di massa è frammentato secondo una regola di frammentazione esterna: le zone che noi stiamo lasciando libere all'interno di questo dispositivo, ossia quei 3 blocchi, sono troppo sparse per poter essere utilizzate correttamente per allocare un file che ha bisogno di tre blocchi all'interno del dispositivo.

- ricompattazione per affrontare il problema della frammentazione esterna

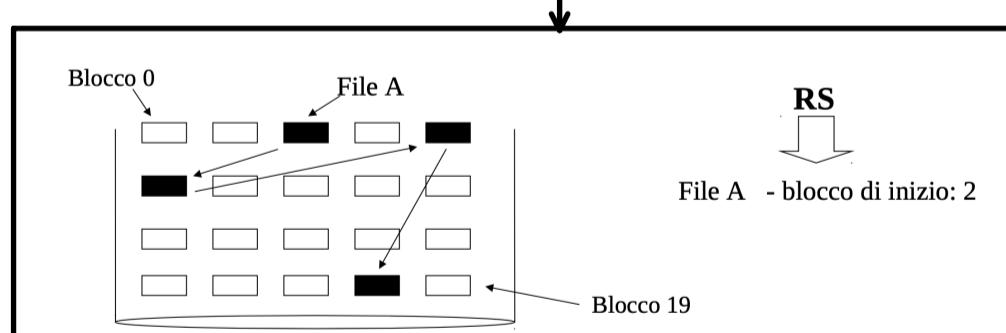
Ma in tutto ciò, il RECORD DI SISTEMA (RS), dove viene mantenuto?
Da qualche parte, su un dispositivo di memoria di massa deve andare a finire.
Perché se si effettua lo shutdown dell'intero sistema e dopo si ri-effettua lo start up, allora vorremmo che l'informazione non si perdesse.

Allocazione di file a catena

Quando rappresentiamo un file all'interno di un file system, oppure, quando rappresentiamo i record del file all'interno di un dispositivo di memoria di massa, tramite il RECORD DI SISTEMA riusciamo a riconoscere qual è il primo dei blocchi del dispositivo (blocco iniziale) che mantiene i dati del FILE.

Ma poi per sapere quali altri blocchi del dispositivo utilizziamo per mantenere i dati del file, dobbiamo leggere informazioni interne a questi blocchi per sapere dove il file continua. Noi tramite il RS sappiamo solo dove inizia il file, ma non sappiamo dove continua, e questa informazione è scritta all'interno dei blocchi. All'interno dei blocchi abbiamo l'indice del successivo elemento: LISTA.

- i blocchi di un file sono collegati come in una lista
- l'occupazione reale e' sempre pari al numero di blocchi realmente nella lista
- RS mantiene informazioni sul primo blocco
- accesso potenzialmente costoso
- ricompattazione utile per diminuire il costo di accesso

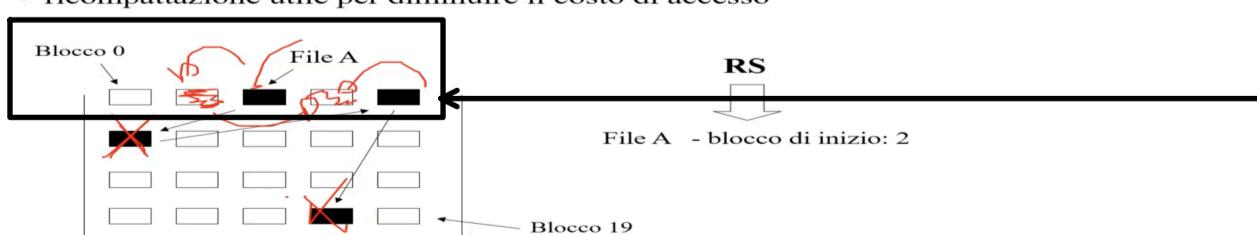


Un blocco è formato dalla zona dati e una zona a puntatore a successivo, quindi internamente non stiamo utilizzando tutto il blocco per i dati del file, nell'allocazione indicizzata non è così. Usiamo tutto il blocco per i dati del file.

Se vogliamo espandere il file A, basta andare in un blocco vuoto, scrivere l'informazione, andare nell'ultimo blocco precedente pieno, ed effettuare il collegamento.

Abbiamo una flessibilità molto superiore, non abbiamo una frammentazione esterna perché qualsiasi blocco libero può essere usato per collegarlo al file.

- ricompattazione utile per diminuire il costo di accesso

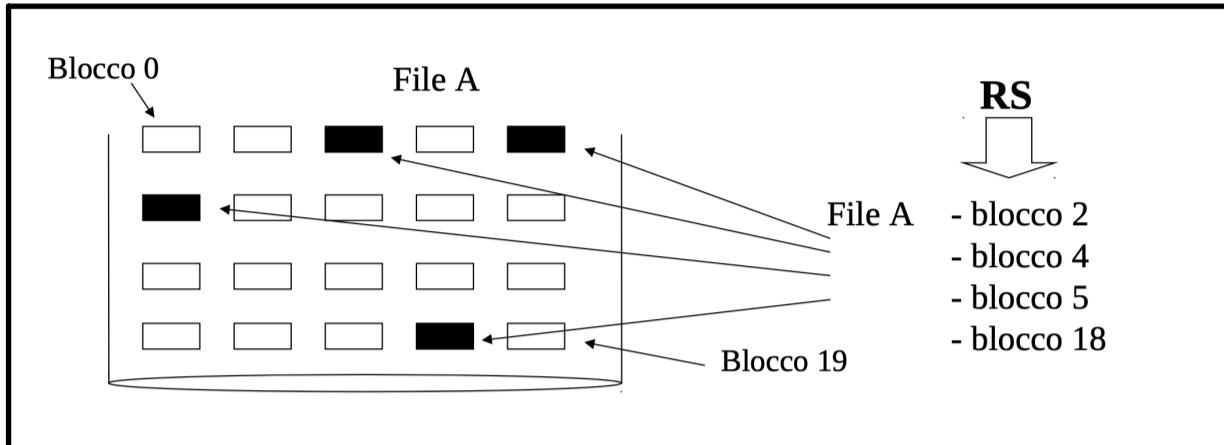


QUESTO È FONDAMENTALE: SE ABBIAMO UNA RICOMPATTAZIONE ALL'INTERNO DELLA STESSA ZONA DI UN DISPOSITIVO DI MEMORIA DI MASSA, PER POTER ACCEDERE A QUELLE INFORMAZIONI NON DOBBIAMO SPOSTARE LA TESTINA. MANTENIAMO LA TESTINA FERMA SU QUESTA TRACCIA QUI.

Allocazione di file indicizzata

È quella che i sistemi operativi adottano.

I dati di un file possono essere collegati ovunque, in qualsiasi blocco. Non abbiamo il vincolo che i dati devono essere contigui. Per sapere dove sono questi dati manteniamo un indice all'interno del RECORD DI SISTEMA: l'indice di tutti i blocchi che utilizziamo.



All'interno del record di sistema andiamo ad indicare questo indice che utilizziamo il blocco 2, il blocco 4, il blocco 5 e il 18. Tramite questa informazione noi sappiamo esattamente dove sono i dati del file all'interno del dispositivo di memoria di massa, il che implica dire che se volessimo supportare un metodo di accesso DIRETTO, lo potremmo fare. Se un'applicazione mi richiede di riposizionarsi su un certo record il cui indice mi va ad indicare che quel record è presente nel blocco 18, io so esattamente dove è questo blocco tramite il record di sistema.

Mentre nel metodo precedente, per sapere dove era posizionato l'ultimo blocco dovevo leggere prima tutti i precedenti. $O(n)$.

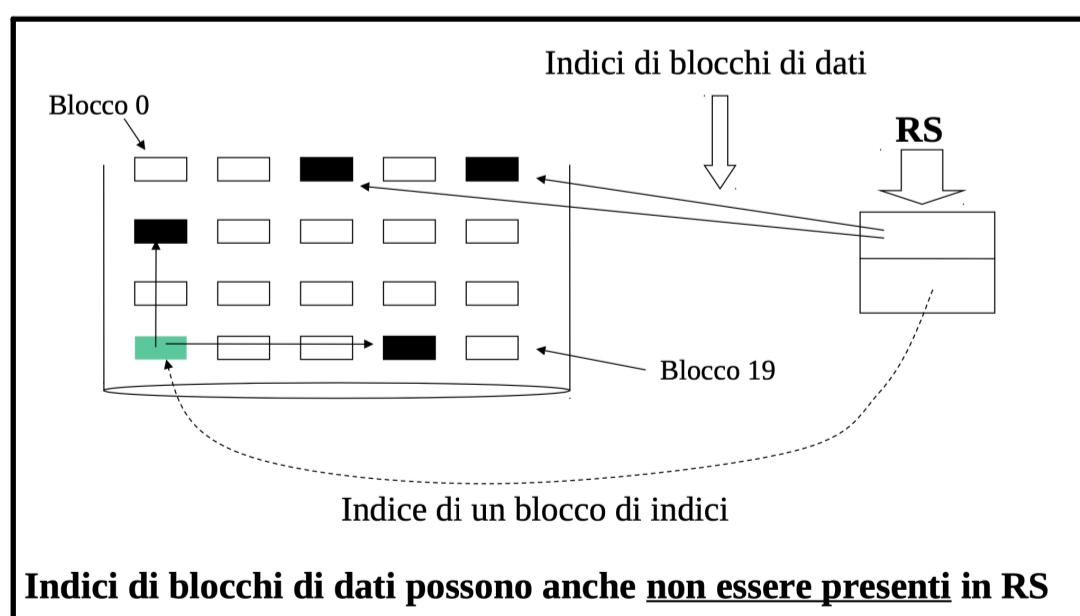
- i blocchi di un file sono rintracciati tramite un indice
- l'occupazione reale è sempre pari al numero di blocchi realmente allocati per record del file
- RS mantiene informazioni sugli indici dei blocchi (maggiore occupazione di spazio per RS rispetto ad altri schemi)

Il record di sistema ha una taglia predeterminata T , e se è predeterminata, la crescita del file è determinata dal numero di elementi che io posso andare ad inserire all'interno di RS. Questo è un problema che abbiamo sull'allocazione indicizzata.

Per gestire file molto grandi, senza far crescere questa taglia - ossia il numero degli elementi che noi manteniamo all'interno del RECORD DI SISTEMA, viene usata l'allocazione indicizzata a LIVELLI MULTIPLI.

Indicizzazione a livelli multipli

È vero che nel record di sistema possiamo mantenere un numero finito di indici per sapere dove è posizionato il file in memoria, però alcuni indici sono DIRETTI che in qualche modo sono blocchi dove noi manteniamo i dati del file, e se il file è piccolo ci va bene così. Supponiamo che il file cresce, a questo punto abbiamo un indice INDIRETTO CHE CI PORTA A PUNTARE AD UN BLOCCO DEL DISPOSITIVO dove noi non manteniamo i dati del file, ma andiamo a scrivere gli indici dei blocchi dove i dati del file sono mantenuti.



Quindi se nel record di sistema siamo in grado di mantenere 3 indici soltanto, siamo in grado di poter allocare un file che eventualmente ha i record che devono essere mantenuti all'interno di quattro blocchi? Sì. Perché utilizziamo il terzo degli indici, quello sotto, per puntare ad un blocco dove inseriamo gli indici di altri blocchi.

Quello si chiama "BLOCCO DI INDICI", e sono molti nella realtà.

Se ci pensiamo è anche inutile mantenere i primi indici, a volte manteniamo soltanto l'indice al blocco di indici. Ci permette di ridurre la TAGLIA DEL RECORD DI SISTEMA. Ma per arrivare sui dati del file dobbiamo fare più

operazioni.