



Corso di Laurea Triennale in Informatica
Università Degli Studi della Basilicata

Sistemi Operativi

Docente:
ugo.erra@unibas.it

12° Lezione

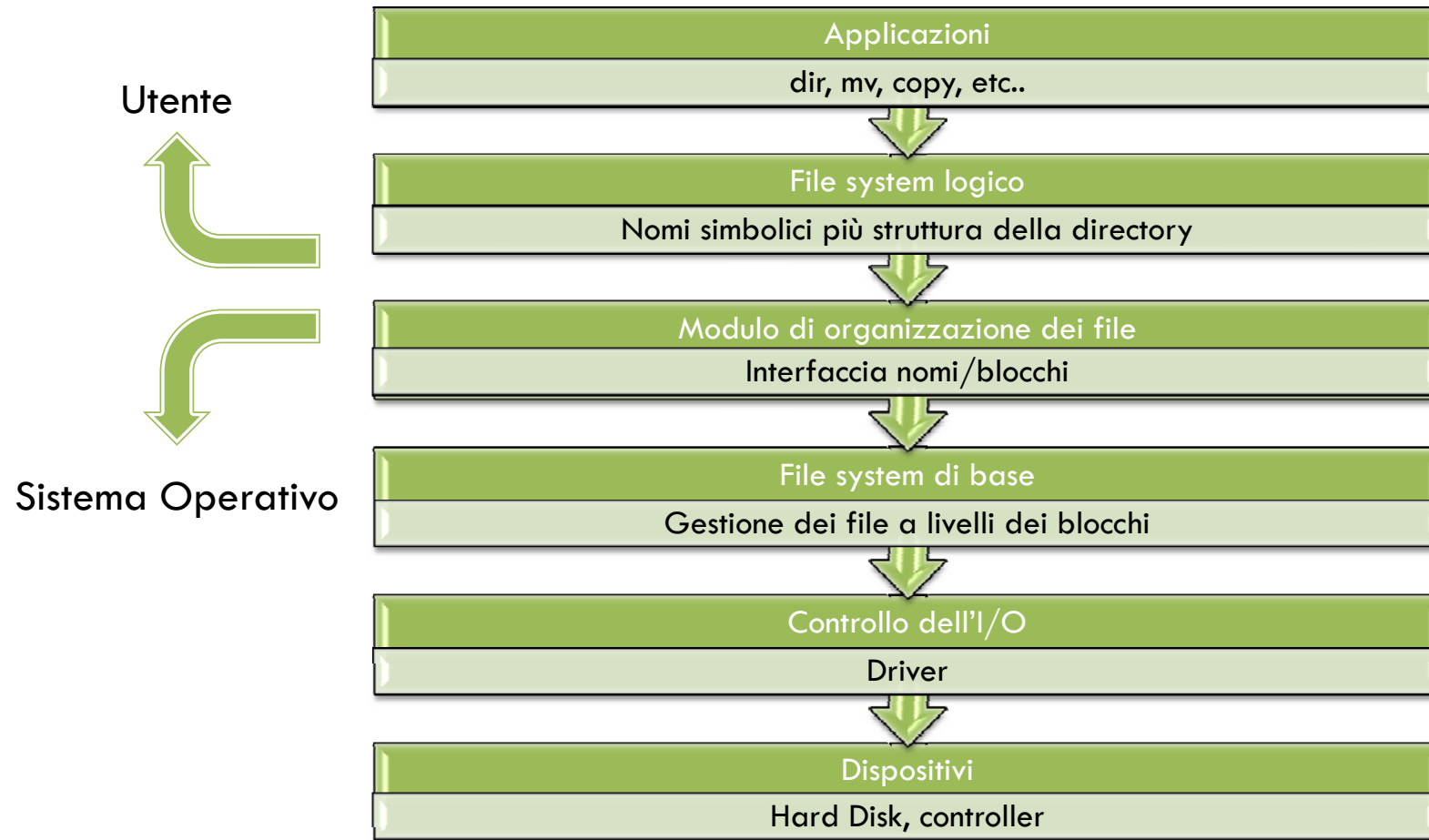
Realizzazione del file system

Sommario della lezione



- Struttura del file system
- Metodi di allocazione
 - ▣ Allocazione contigua
 - ▣ Allocazione concatenata
 - ▣ Allocazione indicizzata
- Gestione dello spazio libero

Struttura a livelli



Blocchi di un disco

- La memoria secondaria è una memoria permanente gestita solitamente con un disco fisso
- Lo spazio del disco fisso è logicamente suddiviso in blocchi, ad esempio di 512 byte
 - ▣ Altri valori possibili sono 256 o 1024 byte
- Ogni blocco è numerato a partire da 0, per cui un disco può essere visto come un array in cui ogni elemento ha una dimensione di 512 byte
- Il controller del disco a partire dal numero di blocco può accedere in lettura e scrittura direttamente
 - ▣ Per ottimizzare gli accessi solitamente lettura e scrittura su disco avvengono in più blocchi

Allocazione di un file

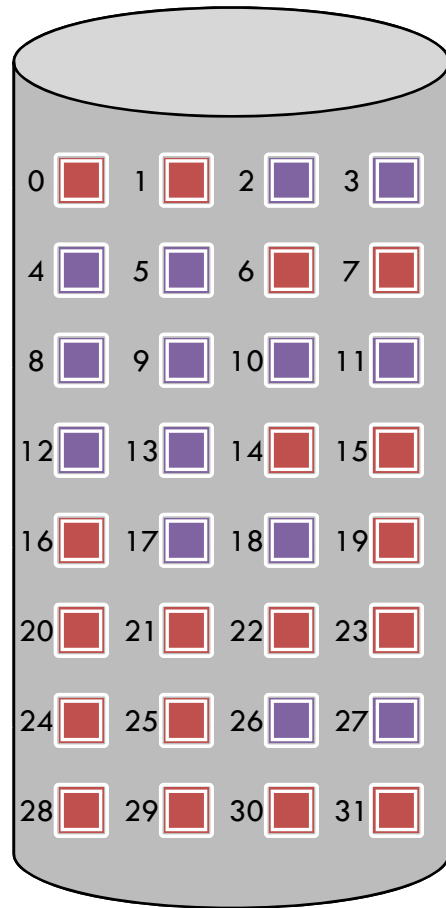


- Per poter memorizzare un file la cui dimensione supera quella di un singolo blocco è necessario dividere il file in più blocchi
- Per allocare il numero di blocchi necessari su disco sono possibili le seguenti strategie:
 - ▣ Allocazione contigua
 - ▣ Allocazione concatenata
 - ▣ Allocazione indicizzata

Allocazione contigua - 1

- Nell'**allocazione contigua** ogni file è allocato in un insieme di blocchi contigui
- Per accedere ad un file è necessario avere le seguenti informazioni:
 - ▣ Il numero del primo blocco del file
 - ▣ Quanti blocchi contigui sono occupati dal file
- Se b è il primo blocco occupato ed n la dimensione del file allora il file è allocato nei blocchi $b, b+1, b+2, \dots, b+n-1$
- Usato nei sistemi IBM VM/CMS

Allocazione contigua - 2



file	Blocco iniziale	lunghezza
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Allocazione contigua: vantaggi e svantaggi

□ Vantaggi:

- L'accesso ai file è veloce e semplice
- Le informazioni aggiuntive da salvare insieme al file sono poche

□ Svantaggi:

- Per allocare un file è necessario trovargli uno spazio libero sul disco che sia contiguo (cosa vi ricorda?)
- È necessaria una strategia di allocazione dello spazio di tipo first/best/worst fit che provoca **frammentazione esterna del disco**
 - Si può ricorrere ad una **ricompattazione periodica** del disco che è più costosa rispetto alla memoria primaria

Come gestire la crescita del file?



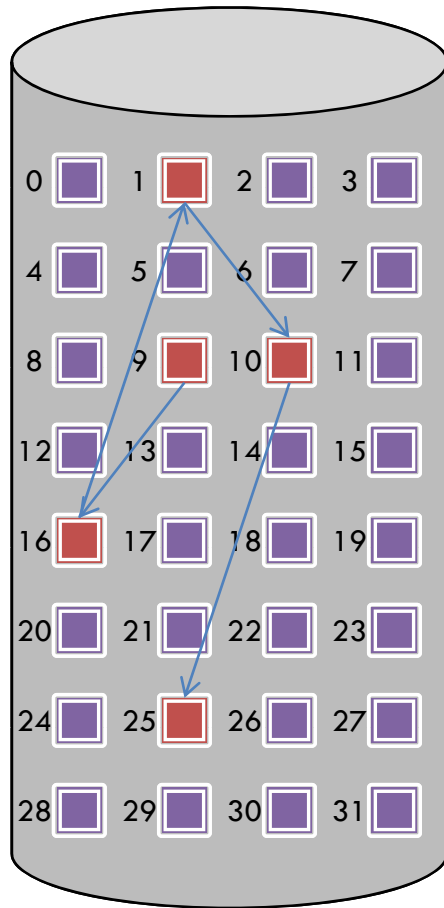
- Cosa succede se il file cresce oltre la dimensione che gli era stata allocata?
- Sono possibili due strategie:
 - ▣ Riallocare lo spazio originale
 - ▣ Sovradimensionare il file in fase di allocazione aumentando però la frammentazione interna
- I problemi sono gli stessi dell'allocazione contigua nella memoria principale

Allocazione concatenata - 1



- Nell'allocazione concatenata ogni blocco contiene un puntatore al blocco successivo del file
 - ▣ Solitamente si utilizzano gli ultimi byte
 - ▣ Soluzione simile a quella della memoria principale
- L'informazione addizionale da memorizzare è il numero del blocco iniziale
 - ▣ Si potrebbe anche memorizzare il numero di blocchi usati e/o il numero del blocco finale
 - ▣ Nell'ultimo blocco viene scritto un numero negativo per segnalare al sistema che quello è l'ultimo blocco del file

Allocazione concatenata - 2



file	Start	end
jeep	9	25

Allocazione concatenata: vantaggi e svantaggi

- Vantaggi:
 - ▣ Non sono necessari blocchi contigui, per cui non c'è bisogno di ricompattare il disco e non si verifica frammentazione esterna
 - ▣ Qualsiasi blocco libero può essere utilizzato per memorizzare un pezzo del file
- Svantaggi:
 - ▣ Per ogni blocco gli ultimi byte sono utilizzati per memorizzare il puntatore al (ossia il numero del) blocco successivo
 - ▣ Nel caso di blocchi da 512 byte, con 4 byte per puntatore, circa lo 0,78% di un blocco non può essere utilizzato per memorizzare dati del file

Accesso diretto nell'allocazione concatenata

- L'accesso diretto ad un blocco del file non è efficiente:
 - ▣ Per accedere ad un blocco è necessario leggere tutti i blocchi precedenti
 - ▣ Per leggere l' n -esimo blocco sono necessari $n-1$ accessi al disco
- Il sistema di allocazione è poco affidabile
 - ▣ Se un blocco del file viene danneggiato si perde tutta la parte del file dal blocco danneggiato fino alla fine del file

Soluzioni ai problemi - 1



- Piuttosto che usare una lista a singolo puntatore possiamo utilizzare una lista doppiamente concatenata
 - ▣ Se si danneggia un blocco possiamo recuperare i successivi ripercorrendo la catena all'indietro
- Per ogni blocco possiamo memorizzare il nome del file a cui appartiene e la sua posizione all'interno della lista
 - ▣ Nel caso in cui un blocco si “perde” siamo sempre in grado di sapere a chi apparteneva e dove era collocato

Soluzioni ai problemi - 2

- Possiamo cercare di mantenere i vantaggi dell'allocazione contigua considerando il disco formato da cluster di blocchi adiacenti
 - ▣ Ad esempio, ciascun cluster può essere composto da 4 blocchi del disco adiacenti, e avere quindi una dimensione di 2 o 4 Kbyte
- I vantaggi di avere blocchi di dimensione maggiore sono:
 - ▣ I tempi di accesso al file diminuiscono perché la testina deve essere riposizionata meno volte
 - ▣ Lo spazio non è sprecato come prima per i puntatori ma aumenta la frammentazione interna

FAT – File Allocation Table



- L'allocazione concatenata è stata adottata dai sistemi operativi MS-DOS, OS2 e nei sistemi windows
- La FAT è un array allocato all'inizio del disco in cui ogni entry corrisponde ad un blocco
 - ▣ Il numero contenuto nell'entry rappresenta un blocco occupato
- Le entry il cui valore è zero corrispondono a blocchi liberi

FAT – File Allocation Table

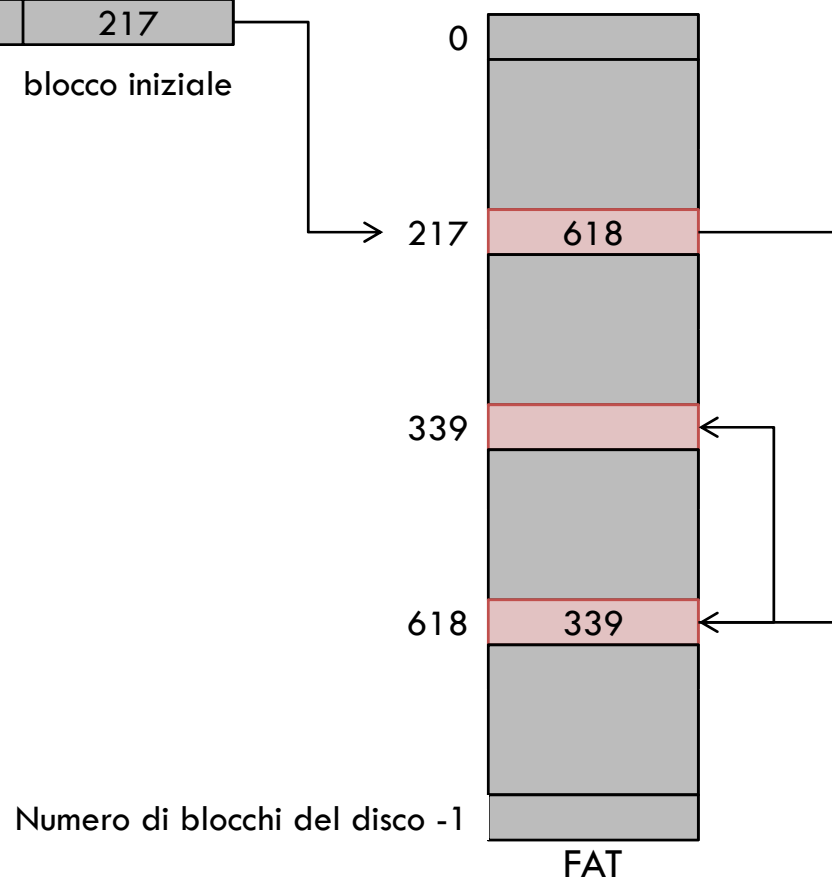
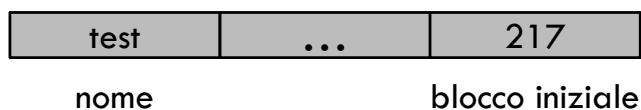


- Lo scopo della FAT è di:
 - ▣ Registrare lo stato di allocazione di tutti i blocchi del disco
 - ▣ Riprodurre la lista concatenata di blocchi di ogni file
- Se il blocco i punta al blocco n , allora nella FAT l' i -esima entry contiene il numero n
- Se l'ultimo blocco del file è il numero j , la j -esima entry della FAT contiene un marker speciale di fine file

Esempio di FAT

- Il file test ha tra gli attributi il puntatore al primo blocco del file
- I restanti blocchi sono accessibili seguendo gli indici all'interno della catena dei puntatori all'interno della FAT

Elemento della directory



FAT: vantaggi e svantaggi

□ Vantaggi:

- ▣ L'accesso diretto al file migliore se la FAT è tenuta costantemente in memoria
- ▣ Se un blocco si perde il sistema è ancora sicuro
- ▣ La gestione dei blocchi liberi è automatica

□ Svantaggi:

- ▣ La quantità di memoria occupata dalla FAT è pari al numero di blocchi del disco. Ogni entry della FAT memorizza un numero di blocco
- ▣ All'aumentare della dimensione dei dischi aumenta in maniera proporzionale la dimensione della FAT
- ▣ La perdita della FAT comporta l'impossibilità di accedere ai file, quindi è necessario salvarla periodicamente su disco fisso

Overhead occupazione di memoria

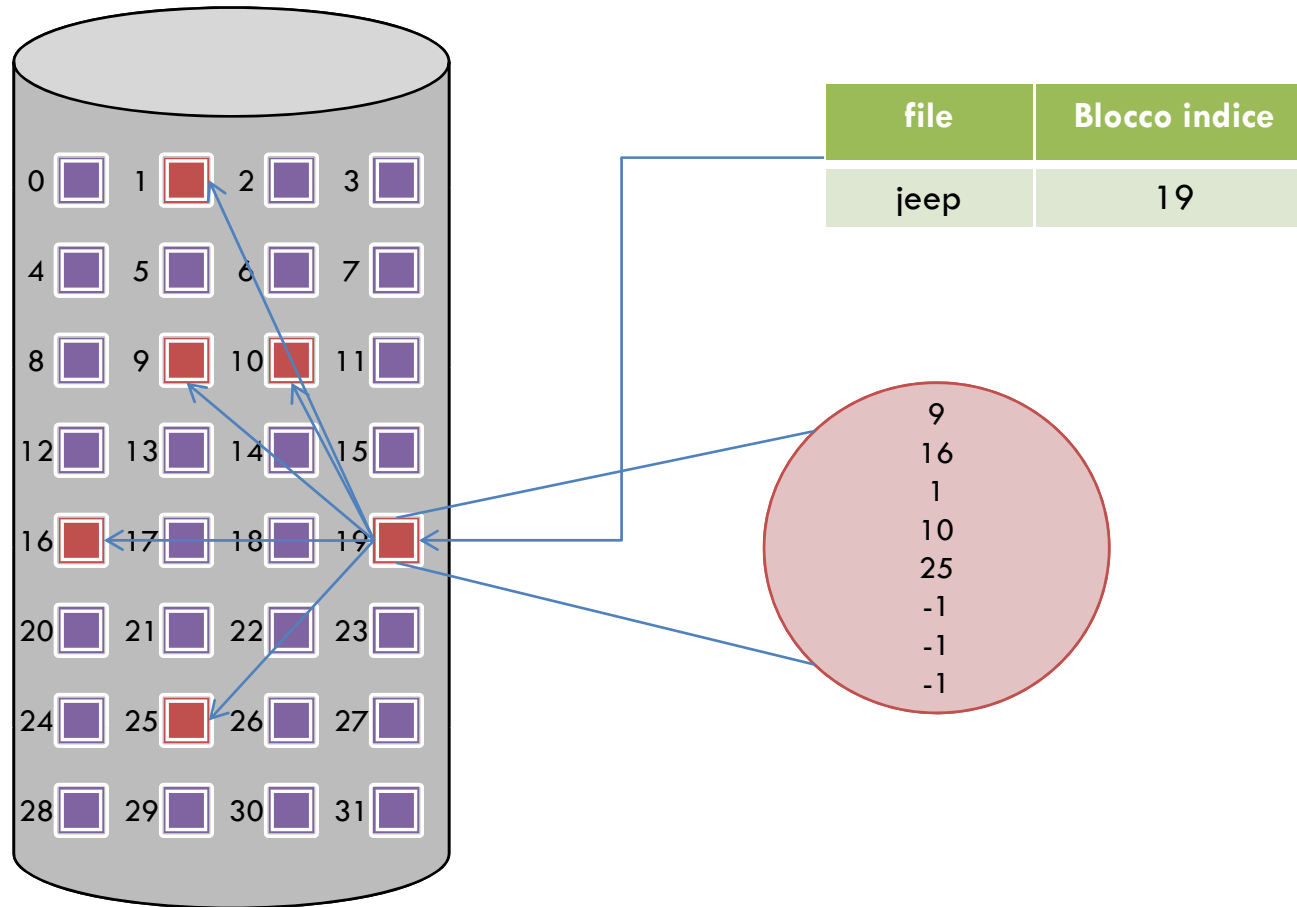
- Supponiamo di avere un disco fisso da 20 GB con 1Kb per blocco
 - ▣ $20\text{GB} = 21.474.836.480$ byte
 - ▣ $1\text{ kb} = 1.024$ byte
- Quanto spazio occupa la FAT?
- Numero di blocchi necessari 20.971.520
- Ogni blocco è indicizzato con un intero a 32 bit = 4 byte
- Spazio occupato dalla FAT
 - $20.971.520 \times 4 = 81920\text{ Kb} = 80\text{ Mb}$ circa 0.4 %

Allocazione indicizzata - 1



- Nell'**allocazione indicizzata** memorizziamo all'interno di un blocco chiamato **blocco indice** tutti i blocchi del file
- Per recuperare tutti i blocchi del file è necessario un solo accesso al blocco indice da cui ricaviamo tutti i blocchi memorizzati
- Tra gli attributi del file da memorizzare è necessario memorizzare anche il blocco indice del file

Allocazione indicizzata- 2



Allocazione indicizzata: vantaggi e svantaggi

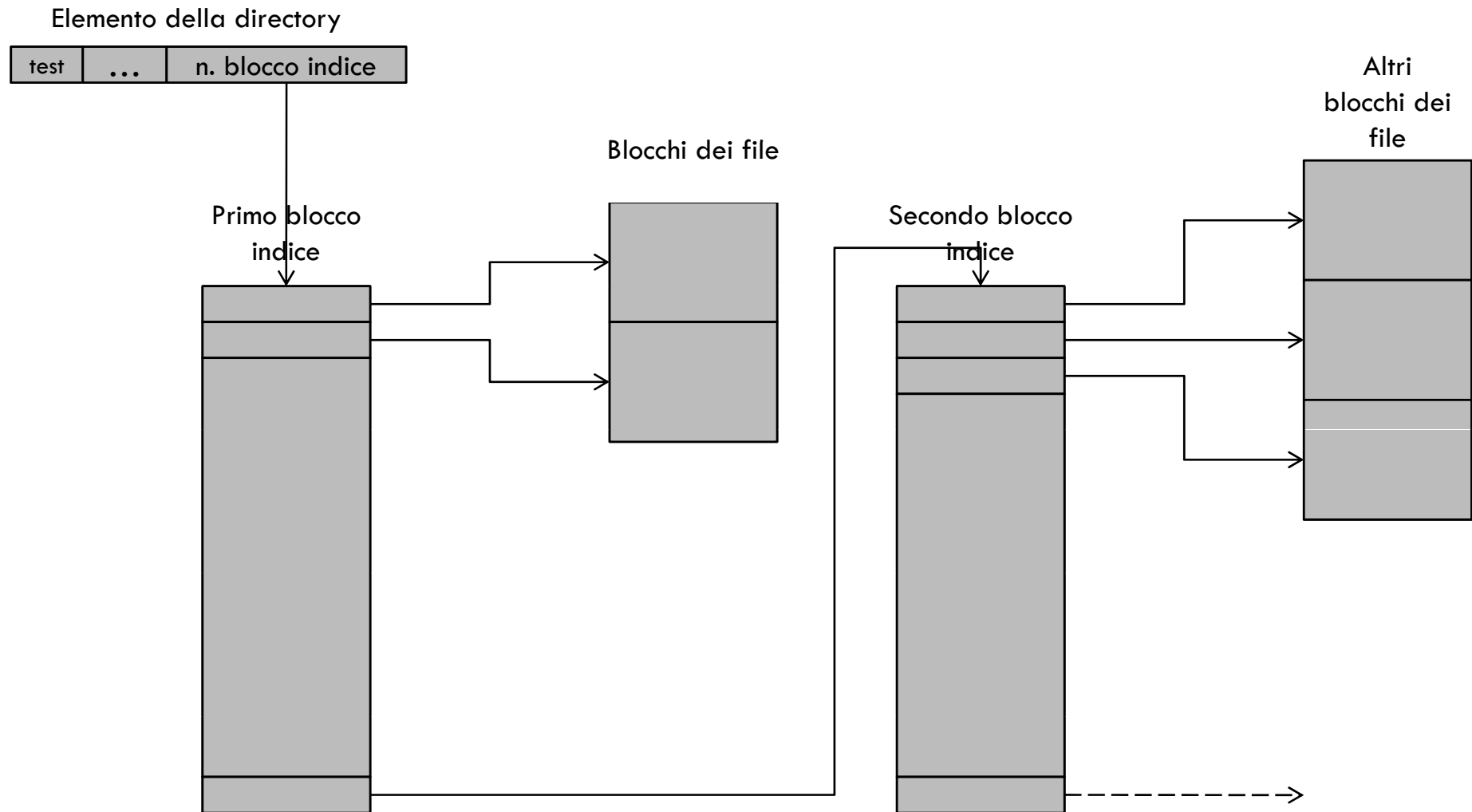
□ Vantaggi:

- ▣ La frammentazione esterna non si verifica in quanto non sono necessari blocchi contigui
- ▣ L'accesso diretto e l'accesso sequenziale sono efficienti
- ▣ Abbiamo bisogno di sprecare almeno un blocco indice
- ▣ Se il file è piccolo il blocco indice è quasi vuoto

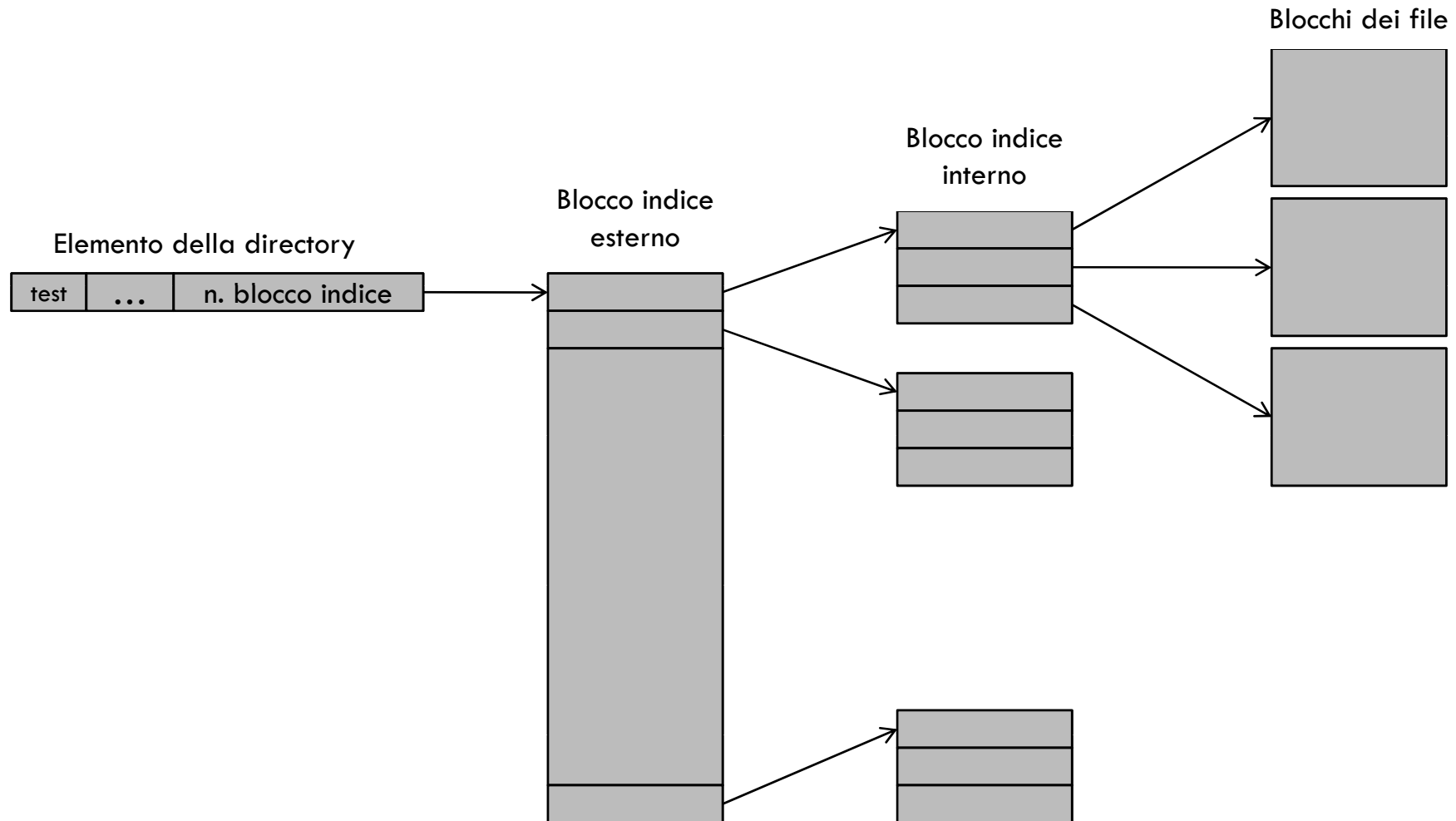
□ Svantaggi:

- ▣ Ogni blocco indice può gestire file di una dimensione fissata. Che succede se il file eccede questa dimensione? Le soluzioni sono:
 - L'ultima entry del blocco indice punta ad un secondo blocco indice (**schema concatenato**)
 - Il blocco indice contiene solo puntatori ad altri blocchi indice (**schema a più livelli**)

Allocazione indicizzata: schema concatenato



Allocazione indicizzata: schema a più livelli



Gli i-nodi di Unix



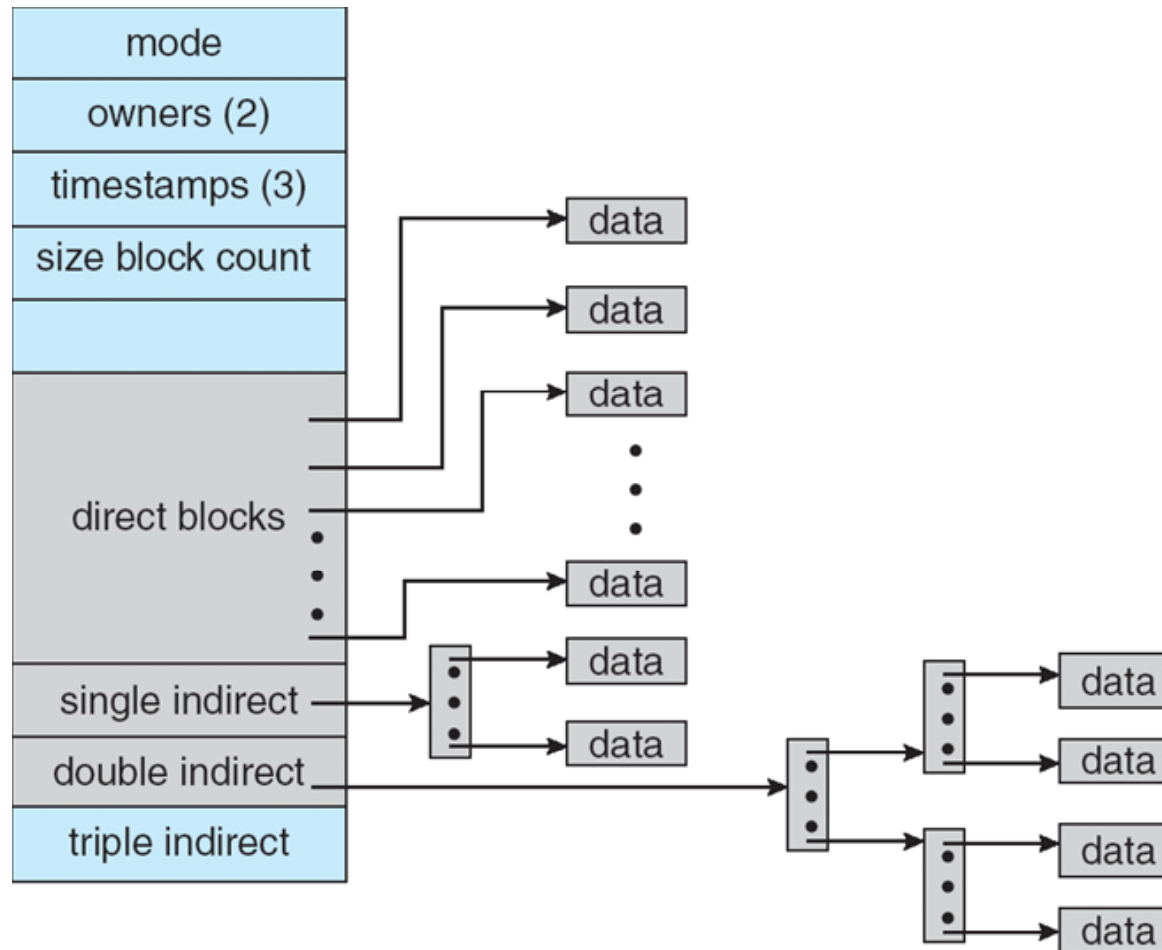
- Nel sistema operativo Unix, ad ogni file è associato un i-node che fa da blocco indice
 - ▣ Ogni i-node contiene anche gli attributi del file
- Gli i-node sono gestiti dal Sistema Operativo e sono memorizzati sul disco fisso in una porzione riservata
 - ▣ Di solito nella prima parte dell'hard disk

i-node



- Ogni i-node contiene:
 - ▣ 12 puntatori **diretti a blocchi di dati del file**
 - ▣ Un puntatore **single indirect che punterà ad un blocco indice che** conterrà puntatori a blocchi di dati file
 - ▣ Un puntatore **double indirect che punterà ad un blocco indice che** conterrà puntatori a blocchi indice ognuno dei quali conterrà puntatori a blocchi di dati del file
 - ▣ Un puntatore **triple indirect che punterà ad un blocco indice che** conterrà puntatori a blocchi indice ognuno dei quali conterrà puntatori a blocchi indice, ognuno dei quali conterrà puntatori a blocchi di dati del file

i-node



Gestione dello spazio libero



- Il Sistema Operativo deve tenere traccia di tutti i blocchi liberi del disco fisso
- Il file system deve gestire una struttura dati opportuna memorizzata su disco che gli permetta di risalire ai blocchi liberi
 - ▣ In Unix, queste informazioni sono memorizzate nel superblocco, contenuto in alcuni blocchi iniziali dell'HD, a partire dal blocco numero 1.
 - ▣ In Windows XP queste informazioni stanno nella MFT

Utilizzo della spazio libero



- Questa struttura dati sarà utilizzata ogni qualvolta cerchiamo dei blocchi liberi per:
 - ▣ Creare un file
 - ▣ Estendere un file
- Se un file viene cancellato i suoi blocchi diventano liberi e quindi inseriti in questa struttura dati

Vettori di bit

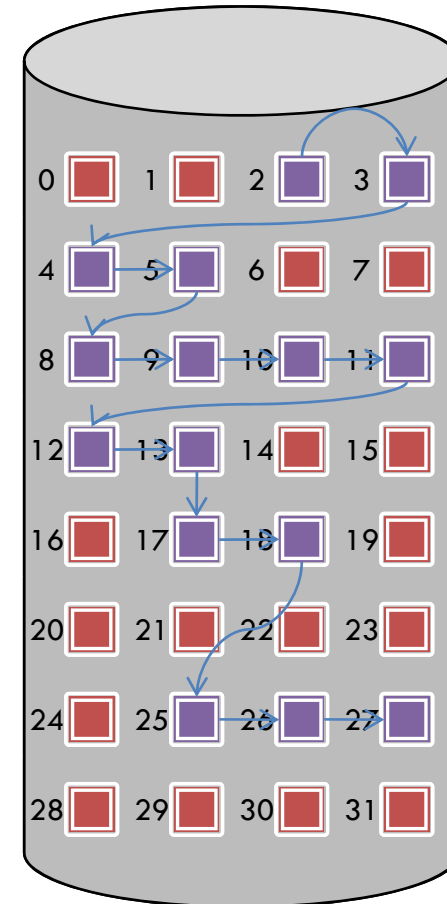
- Per ogni blocco usiamo un bit per indicare se è libero o occupato
 - ▣ $\text{bit}[i] = 1 \Rightarrow$ blocco i -esimo libero
 - ▣ $\text{bit}[i] = 0 \Rightarrow$ blocco i -esimo occupato



- Affinché il vettore di bit sia efficiente deve essere tenuto nella memoria principale e periodicamente salvato su disco
- Per ridurre l'occupazione del vettore di bit è possibile usare un bit per dei cluster di blocchi
- Approccio utilizzato in Mac/OS

Lista concatenata

- Si utilizza una lista concatenata di blocchi liberi
 - ▣ Ogni blocco libero punta al successivo blocco libero
 - ▣ Nessuno spreco di spazio
- In alternativa si può usare una variante della FAT per gestire i blocchi liberi in modo da ridurre i tempi per cercare molti blocchi liberi



Raggruppamento/Conteggio

□ Raggruppamento

- ▣ Raggruppare in un blocco più puntatori a blocchi liberi
- ▣ L'ultimo puntatore punta ad un altro blocco di blocchi liberi come nell'allocazione indicizzata
- ▣ Efficiente se dobbiamo trovare molti blocchi liberi velocemente

□ Conteggio

- ▣ Mantenere il numero di un blocco e quanti blocchi consecutivi liberi lo seguono
- ▣ Simile alla lista ma con meno entry