

Architettura dei Calcolatori e Sistemi Operativi

Filesystem

Chair

Politecnico di Milano

Prof. C. Brandolese

e-mail: carlo.brandolese@polimi.it
phone: +39 02 2399 3492
web: home.dei.polimi.it/brandolese

Teaching Assistant

D. Iezzi

e-mail: [domenico.iezzi \[at\] polimi \[dot\] it](mailto:domenico.iezzi@polimi.it)
material: github.com/NoMore201/polimi_cr_acso_2019

Outline

- **Filesystem**

- Accesso al filesystem
- Schemi di allocazione
 - Allocazione contigua
 - Allocazione concatenata
 - Allocazione indicizzata
- Accessi sequenziali / accessi casuali

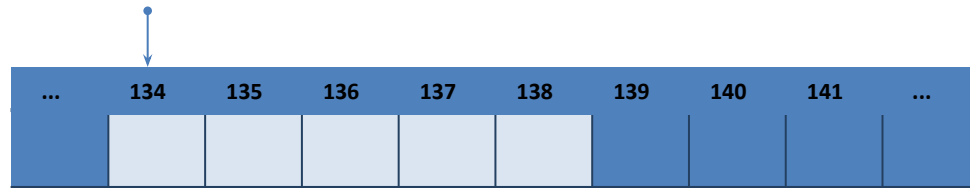
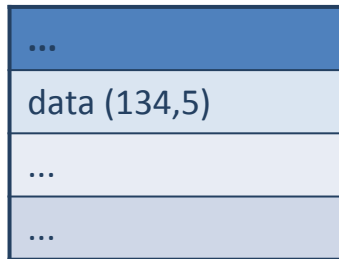
Allocazione Contigua

- **I blocchi di un file sono adiacenti**

- Un file di n blocchi è memorizzato nelle posizioni adiacenti $b, b+1, \dots, b+n-2, b+n-1$
- Un descrittore deve indicare solo la coppia (b, n)
- I tempi d'accesso sono contenuti poiché
 - Accesso a due blocchi successivi b e $b+1$ raramente richiede lo spostamento della testina in quanto hanno una probabilità elevata di trovarsi sulla stessa traccia
 - L'accesso al blocco logico i -esimo comporta l'accesso diretto al blocco fisico $b+i$

- **Problema**

- Allocazione spazio per un nuovo file



Allocazione Contigua

- **Dovendo creare un nuovo file di m blocchi**
 - E' necessario individuare una porzione di disco costituita da almeno m blocchi contigui
 - **Si usano tre politiche:**
 - First-Fit
 - La prima zona, di almeno m blocchi, viene usata
 - Best-Fit
 - La zona più piccola, di almeno m blocchi, viene usata
 - Worst-Fit
 - La zona più grande, di almeno m blocchi, viene usata
 - **Le tecniche migliori**
 - Sono le prime due
 - In particolare il metodo first-fit risulta più veloce
 - **L'allocazione contigua**
 - Crea, nel tempo, zone inutilizzate di piccole dimensioni
 - Tali zone hanno una bassa probabilità di contenere un file
 - Questo fenomeno viene detto frammentazione esterna
-

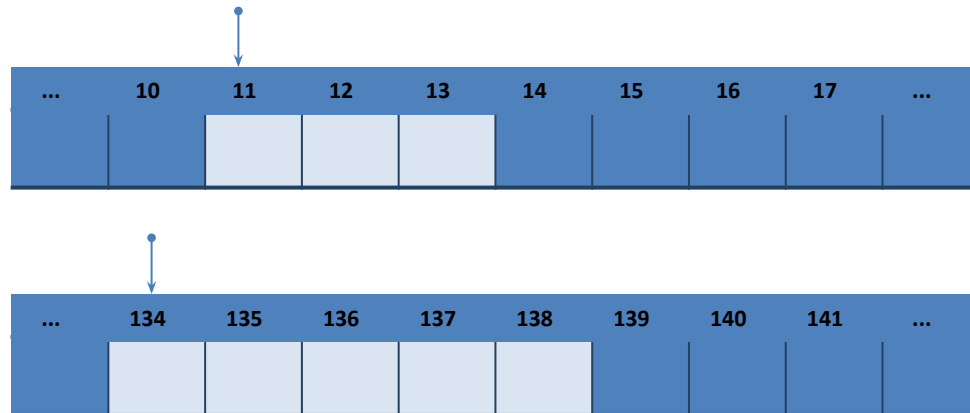
Allocazione Contigua

- **Una soluzione consiste nella compattazione dei dischi**
 - I file su un disco vengono letti e memorizzati temporaneamente altrove
 - Su un secondo disco, in una porzione libera del disco in esame o in memoria centrale
 - **Si procede come segue**
 - Il disco originale viene cancellato
 - Completamente, o più spesso una porzione alla volta
 - I file vengono riscritti in sequenza, eliminando gli spazi vuoti
 - **Questa operazione**
 - È molto costosa in termini di tempo
 - Deve essere compiuta con una certa frequenza
 - **Una soluzione migliore**
 - Memorizzazione di un file in due zone differenti
 - Ogni zona formata da blocchi contigui
 - La zona aggiuntiva prende il nome di "extent"
 - Ancora necessari algoritmi per la ricerca di spazio disponibile
-

Allocazione Contigua

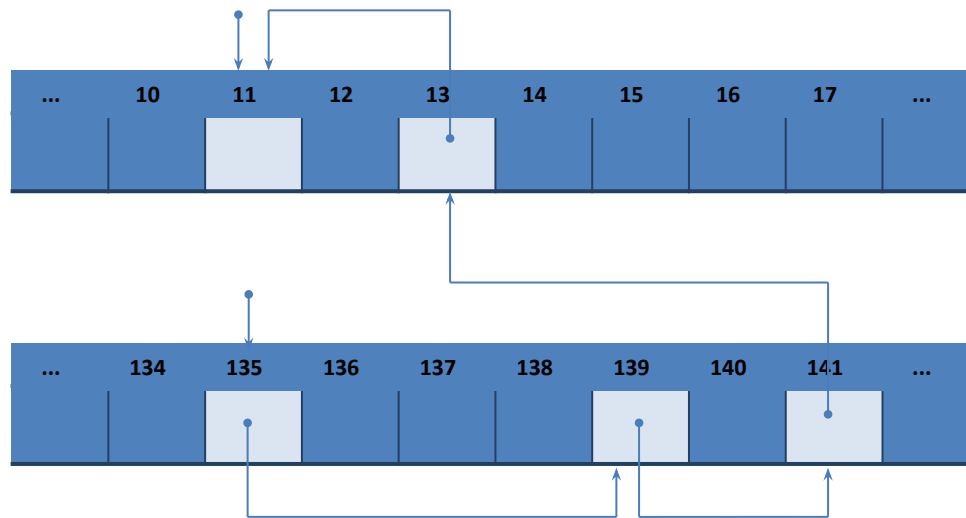
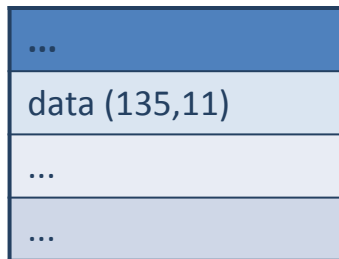
- Un descrittore di file indica (b, n1, e, n2):
 - b base della sezione principale
 - n1 dimensioni della sezione principale
 - e base dell'extent
 - n2 dimensione dell'extent

...
data (134,5,11,3)
...
...



Allocazione Concatenata

- **L'idea dell'uso di un extent**
 - Può essere estesa a ad un numero maggiore di estensioni
- **In questo modo**
 - Un file è costituito da una sequenza di blocchi in cui:
 - Il descrittore contiene il riferimento al primo ed all'ultimo blocco
 - Ogni blocco contiene
 - I dati
 - Un riferimento al blocco successivo



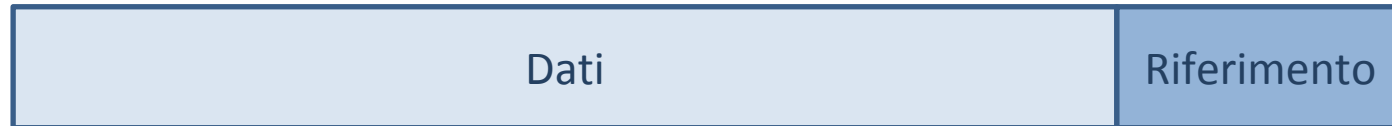
Allocazione Concatenata

- **Questa soluzione**

- Risolve tutti i problemi tipici della allocazione contigua
- Non si ha frammentazione esterna

- **In ogni blocco**

- Una parte è dedicata a contenere i dati
- Una parte è dedicata a contenere un riferimento al blocco successivo



- **La memorizzazione dei riferimenti**

- Riduce lo spazio disponibile per i dati
- Consideriamo un esempio
 - Blocchi di 512 byte e riferimenti di 4 byte
 - Dimensione del disco $512 \times 2^{32} = 2\text{Tbyte}$
 - Spreco di spazio pari allo 0.78% del disco, ovvero circa 16Gbyte

Allocazione Concatenata

- **Anche questo metodo presenta alcuni svantaggi**
 - Per l'accesso casuale è necessario scorrere il file dall'inizio fino al blocco desiderato
 - L'accesso ad un file è meno efficiente
 - In generale comporta molti riposizionamenti della testina
 - **Una soluzione consiste**
 - Nel raggruppare più blocchi in un "cluster"
 - Prevedere l'accesso ai tali gruppi, piuttosto che ai singoli blocchi
 - **In questo modo si ha**
 - Miglioramento delle prestazioni
 - Minor numero di riposizionamenti della testina
 - Riduzione dello spazio utilizzato per i riferimenti
 - Uno per cluster invece di uno per blocco
 - Maggiore frammentazione interna
 - L'unità di allocazione è un cluster, che è più grande di un blocco
 - **Questo approccio è utilizzato in molti sistemi operativi**
-

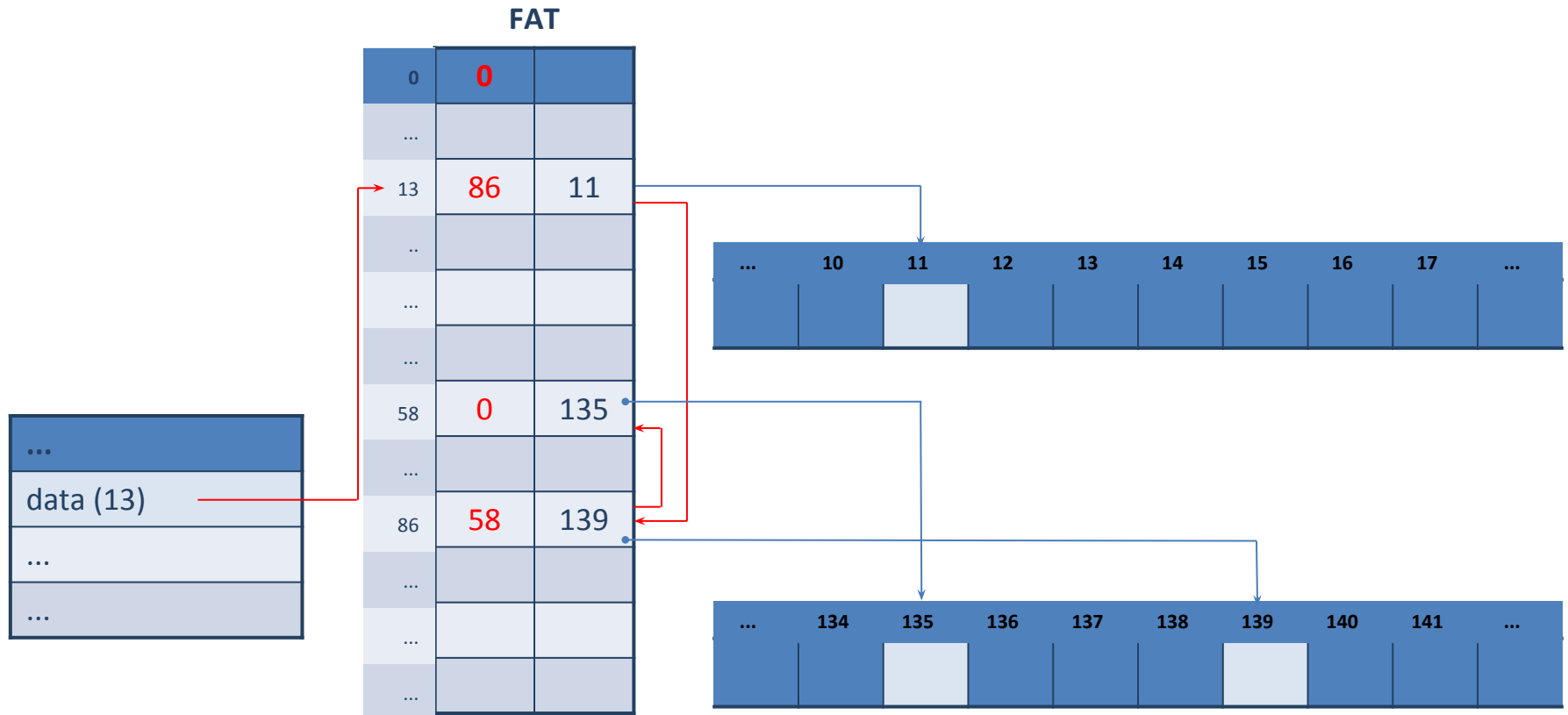
Allocazione Concatenata

- **Le liste di blocchi dell'allocazione concatenata sono fragili**
 - La perdita di un solo riferimento può rendere inaccessibili grandi quantità di dati
 - **Una soluzione consiste nella costruzione di un indice per ogni partizione**
 - Detto File Allocation Table o FAT
 - **La FAT**
 - Contiene tanti elementi quanti sono i blocchi del disco
 - Un blocco disponibile è indicato da uno 0 nella tabella
 - L'ultimo blocco di un file è indicato da uno speciale valore EOF
 - Ogni elemento della tabella contiene l'indice dell'elemento della FAT che contiene il blocco successivo
 - La robustezza può essere facilmente aumentata mediante
 - Tecniche di codifica dei dati della FAT
 - Ridondanza
 - **Questo metodo**
 - Comporta un tempo di accesso mediamente maggiore con allocazione concatenata
 - E' stato adottato da MS-DOS, OS/2 e diversi file system per memorie Flash
-

Allocazione Concatenata

■ Utilizzando allocazione concatenata e FAT

- Un descrittore di file deve semplicemente indicare il numero del primo blocco del file
- Le informazioni sulla posizione dei blocchi successivi sono contenute nella FAT



Allocazione Indicizzata

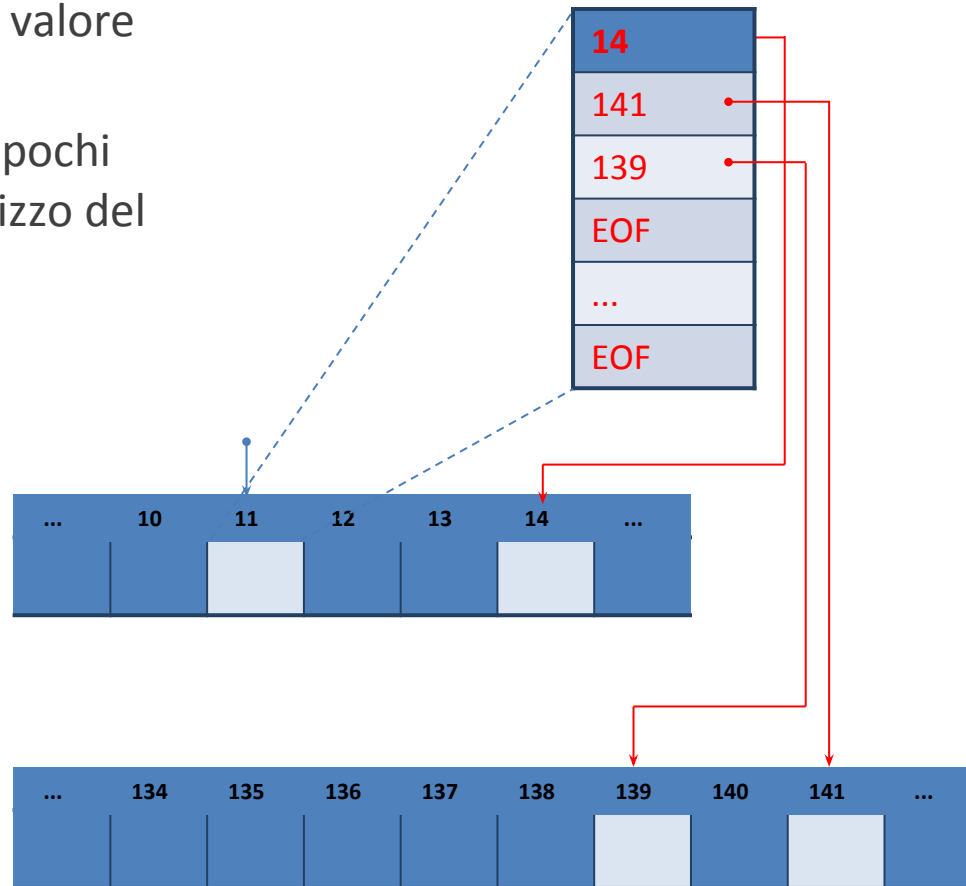
- **L'allocazione indicizzata**
 - Si basa sul raggruppamento di tutti i riferimenti
 - Risolve i problemi di scarsa efficienza della allocazione concatenata
 - **I riferimenti ai blocchi di un file sono memorizzati in un unico blocco speciale**
 - Detto blocco indice
 - Il blocco indice appartiene al file stesso
 - **Un blocco indice è quindi**
 - Un vettore di riferimenti ai blocchi del file
 - L'*i*-esimo elemento del vettore si riferisce all'*i*-esimo blocco del file
 - Il descrittore del file contiene il riferimento al blocco indice
 - **In questo modo si ottiene**
 - Eliminazione della frammentazione esterna
 - Accesso casuale ai blocchi molto efficiente
 - **Lo spazio richiesto per i riferimenti è in generale maggiore che nel caso di allocazione concatenata**
 - Per pochi riferimenti è comunque necessario un intero blocco
-

Allocazione Indicizzata

■ Nel blocco indice

- Gli elementi che non si riferiscono a nessun blocco hanno il valore speciale "EOF"
- Se un file è costituito da pochi blocchi si ha un sottoutilizzo del blocco indice

...
data (11)
...
...

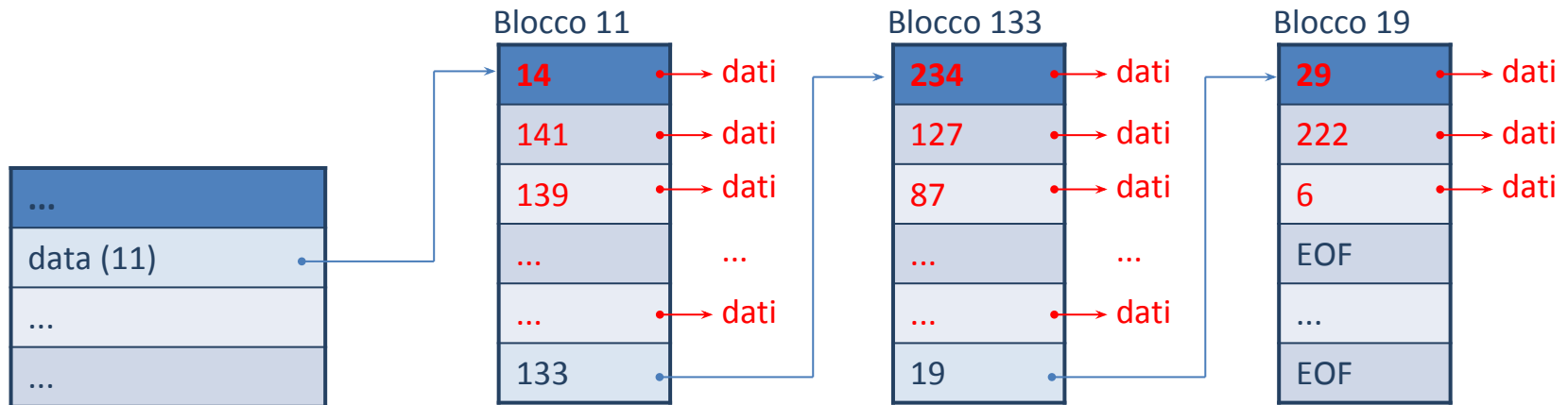


Allocazione Indicizzata

- **È importante scegliere una dimensione opportuna per il blocco indice**
 - Se troppo grande
 - Viene sprecato molto spazio nel caso di file piccoli (pochi blocchi da indicizzare)
 - Se troppo piccolo
 - Non consente di trattare file di grandi dimensioni
 - **In genere il blocco indice**
 - Coincide con un blocco fisico
 - **Esistono tre schemi possibili per risolvere i problemi legati alla gestione di file di dimensioni molto variabile**
 - Schema concatenato
 - Schema multilivello
 - Schema combinato
 - **Tutti questi metodi sono utilizzati in sistemi operativi e file system reali**
-

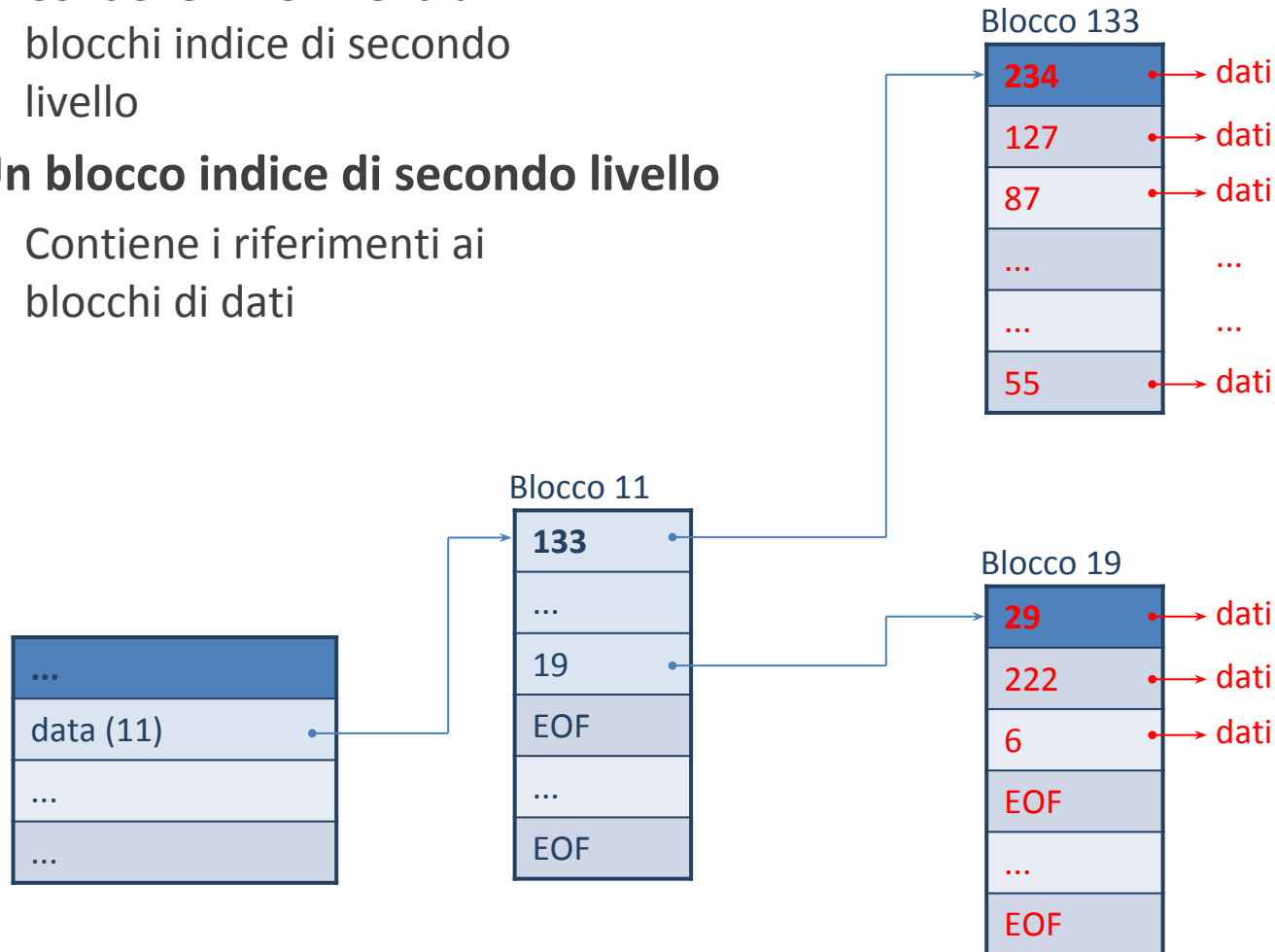
Allocazione Indicizzata – Schema concatenato

- **Il blocco indice**
 - Contiene i riferimenti ai blocchi del file
- **L'ultimo elemento del blocco indice**
 - Contiene un riferimento ad un secondo blocco indice, se il file è di grandi dimensioni
- **L'ultimo elemento dell'ultimo blocco indice**
 - Contiene il valore speciale EOF ad indicare la fine del file



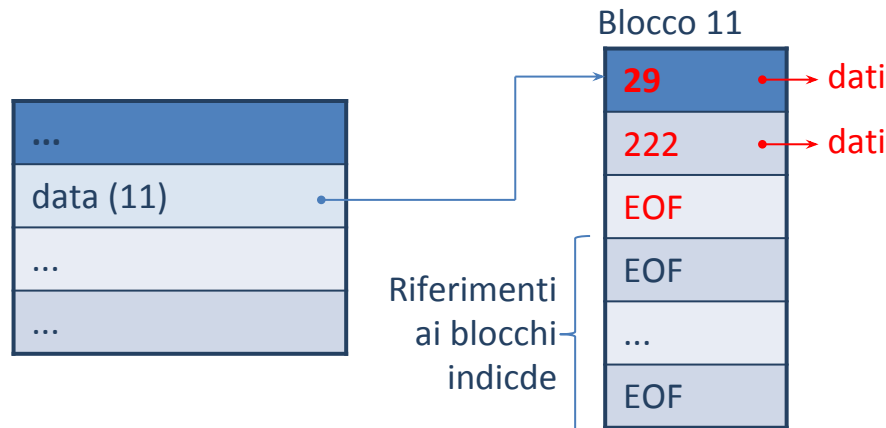
Allocazione Indicizzata – Schema multilivello

- **Un blocco indice di primo livello**
 - Contiene i riferimenti ai blocchi indice di secondo livello
- **Un blocco indice di secondo livello**
 - Contiene i riferimenti ai blocchi di dati



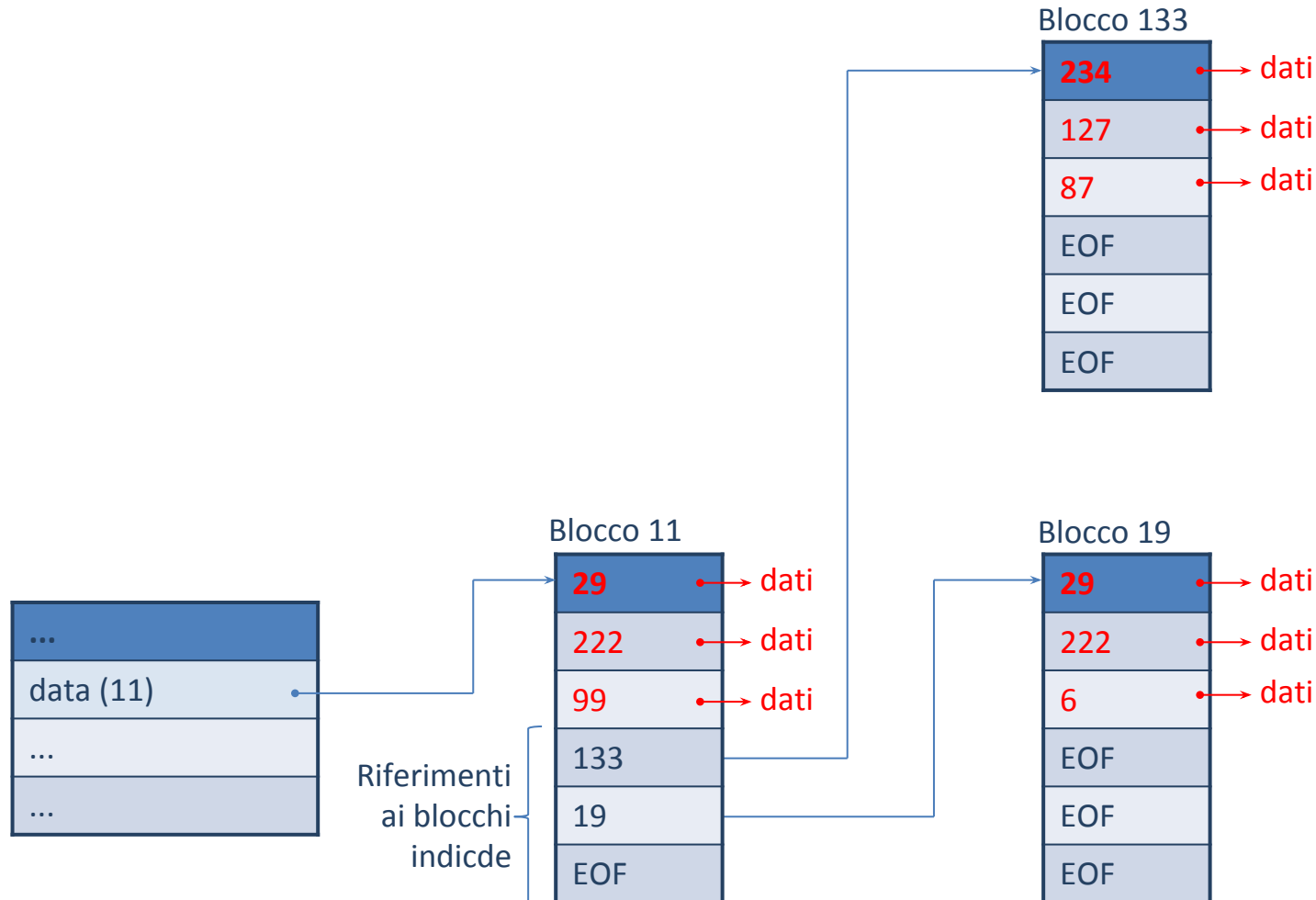
Allocazione Indicizzata – Schema combinato

- In ogni blocco indice
 - La parte iniziale contiene riferimenti a blocchi di dati del file
 - La parte finale contiene riferimenti a blocchi indice di secondo livello
- Se il file ha dimensioni ridotte
 - Non viene utilizzato il secondo livello



Allocazione Indicizzata – Schema combinato

- Per un file di grandi dimensioni
 - Si ricorre ai blocchi indice concatenati



Schemi di allocazione

■ Esercizio 1

- Si supponga di avere lo spazio fisico dell'unità di memorizzazione così frammentato:



- Si vuole allocare un file che occupa **8 blocchi**. Indicare cosa succede a seconda che venga adottata una delle seguenti politiche:

☐ Allocazione contigua – politica first-fit/best-fit/worst fit

No match

☐ Allocazione contigua – politica base+extent (b,n1,e,n2)

(100,5,200,3)

☐ Allocazione concatenata

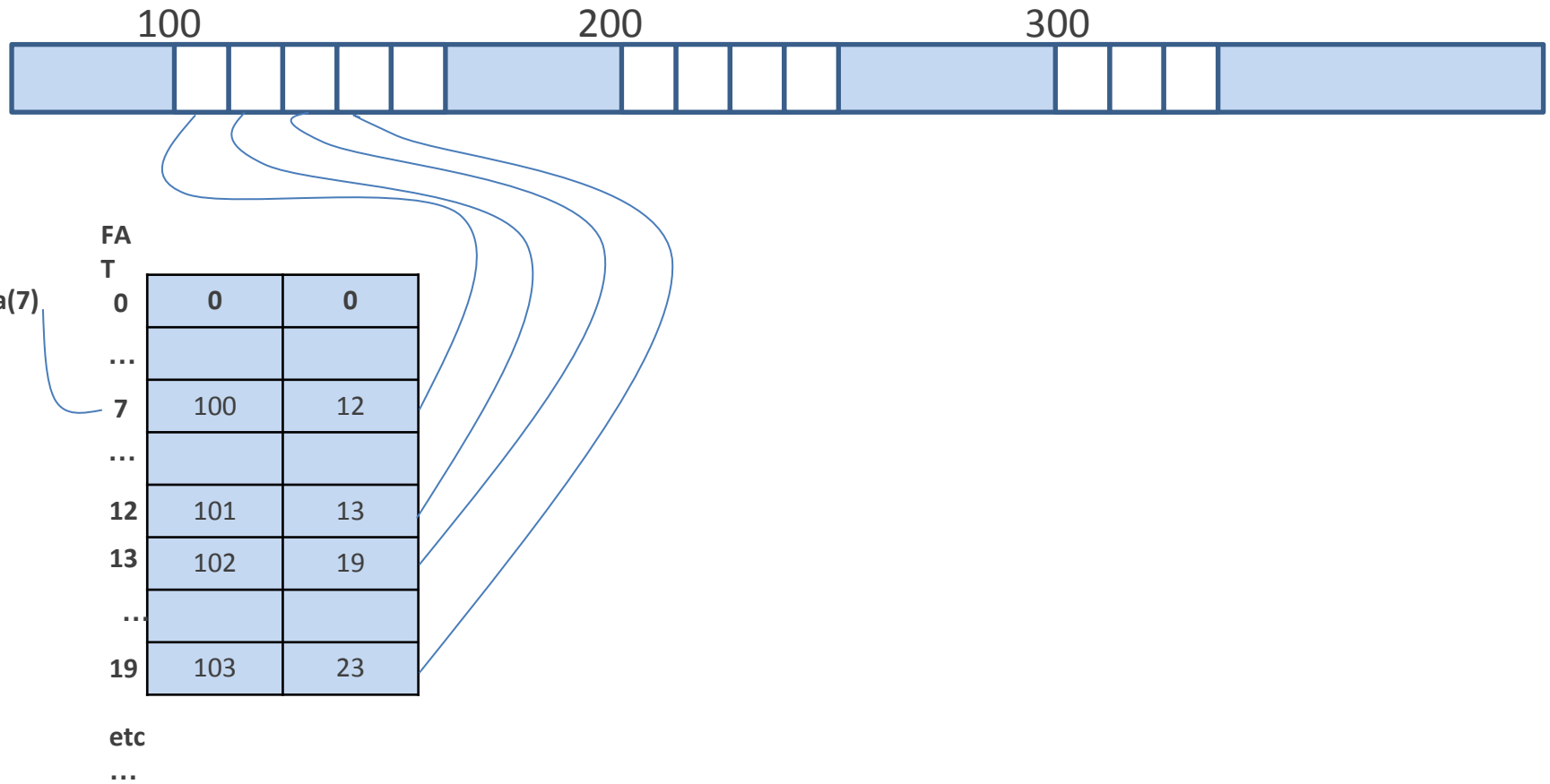
(100, 202)

100		101		102		103		104		200		201		202	
Data	101	Data	102	Data	103	Data	104	Data	200	Data	201	Data	202	Data	EOF

Schemi di allocazione

Continua...

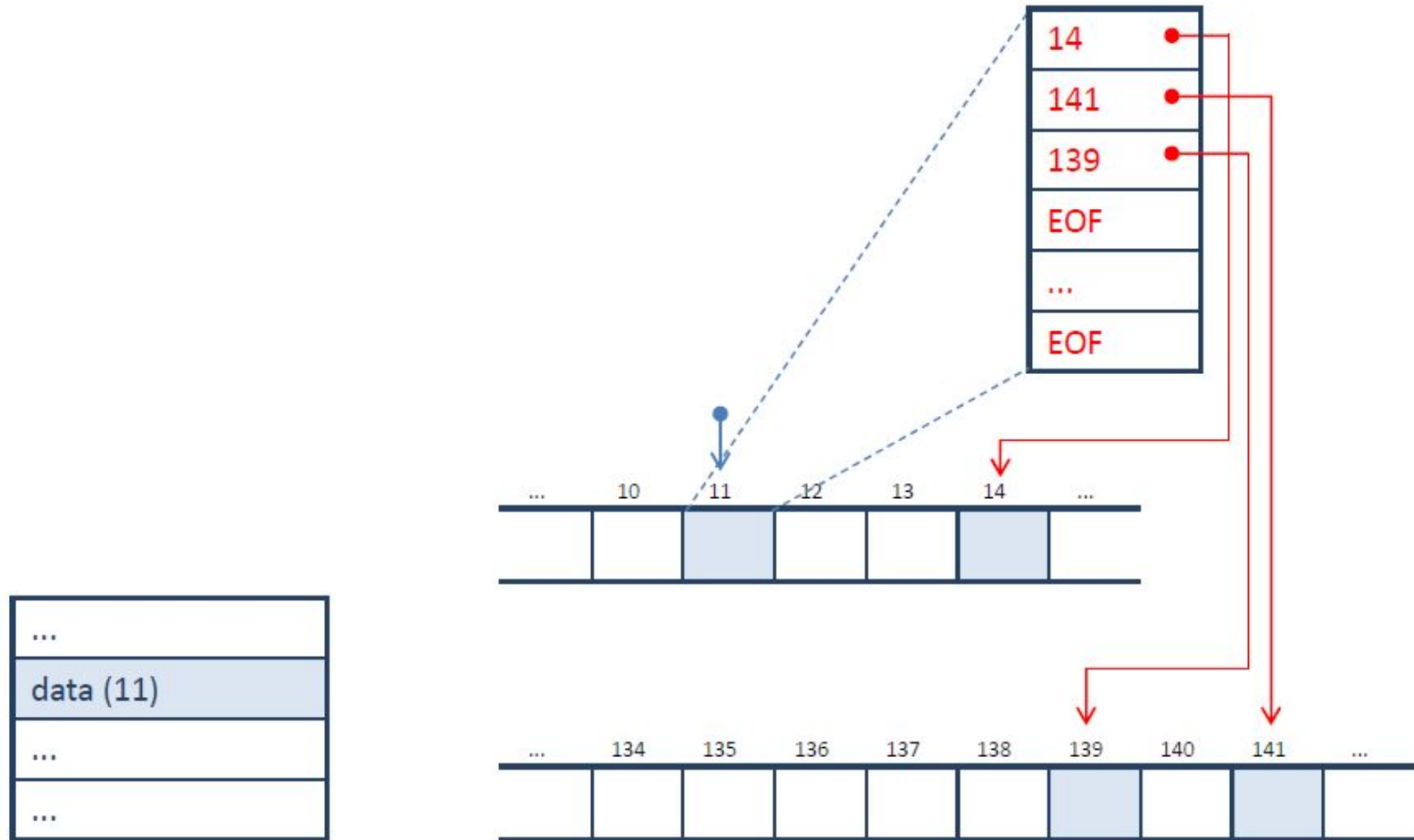
Allocazione concatenata – FAT



■ Esercizio 2

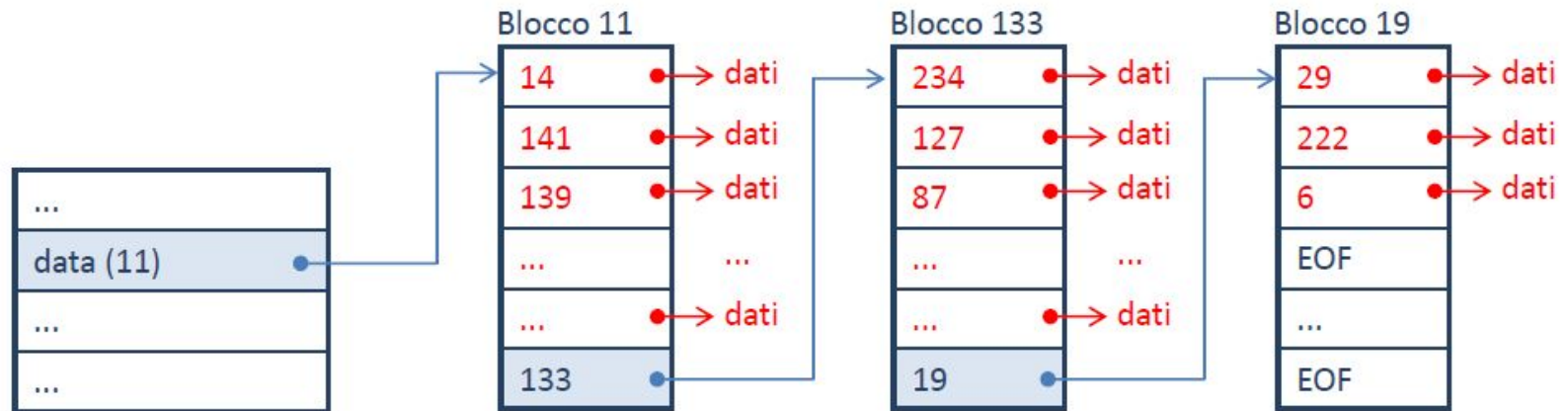
- Si supponga di avere blocchi grandi 1KB e indici di dimensione 4B. Si calcoli il numero di blocchi indirizzabili in accordo con le seguenti politiche:
 - Allocazione indicizzata semplice
 - Allocazione indicizzata concatenata
 - Allocazione indicizzata multilivello (2 e 3 livelli)
 - Allocazione indicizzata combinata (3 livelli, il resto diretto)

Allocazione indicizzata semplice



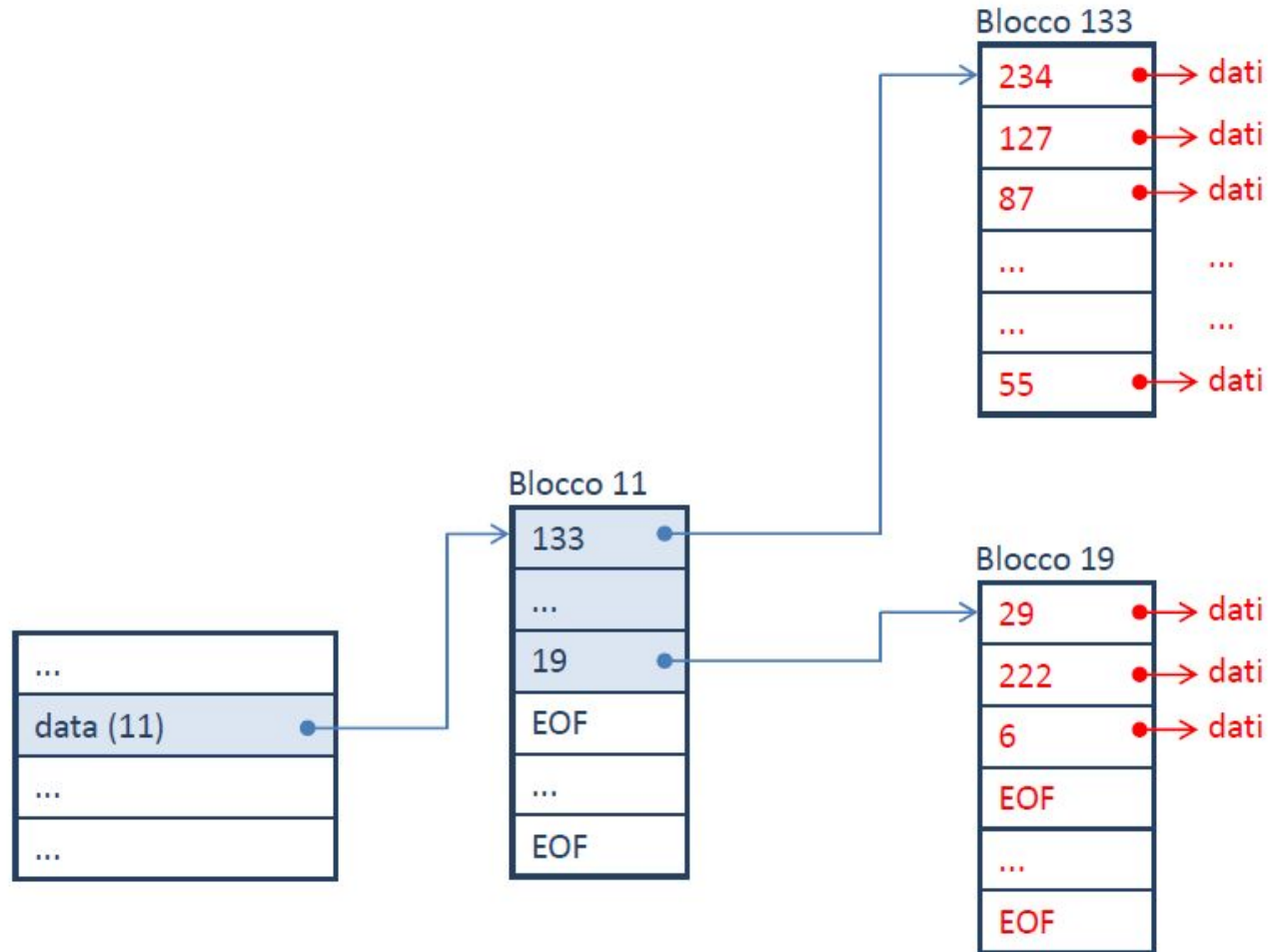
- $\#INDICI = 2^{10}/2^2 = 2^8 = 256$
- $DIMENSIONE\ MASSIMA\ FILE = 256 * 1KB = 256\ KB$

Allocazione indicizzata concatenata



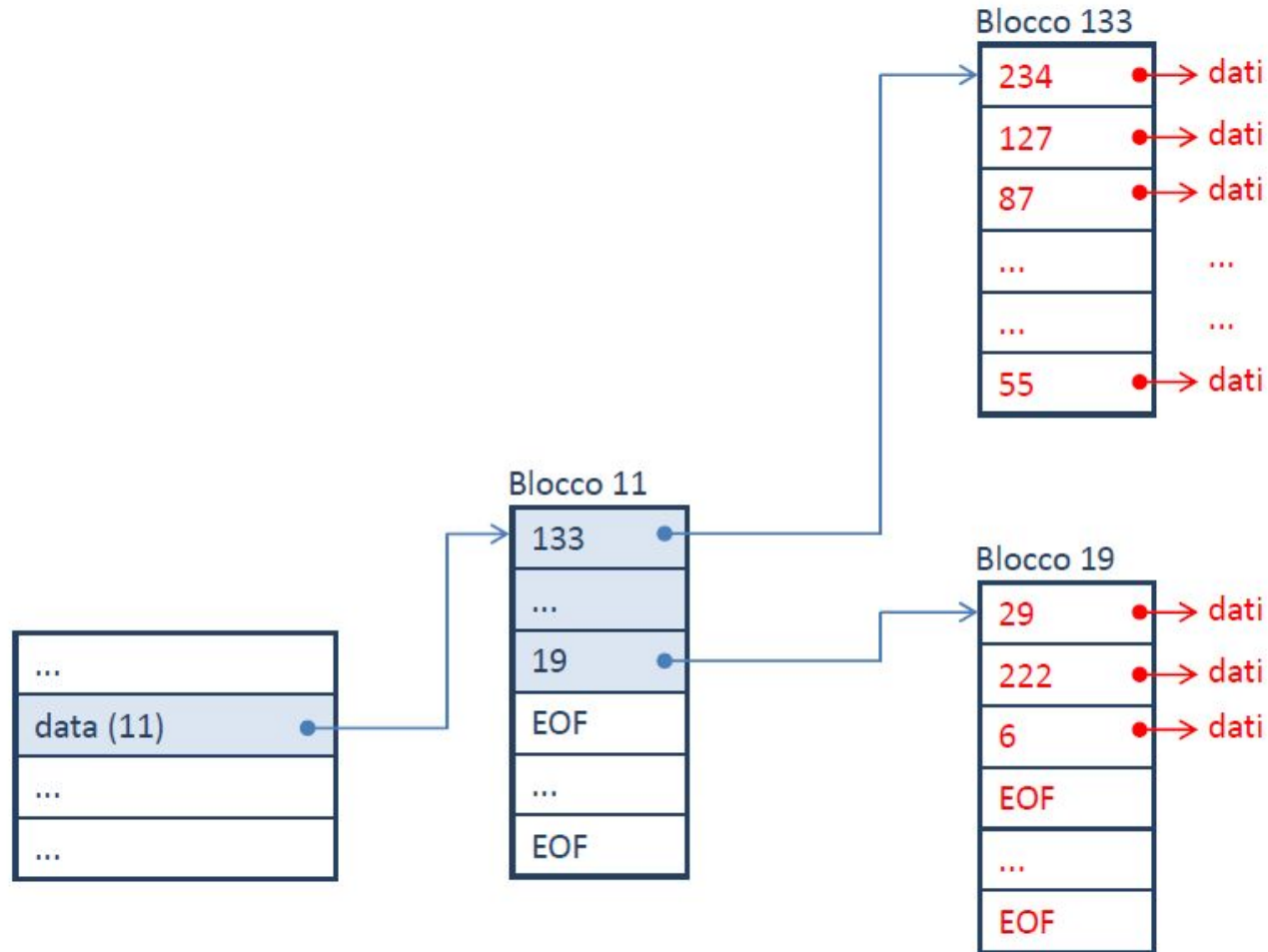
- $\#INDICI_{BLOCCO} = 2^{10}/2^2 - 1 = 2^8 - 1 = 255$
- $\#INDICI = \infty$
- $DIMENSIONE\ MASSIMA\ FILE = \infty$

Allocazione indicizzata multilivello (2 livelli)



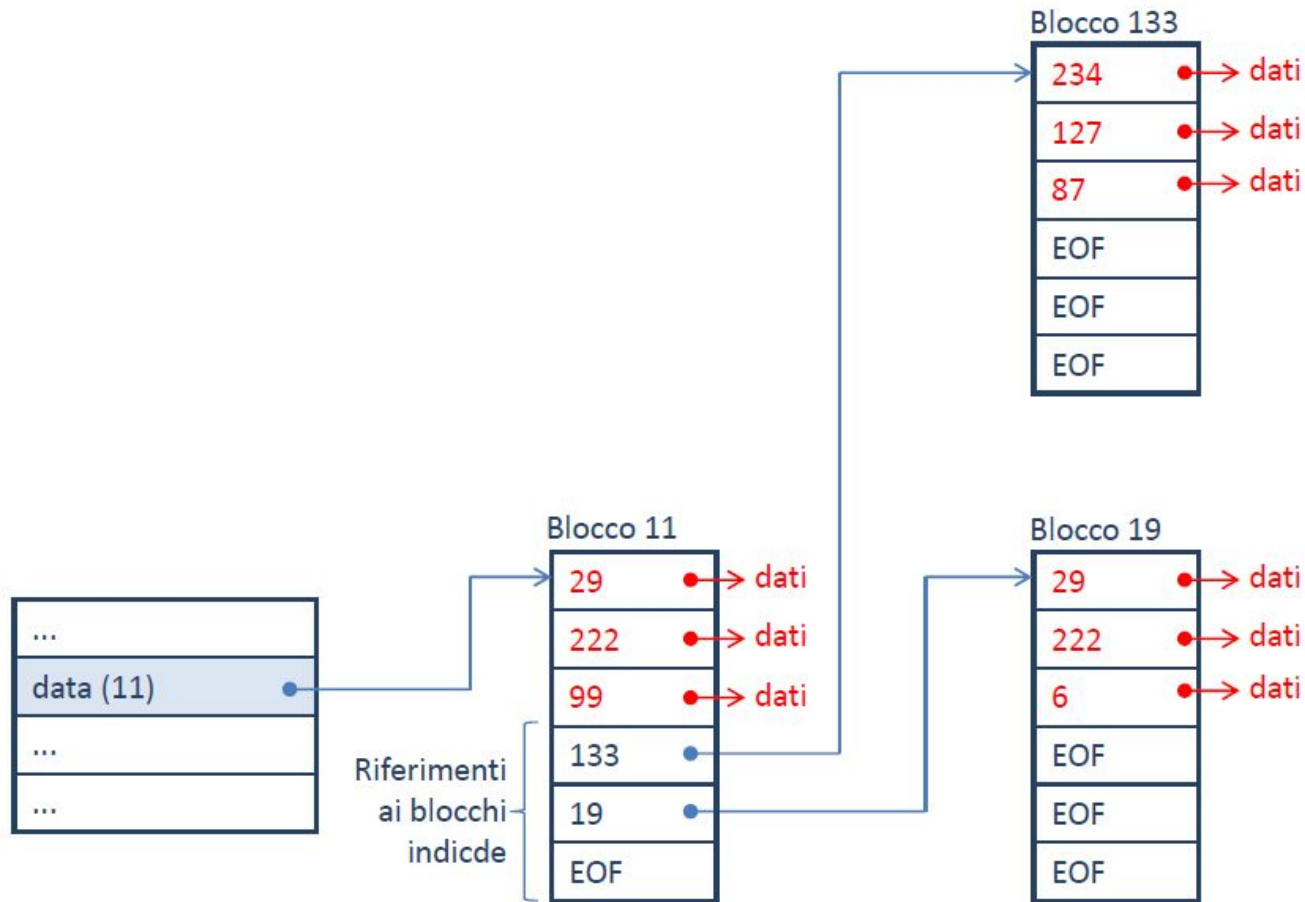
- $\#INDICI = (2^{10}/2^2)^2 = 2^{16} = 65536$
- $DIMENSIONE\ MASSIMA\ FILE = 65536 * 1KB = 64\ MB$

Allocazione indicizzata multilivello (3 livelli)



- $\#INDICI = (2^{10}/2^2)^3 = 2^{24} = 16777216$
- $DIMENSIONE\ MASSIMA\ FILE = 2^{24} * 1KB = 4\ GB$

Allocazione indicizzata combinata



- $\#INDICI_{LIVELLO} = \frac{(2^{10}/2^2)}{2} = 2^7$
- $\#INDICI = 2^7 + 2^7 (2^7 + 2^7 (2^7 + 2^7 2^8)) = 2^7 + 2^{14} + 2^{21} + 2^{29}$
- $DIMENSIONE\ MASSIMA\ FILE = \#INDICI * 1KB \cong 514\ GB$

Filesystem

▪ Esercizio 3

- Si supponga di avere blocchi da 1KB e indici di dimensione 4B.
- Si supponga inoltre di avere un file di 128KB.
- Simulare
 - Accesso sequenziale a tutto il file
 - Accesso casuale alla fine del file
 - Accesso casuale al blocco 127 partendo dal 128 (`lseek(fd,-1024, SEEK_CUR)`)
- Supponendo che il filesystem abbia
 - Allocazione contigua
 - Allocazione concatenata
 - Allocazione indicizzata semplice

Accesso Sequenziale

▪ Allocazione contigua

- $\#BLOCCHI_{FILE} = \frac{2^{17}}{2^{10}} = 2^7 = 128$
- Accedo al filesystem (blocchi (200,128))
- Accedo al blocco dati 200
- Finito l'accesso al blocco dati 200 carico il blocco dati 201
 - Non controllo più il filesystem perché so che i blocchi sono tutti contigui
- In totale faccio solo 128 accessi ai blocchi (da blocco 200 a 327)

▪ Allocazione concatenata

- $\#BLOCCHI_{FILE} = \frac{2^{17}}{2^{10}-2^2} = 128.5 \rightarrow 129$ perché devo memorizzare l'indice
- Accedo al filesystem (primo blocco (200))
- Accedo al blocco dati 200
- Finito l'accesso ai dati del blocco dati 200 leggo il numero del blocco dati successivo
 - Non controllo più il filesystem perché il blocco precedente mi dice quale è quello successivo
- In totale faccio solo 129 accessi ai blocchi

Accesso Sequenziale

- **Allocazione indicizzata semplice**

- $\#BLOCCHI_{FILE} = \frac{2^{17}}{2^{10}} = 2^7 = 128$

- $\#INDICI_{BLOCCO} = \frac{2^{17}}{2^2} = 2^{15} > 128$

- Accedo al filesystem (blocco indice (200))

- Accedo al blocco indice 200

- Accedo al blocco primo dati

- Finito l'accesso al primo blocco dati carico accedo al blocco indice per sapere quale è il successivo

- In totale faccio 256 accessi ai blocchi (128 indice + 128 dati)

Accesso Casuale Fine

▪ Allocazione contigua

- Accedo al filesystem (blocchi (200,128))
- Calcolo il numero del blocco finale (200+128)
- Accedo al blocco dati 327 (il 128 blocco sequenziale)
- In totale faccio solo 1 accesso

▪ Allocazione concatenata

- Accedo al filesystem(primo blocco (200))
- Accedo al blocco dati 200
- Recupero il numero del blocco successivo
- Termino quando ho fatto passare tutti i blocchi e sono arrivato al 129 blocco
- In totale faccio 129 accessi ai blocchi

Accesso Casuale Fine

- **Allocazione indicizzata semplice**
 - Accedo al filesystem (blocco indice (200))
 - Accedo al blocco indice 200
 - Accedo all'ultimo blocco dati nel blocco indice
 - In totale faccio 2 accessi ai blocchi (1 indice + 1 dati)

Accesso Casuale Precedente

- Devo accedere al blocco $-1024/1024 = -1$ rispetto all'attuale
- Calcolo il numero del blocco del file $128-1 = 127$
- **Allocazione contigua**
 - Sono al blocco 327
 - Accedo direttamente al blocco dati $327 - 1 = 326$
 - In totale faccio solo 1 accesso
- **Allocazione concatenata**
 - Accedo al filesystem (primo blocco (200))
 - Accedo al blocco dati 200
 - Finito l'accesso ai dati del blocco dati 200 leggo il numero del blocco dati successivo fino a che non arrivo al 127esimo blocco del file
 - Non controllo più il filesystem perché il blocco precedente mi dice quale è quello successivo
 - In totale faccio 127 accesso ai blocchi
 - **ATTENZIONE:** Devo sempre passare dal filesystem per un "accesso all'indietro"

Accesso Casuale Precedente

- **Allocazione indicizzata semplice**
 - Accedo al filesystem (blocco indice (200))
 - Accedo al blocco indice 200
 - Accedo al blocco 127 del blocco indice
 - In totale faccio 2 accessi ai blocchi (1 indice + 1 dati)