File System

- Cos'è un File System
- File e Directory
- Implementazione di un File System

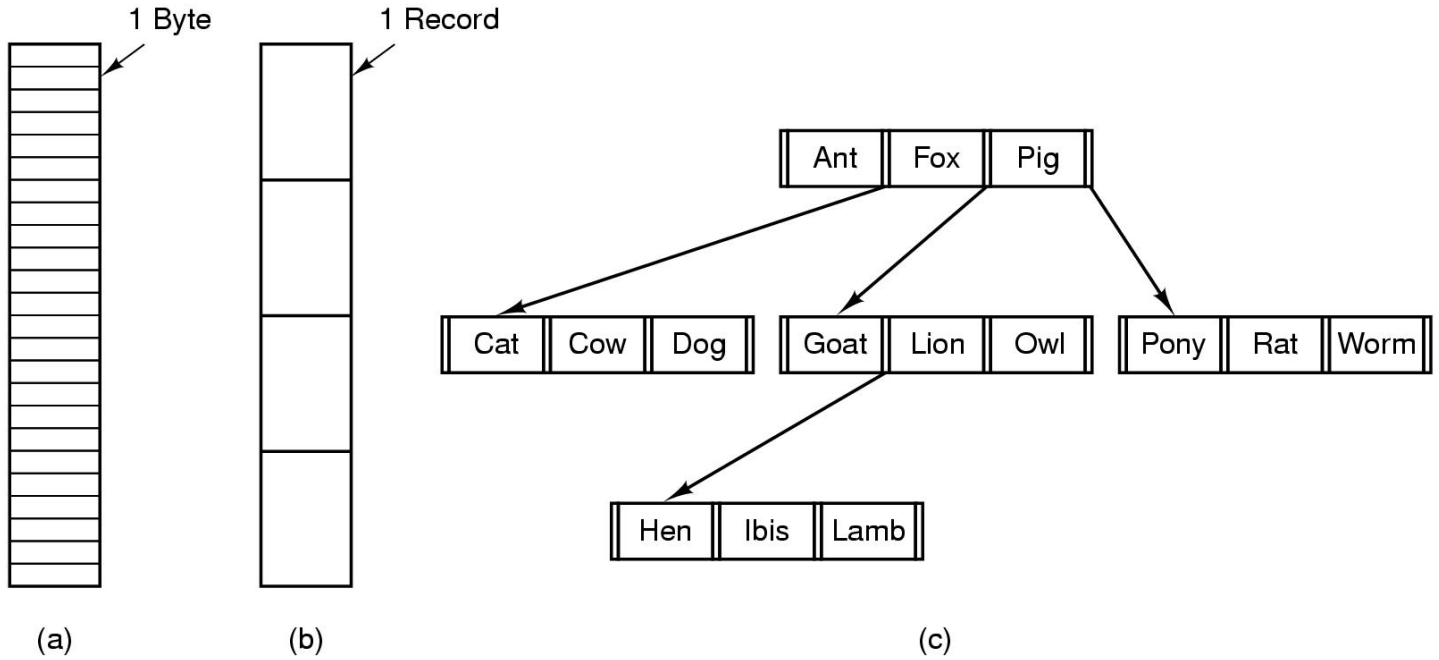
Memorizzazione di informazione a lungo termine

1. È necessario memorizzare grandi quantità di dati
2. È necessario che l'informazione memorizzata sopravviva alla terminazione del processo che l'ha generata (*persistenza*)
3. Più processi devono poter accedere concorrentemente all'informazione memorizzata

Il File System

- È la parte del SO che si occupa di memorizzare informazioni in modo persistente in memoria secondaria
- *File* : unità di informazione memorizzata in modo persistente
- *Directory* - *Folder* : astrazione che permette di raggruppare assieme più file

Struttura di un File



- Come può essere strutturata l'informazione all'interno di un file
  - sequenze di byte, sequenze di record , alberi con chiave

Accesso ai File

- Accesso diretto (*random*)
  - i byte/record possono essere letti in qualsiasi ordine
  - una *read* può essere specificata …
    - specificando la posizione del dato da accedere ad ogni chiamata, …
    - usando una speciale operazione (la *seek*) per posizionare la testina prima di iniziare più letture
  - nei moderni sistemi operativi tutti i file sono automaticamente ad accesso diretto

Attributi di un file

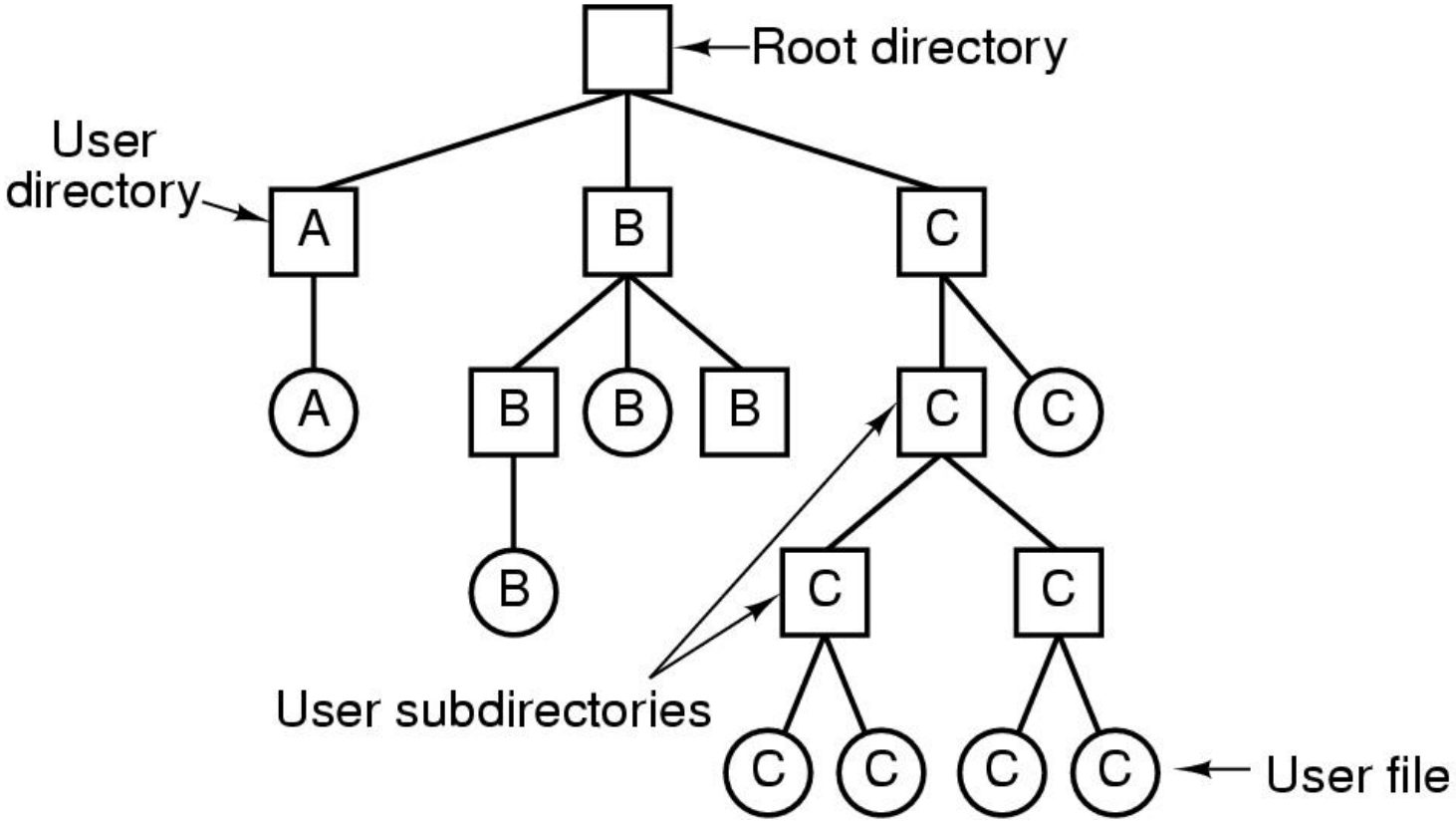
Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

- File = nome + dati + attributi
- Tipici attributi di un file

Operazioni su File

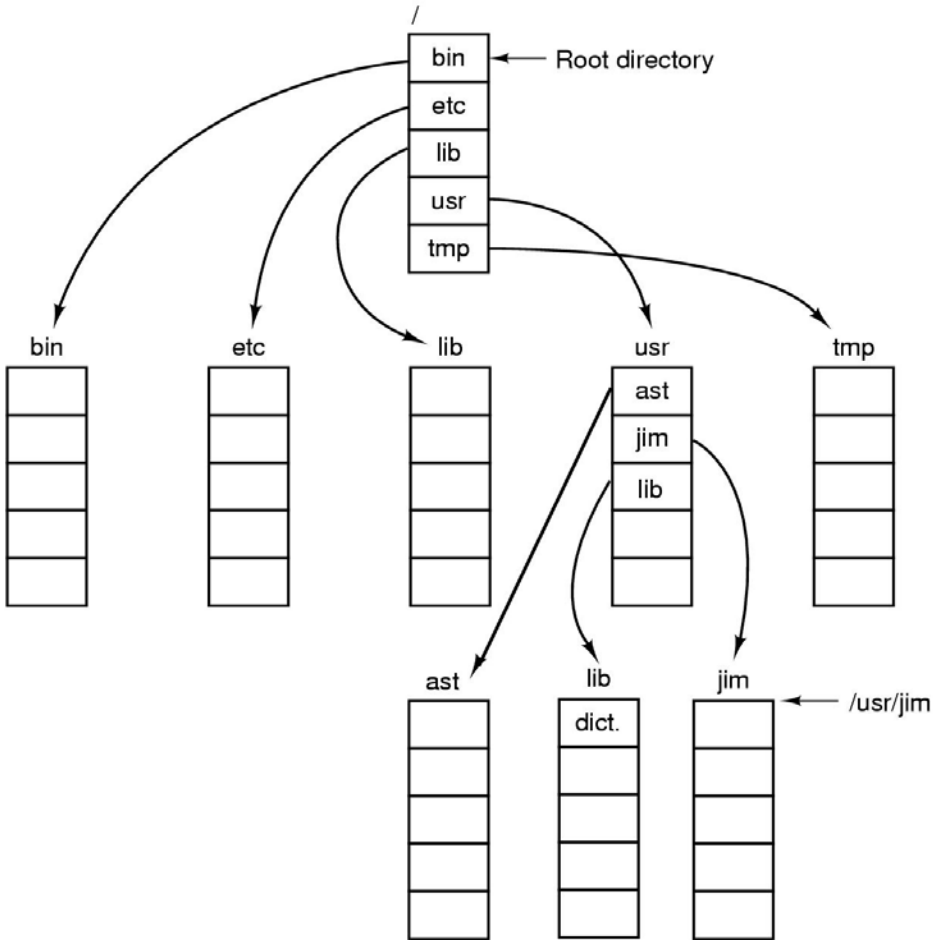
1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

Sistemi con directory gerarchiche



Una gerarchia di directory

I *Path Name*



- Come vengono specificati i file in UNIX
- */usr/lib/dictionary* (ass.) *../lib/dictionary* (rel.)

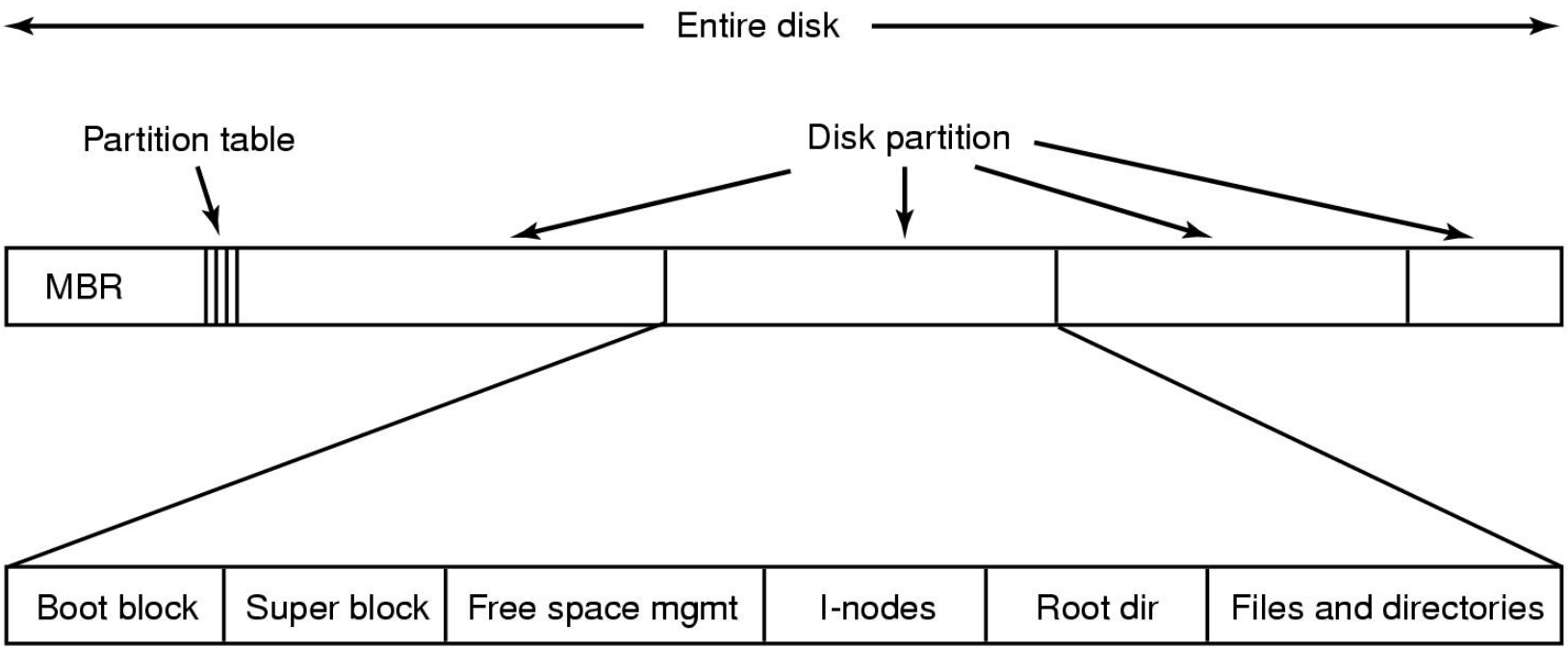
Operazioni sulle directory

1. Create
2. Delete
3. Opendir
4. Closedir
5. Readdir
6. Rename
7. Link
8. Unlink

Implementazione di un File System (1)

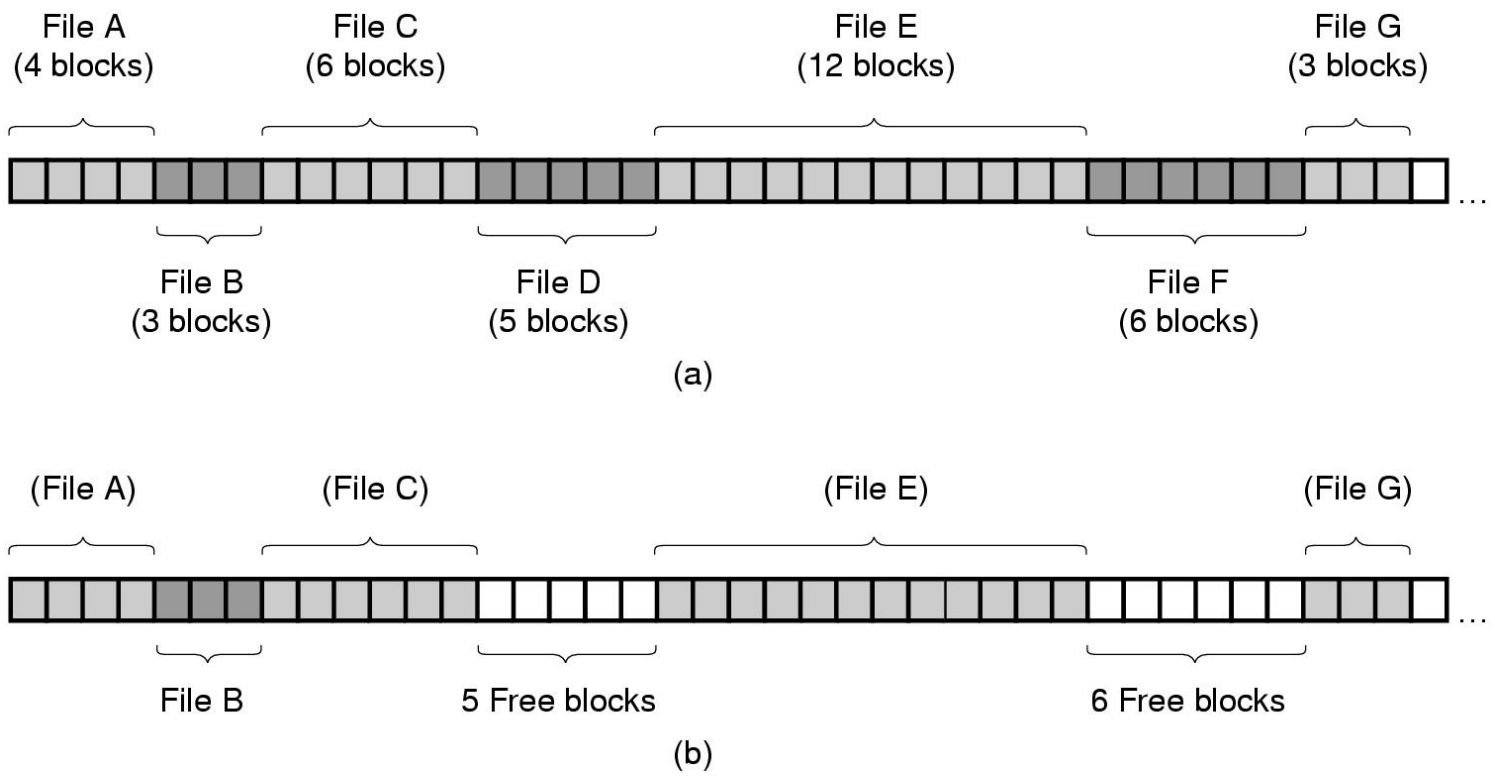
- Come rappresentare i file
  - i dati sono memorizzati in unità (*blocchi*) di ampiezza fissa (tipicamente 1,2 KB)
  - si devono memorizzare gli attributi e la posizione dei singoli blocchi
- Come rappresentare le directory
  - generalmente sono file con uno speciale formato
- Come organizzare lo spazio disco
  - allocazione dei blocchi relativi ad un singolo file
  - gestione blocchi liberi
  - tenere traccia della root directory

Implementazione di un File System (2)



Possibile organizzazione un File System su disco

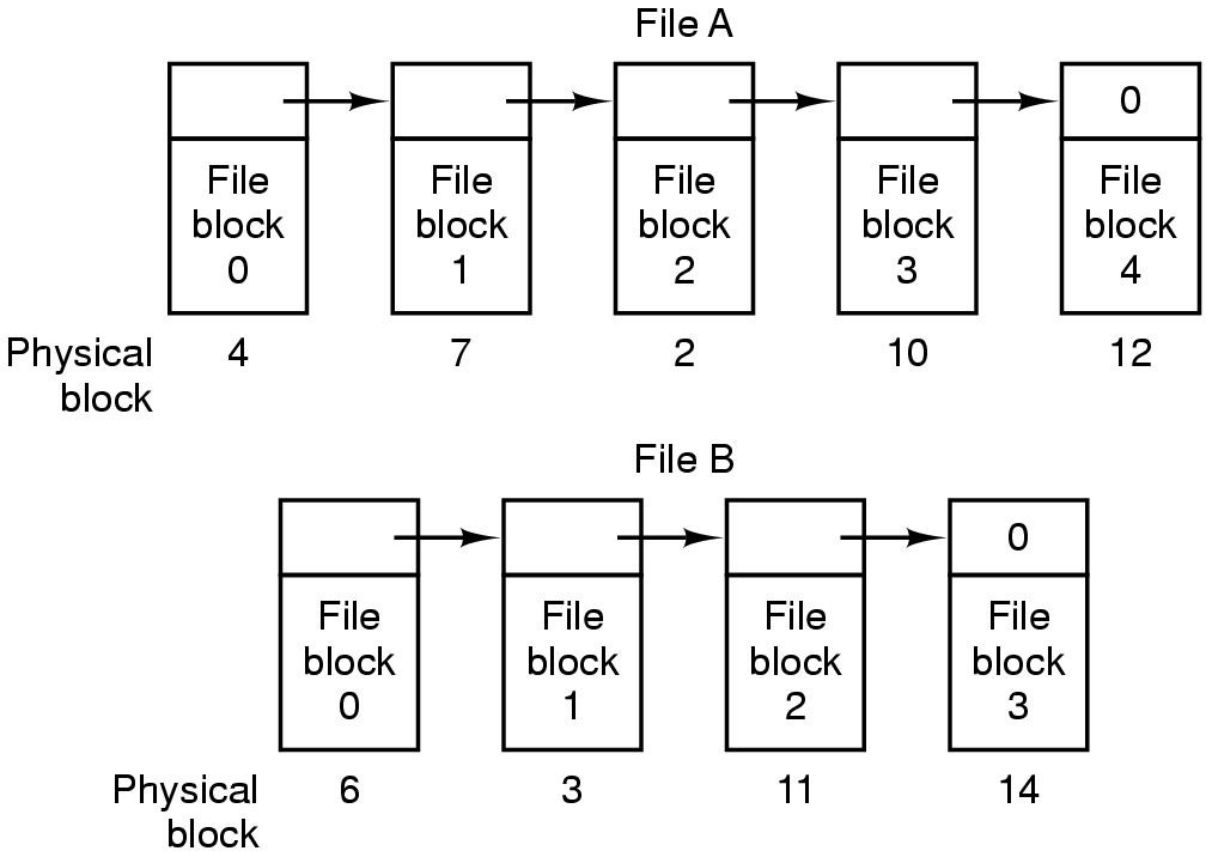
Implementazione dei File (1)



(a) *Allocazione contigua* dello spazio disco per 7 file

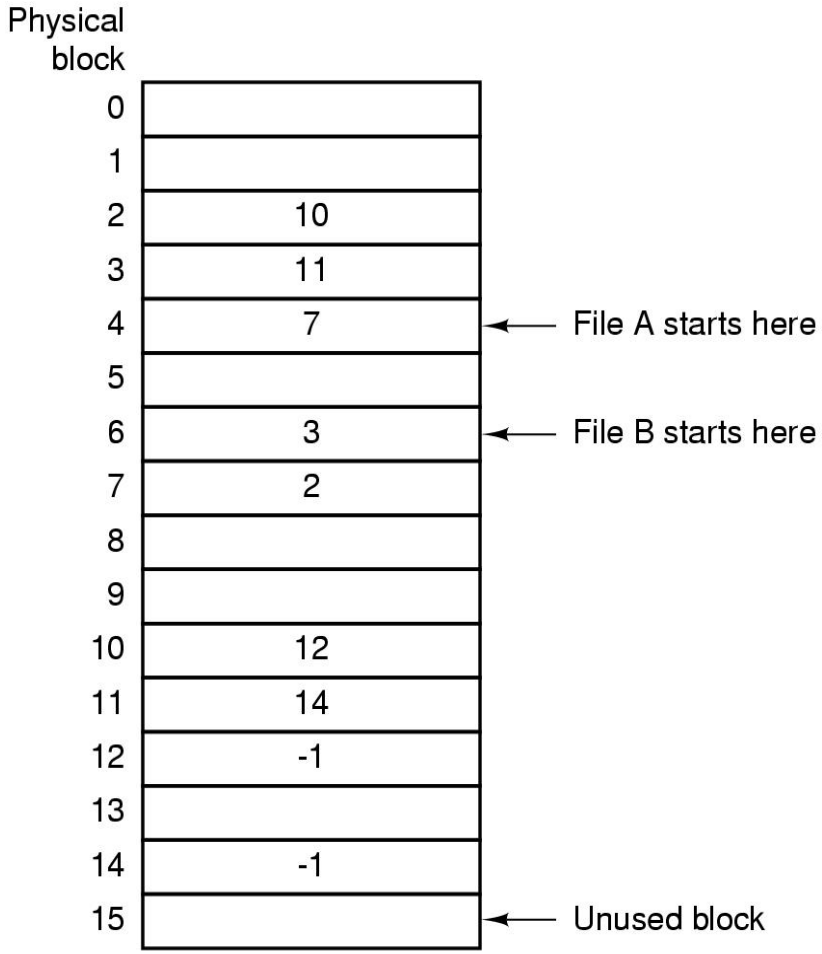
(b) Stato del disco dopo la rimozione di *D* ed *E*

Implementazione dei File (2)



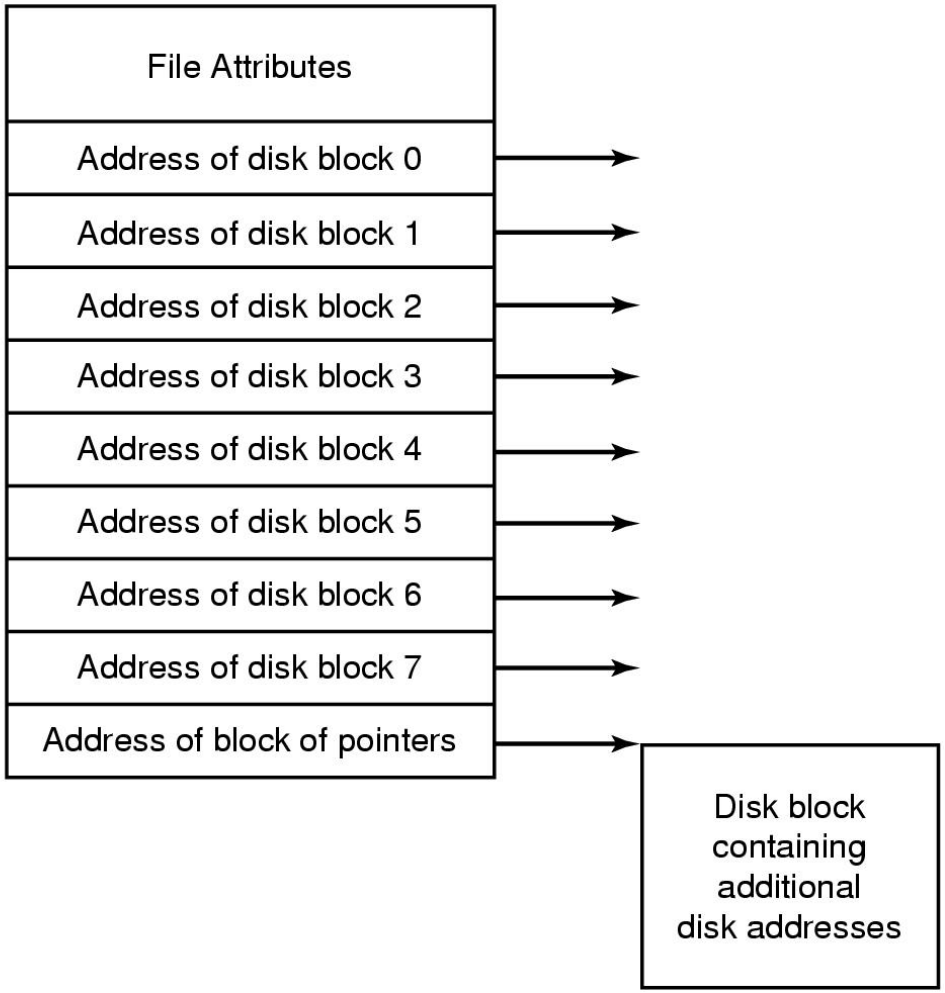
Memorizzazione come lista concatenata di blocchi

Implementazione dei File (3)



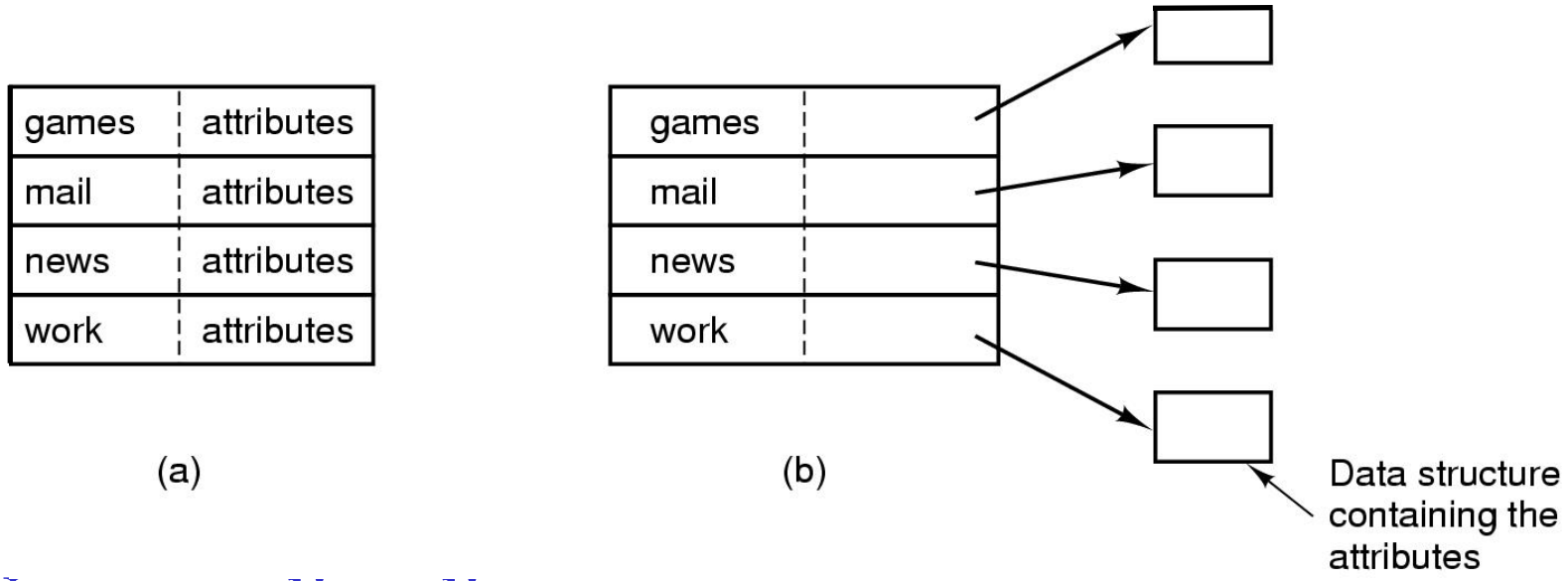
Allocazione con lista concatenata che utilizza una *file allocation table* (FAT) nella RAM

Implementazione dei File (4)



Un esempio di *i-node*

Implementazione delle Directory (1)

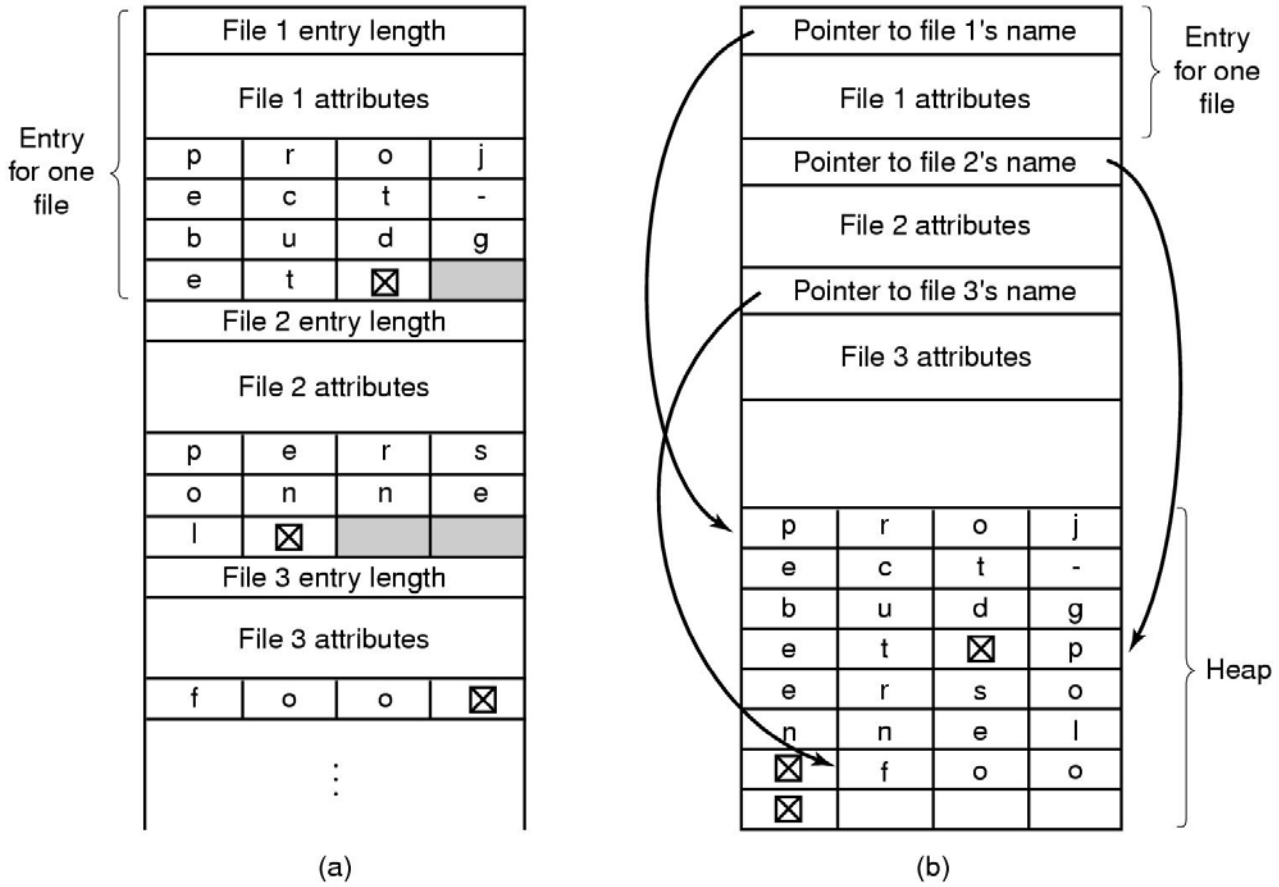


(a) Una semplice directory

- elementi di ampiezza fissa
- attributi e indirizzi su disco tutti memorizzati in un singolo elemento

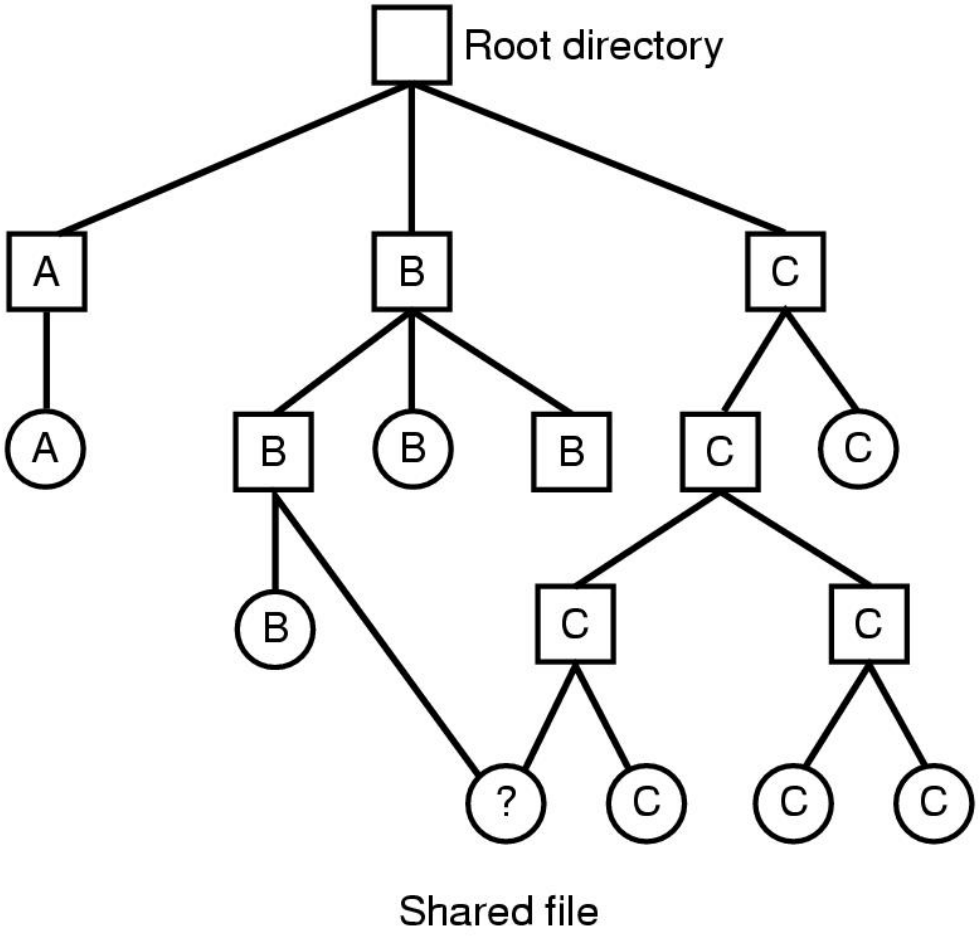
(b) Una directory che contiene solo i puntatori agli *i-node*

Implementazione delle Directory (2)



- Due modi di trattare i nomi di file "lunghi"
  - (a) In linea
  - (b) In un heap

File condivisi (1)

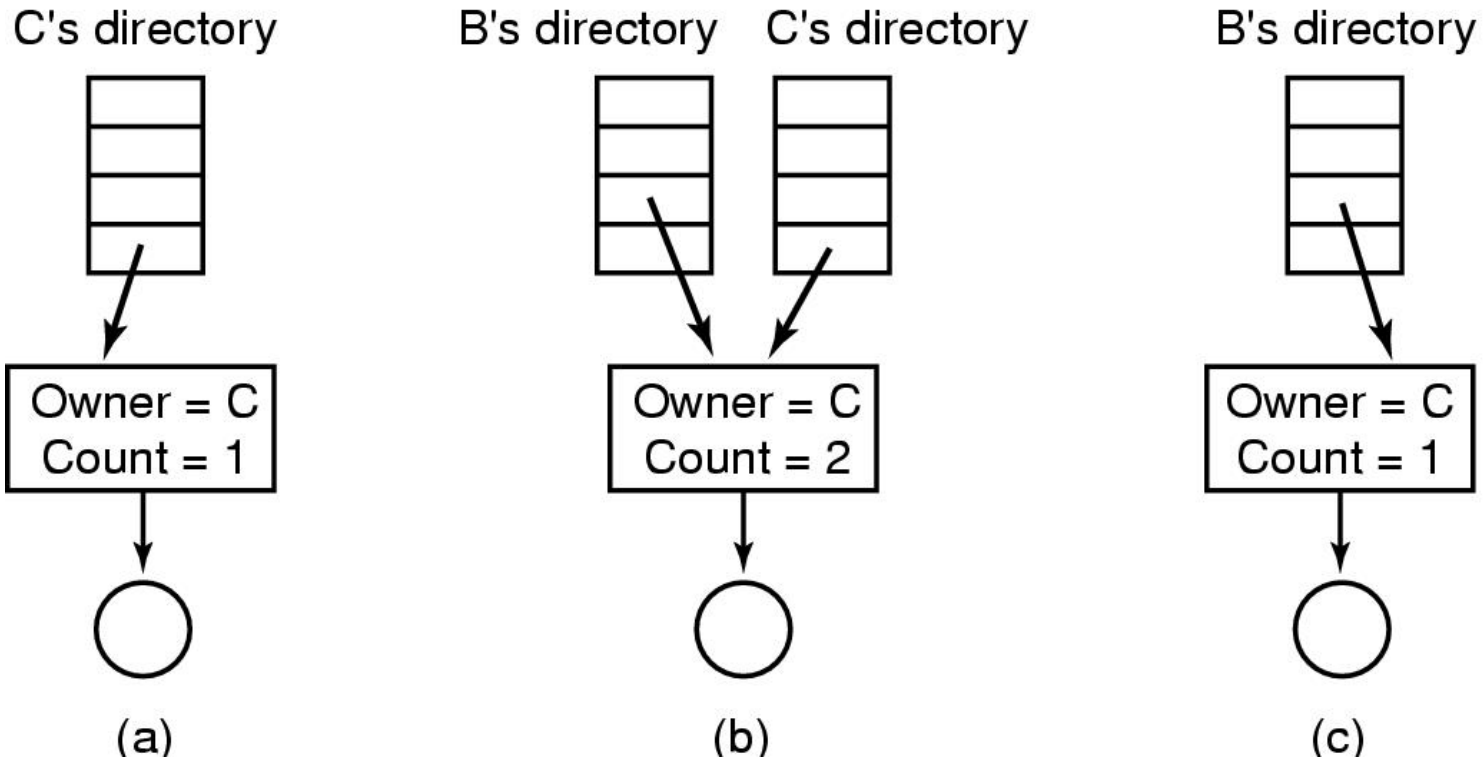


Un file system contenente un file condiviso

File condivisi (2)

- Hard link :
  - le due directory condividono la struttura dati relativa al file
  - paradosso della rimozione da parte dell'owner
- Symbolic Link :
  - la seconda directory contiene un file speciale (LINK) con il path name del file condiviso
  - accesso più lento (il path name deve essere seguito ogni volta che accediamo al file)

File condivisi (3)

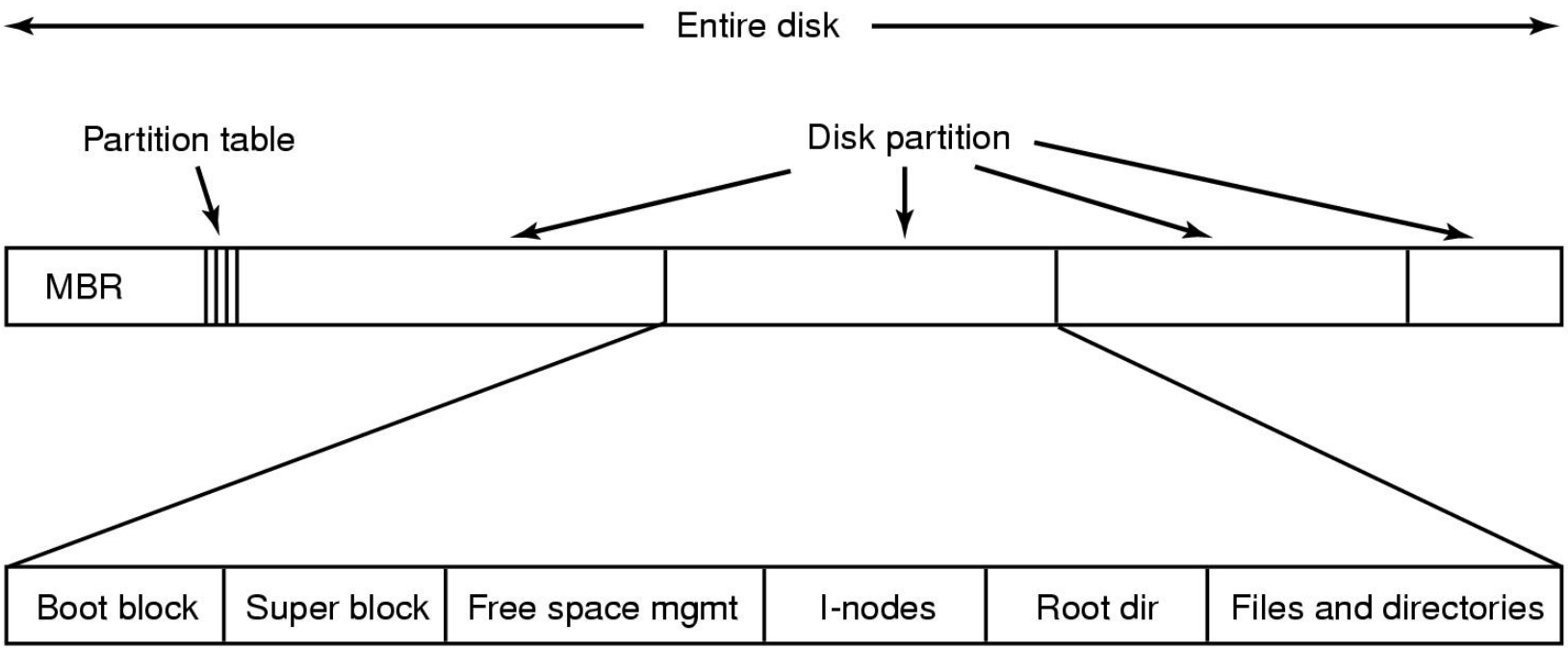


(a) situazione precedente al linking (hard)

(b) dopo la creazione del link

(c) dopo che l'*owner* originale ha rimosso il file

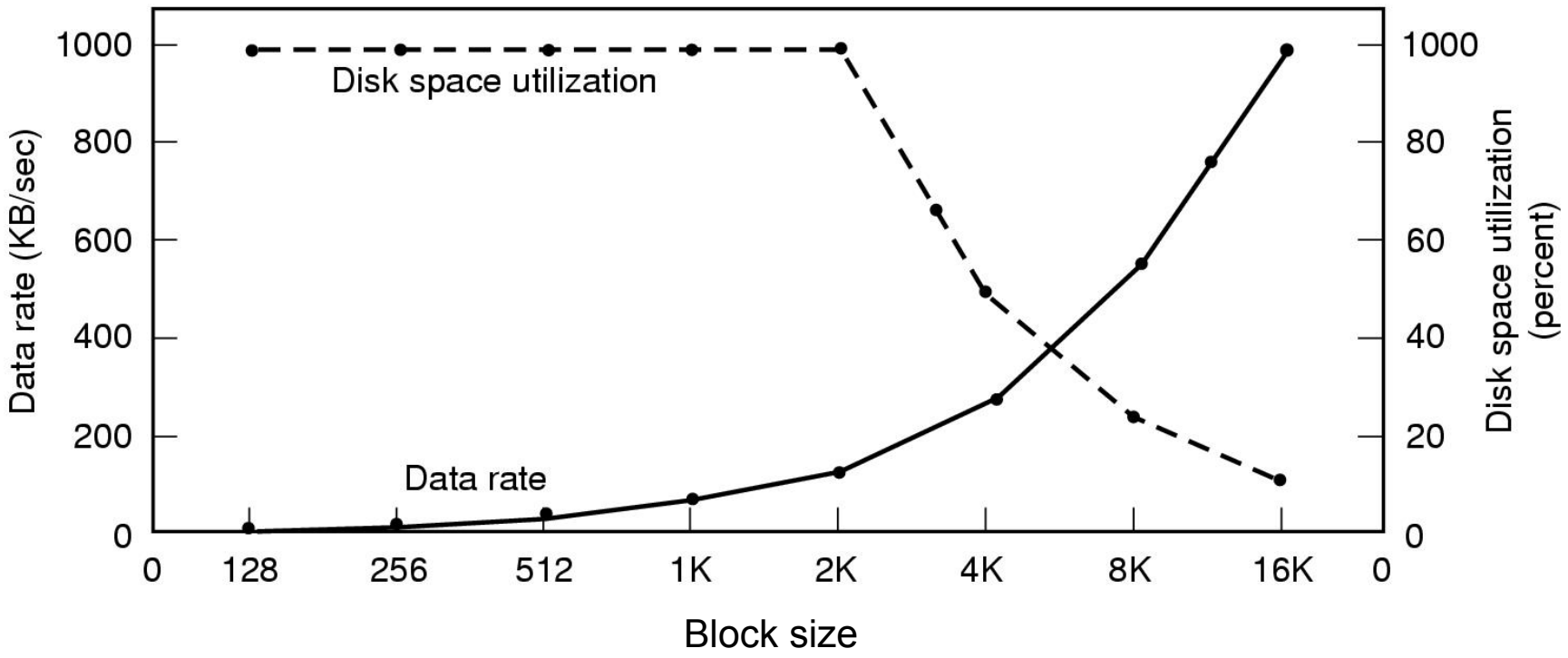
Implementazione di un File System (2)



Possibile organizzazione un File System su disco

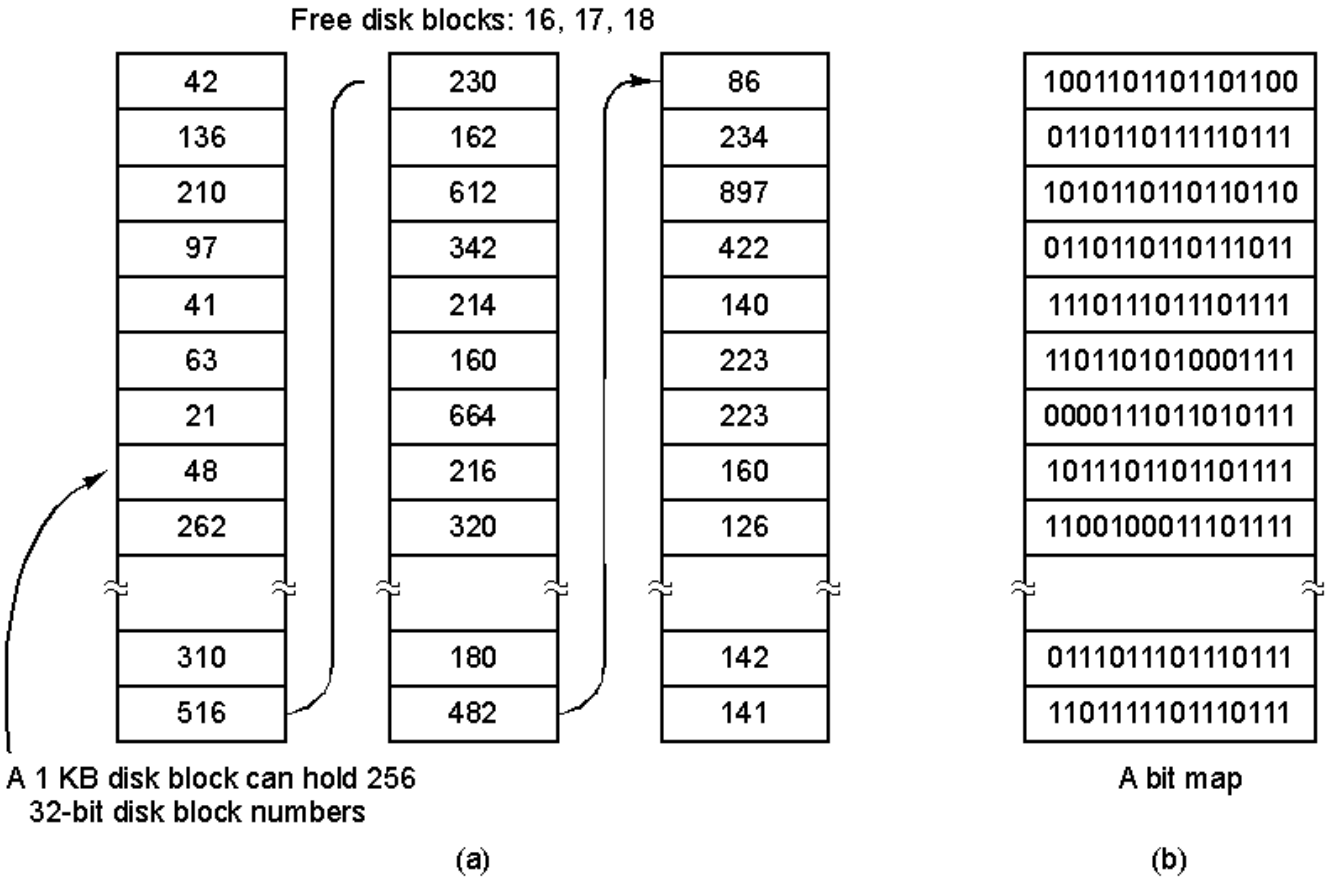
Gestione dello spazio disco (1)

Scelta dell'ampiezza dei blocchi



- La linea intera (scala a sin.) fornisce la velocità del disco
- La linea tratteggiata (scala a ds.) fornisce l'efficienza nell'utilizzo dello spazio disco
- Tutti i file sono di 2KB

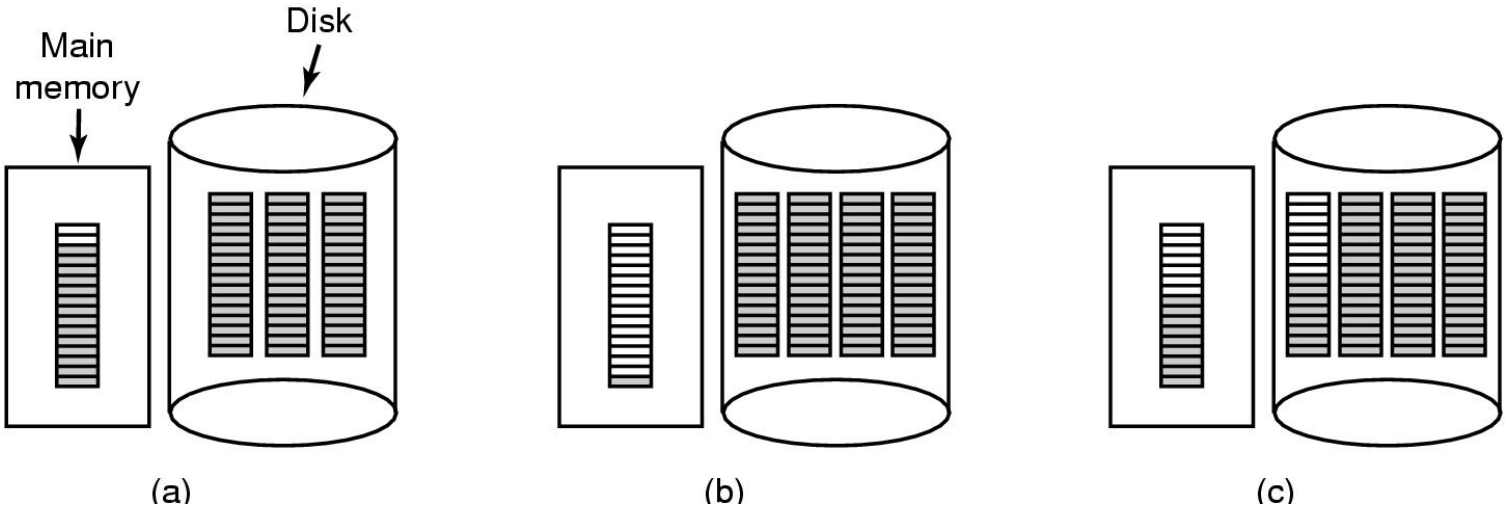
Gestione dello spazio disco (2)



(a) memorizzazione della *lista libera* come lista concatenata

(b) memorizzazione come bitmap

Gestione dello spazio disco (3)



(a) blocco di puntatori ai blocchi liberi quasi pieno (RAM)

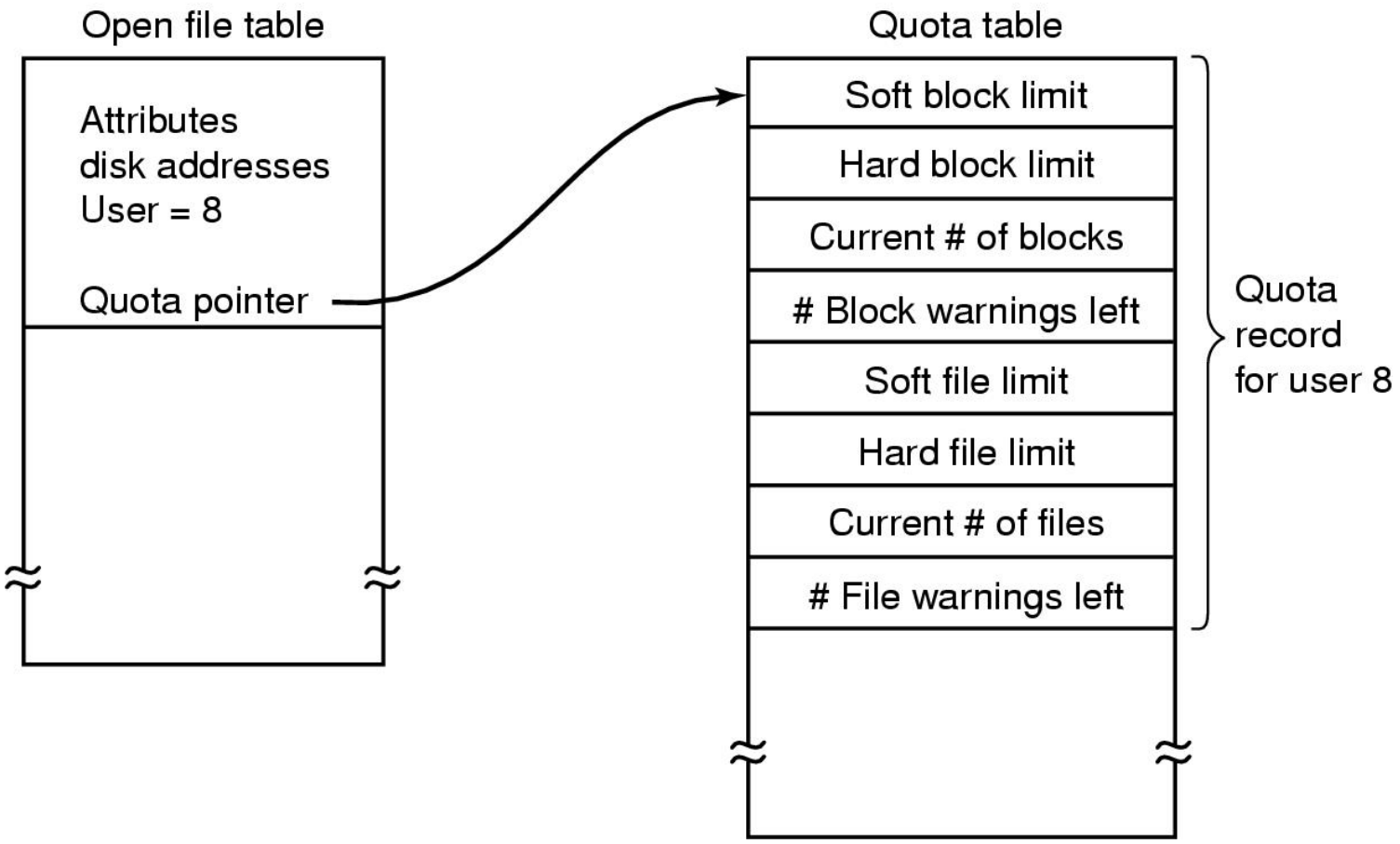
- tre blocchi di puntatori su disco

(b) situazione dopo aver liberato un file di 3 blocchi

(c) strategia alternativa per gestire i 3 blocchi

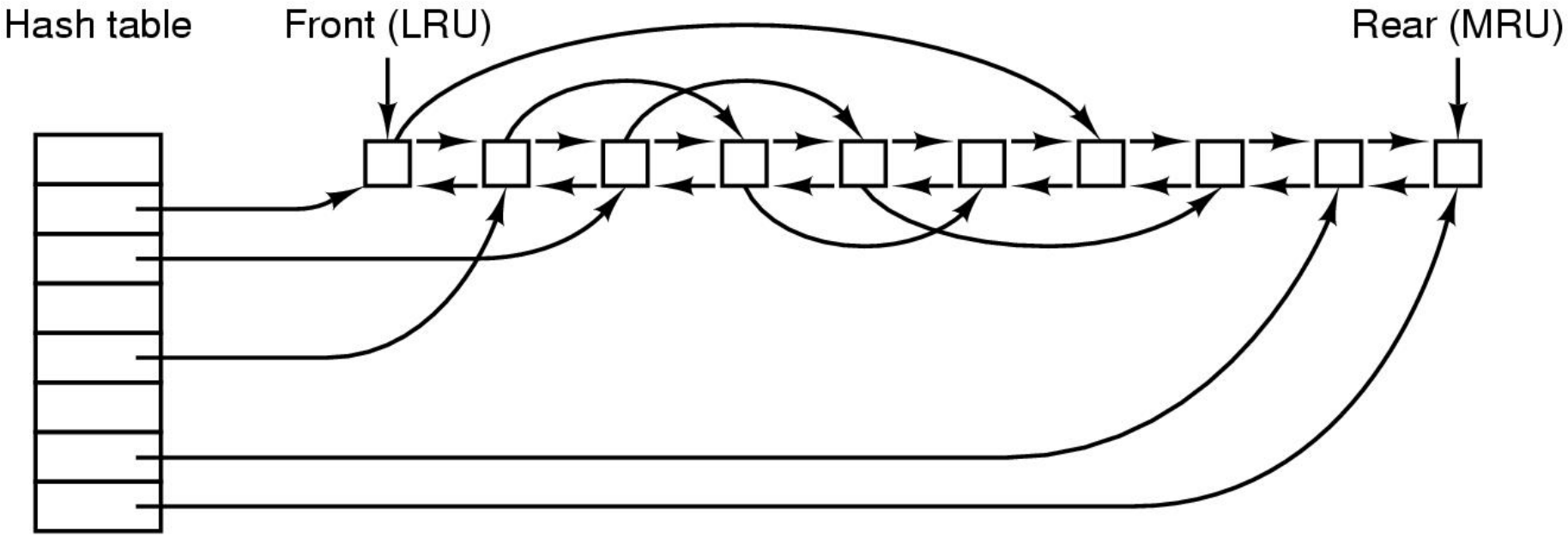
- gli elementi in grigio puntano a blocchi di disco liberi

Gestione dello spazio disco (4)



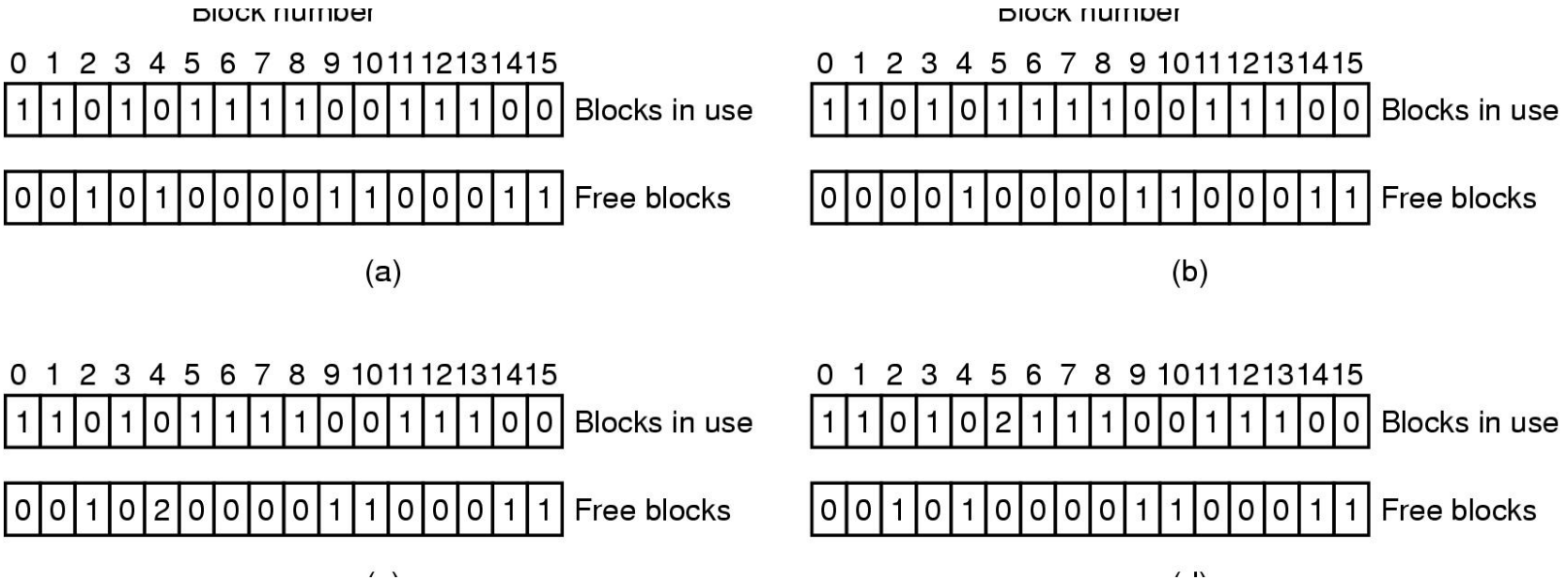
Il meccanismo delle *quote* per tener traccia dello spazio disco utilizzato da ciascun utente

Prestazioni di un File System (1)



- Strutture dati per il cache dei blocchi
- Hash su *dispositivo::indirizzo del blocco*
- I blocchi critici per la consistenza del FS vengono scritti subito (i-node, directory, lista libera)

Consistenza di un File System (1)



- Stati di un file system
  - (a) consistente (*consistent*)
  - (b) blocco mancante (*missing* block)
  - (c) blocco duplicato nella lista libera (*duplicate free* block)
  - (d) blocco dati duplicato (*duplicate data* block)

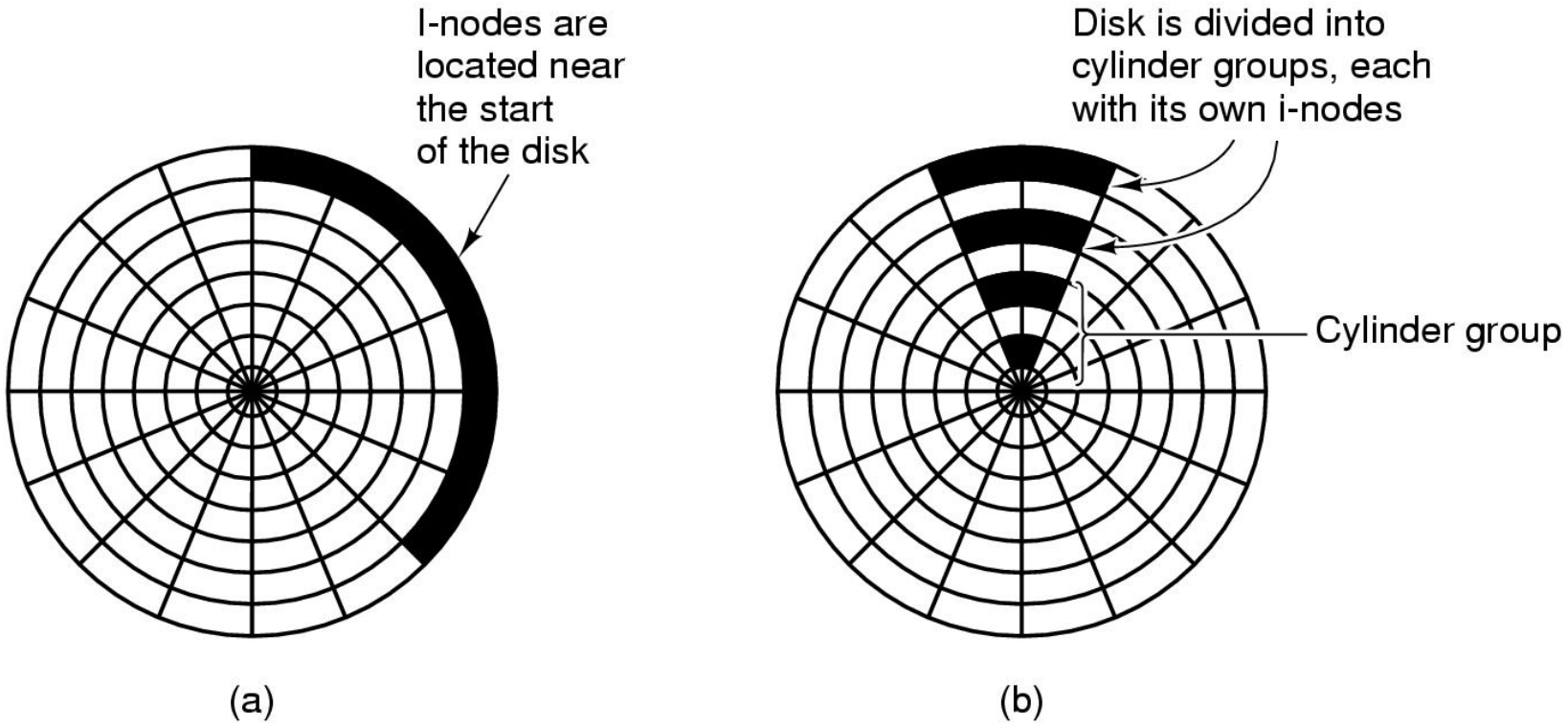
Consistenza di un File System (2)

- Funzionamento di *fsck*
- *Consistenza dei blocchi*
  - scansione i-node e blocchi liberi
  - costruzione tabella blocchi liberi e tabelli blocchi in uso
    - missing block - viene aggiunto alla lista libera
    - duplicate free block - viene ricostruita la lista libera
    - duplicate data block - viene duplicato il blocco per avere una copia diversa in ciascun file
- *Consistenza delle directory*
  - scansione delle directory
  - costruzione tabella di occorrenza file
  - controllo di consistenza fra la tabella di occorrenza ed il conto degli hard link nell'i-node
    - se differiscono si modifica l'i-node

Prestazioni di un File System (2)

- Lettura anticipata (read ahead)
  - si controlla il pattern di accesso del disco e se ne tiene traccia nell'i-node
  - se l'accesso è sequenziale si leggono in anticipo i prossimi blocchi della sequenza e si memorizzano nella cache
- Si cerca di allocare blocchi di disco "vicini" per blocchi logici vicini di uno stesso file
- Si ottimizza l'allocazione degli i-node

Prestazioni di un File System (3)



- Gli *i-node* sono piazzati all'inizio del disco
- Il disco e diviso in gruppi di cilindri
  - ognuno con i suoi blocchi ed i suoi *i-node*