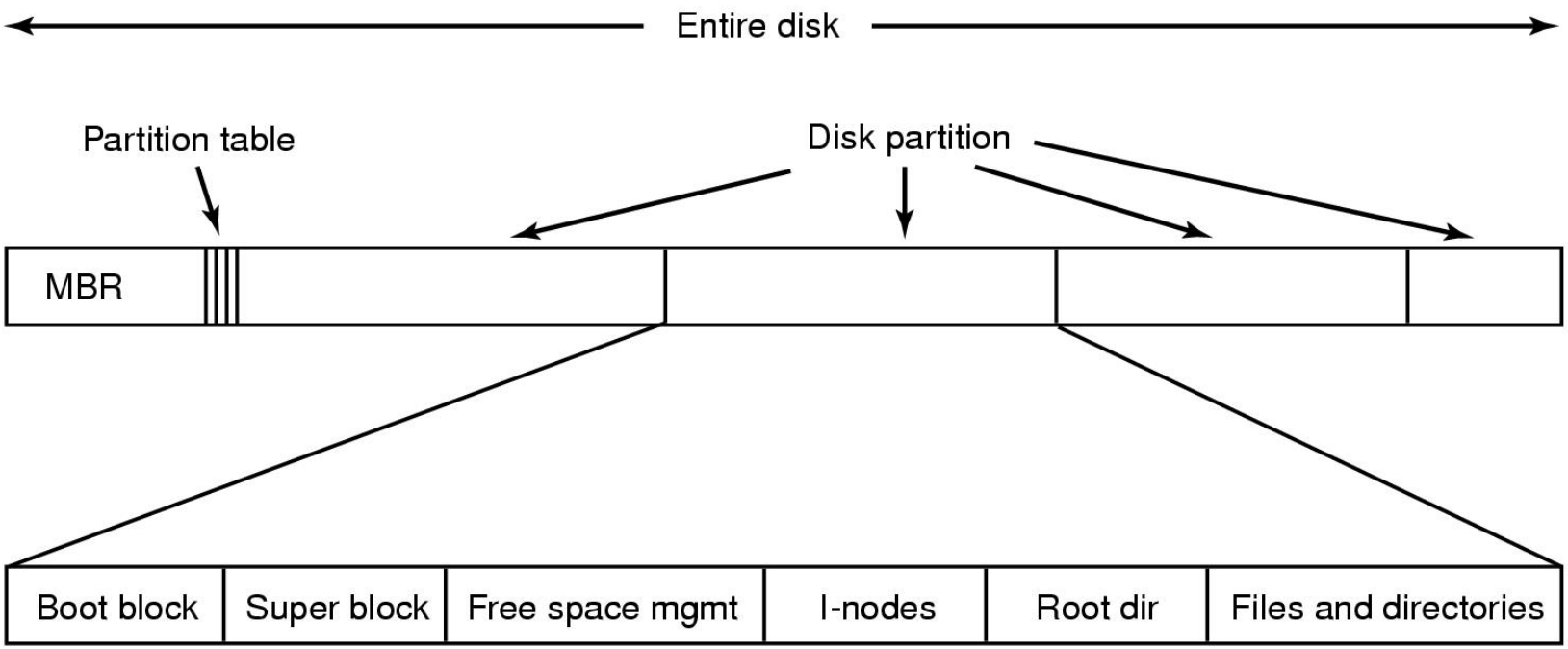
Sistemi Operativi

Lez. 14
File System: aspetti implementativi

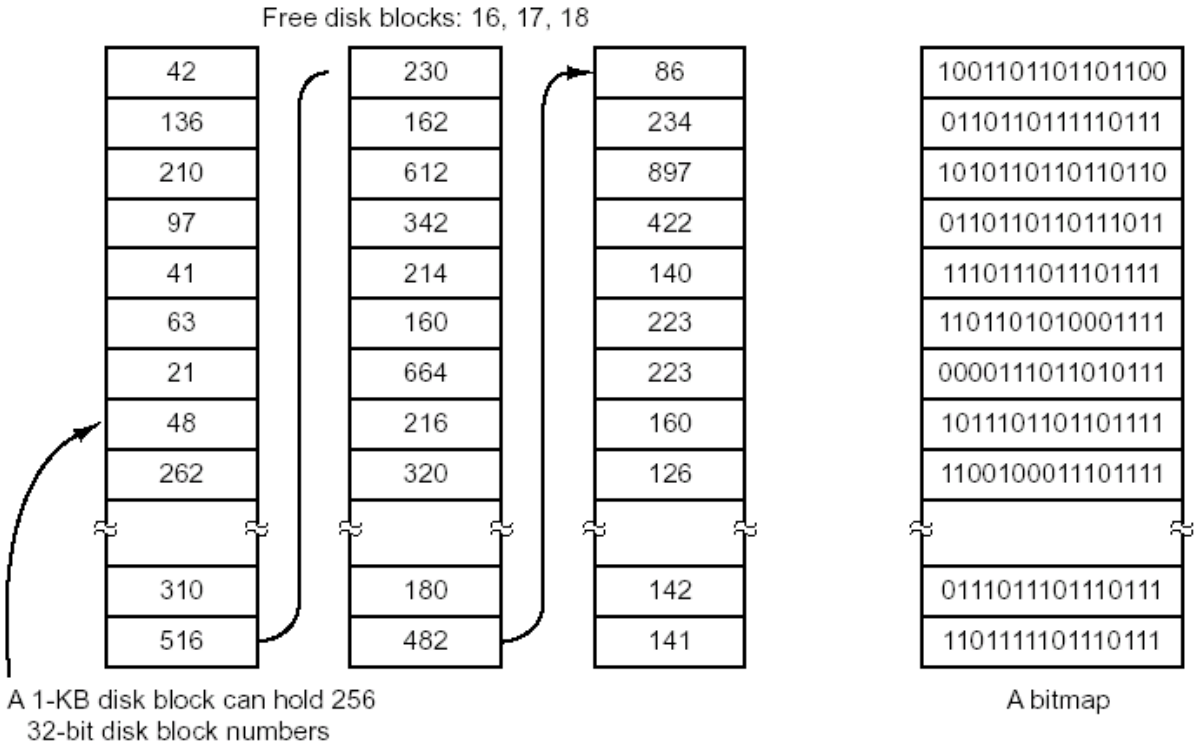
Layout disco

- Tutte le informazioni necessarie al file system per poter operare, sono memorizzate sul disco di boot
  - MBR: settore 0 del disco, contiene la tabella delle partizioni con indicazione delle partizioni attive
  - Boot block: primo settore della partizione attiva, contiene un programma che carica il sistema operativo
  - Superblock: contiene i parametri di inizializzazione per il file system
  - Free Blocks: elenco dei blocchi liberi
  - I-nodes: insieme degli i-node

File System Unix



Gestione spazio su disco

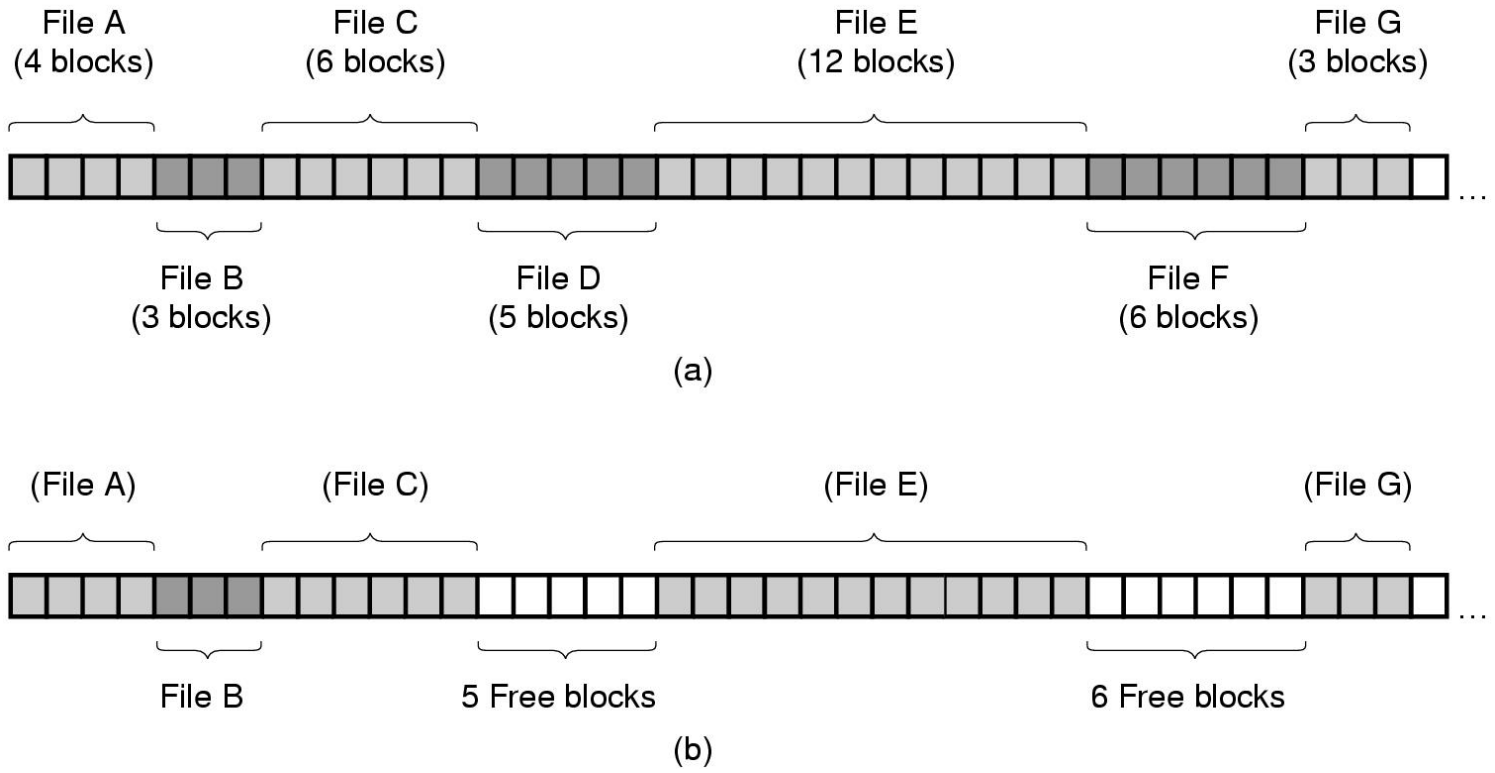


Per un disco di 256 GB, con blocchi da 1K ho bisogno di 1.052.689 blocchi per contenere la free-list dei 2^28 blocchi

Tecniche di allocazione

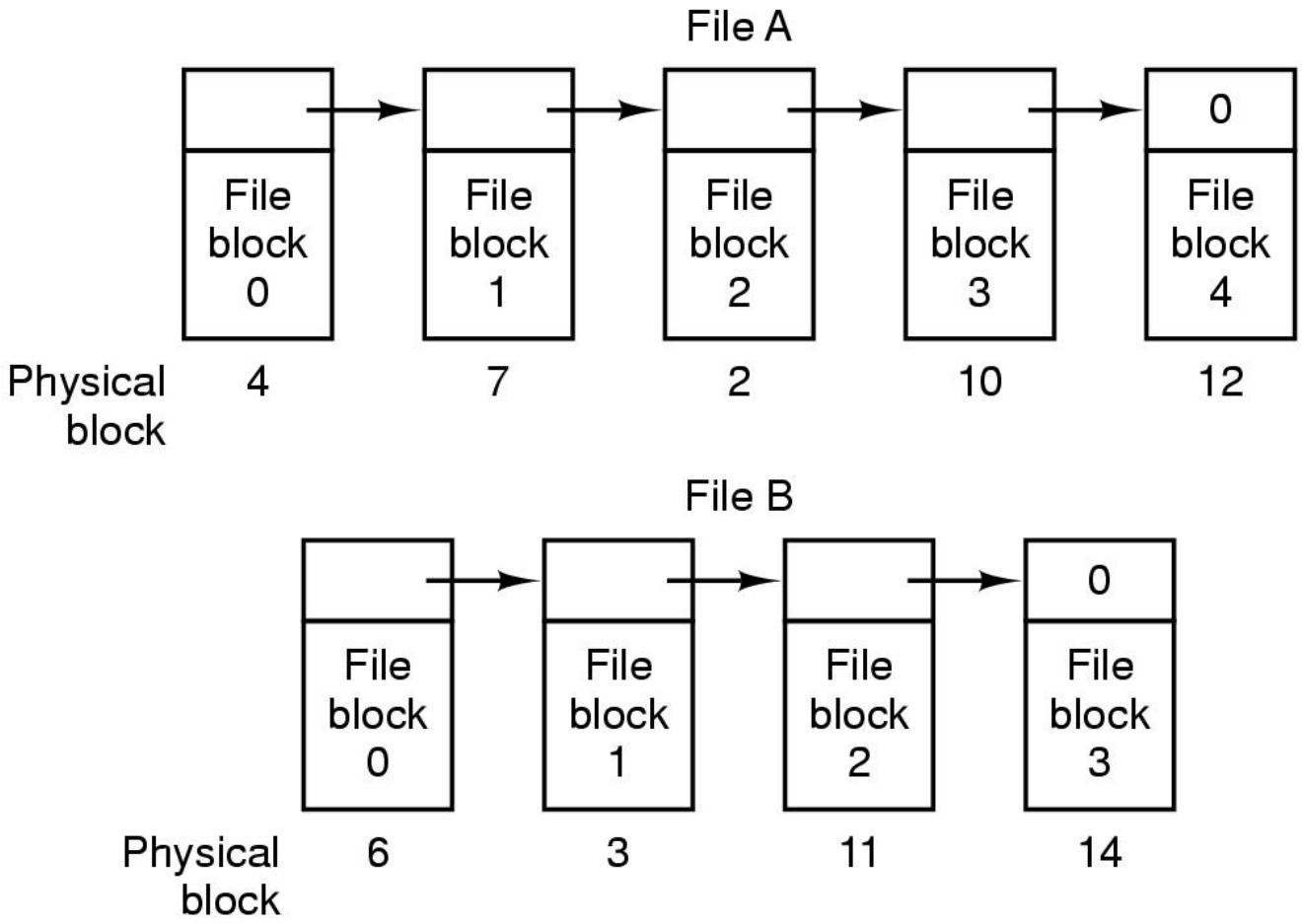
- Il FS deve preoccuparsi di tenere traccia dei contenuti dei diversi blocchi del disco, così come dei blocchi liberi, questa attività presuppone però che sia stata definita la modalità di allocazione dei blocchi ai file
  - Contigua
  - Lista linkata
  - FAT
  - I-node

Contigua

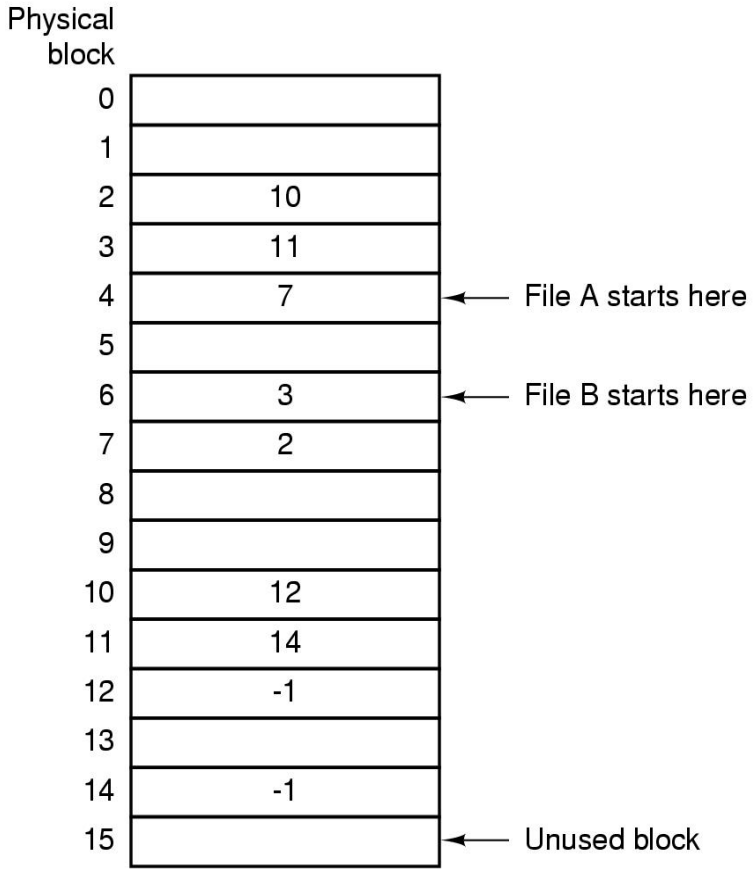


a. Allocazione contigua dello spazio su disco per 7 files
b. Situazione del disco dopo la rimozione di *D* e *E*

Lista linkata



FAT

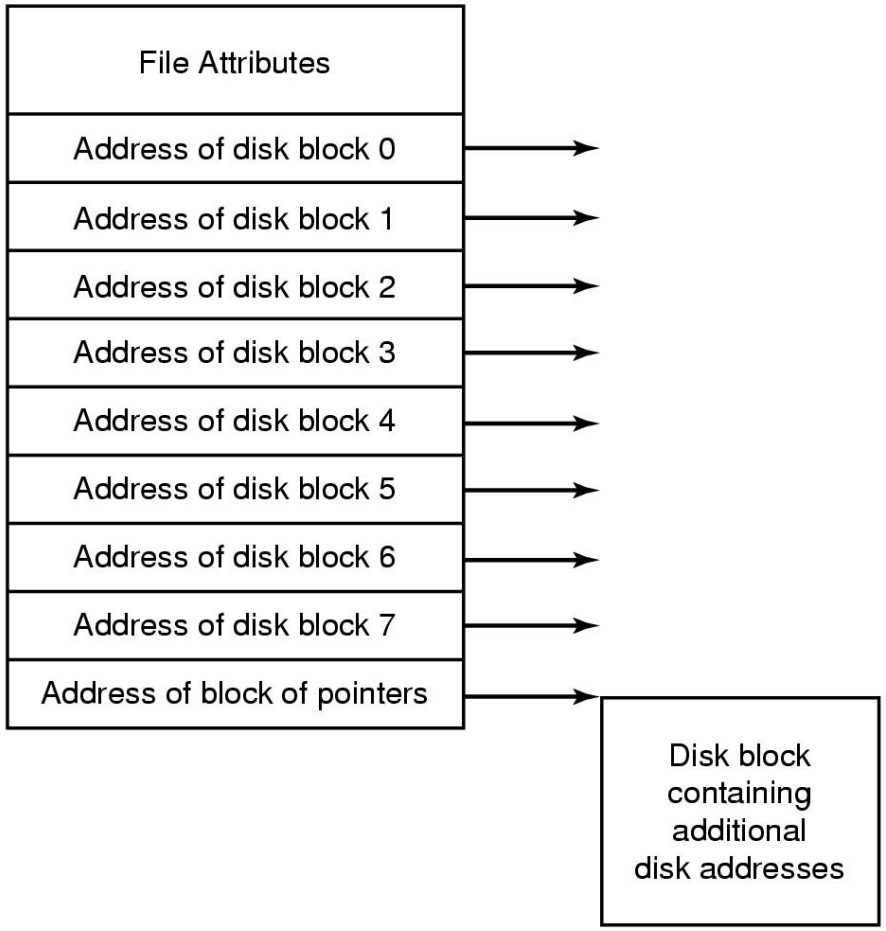

Physical block
0
1
2
10
3
11
4
7
File A starts here
5
6
3
File B starts here
7
2
8
9
10
12
11
14
12
-1
13
14
-1
15
Unused block

FAT

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

- Dimensione massima dei file in funzione della dimensione del blocco
- Le entry sono vuote in corrispondenza delle combinazioni non consentite

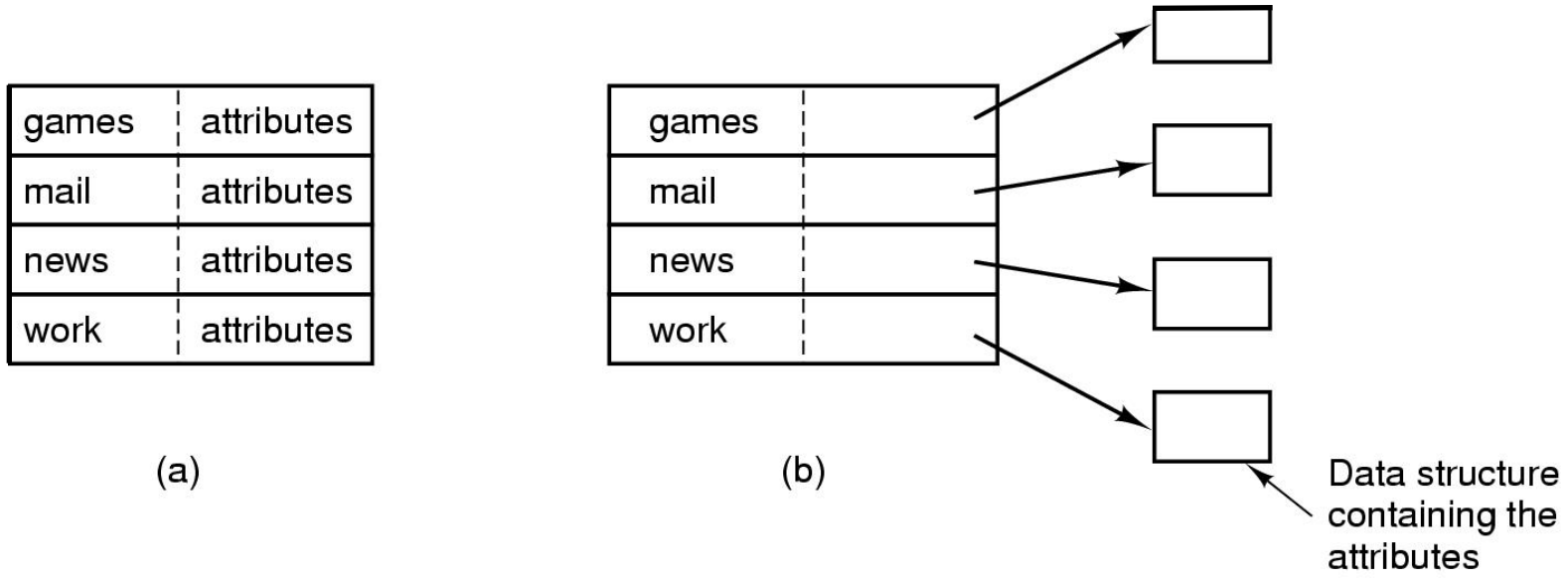
I-node



Directory

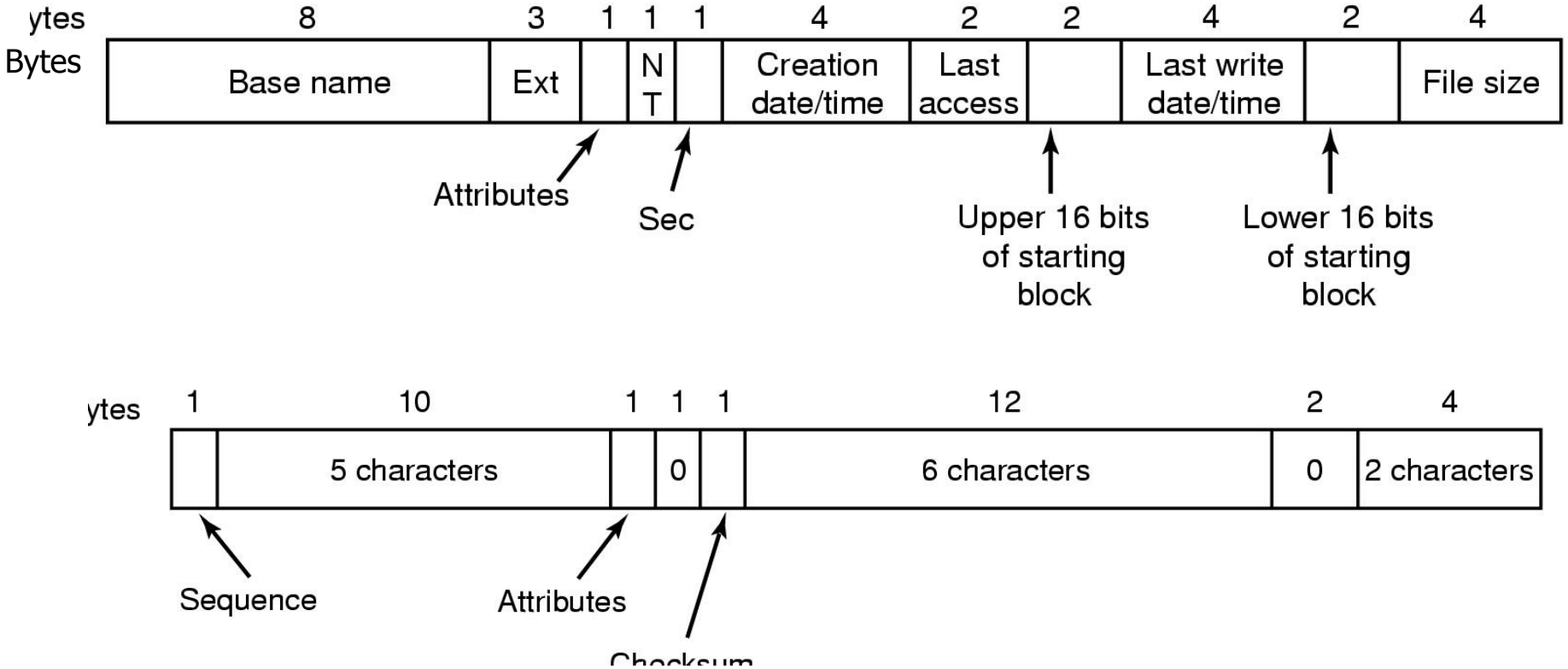
- Struttura dati usata dal FS per associare al nome simbolico di un file il suo indirizzo fisico
- Per recuperare la directory di un file è necessario sapere dov'è la directory root, è di solito in una posizione fissa all'interno della partizione di boot in altri casi è il superblock a fornire informazioni su dove recuperarla
- Si sono sinora utilizzate due strategie per implementare le directory:
  - Inserire tutti gli attributi di un file in directory
  - Inserire nella directory un puntatore ad un file che contiene gli attributi del file

Implementazione di Directory



a. Directory semplice
   a. Dimensione fissa
   b. Indirizzi e attributi del file sono nell'entry di directory
b. Directory dove ogni entry punta a un i-node

Directory in Windows 98



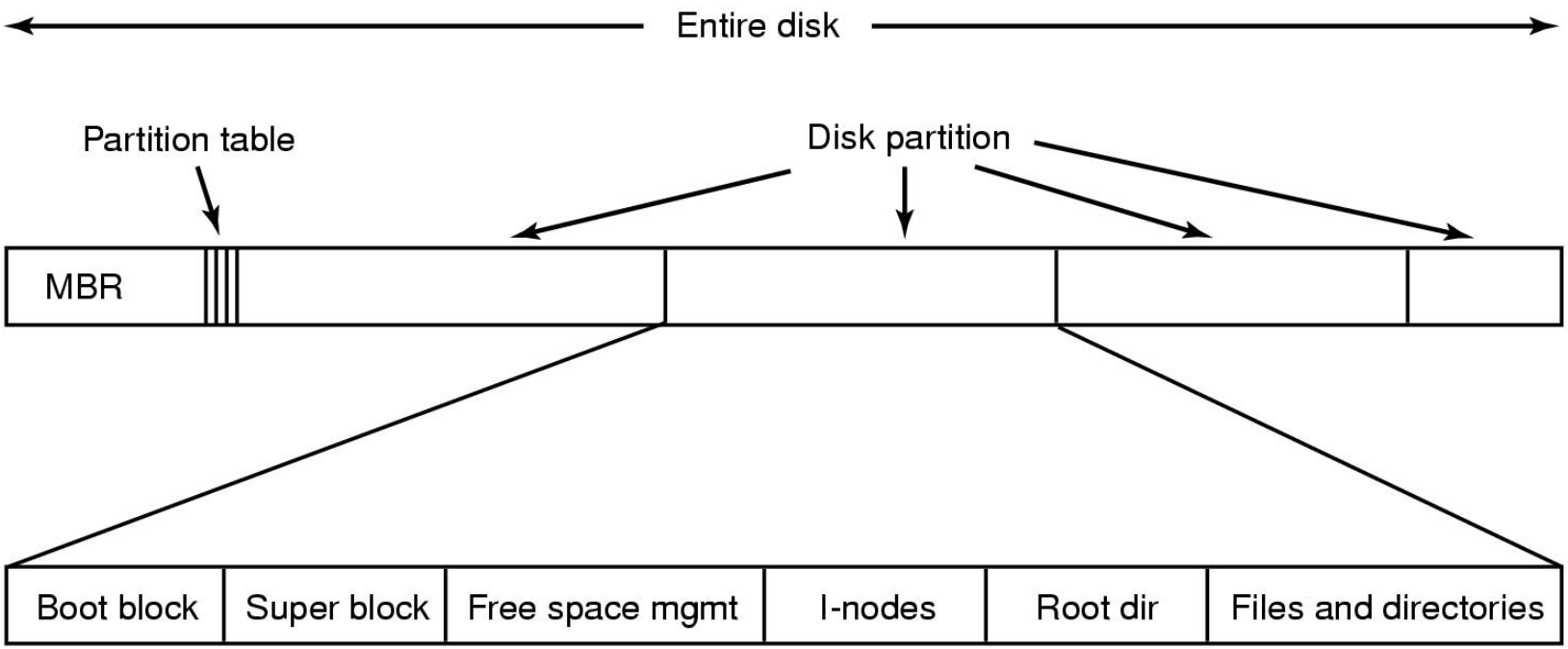
Directory in Unix

- È un file con la seguente struttura:

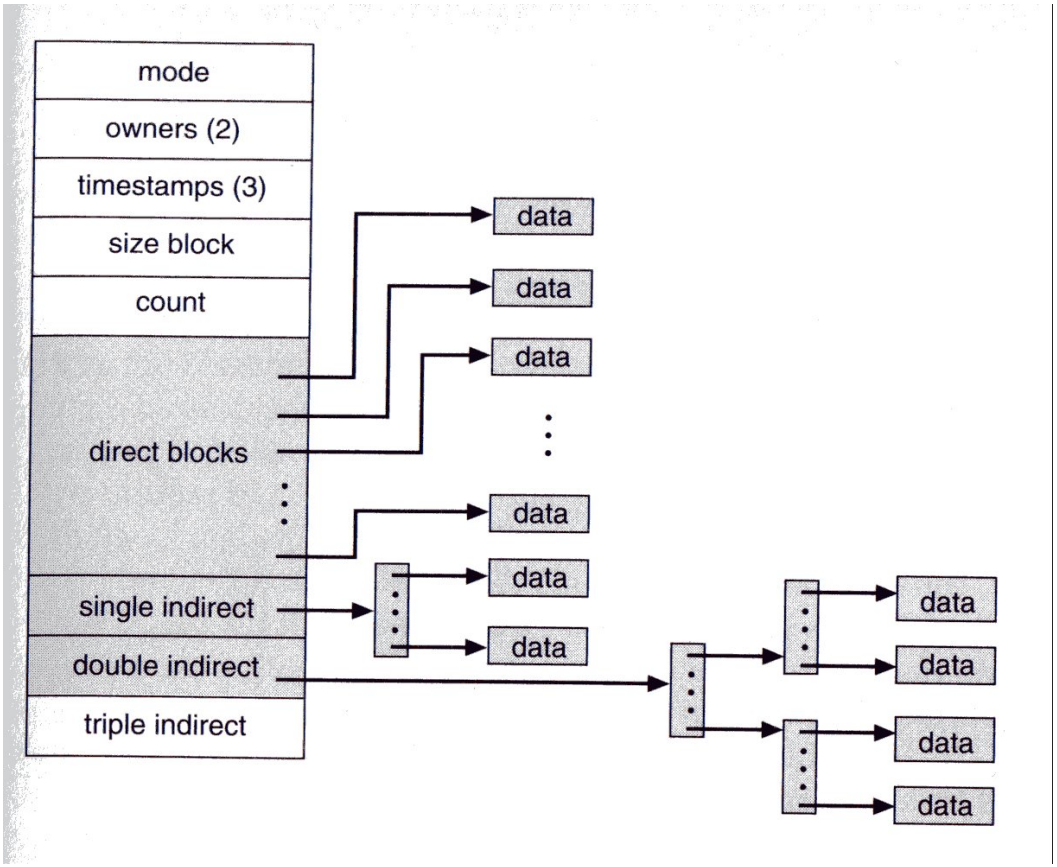
```
typedef struct{
        unsigned inode_number ;  /* 2 bytes */
        char file_name[14] ;  /* 14 bytes */
} DIRECTORY_ENTRY;
```

- Per ogni file contiene nome e i-number
  - Ogni i-number identifica univocamente un i-node
  - Gli i-node sono memorizzati in un zona del disco nota
  - I-node di root è noto

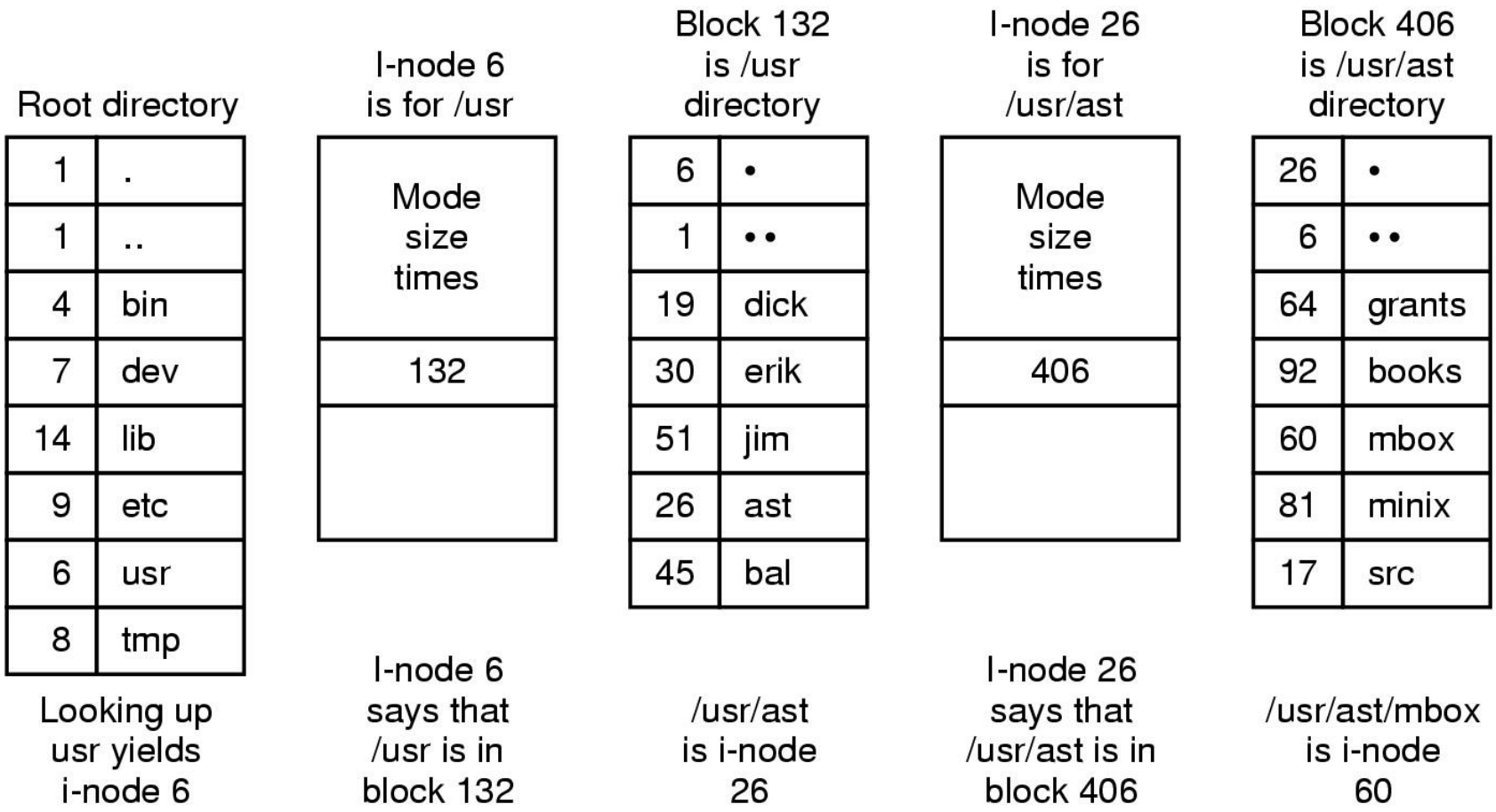
File System Unix



i-node


mode
owners (2)
timestamps (3)
size block
count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

Name Resolution



Come avviene la ricerca del file: */usr/ast/mbox*

FS Reliability

- **Problema**: i dispositivi di memorizzazione hanno un indice di affidabilità abbastanza basso, e sono tra i primi componenti che si guastano di un sistema
- **Soluzione**: il FS offre oltre ad alcune tecniche di recovery automatico, strumenti di back-up e recovery per consentire all'utente di ovviare ad eventuali danni
  - Back-up incrementali: solo i dati modificati dall'ultimo back-up sono ricopiati

File System Backup

- Il Backup consiste nella copia di un intero (o parti) file system su un supporto alternativo a quello che ospita la versione originale
- Le principali motivazioni sono:
  - Ripristinare il sistema a seguito di un disastro (rottura disco, calamità naturale)
  - Ripristinare il sistema a seguito di errori accidentali (rimozione involontaria di file)
- Operazione costosa in termini di tempo e spazio

Backup: questioni critiche

- Tutto il file system o solo parti
  - Eseguibili ha senso farne il backup?
  - File non modificati dall'ultimo backup → incrementale
- Compressione file prima del backup
  - In caso di errore minimo sul file copia può essere difficile ripristinare l'originale
- Protezione dei dati di backup

Backup fisico

- Viene effettuata la copia bit a bit del dispositivo fisico, senza consapevolezza della sua struttura logica:
  - Blocchi liberi sono copiati
  - Blocchi danneggiati: è necessario che il back up ne sia a conoscenza per evitare di tentare di leggerne il contenuto
- Vantaggi: semplice e veloce (opera alla velocità del disco)

Backup logico

- Consapevole della struttura del disco e in grado di interpretare le varie strutture dati
- Può limitarsi a copiare specifiche directory o file modificati a partire da un certo istante
- Il tipo di backup più usato

OTTIMIZZAZIONI

Buffer cache

- Quando si effettua un'operazione su disco i dati contenuti su uno o più blocchi sono portati in memoria
- La buffer cache è un'area di transito della memoria centrale dove questi blocchi restano per un certo periodo al fine di ottimizzare i tempi di trasferimento tra disco e memoria centrale

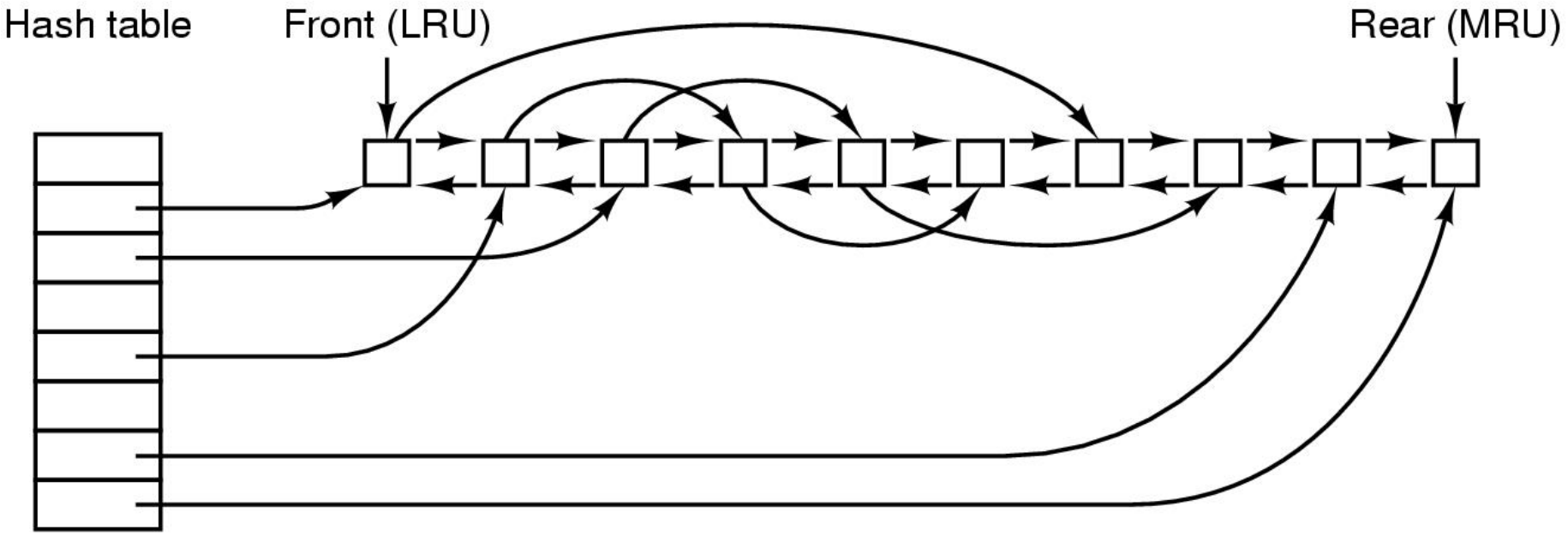
Buffer cache

- Le applicazioni sono caratterizzate da una certa località nei pattern di accesso a disco
- Si introduce allora, all'interno della memoria centrale una cache in cui memorizzare i blocchi di disco acquisiti dai diversi processi
- Questa cache è chiamata buffer cache:
  - system wide, usata e condivisa da tutti i processi
  - Obiettivo principale avvicinare i tempi di accesso a disco a quelli della memoria centrale
  - Buffer cache compete con VM

Buffer cache

- Problema: la buffer cache è volatile, cosa succede in caso di spegnimento di un sistema dei dati in buffer cache?
- Due strategie:
  - Windows: dopo una write riscrive immediatamente il blocco su disco (write-through)
  - UNIX: riscrive i blocchi di buffer cache modificati ogni 30 secondi (write back)

Gestione buffer cache

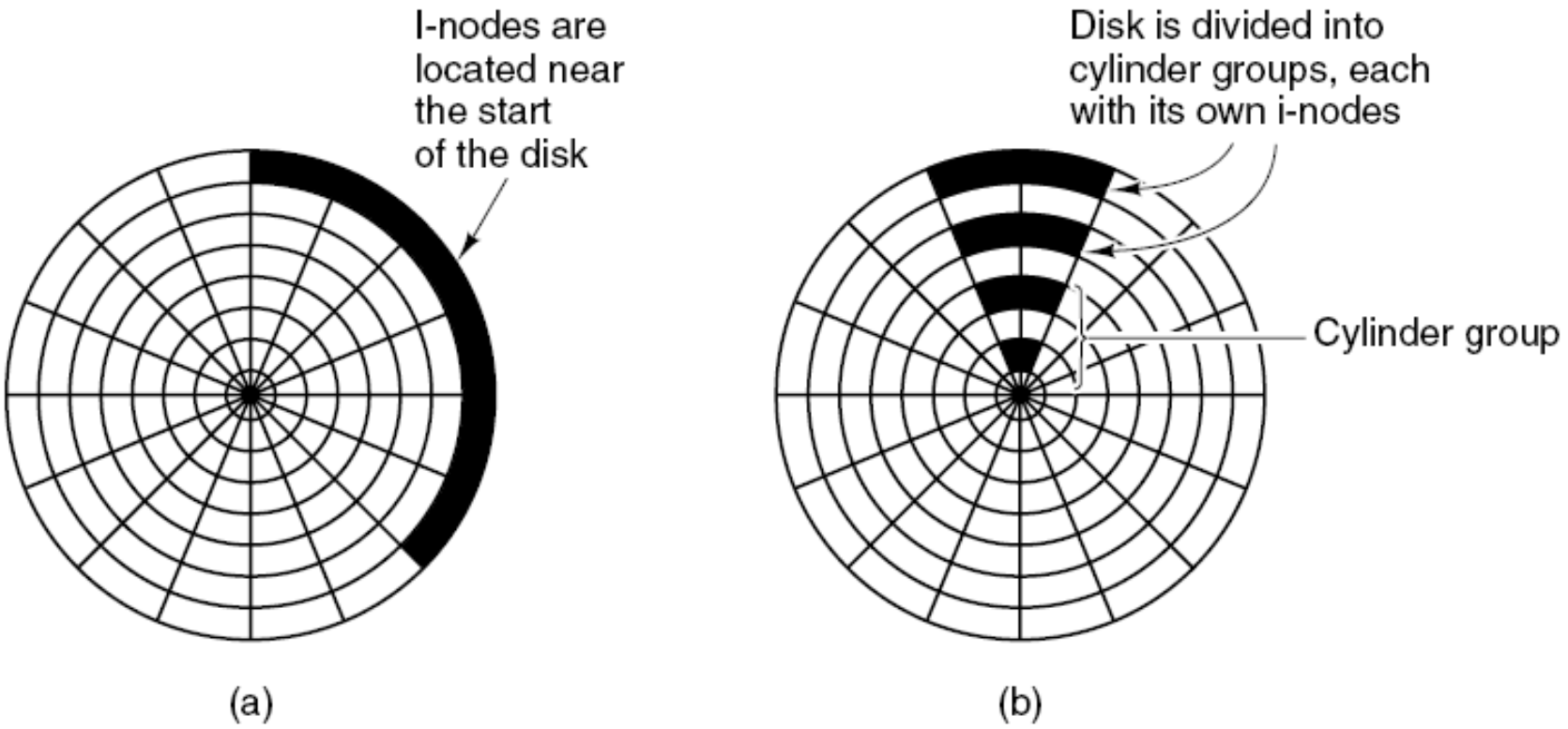


- La buffer cache è gestita tramite una tabella hash con liste linkate per la gestione delle collisioni
- A loro volta tutti i blocchi di buffer cache sono gestiti anche attraverso una lista doppia, ordinata secondo un criterio LRU (Least Recently Used)

Read Ahead

- Implementata su molti file system
- Quando un processo chiede l'accesso al blocco k, il FS acquisisce anche il blocco k+1, mentre il processo elabora il blocco k
  - Ottimo effetto per file sequenziali
  - Nel caso di file ad accesso causale può provocare l'effetto opposto saturando la banda con dati inutili
- Il FS può cercare di inferire il comportamento di un file
- Non funziona con le write

Riduzione spostamenti testine



Consistenza FS

- La `write()` di un nuovo record di un file avviene eseguendo i seguenti passi:
  - Il record da scrivere viene messo in cache
  - Successivamente sarà avviata la procedura di scrittura su disco che implica:
    - write della bit map dei blocchi liberi
    - write dell' i-node del file
    - write del blocco dati

Consistenza FS

- L'unità disco è in grado di garantire l'atomicità di una write alla volta
- Un crash del disco si può verificare in un qualunque momento
  - Ad esempio nel bel mezzo di un'operazione di scrittura
- È quindi opportuno richiedere che un'operazione come quella descritta sia eseguita atomicamente

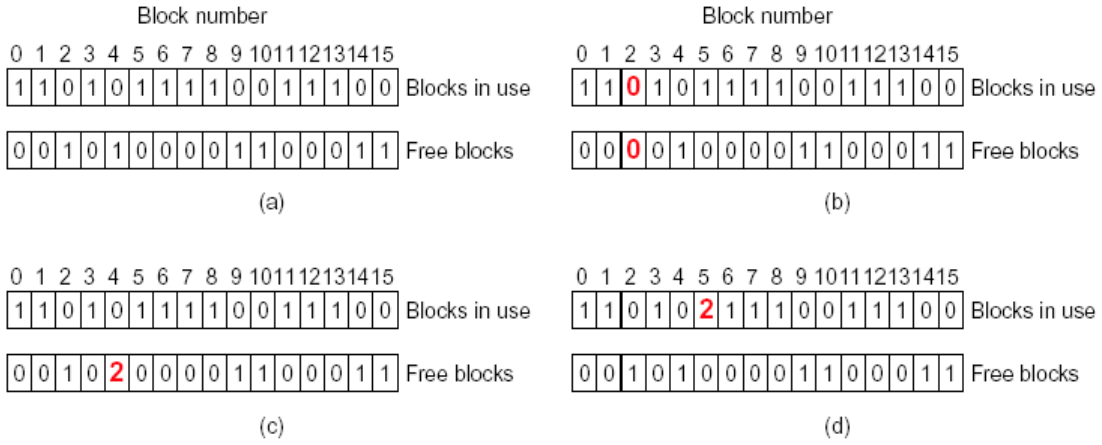
Crash dopo aggiornamento bitmap

- Un blocco risulta in uso
- Non vi è però corrispondenza del suo uso negli inode
- Il blocco non è nemmeno stato usato
- Cosa succede se cambiamo l'ordine delle operazioni
  - I-node, Bitmap, Data
  - Data, Bitmap, I-node

Consistenza FS

- Verifica periodicamente la consistenza delle strutture dati di riferimento
- Per fare questa attività fa ricorso ad un programma (fsck in Unix e chkdsk in Windows) che viene eseguito periodicamente in fase di boot, o dopo un guasto o lo spegnimento accidentale di un computer
  - Usa due tabelle
    - La prima costruita leggendo gli i-node per recuperare i blocchi in uso
    - La seconda scorrendo la tabella dei blocchi liberi
- Poco efficiente !!!

Controllo consistenza dei blocchi



- a) FS consistente,
- b) Missing block,
- c)double free block (non possibile con bit-map)
- d) double used block

Evitare FSCK

- Scrivere un “qualcosa” (write ahead log/ journal) sul disco prima di modificare le sue strutture dati
- Svolgi le operazioni richieste e solo al termine elimina il log
- In caso di crash dall’analisi del log emergono le operazioni che devono ancora essere svolte
- Questo modo di operare dei FS viene chiamato journaling

Esempio

- Scrive sul journal:
  - Inizio transazione
  - I blocchi relativi ad B, I e D
  - Fine transazione
- I blocchi coinvolti nella transazione sono scritti nelle loro posizioni su disco
- La transazione viene rimossa dal journal

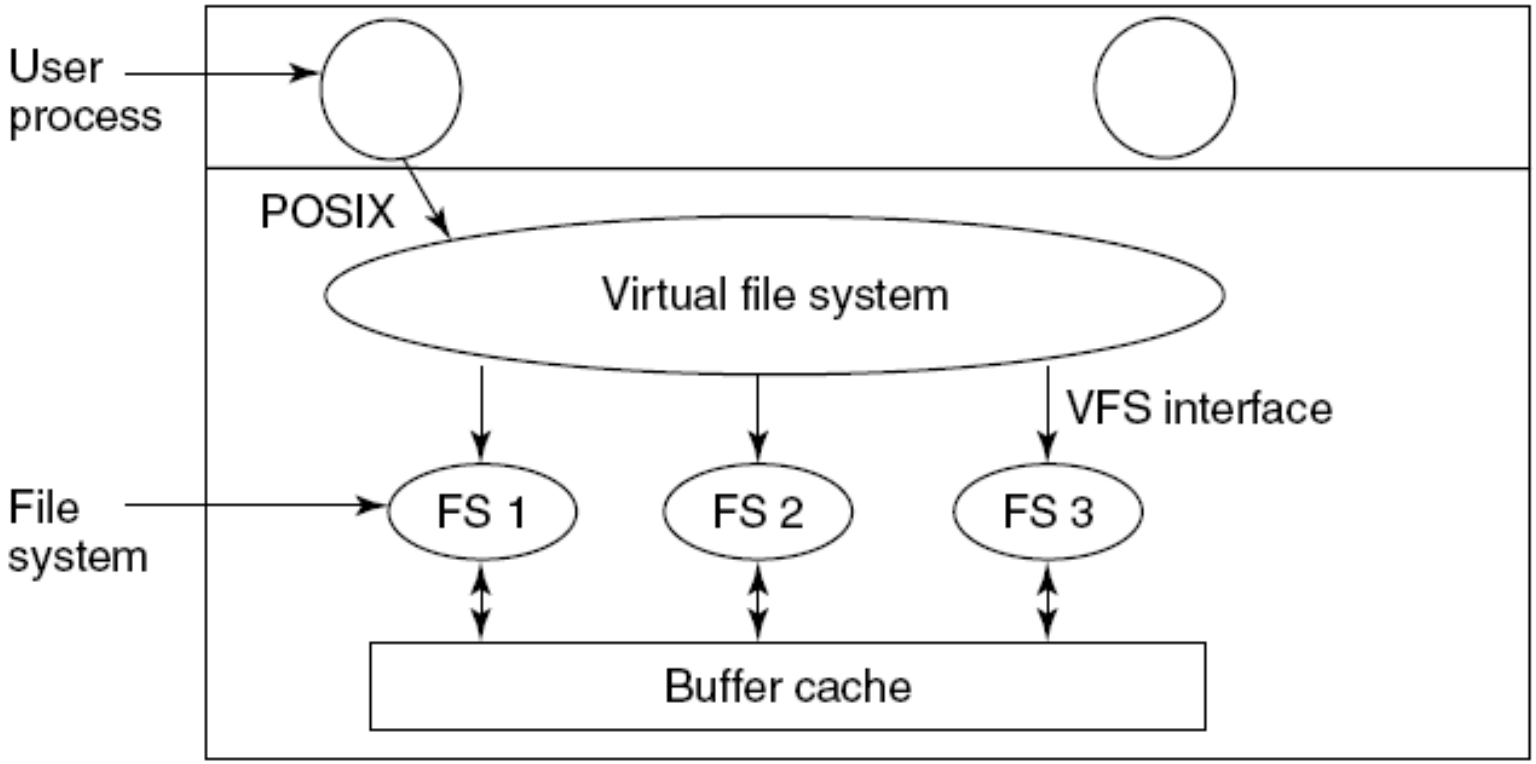
Problemi con il journaling

- Molte extra write
  - Tutti i dati sono scritti su disco due volte
- Possibile ottimizzazione:
  - Scrivere sul journal le sole transazioni relative ai metadati (i-node, bitmap)
  - Scrivere immediatamente i blocchi dati direttamente alla destinazione finale
- Tutti i file system moderni usano journaling: ext3, NTFS, IBM JFS, ReiserFS

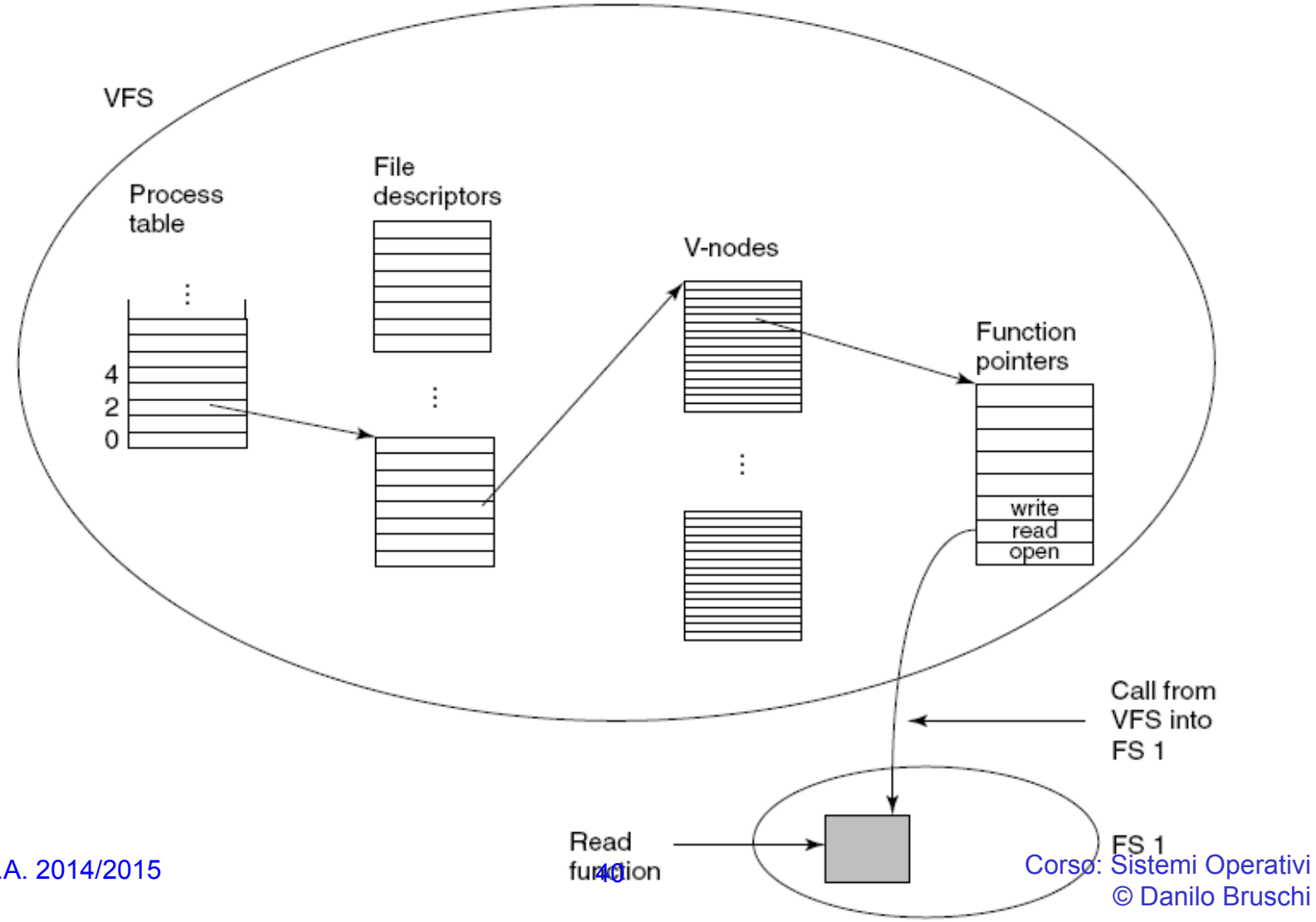
VFS

- Linux fornisce anche un potente strumento per l'integrazione di diverse tipologie di file system
- VFS è l'unica interfaccia verso le applicazioni in merito alla gestione dei file
- Le applicazioni possono accedere a diversi file system su media diversi usando un insieme omogeneo di system call

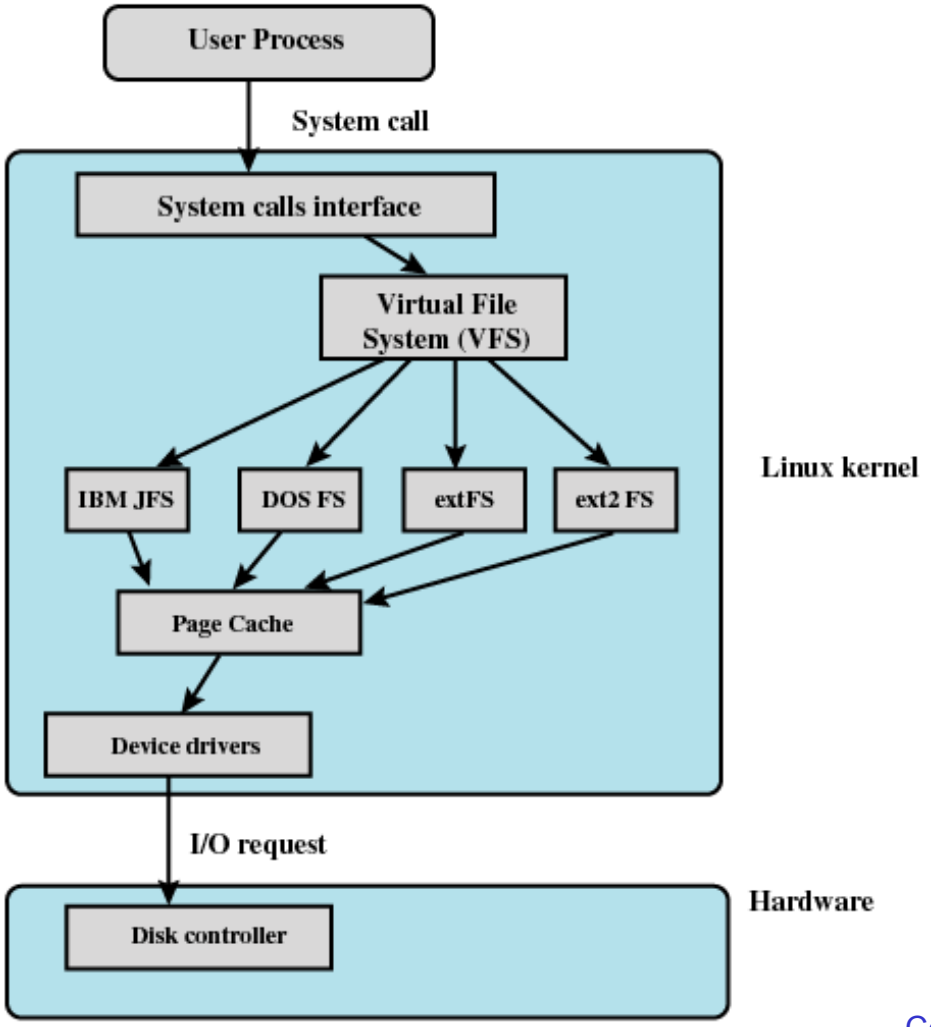
Virtual File Systems



Virtual File Systems

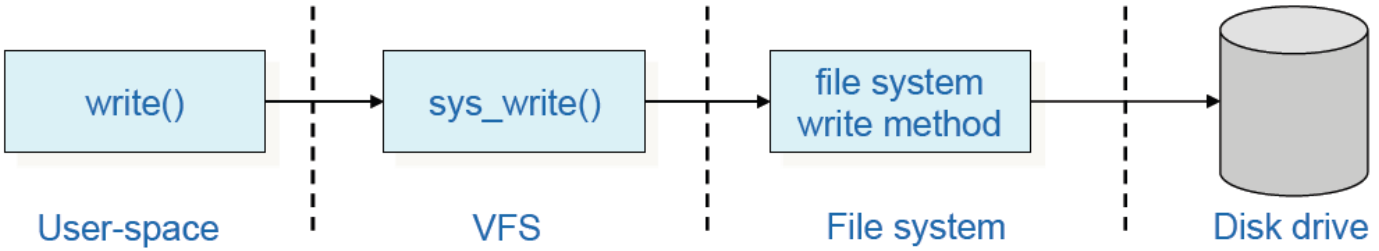


VFS


User Process
System call
System calls interface
Virtual File System (VFS)
IBM JFS
DOS FS
extFS
ext2 FS
Page Cache
Device drivers
Linux kernel
I/O request
Disk controller
Hardware

Esempio

- `write (f, &buf, len);`
- Trasferisce `len` byte dall'area di memoria `buf` al file il cui descrittore è f
- La system call è intercettata dal VFS che la traduce nell'appropriato comando per il file system di destinazione



FS supportati da Linux (alcuni)

Linux Filesystems

- Media based
  - ext2 - Linux native
  - ufs - BSD
  - fat - DOS FS
  - vfat - win 95
  - hpfs - OS/2
  - minix - well....
  - Isofs - CDROM
  - sysv - Sysv Unix
  - hfs - Macintosh
  - affs - Amiga Fast FS
  - NTFS - NT's FS
  - adfs - Acorn-strongarm
- Network
  - nfs
  - Coda
  - AFS - Andrew FS
  - smbfs - LanManager
  - ncpfs - Novell
- Special ones
  - procfs -/proc
  - umsdos - Unix in DOS
  - userfs - redirector to user

P.J.Braam/CMU -- 4