Il File System

Il File System

Il file system e` quella parte del Sistema Operativo che fornisce i meccanismi necessari per l'accesso e l'archiviazione delle informazioni in memoria secondaria.

Realizza i concetti astratti:

- ✓ di **file**: unità logica di memorizzazione
- ✓ di **direttorio**: insieme di file (e direttori)
- ✓ di **partizione**: insieme di file associato ad un particolare dispositivo fisico (o porzione di esso)

Le caratteristiche di file, direttorio e partizione sono del tutto indipendenti dalla natura e dal tipo di dispositivo utilizzato.

6.1 Organizzazione del File System

La struttura di un file system puo`, in generale, essere rappresentata da un insieme di componenti organizzate in vari livelli:

Struttura logica: Il File

Il file è un insieme di informazioni, rappresentate mediante insieme di *record logici* (bit, byte, linee, record, etc.)

- Ogni file è caratterizzato da un insieme di attributi.

Ad esempio:

- tipo: stabilisce l'appartenenza a una classe (eseguibili, batch, testo, etc)
- indirizzo: puntatore/i al record logico corrente (lettura/scrittura)
- dimensione: numero di byte contenuti nel file
- data e ora (di creazione e/o di modifica)

In S.O. multiutente anche:

- utente proprietario
- protezione: diritti di accesso al file per gli utenti del sistema

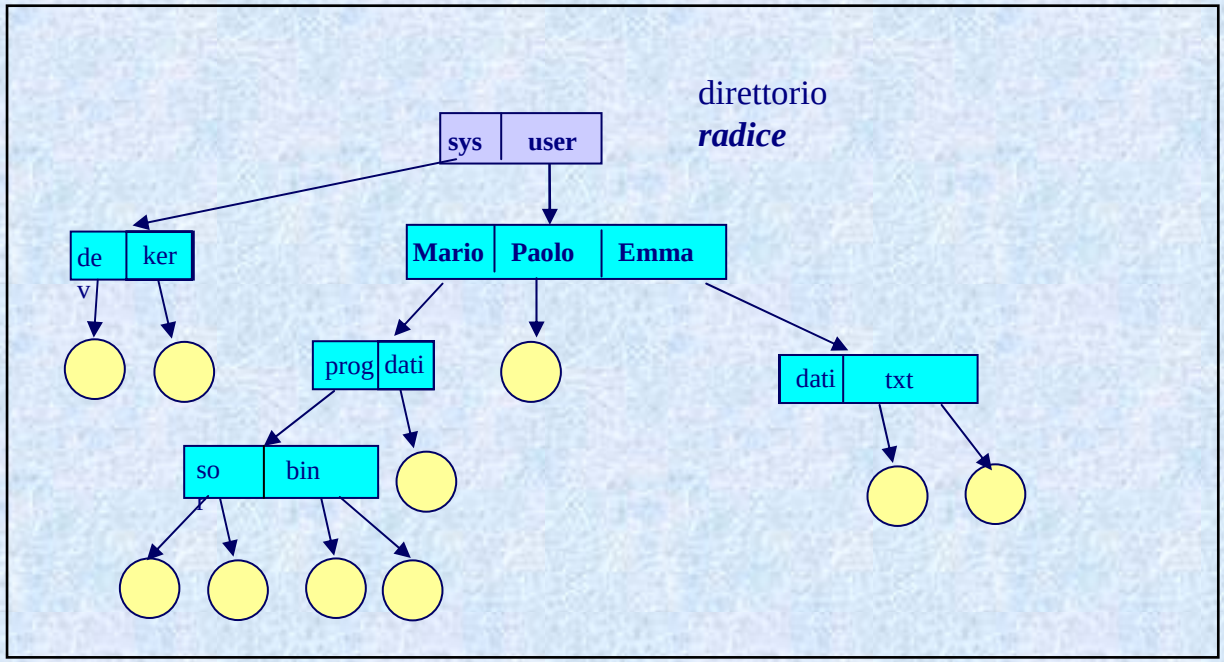
Struttura logica: Il Direttorio

E` un'astrazione che consente di raggruppare più file :

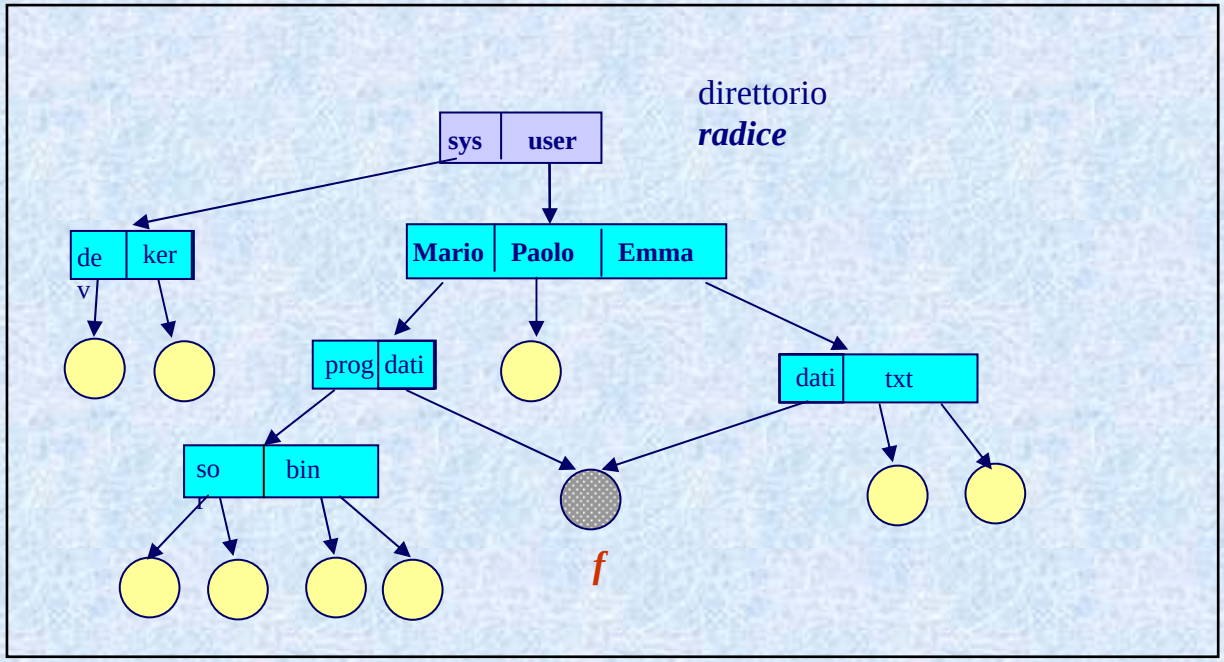
- **un direttorio può contenere più file**
- **può contenere a sua volta altri direttori: struttura logica del file system come una composizione di file e direttori.**

➔ **La struttura logica del file system è un'aggregazione di file e direttori le cui caratteristiche dipendono dalle caratteristiche del *direttorio* stesso.**

Direttorio ad albero



Direttorio a grafo diretto aciclico



Struttura logica: Gestione del File System

Operazioni fondamentali per la gestione del file system:

- **Creazione e cancellazione di direttori: modificano la struttura logica del file system, aggiungendo/eliminando rami al grafo che rappresenta il file system.**
- **Aggiunta/cancellazione di file**
- **Listing: ispezione del contenuto di uno (o più) direttori**
- **Attraversamento del direttorio: *navigazione* attraverso la struttura logica del file system.**

Accesso al file system

Strutture dati: rappresentazione del file

- ogni file è rappresentato concretamente dal *descrittore di file*, che memorizza gli attributi
- i descrittori di file devono essere memorizzati in modo persistente, mediante apposite strutture in memoria secondaria (Unix: *i-list*)

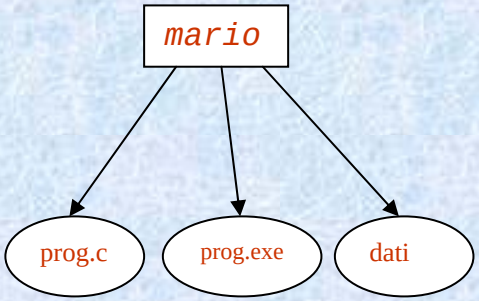
Strutture dati

Rappresentazione del direttorio

Poiché ogni file appartiene ad un direttorio, ogni direttorio mantiene il collegamento con i descrittori dei file contenuti in esso.

Ad esempio (Windows-FAT):

- **il direttorio e` una struttura dati di tipo tabellare che contiene i descrittori dei file:**

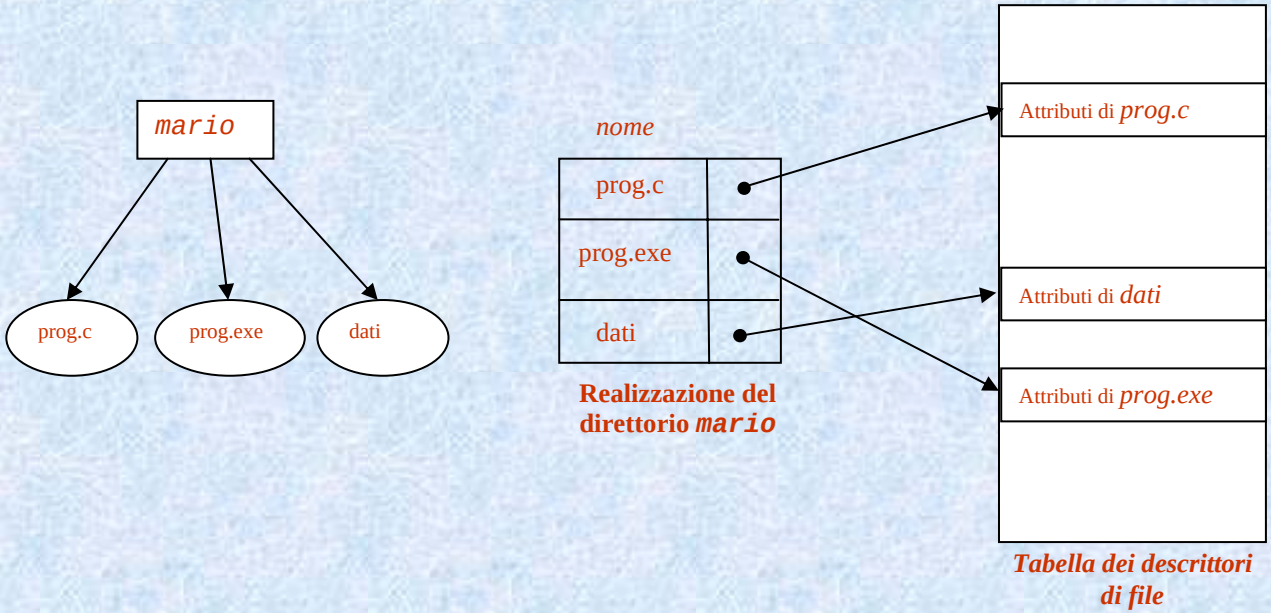


nome	*descrittore*
prog.c	**Attributi**: tipo, indirizzi, ecc.
prog.exe	**Attributi**: tipo, indirizzi, ecc.
dati	**Attributi**: tipo, indirizzi, ecc.

Realizzazione del direttorio `mario`

In alternativa (*Unix*):

- la tabella che rappresenta il direttorio contiene i riferimenti ai descrittori dei file (memorizzati in una struttura separata)



Accesso a file

Compito del S.O. è consentire l'accesso *on-line* ai file.

Operazioni di accesso:

- Lettura **di record logici dal file.**
- Scrittura**: inserimento di nuovi record logici all'interno di file.**

Ogni operazione richiederebbe la localizzazione di informazioni su disco; ad esempio:

- **gli indirizzi dei record logici a cui accedere**
- **gli altri attributi del file**
- **i record logici**

➔ **costo elevato!**

Accesso a file

Per migliorare l'efficienza:

- **il S.O. mantiene in memoria una struttura che registra i file attualmente in uso (file *aperti*):**
  - **per ogni file aperto:** {*puntatore al file*, posizione su disco,..}
- ***Memory mapping* dei file aperti:**
  - **i file aperti (o porzioni di essi) vengono temporaneamente copiati in memoria centrale -> accesso più veloce.**

Operazioni necessarie:

- **Apertura: introduzione di un nuovo elemento nella tabella dei file aperti e eventuale memory mapping del file.**
- **Chiusura: salvataggio del file in memoria secondaria e eliminazione dell'elemento corrispondente dalla tabella dei file aperti.**

Metodi di accesso

L'accesso a file può avvenire secondo varie modalità:

- **accesso *sequenziale***
- **accesso *diretto***
- **accesso a *indice***

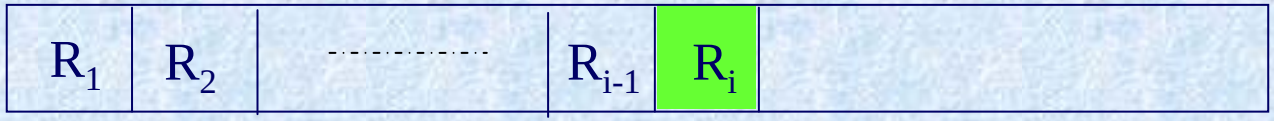
Il metodo di accesso è indipendente:

- dal tipo di dispositivo **utilizzato,**
- dalla tecnica di allocazione **dei blocchi in memoria secondaria.**

Accesso sequenziale

Il file è una sequenza **[R_1, R_2,.. R_N] di record logici:**

- **per accedere ad un particolare record logico R_i , è necessario accedere prima agli (*i-1*) record che lo precedono nella sequenza:**



- **Operazioni di accesso:**
  - ✓ ***readnext*: lettura del prossimo record logico della sequenza**
  - ✓ ***writenext*: scrittura del prossimo record logico**

➢ **ogni operazione di accesso (lettura/scrittura) posiziona il puntatore al file sull'elemento successivo a quello letto/scritto.**

Accesso diretto

Il file è un insieme non ordinato $\{R_1, R_2, .. R_N\}$ di record logici:

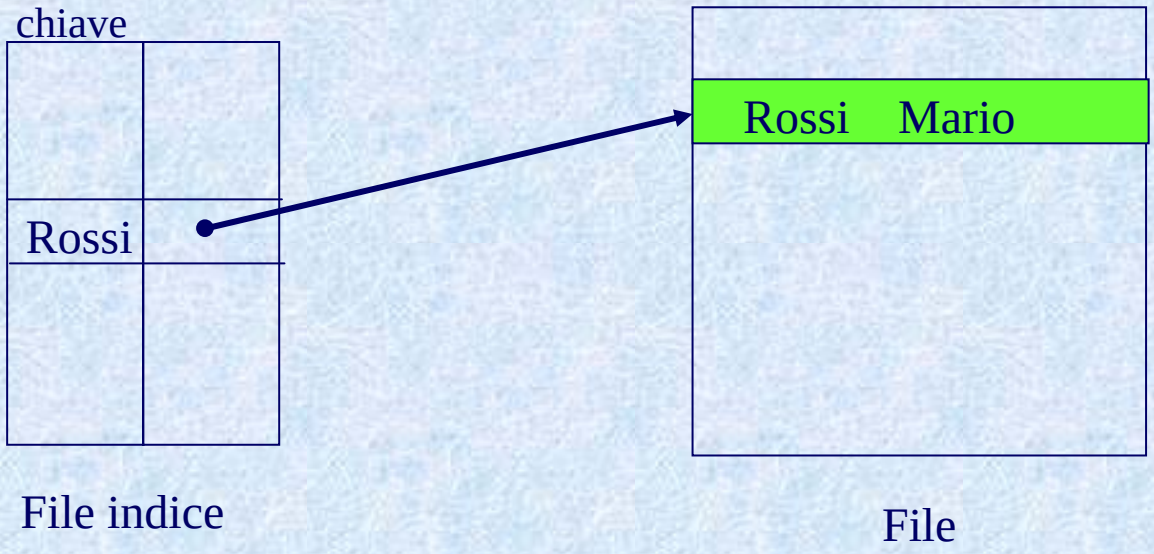
- **si può accedere direttamente ad un particolare record logico.**

- **Operazioni di accesso:**
  - ✓ ***readd(f, i, &V):*** **lettura del record logico R_i dal file *f* ; il valore letto viene assegnato alla variabile *V*;**
  - ✓ ***writed(f, i, V):*** **scrittura del valore della variabile *V* nel record logico R_i del file *f* .**

Accesso a indice

Ad ogni file viene associata una struttura dati (un file) contenente l'indice delle informazioni contenute nel file.

- **Operazioni di accesso:**
  - ✓ ***readk(f, key, &V):*** **lettura del record logico con chiave uguale a *key* dal file *f* nella variabile *V*,**
  - ✓ ***writek(f, key, V):*** **scrittura del valore della variabile V nel record logico con chiave *key*.**
- **per accedere a un record logico, si esegue una ricerca nell'indice (utilizzando una *chiave*): *readk(f, "Rossi", &V);***



6.3.3 Protezione

Le politiche di protezione delle risorse (file e direttori) possono essere specificate mediante una **matrice di protezione**:

	File1	File2	File3	File4	Dir01	Dir02	Stampante
Utente Mario	r w	r				r w	w
Utente Paola				r w x	r		
Utente Elena		r w	r w x	r		r w	w

- Ogni riga corrisponde a un ***dominio di protezione***
- Ogni colonna rappresenta una ***risorsa*** del sistema

Rappresentazione delle politiche di protezione

Due approcci:

- **Liste di controllo degli accessi (ACL):**
- **Liste di capability (C-list)**

ACL

- una *ACL* esprime la politica di protezione associata a una particolare risorsa: per ogni risorsa R vengono elencati i permessi concessi ad ogni utente per l'accesso a R
- rappresenta una colonna della matrice di protezione:

	File1	File2	File3	File4	Dir01	Dir02	Stampante
Utente Mario	r w	r				r w	w
Utente Paola				r w x	r		
Utente Elena		r w	r w x	r		r w	w

- Ad esempio, l'ACL associata a *File4* è

  [Paola: r-w-x, Elena: r].

C-list

- per ogni processo P il sistema costruisce una lista di permessi riferiti alle diverse risorse del sistema alle quali l'utente a cui P appartiene è abilitato ad accedere.
- rappresenta una riga della matrice di protezione:

	File1	File2	File3	File4	Dir01	Dir02	Stampante
Utente Mario	r w	r				r w	w
Utente Paola				r w x	r		
Utente Elena		r w	r w x	r		r w	w

- Ad esempio, ad ogni processo dell'utente Mario verrà associata la C-list:

[File1:r-w, File2:r, Dir01:r-w, Stampante1:w].

Organizzazione fisica del File System

Ogni dispositivo di memorizzazione secondaria viene partizionato in blocchi **(o** *record fisici***):**

Blocco**: unità di trasferimento nelle operazioni di I/O da/verso il dispositivo; la sua dimensione è costante.**

I processi vedono ogni file come un insieme di record logici:

Record logico**: unità di trasferimento nelle operazioni accesso al file.**

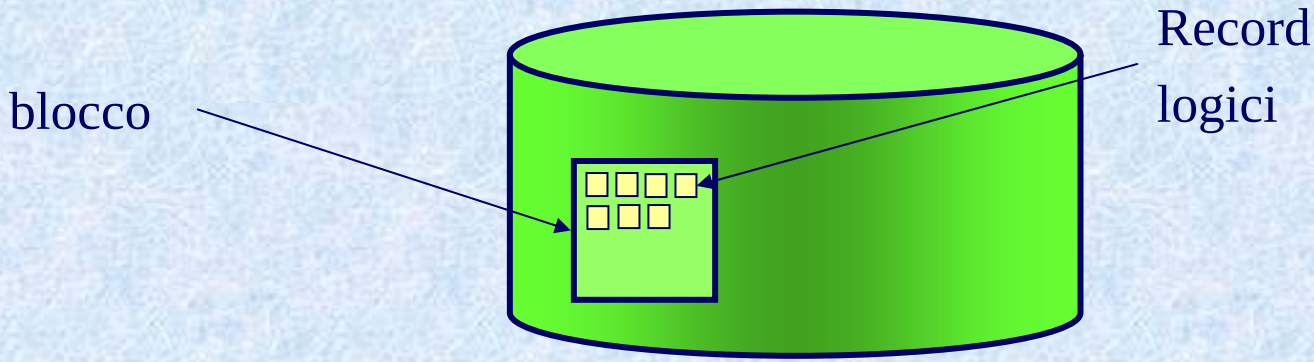
6.4.1 Blocchi & record logici

Uno dei compiti del file system è stabilire una corrispondenza tra record logici e blocchi.

Di solito:

Dimensione(blocco) >> Dimensione(record logico)

➔ **ogni blocco può contenere più record logici.**



Metodi di Allocazione dei file

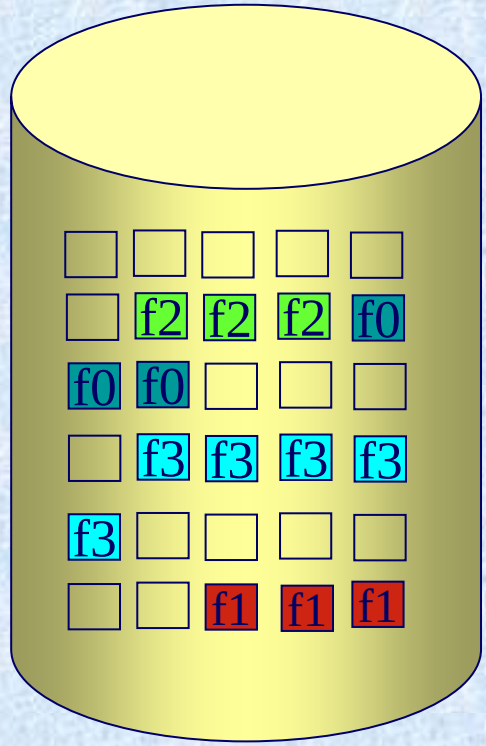
Ogni blocco **contiene un insieme di** record logici contigui.

Quali sono le tecniche più comuni per l'allocazione dei blocchi sul disco?

- **allocazione** contigua
- **allocazione a** lista
- **allocazione a** indice

Allocazione contigua

Ogni file è mappato su un insieme di blocchi fisicamente contigui.



Vantaggi:

- ✓ costo della ricerca di un blocco
- ✓ possibilità di accesso sequenziale e diretto

Svantaggi:

- ✓ *Frammentazione esterna*: man mano che si riempie il disco, rimangono zone contigue sempre più piccole, a volte inutilizzabili:
  - ➢ *compattazione*
- ✓ costo della ricerca dello spazio libero per l'allocazione di un nuovo file
- ✓ crescita delle dimensioni di file

Allocazione a lista concatenata

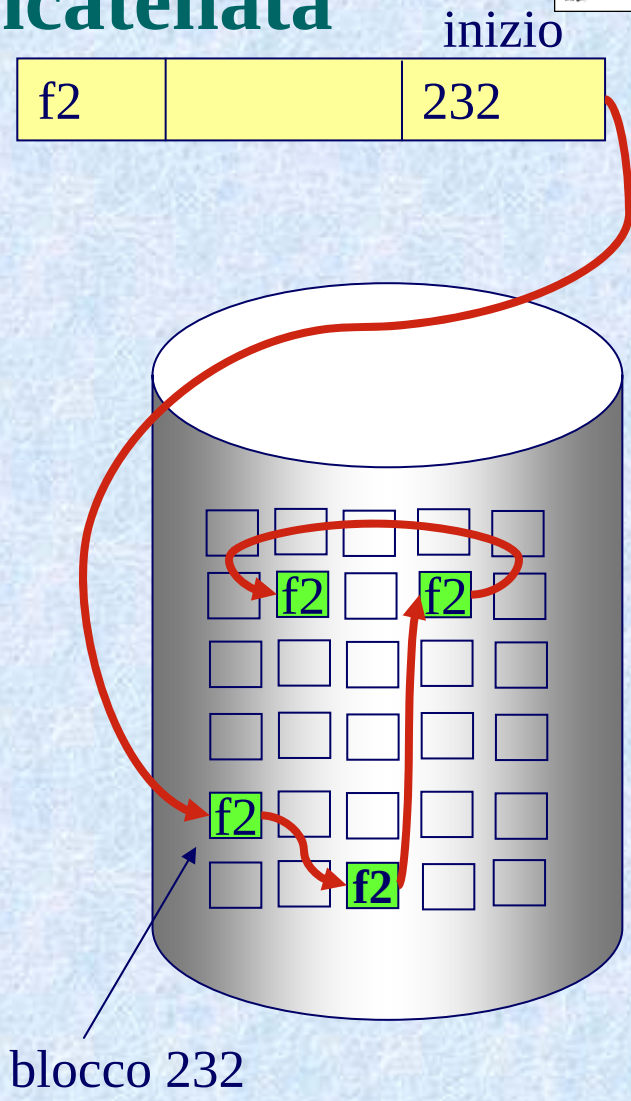
I blocchi sui quali viene mappato ogni file sono organizzati in una lista concatenata.

Vantaggi:

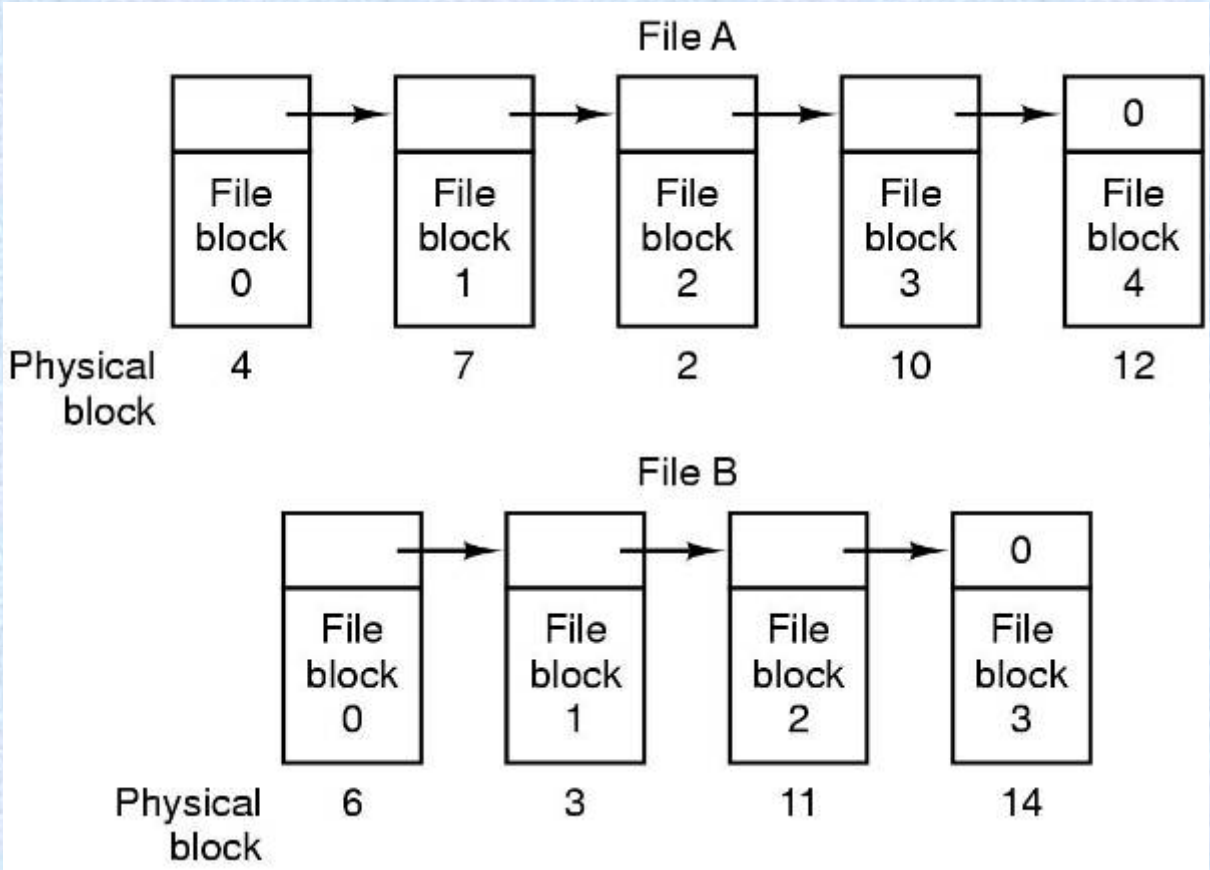
- ✓ non c'è frammentazione esterna
- ✓ minor costo di allocazione

Svantaggi:

- ✓ possibilità di errore se un link viene danneggiato
- ✓ maggior occupazione (spazio occupato dai puntatori)
- ✓ difficoltà di realizzazione dell'accesso diretto
- ✓ costo della ricerca di un blocco

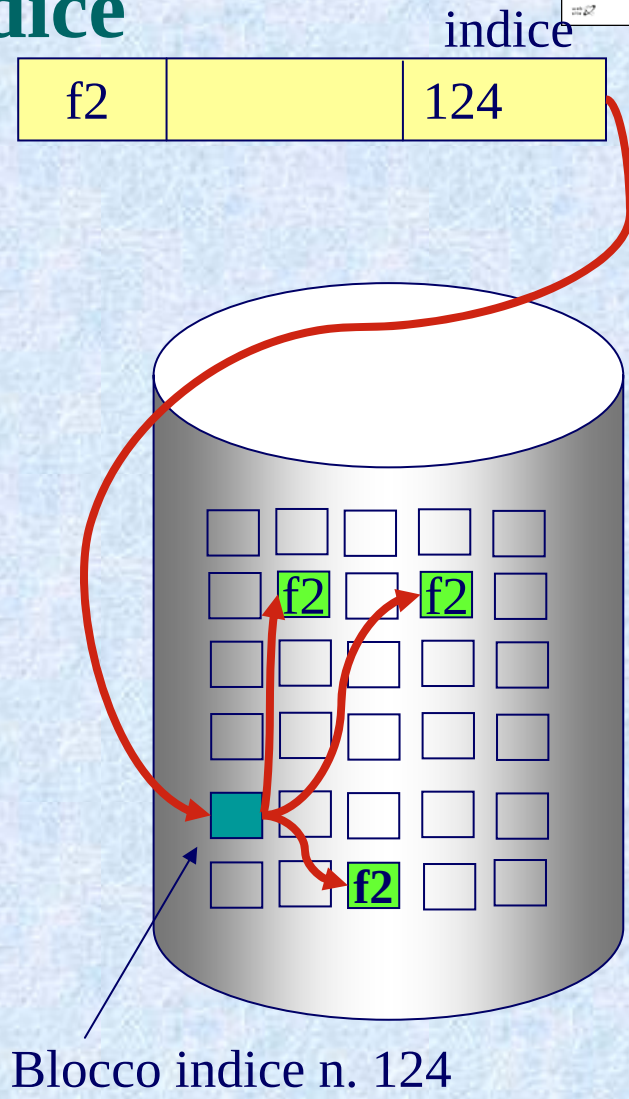


Allocazione a lista concatenata (2)



Allocazione a indice

A ogni file è associato un blocco (*indice*) in cui sono contenuti tutti gli indirizzi dei blocchi su cui è allocato il file.



Vantaggi:

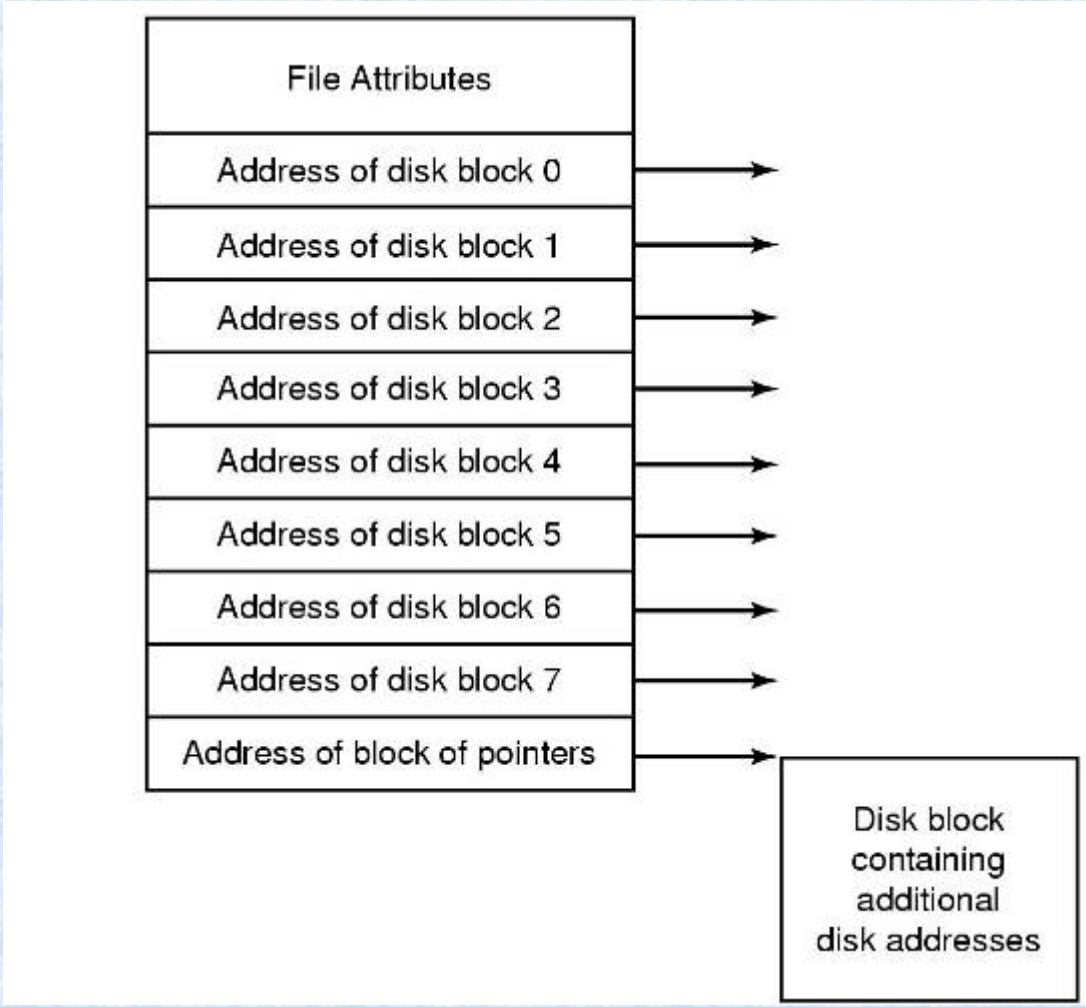
gli stessi dell'allocazione a lista, più

- ✓ possibilità di accesso diretto
- ✓ maggiore velocità di accesso (rispetto a liste)

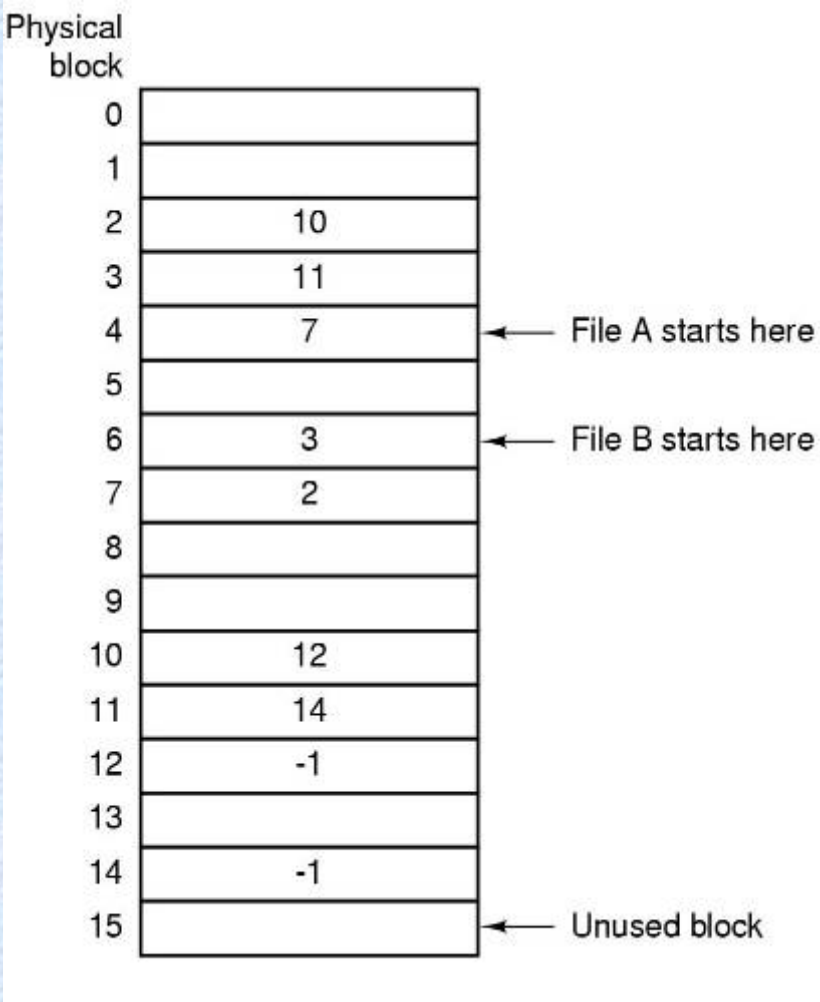
Svantaggi:

scarso utilizzo dei blocchi indice

Allocazione a indice (2)



File Allocation Table (FAT)
Soluzione ibrida tra lista concatenata e indice



Limitazioni della FAT (MS-DOS)

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

Massima dimensione del File System per diverse ampiezze dei blocchi

Metodi di Allocazione

Riassumendo, gli aspetti caratterizzanti sono:

- **grado di** utilizzo **della memoria**
- **tempo di** accesso **medio al blocco**
- **realizzazione dei** metodi di accesso

- Alcuni sistemi operativi che adottano più di un metodo di allocazione; spesso:
  - **file** piccoli ➔ **allocazione contigua**
  - **file** grandi ➔ **allocazione a indice**

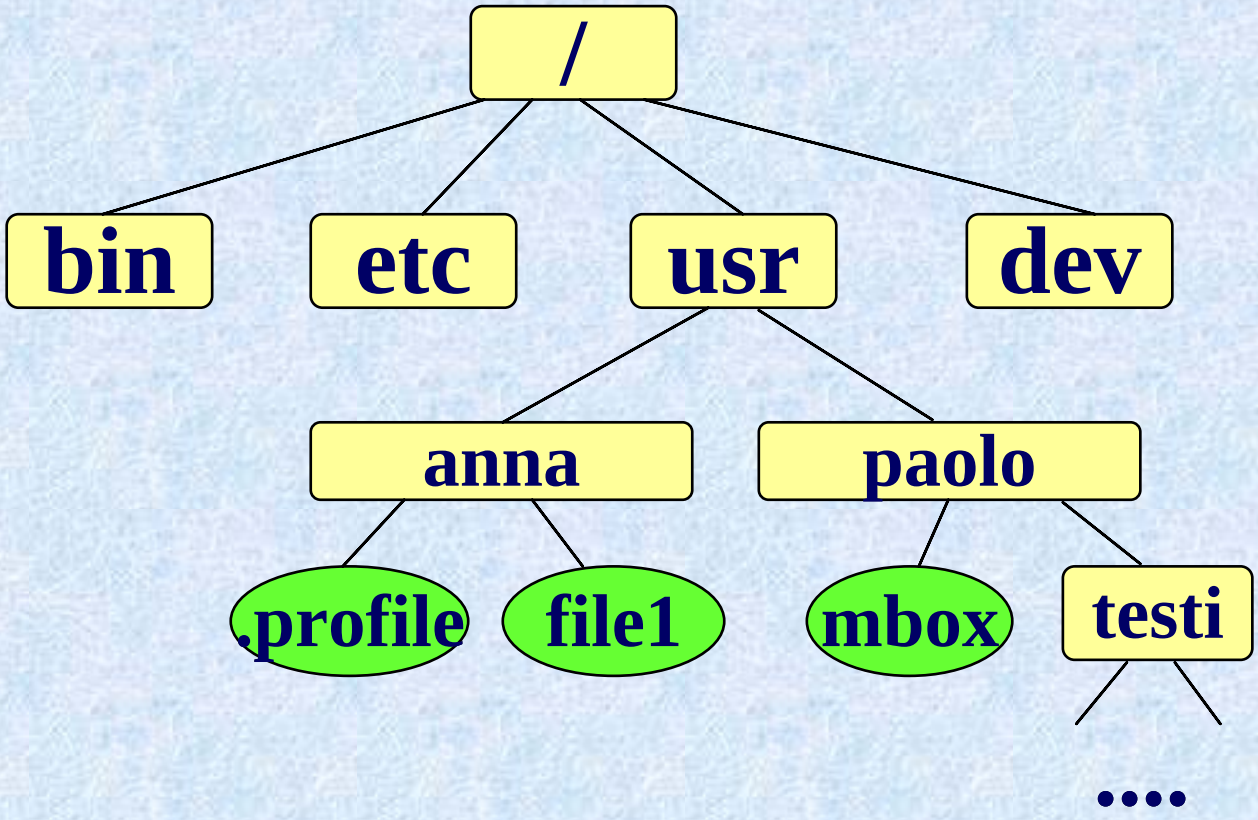
Il File System di UNIX

Omogeneità: tutto è file

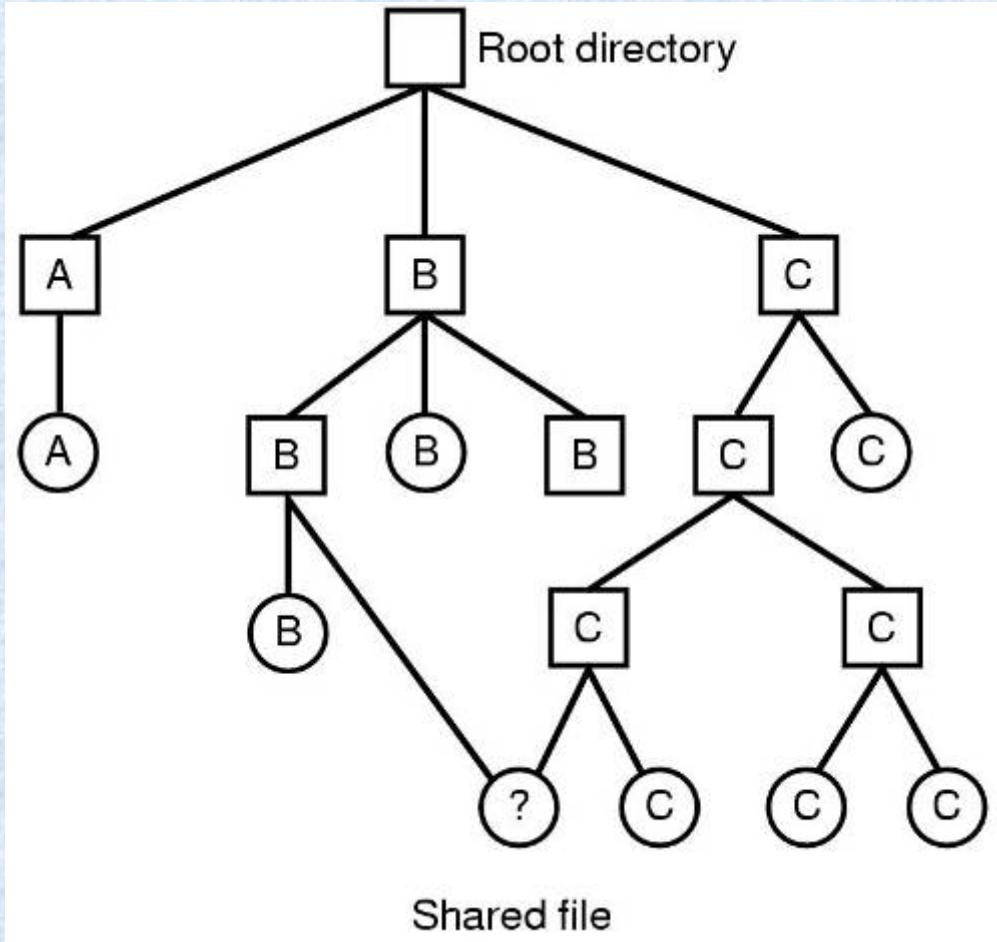
Tre categorie di file:

- file ordinari
- direttori
- dispositivi (file speciali)

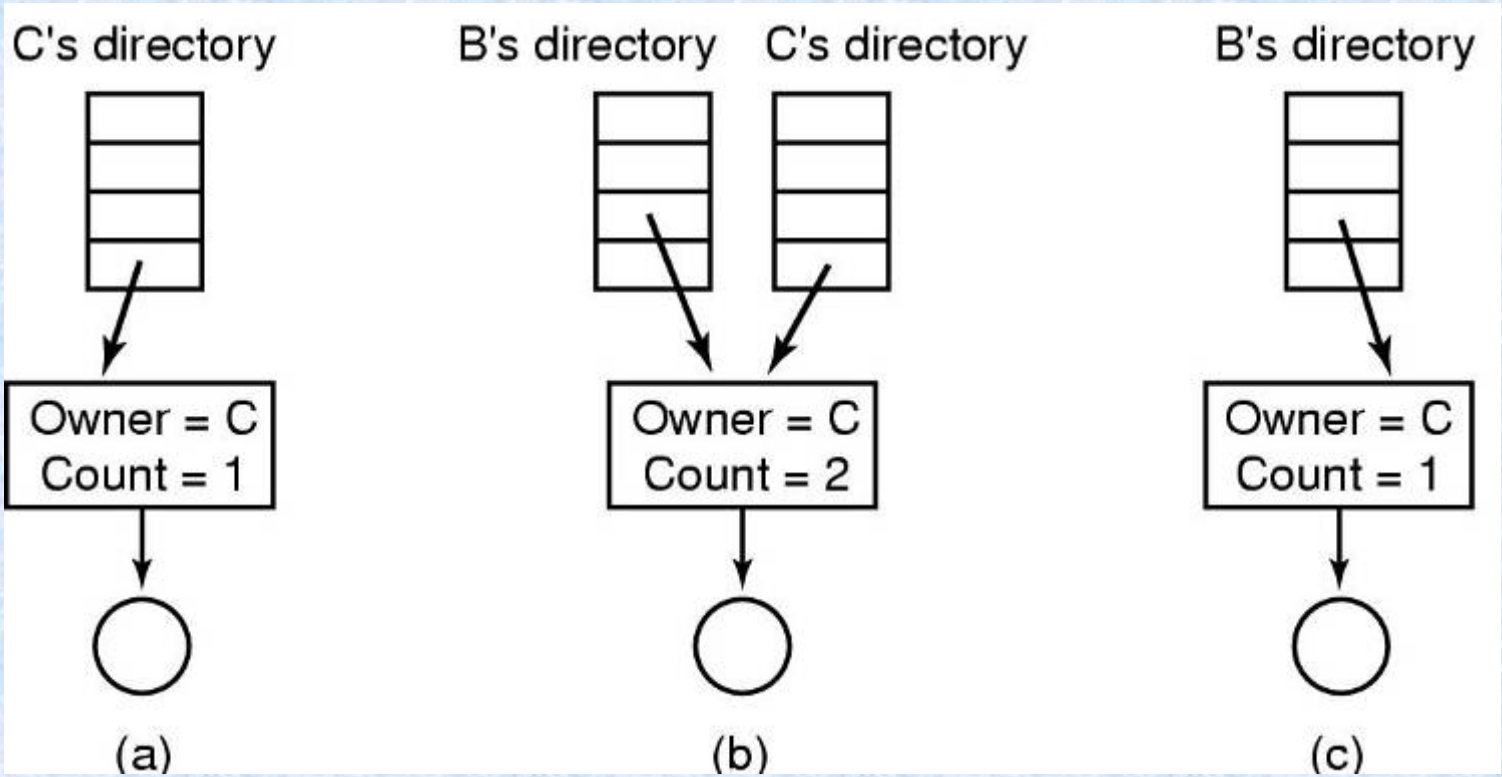
Il File System Unix
Organizzazione logica



Collegamento (linking) di file (1)



Collegamento (linking) di file (2)

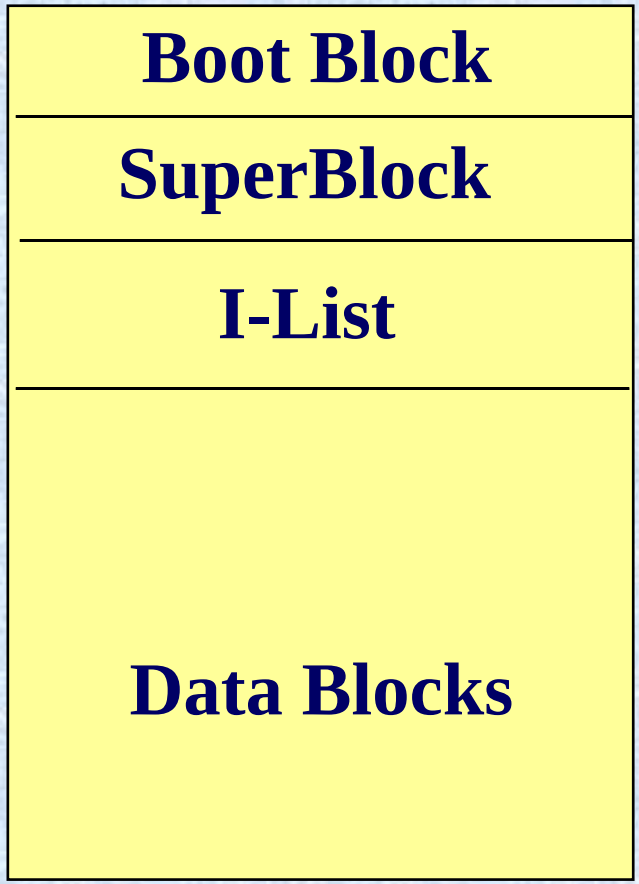


(a) situazione precedente al linking

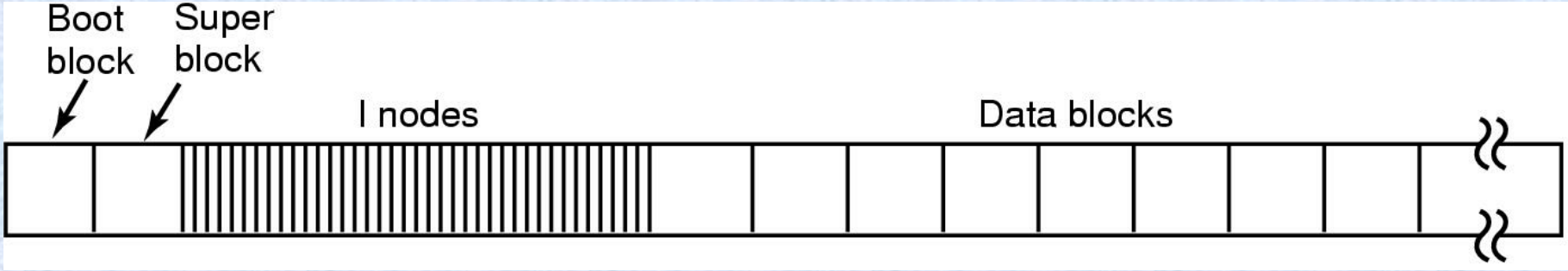
(b) dopo la creazione del link

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

Il File System
Organizzazione Fisica



Organizzazione fisica del disco nei sistemi UNIX



i-node

È il descrittore del file.

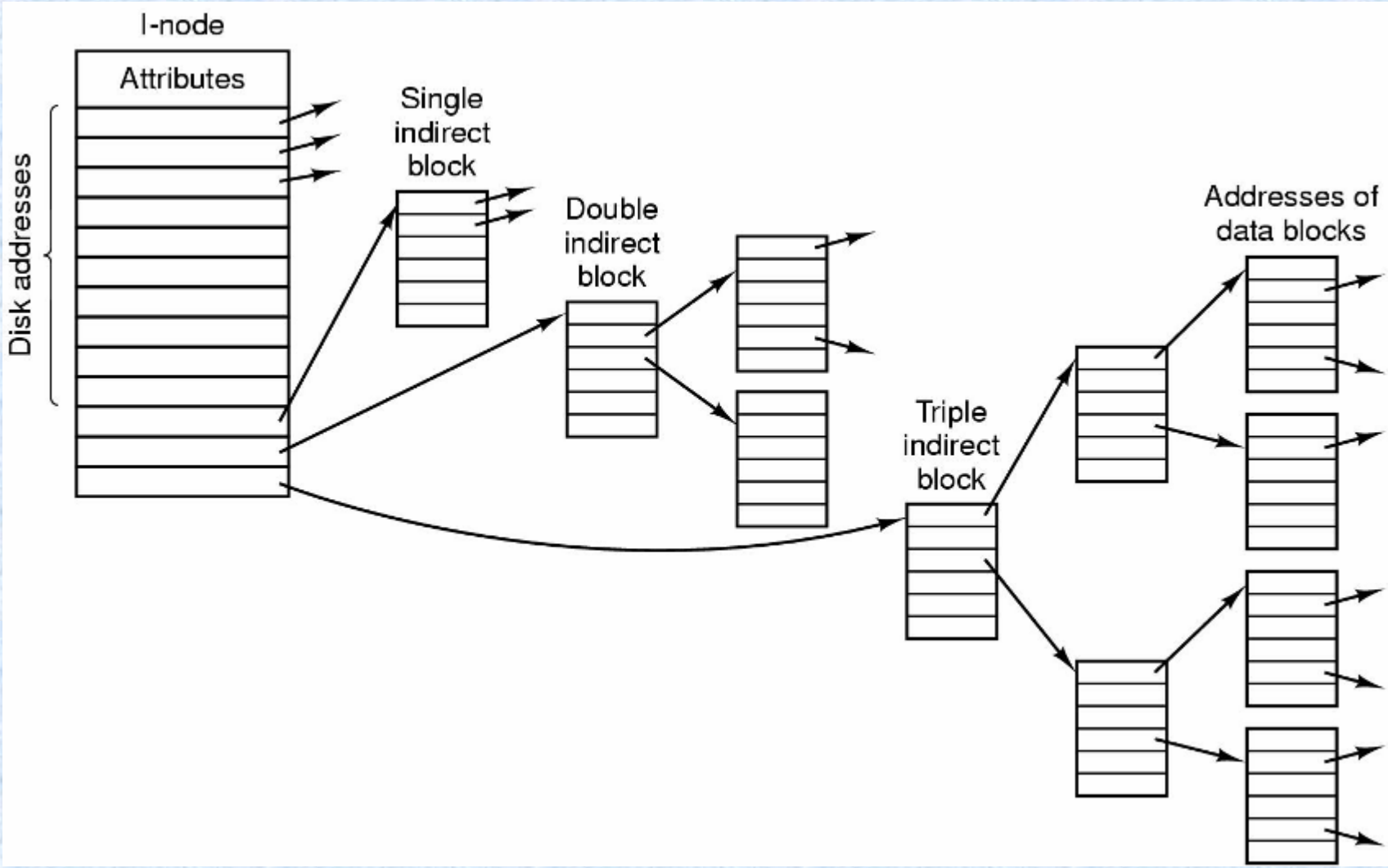
Tra gli attributi contenuti nell'*i-node* vi sono:

- **tipo** di file:
  - ✓ ordinario
  - ✓ direttorio
  - ✓ file speciale
- proprietario, gruppo (***user-id, group-id***)
- dimensione
- data
- 12 bit di **protezione**
- numero di **links**
- **13 - 15 indirizzi di blocchi (a seconda della realizzazione)**

Struttura di un i-node

Field	Bytes	Description
Mode	2	File type, protection bits, setuid, setgid bits
Nlinks	2	Number of directory entries pointing to this i-node
Uid	2	UID of the file owner
Gid	2	GID of the file owner
Size	4	File size in bytes
Addr	39	Address of first 10 disk blocks, then 3 indirect blocks
Gen	1	Generation number (incremented every time i-node is reused)
Atime	4	Time the file was last accessed
Mtime	4	Time the file was last modified
Ctime	4	Time the i-node was last changed (except the other times)

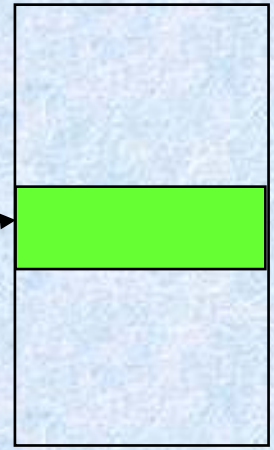
Indirizzamento dei blocchi tramite i-node e blocchi indiretti



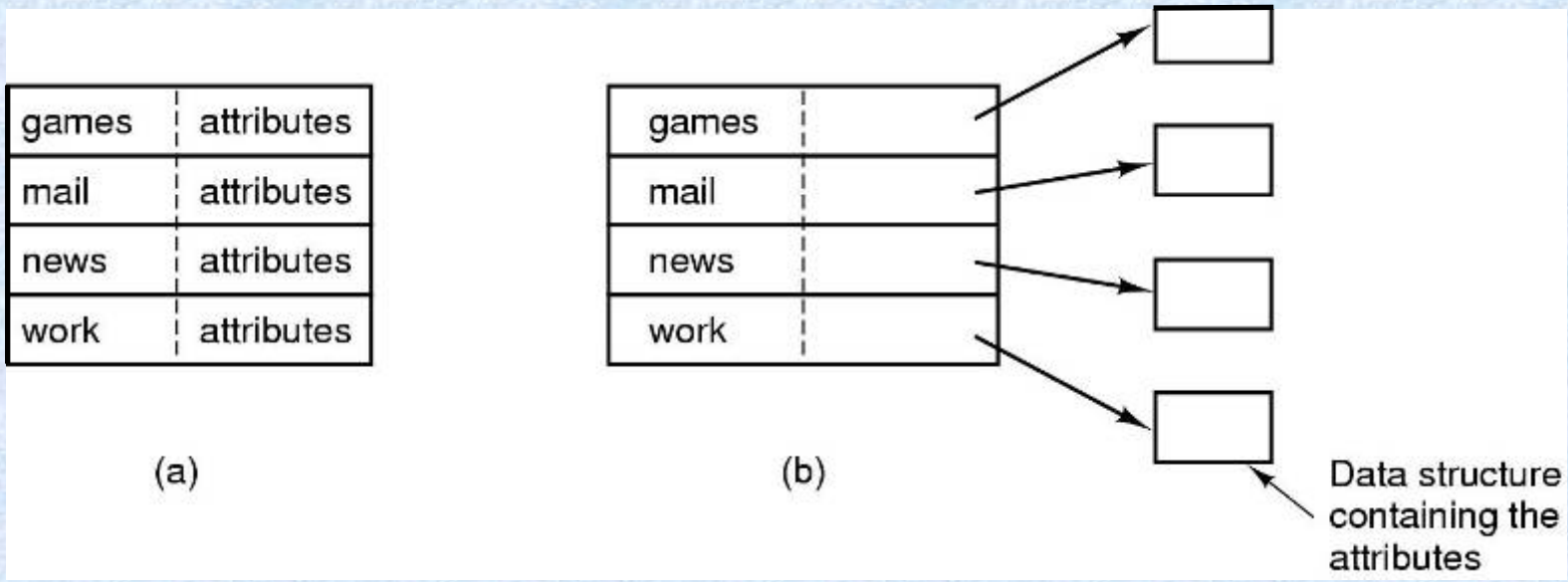
Il Direttorio

il file (direttorio) `/usr/mario` contiene:

file1	189
file2	133
miodir	121
.	110
..	89



Alternative per l'implementazione delle Directory

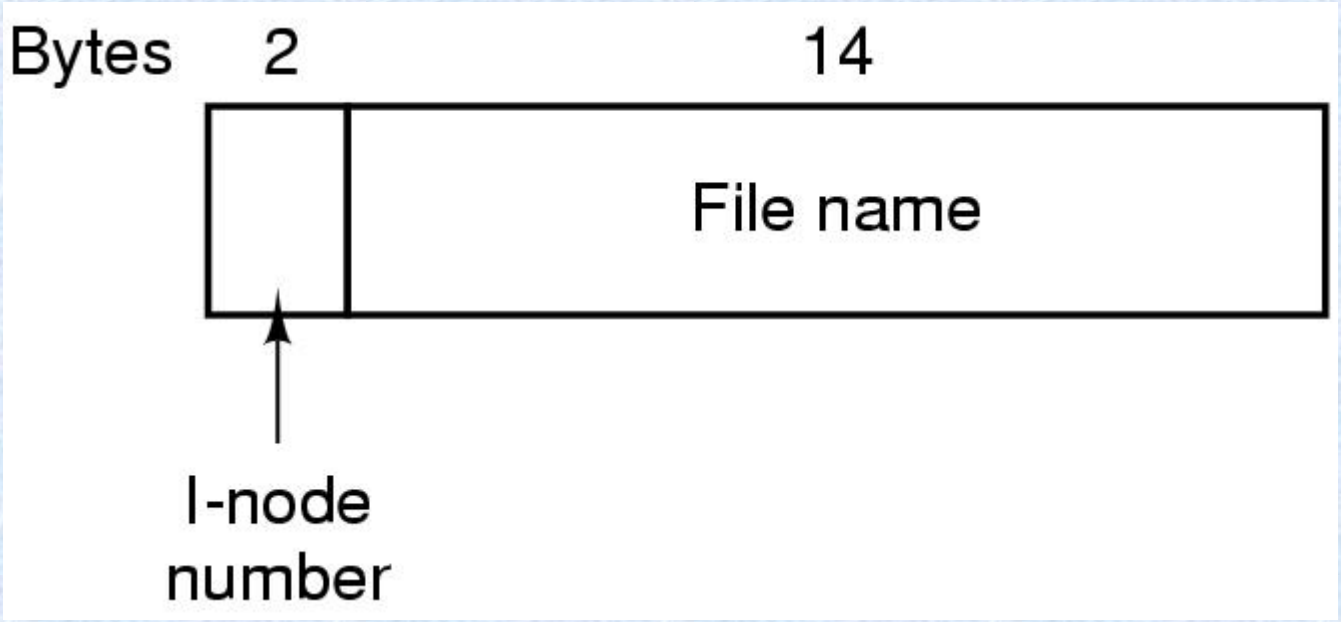


(a) Una semplice directory

- **elementi di ampiezza fissa**
- **contiene sia attributi, sia indirizzi su disco**

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

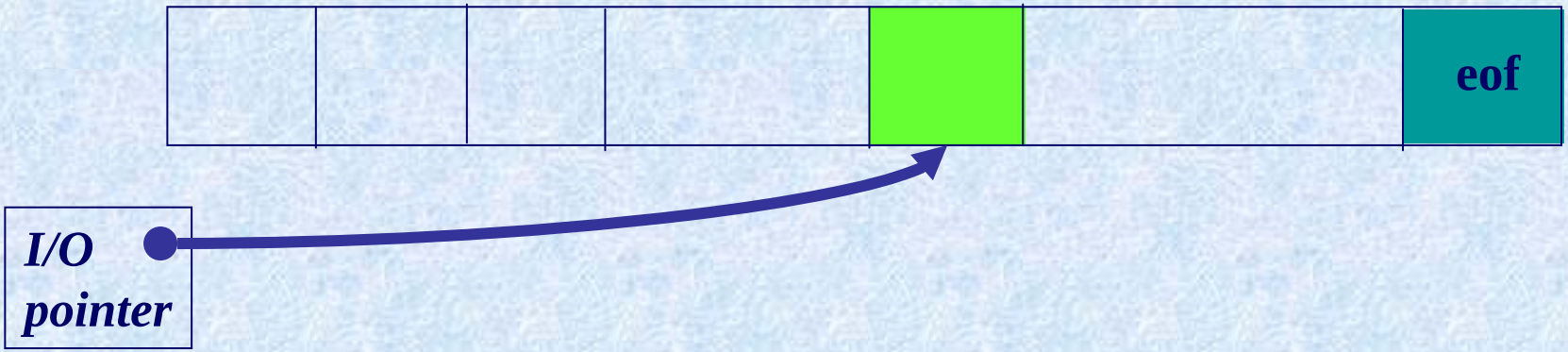
Il File System di UNIX V7 (1)



La rappresentazione di una directory in UNIX V7

Accesso a File: Concetti Generali

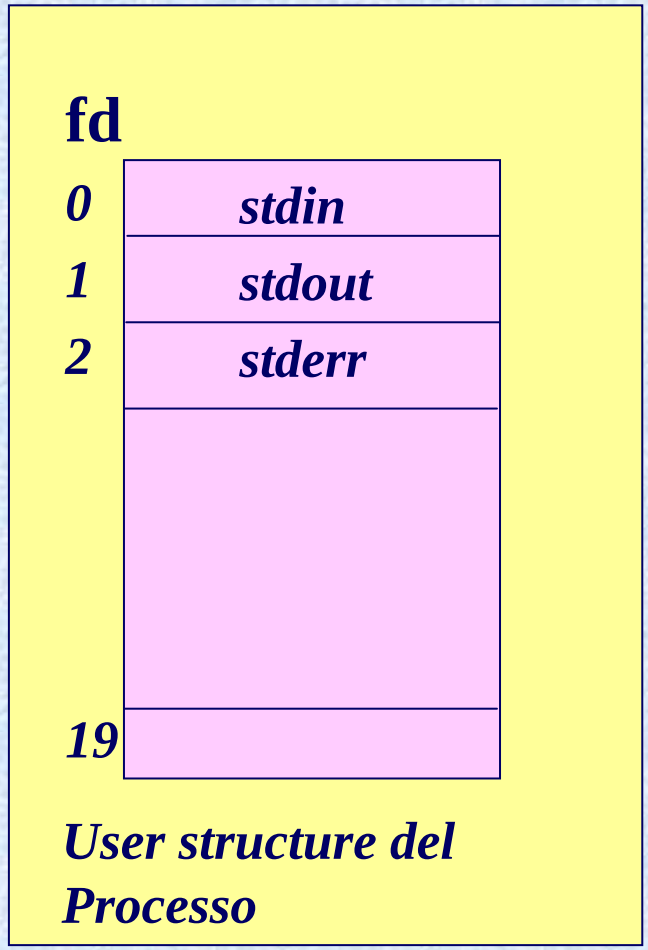
- accesso sequenziale
- assenza di strutturazione: file = sequenza di bytes
- posizione corrente: *I/O Pointer*



- vari modi di accesso (*lettura*, *scrittura*, *lettura/scrittura*, etc.)
- accesso subordinato all'operazione di apertura

Strutture dati del Kernel per l'accesso a file

- A ogni processo è associata una *tabella dei file aperti*
- ogni elemento della tabella rappresenta un file aperto dal processo, individuato da un indice intero: *file descriptor*
- i file descriptor 0,1,2 individuano rispettivamente *standard input, output, error* (aperti automaticamente)
- la tabella dei file aperti del processo è allocata nella sua *user structure*

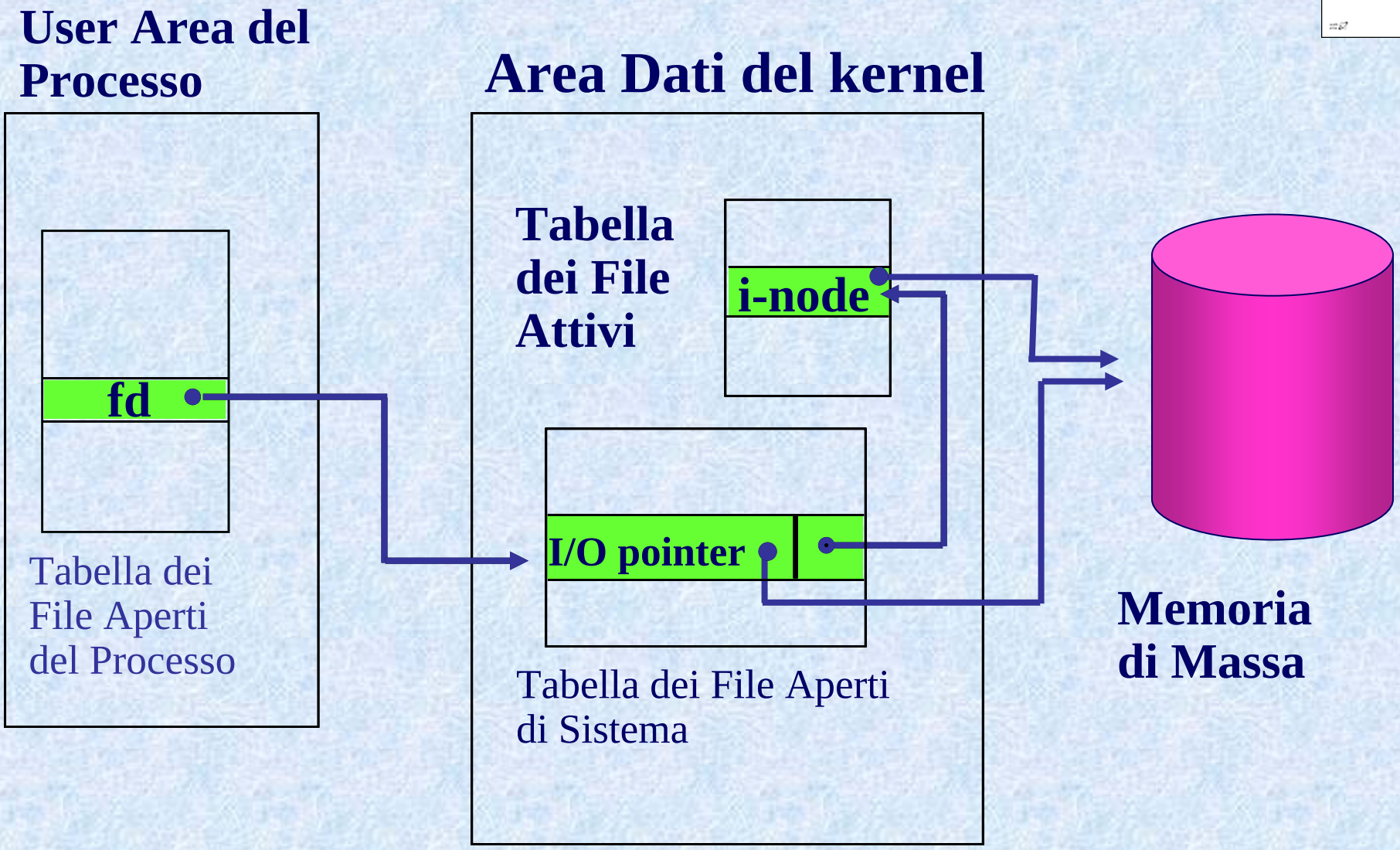


Strutture dati del kernel per l'accesso a file

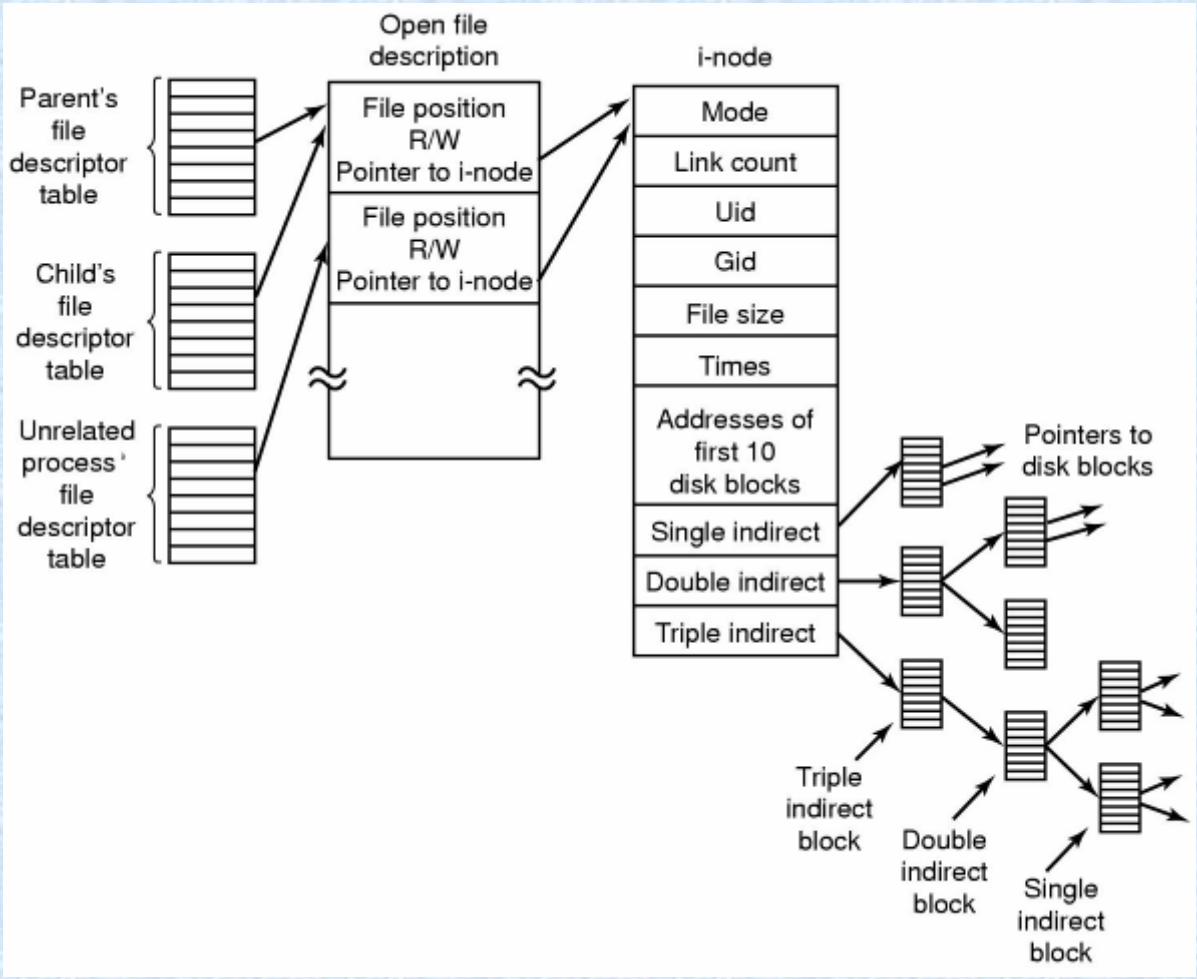
Riassumendo:

- tabella dei file aperti di processo
- tabella dei file aperti di sistema:
- tabella dei file attivi

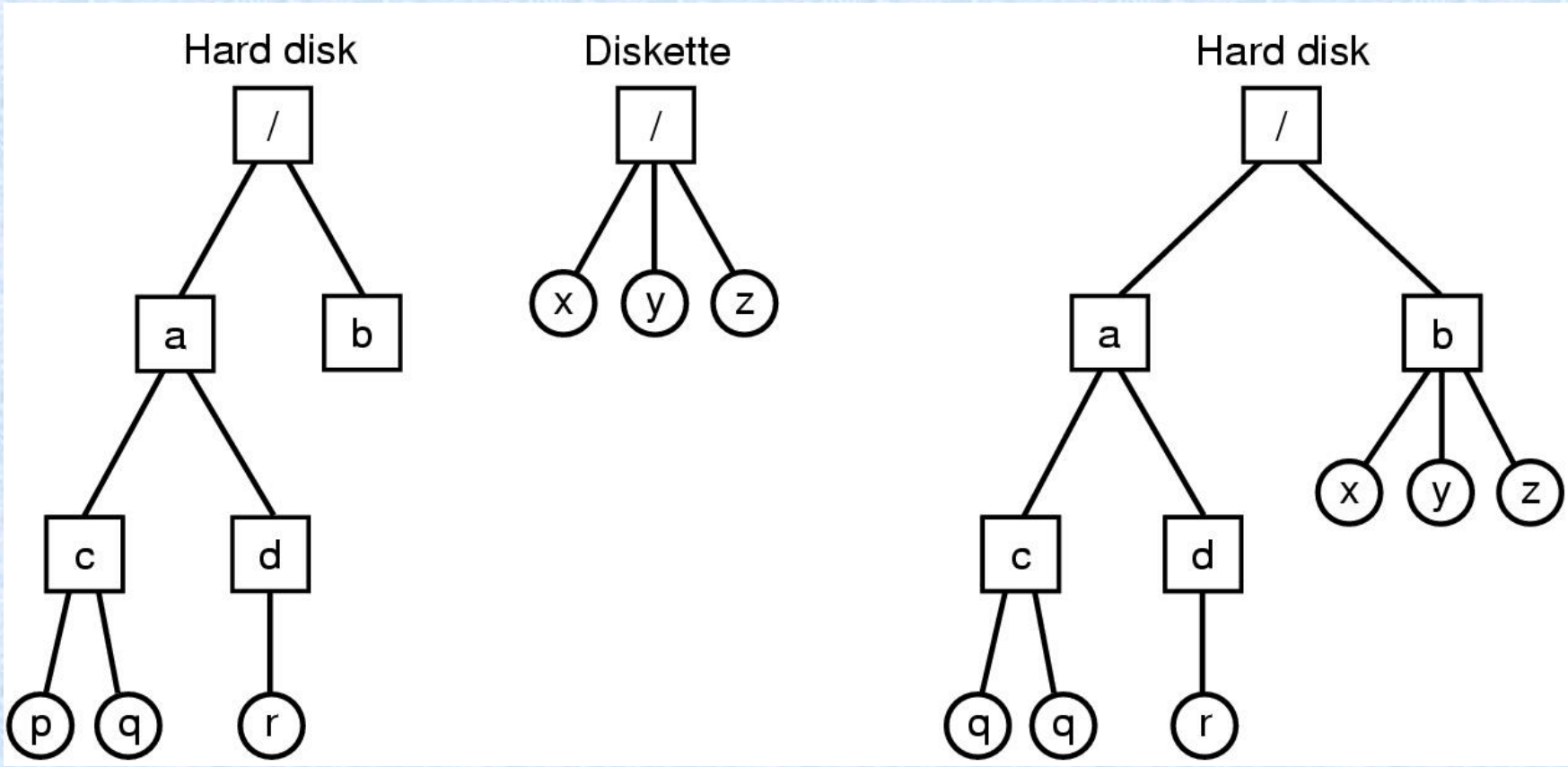
Quali sono le relazioni tra queste strutture?



Relazione fra *file descriptor* e *open file descriptor*



Montaggio di un File System



(a) (b)

(a) prima del *mounting*. (b) dopo il *mounting*

7.6.5 System Call per l'accesso a file

Unix **permette ai processi di accedere a file, mediante un insieme di** *system call*, **tra le quali:**

- ✓ **apertura/creazione:** `open, creat`
- ✓ **chiusura:** `close`
- ✓ **lettura:** `read`
- ✓ **scrittura:** `write`
- ✓ **cancellazione:** `unlink`
- ✓ **linking:** `link`
- ✓ **accesso diretto:** `lseek`

Apertura di File: open

```
int open(char nomefile[],int flag, [int mode]);
```

- `nomefile` è il nome del file (relativo o assoluto)
- `flag` esprime il modo di accesso (ad es. `O_RDONLY`, per accesso in lettura, `O_WRONLY`, per accesso in scrittura)
- `mode` è un parametro richiesto soltanto se l'opzione di apertura determina la **creazione** del file(`O_CREAT`): in tal caso, `mode` specifica i bit di protezione

- il valore restituito dalla `open` è il *file descriptor* associato al file, o -1 in caso di errore.

Chiusura di File: `close`

Per chiudere un file aperto:

```
int close(int fd);
```

- `fd` è il file descriptor del file da chiudere.

- Restituisce l'esito della operazione (0 in caso di successo, <0 in caso di insuccesso).
- Se la `close` ha successo: il file viene memorizzato sul disco e vengono eliminati gli elementi associati ad esso nelle tabelle del kernel.

Lettura e Scrittura di File

Caratteristiche:

- accesso mediante il file descriptor
- ogni operazione di accesso (lettura o scrittura) agisce sequenzialmente sul file, a partire dalla posizione corrente del puntatore (I/O pointer)
- atomicità della singola operazione.
- operazioni sincrone, cioè con attesa del completamento dell'operazione.
- possibilità di alternare operazioni di lettura e scrittura

Lettura di File: system call `read`

```
int read(int fd,char *buf,int n);
```

- `fd` è il file descriptor del file
- `buf` è l'area in cui trasferire i byte letti
- `n` è il numero di caratteri da leggere

- in caso di successo, restituisce un intero positivo (≤ n) che rappresenta il numero di caratteri effettivamente letti

Scrittura di File: system call `write`

```
int write(int fd,char *buf,int n);
```

- `fd` è il file descriptor del file
- `buf` è l'area da cui trasferire i byte scritti
- `n` è il numero di caratteri da scrivere

- in caso di *successo*, restituisce un intero positivo (= = n) che rappresenta il numero di caratteri effettivamente scritti

System call `lseek`

```
off_t lseek(int fd, off_t offset, int whence);
```

- `fd` è il file descriptor del file
- `offset` è lo scostamento da aggiungere alla posizione del file rispetto al parametro `whence`
- `whence` può essere:
  - ✓`SEEK_SET` : in questo caso la nuova posizione è `offset`
  - ✓`SEEK_CUR` : in questo caso `offset` è aggiunto alla posizione corrente del file
  - ✓`SEEK_END` : in questo caso la nuova posizione è ottenuta aggiungendo `offset` (che può essere negativo) alla fine del file
- in caso di successo, restituisce la posizione corrente del file
- In caso di fallimento restituisce -1

8.5 File System di Windows

- Il file system delle ultime versioni di Windows (*NTFS - Native NT File System*) è di tipo transazionale con capacià di recupero dopo una caduta del sistema (*crash*).
- Fornisce il supporto al concetto di partizioni logica (*volume*).
- Ogni *volume* è suddiviso in *cluster* (insieme di settori fisici del disco) che rappresentano le unità elementari di allocazione dei dati.

File System

- I file sono organizzati in directory secondo la classica struttura ad albero.
- È possibile creare riferimenti simbolici ai file dando luogo a strutture più complesse quali grafi aciclici.
- Ogni file è caratterizzato da un insieme di *attributi*

File System

- **Attributi di un file:**
  - ✓ **nome;**
  - ✓ **percorso assoluto (*absolute path*);**
  - ✓ **dimensioni;**
  - ✓ **permessi di accesso;**
  - ✓ **vari tipi di statistiche;**
  - ✓ **contenuto del file;**
  - ✓ **……**

File System

- Ogni file viene descritto da un'opportuna struttura dati contenuta in un file speciale detto *MTF* (*Master File Table*).
- Gli attributi di piccole dimensioni sono contenuti direttamente nella *MTF* (*attributi residenti*).
- Gli attributi di grandi dimensioni, come i contenuti del file, sono memorizzati in una serie di cluster sul disco (*estensioni contigue*).

API del File System di Windows (1)

Win32 API function	UNIX	Description
CreateFile	open	Create a file or open an existing file; return a handle
DeleteFile	unlink	Destroy an existing file
CloseHandle	close	Close a file
ReadFile	read	Read data from a file
WriteFile	write	Write data to a file
SetFilePointer	lseek	Set the file pointer to a specific place in the file
GetFileAttributes	stat	Return the file properties
LockFile	fcntl	Lock a region of the file to provide mutual exclusion
UnlockFile	fcntl	Unlock a previously locked region of the file

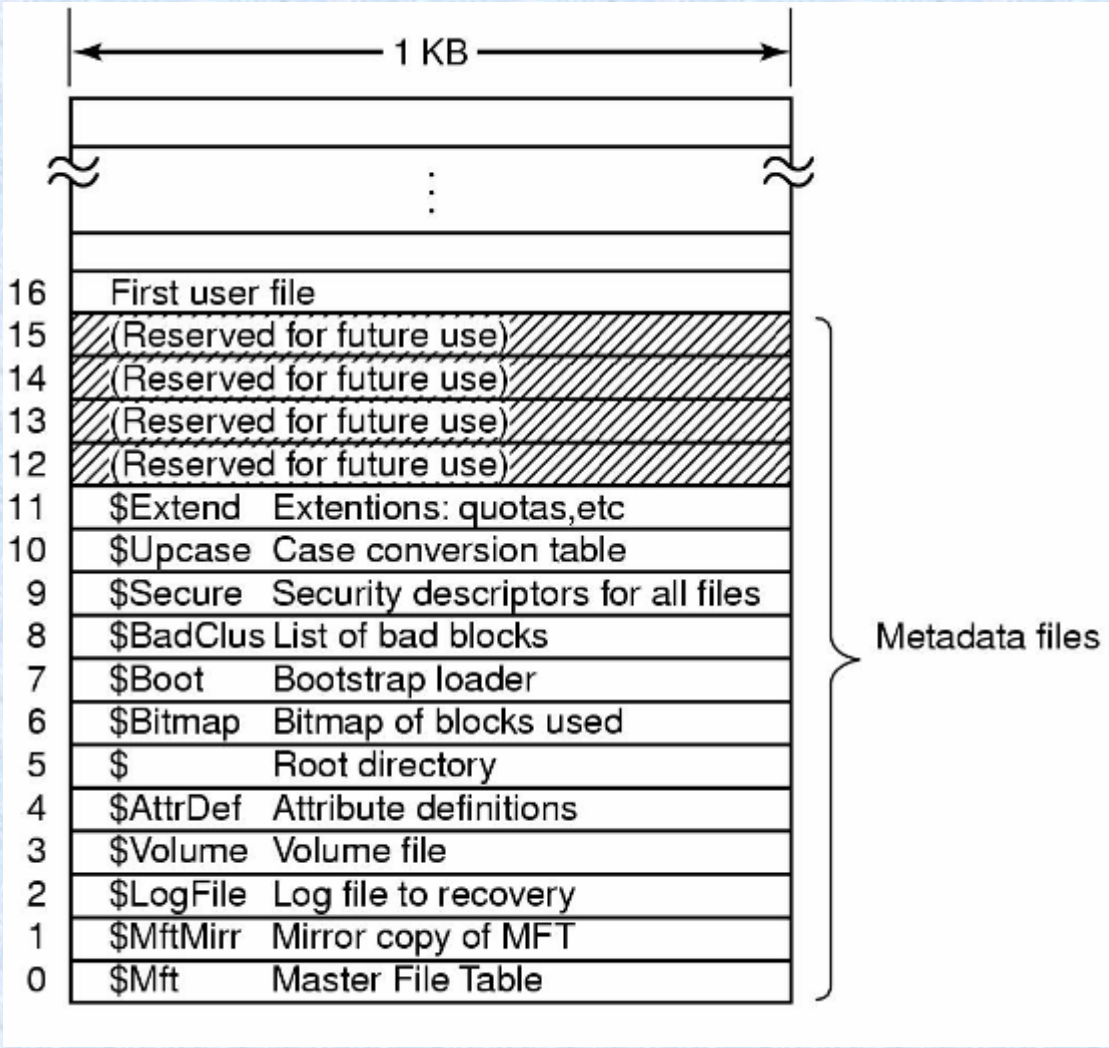
La seconda colonna indica la chiamata di sistema Unix equivalente

API del File System di Windows (2)

Win32 API function	UNIX	Description
CreateDirectory	mkdir	Create a new directory
RemoveDirectory	rmdir	Remove an empty directory
FindFirstFile	opendir	Initialize to start reading the entries in a directory
FindNextFile	readdir	Read the next directory entry
MoveFile	rename	Move a file from one directory to another
SetCurrentDirectory	chdir	Change the current working directory

La seconda colonna indica la chiamata di sistema Unix equivalente

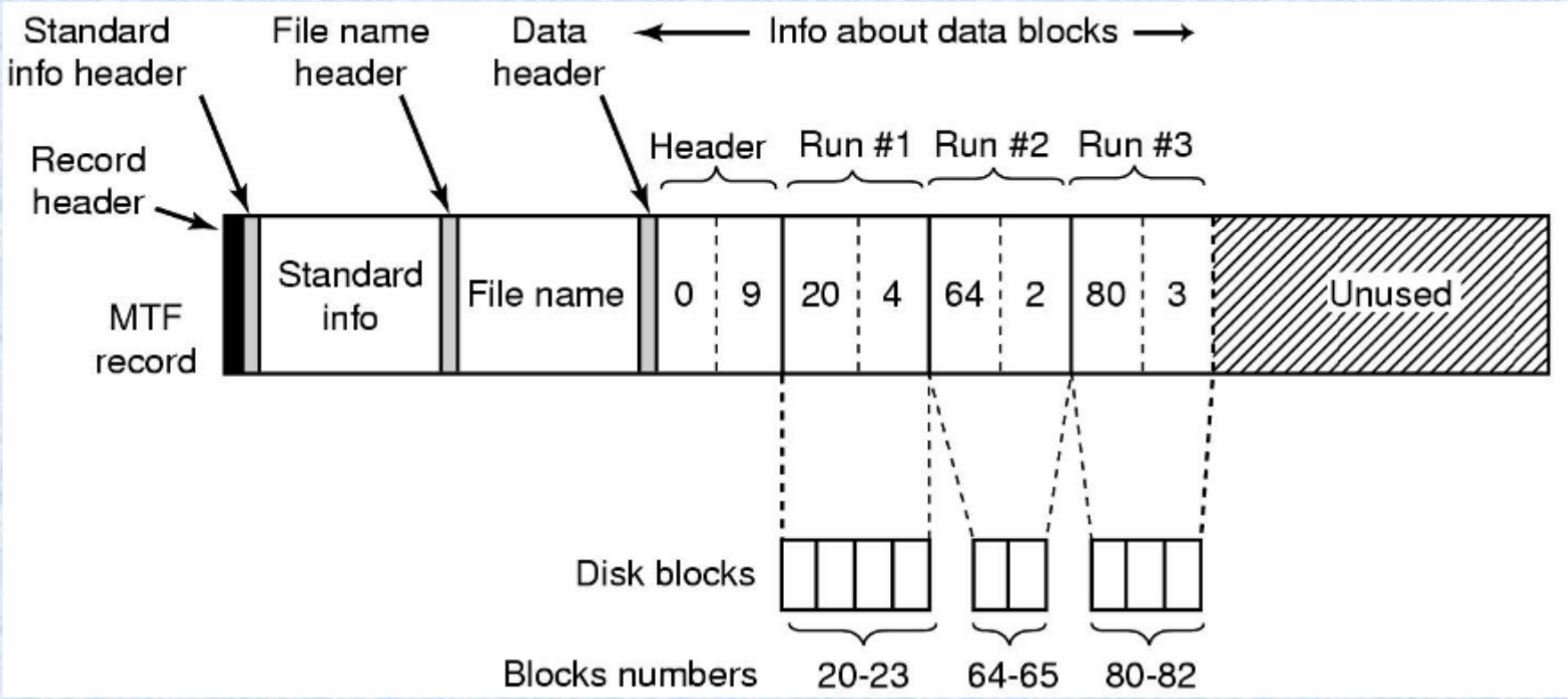
La Master File Table (MFT) di NTFS



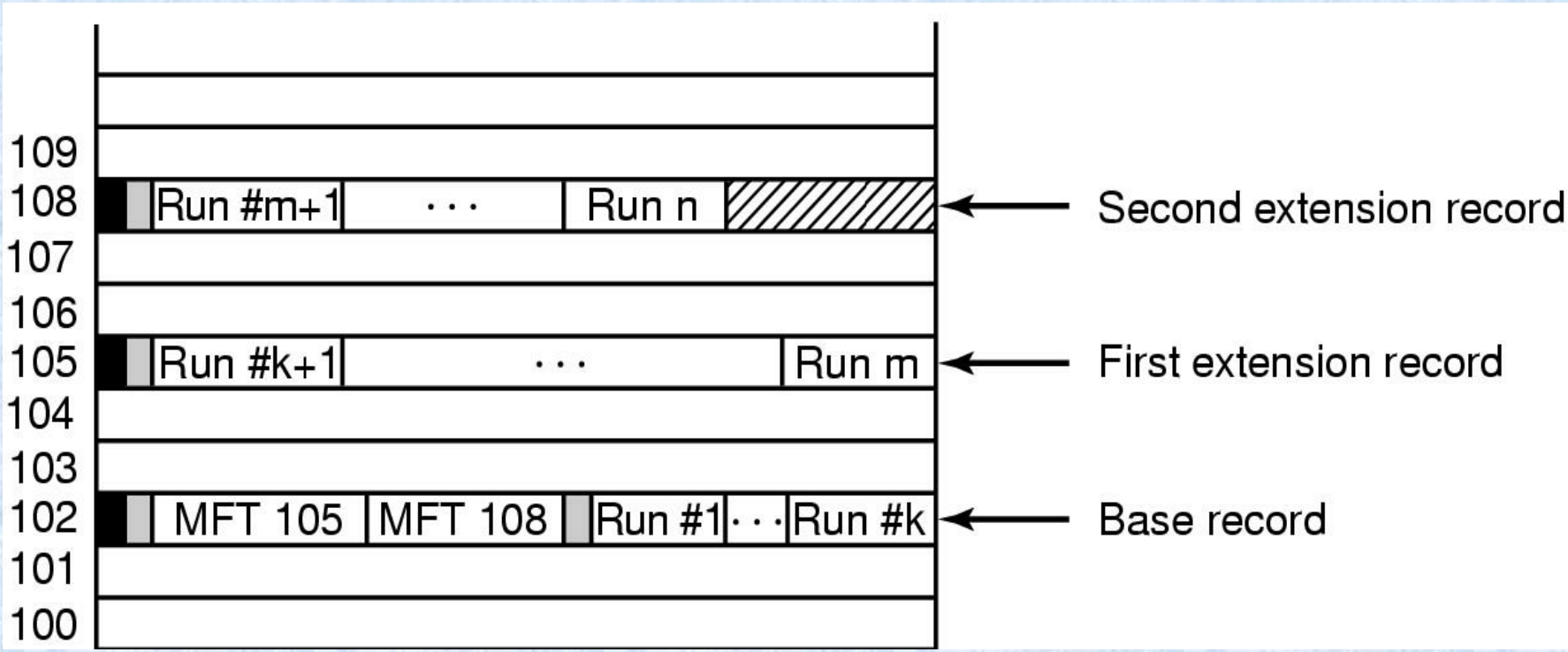
Attributi dei record della MFT

Attribute	Description
Standard information	Flag bits, timestamps, etc.
File name	File name in Unicode; may be repeated for MS-DOS name
Security descriptor	Obsolete. Security information is now in $Extend$Secure
Attribute list	Location of additional MFT records, if needed
Object ID	64-bit file identifier unique to this volume
Reparse point	Used for mounting and symbolic links
Volume name	Name of this volume (used only in $Volume)
Volume information	Volume version (used only in $Volume)
Index root	Used for directories
Index allocation	Used for very large directories
Bitmap	Used for very large directories
Logged utility stream	Controls logging to $LogFile
Data	Stream data; may be repeated

Un record MFT per un file di 3 *run* e di 9 blocchi



Un file che richiede 3 MFT record per memorizzare i suoi *run*



Il record MFT di una piccola directory