

Il File System

Il file system

E' quella componente del SO che fornisce i *meccanismi di accesso e memorizzazione* delle informazioni (programmi e dati) *allocate in memoria di massa*

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, directory e partizione sono **del tutto indipendenti** da natura e tipo di dispositivo utilizzato

Organizzazione del File System

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



Organizzazione del file system

- **Struttura logica:** presenta alle applicazioni una visione astratta delle informazioni memorizzate, basata su **file**, **directory**, **partizioni**, ecc.. Realizza le operazioni di gestione di file e directory: copia, cancellazione, spostamento, ecc.
- **Accesso:** definisce e realizza i meccanismi per accedere al contenuto dei file; in particolare:
 - Definisce l'unità di trasferimento da/verso file: **record logico**
 - Realizza i metodi di accesso (sequenziale, casuale, ad indice)
 - Realizza i meccanismi di protezione
- **Organizzazione fisica:** rappresentazione di file e directory sul dispositivo:
 - Allocazione dei file sul dispositivo (unità di memorizzazione = blocco): mapping di record logici su blocchi. Vari metodi di allocazione.
 - Rappresentazione della struttura logica sul dispositivo.
- **Dispositivo Virtuale:** appoggiandosi direttamente sull'HW presenta una vista astratta del dispositivo, che appare come una sequenza di blocchi di dimensione costante.

File

È un insieme di informazioni; ad es.:

- programmi
 - dati (in rappresentazione binaria)
 - dati (in rappresentazione testuale)
 - ...
-
- Ogni file è *individuato da (almeno) un nome simbolico* mediante il quale può essere riferito (ad esempio, nell'invocazione di comandi o system call)
 - Ogni file è *caratterizzato da un insieme di attributi*

Attributi del file

A seconda del SO, i file possono avere attributi diversi.
Solitamente:

- **tipo**: stabilisce l'appartenenza a una classe (eseguibili, testo, musica, non modificabili, ...)
- **indirizzo**: puntatore/i a memoria secondaria
- **dimensione**: numero di byte contenuti nel file
- **data e ora** (di creazione e/o di modifica)

In SO multiutente anche:

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

Attributi del file

Descrittore del file:

è la struttura dati che contiene gli attributi di un file

Ogni **descrittore** di file deve essere **memorizzato in modo persistente**:

- il SO mantiene *l'insieme dei descrittori di tutti i file presenti nel file system in apposite strutture* in memoria secondaria (ad es. UNIX: *i-list*)

Tipi di file: nomi ed estensioni

In alcuni SO,
l'estensione inclusa nel
nome di un file
rappresenta il suo tipo

NON è il caso di UNIX

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Operazioni sui file

Compito del SO è consentire *l'accesso on-line ai file* (ogni volta che un processo *modifica un file*, tale cambiamento è *immediatamente visibile* per tutti gli altri processi)

Tipiche Operazioni

- *Creazione*: allocazione di un file in memoria secondaria e inizializzazione dei suoi attributi
- *Lettura* di record logici dal file
- *Scrittura*: inserimento di nuovi record logici all'interno di file
- *Cancellazione*: eliminazione del file dal file system

➤ Ogni operazione richiederebbe la localizzazione di informazioni su disco, come:

- indirizzi dei record logici a cui accedere
 - altri attributi del file
 - record logici
- > **costo elevato**

Operazioni sui file

Per migliorare l'efficienza:

- SO mantiene *in memoria una struttura che registra i file attualmente in uso (file aperti)* - *tabella dei file aperti* per ogni file aperto {puntatore al file, posizione su disco, ...}
- viene fatto il *memory mapping dei file aperti*: i file aperti (o porzioni di essi) vengono temporaneamente copiati in memoria centrale → accessi più veloci

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 ed *eliminazione* dell'elemento corrispondente dalla *tabella dei file aperti*

Struttura interna dei file

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

Blocco: unità di **trasferimento fisico** nelle operazioni di I/O da/verso il dispositivo. Sempre di **dimensione fissa**

L'utente vede il file come un **insieme di record logici**:

Record logico: unità di **trasferimento logico** nelle operazioni di accesso al file (es. lettura, scrittura di blocchi). Di **dimensione variabile**

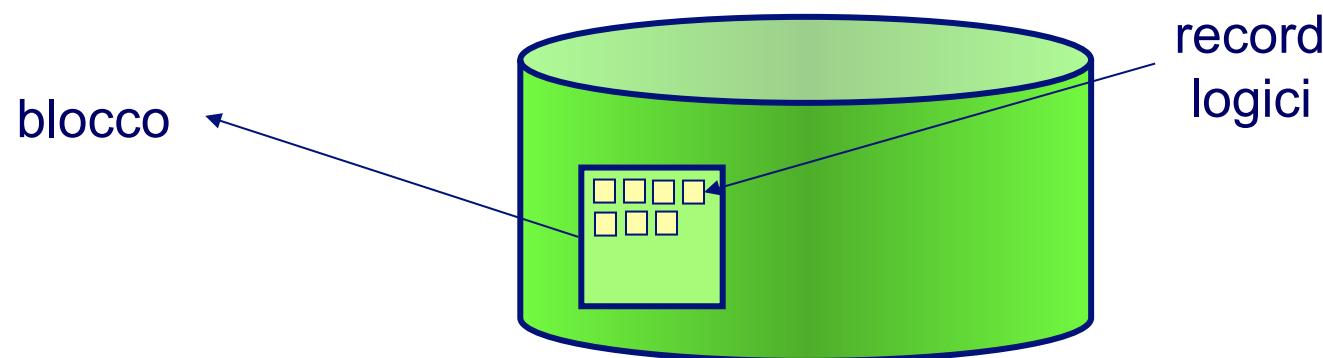
Blocchi & record logici

Uno dei compiti di SO (parte di gestione del file system) è stabilire una *corrispondenza tra record logici e blocchi*

Di solito:

Dimensione(blocco) >> Dimensione(record logico)

➤ *impaccamento* di record logici all'interno di blocchi



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

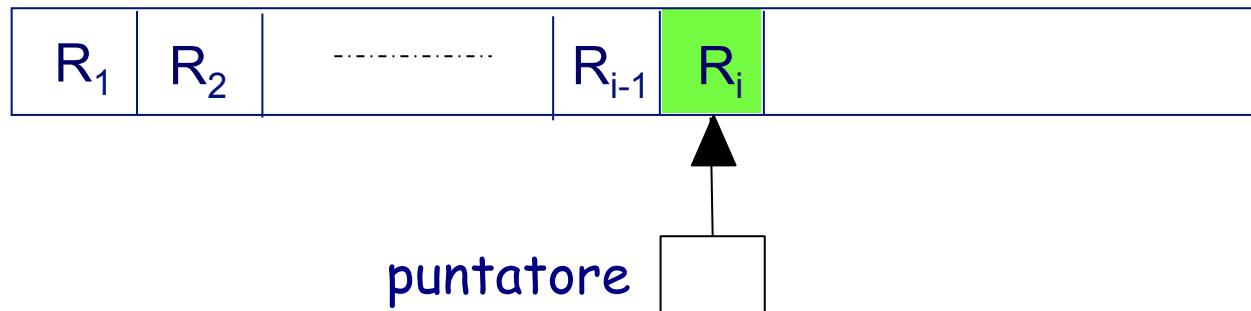
Accesso sequenziale

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

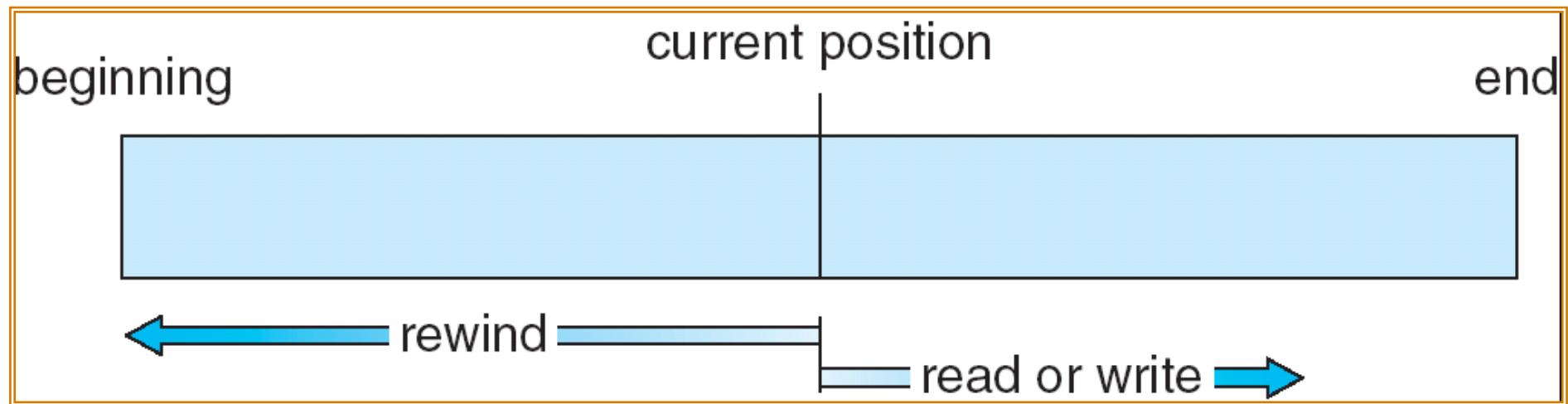
- per accedere ad un particolare record logico R_i , è necessario accedere **prima agli $(i-1)$ record che lo precedono** nella sequenza:



- le operazioni di accesso sono del tipo:
 - ✓ **readnext**: lettura del prossimo record logico della sequenza
 - ✓ **writenext**: scrittura del prossimo record logico
- È necessario registrare la posizione corrente: **puntatore al file**



Accesso sequenziale



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

UNIX prevede questo tipo di accesso

Accesso diretto

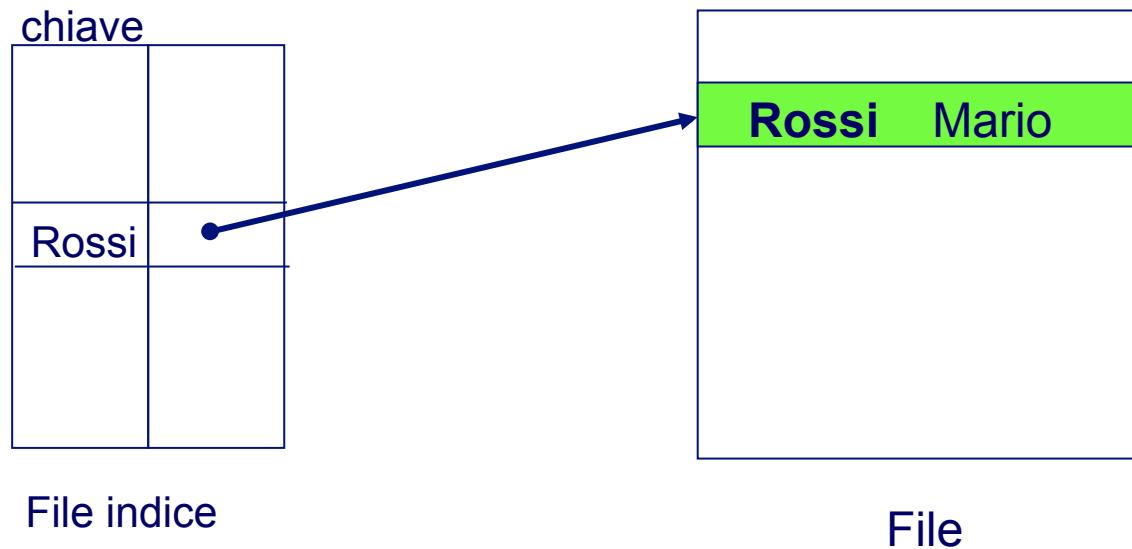
Il file è un *insieme* $\{R_1, R_2, \dots, R_N\}$ di *record logici numerati*:

- ❑ si può accedere direttamente a un particolare record logico specificandone il numero
- operazioni di accesso sono del tipo
 - ✓ *read i*: lettura del record logico i
 - ✓ *write i*: scrittura del record logico i
- ❑ Utile quando si vuole accedere a grossi file per estrarre/aggiornare poche informazioni (ad esempio nell'accesso a database)

Accesso a indice

Ad ogni file viene associata una *struttura dati* contenente l'*indice* delle informazioni contenute

- per accedere a un record logico, si esegue *una ricerca nell'indice (utilizzando una chiave)*



Directory (o direttorio)

Strumento per *organizzare i file all'interno del file system*:

- ❑ una directory può contenere più file
- ❑ è realizzata mediante una **struttura dati** che associa al nome di ogni file come e/o dove esso è allocato in memoria di massa

Operazioni sui direttori:

- **Creazione/cancellazione** di directory
- **Aggiunta/cancellazione** di file
- **Listing**: elenco di tutti i file contenuti nella directory
- **Attraversamento** della directory
- **Ricerca** di file in directory

Tipi di directory

La *struttura logica delle directory* può variare a seconda del SO

Schemi più comuni:

- ❑ *a un livello*
- ❑ *a due livelli*
- ❑ *ad albero*
- ❑ *a grafo aciclico*

Tipi di directory

Struttura a un livello: una sola directory per ogni file system



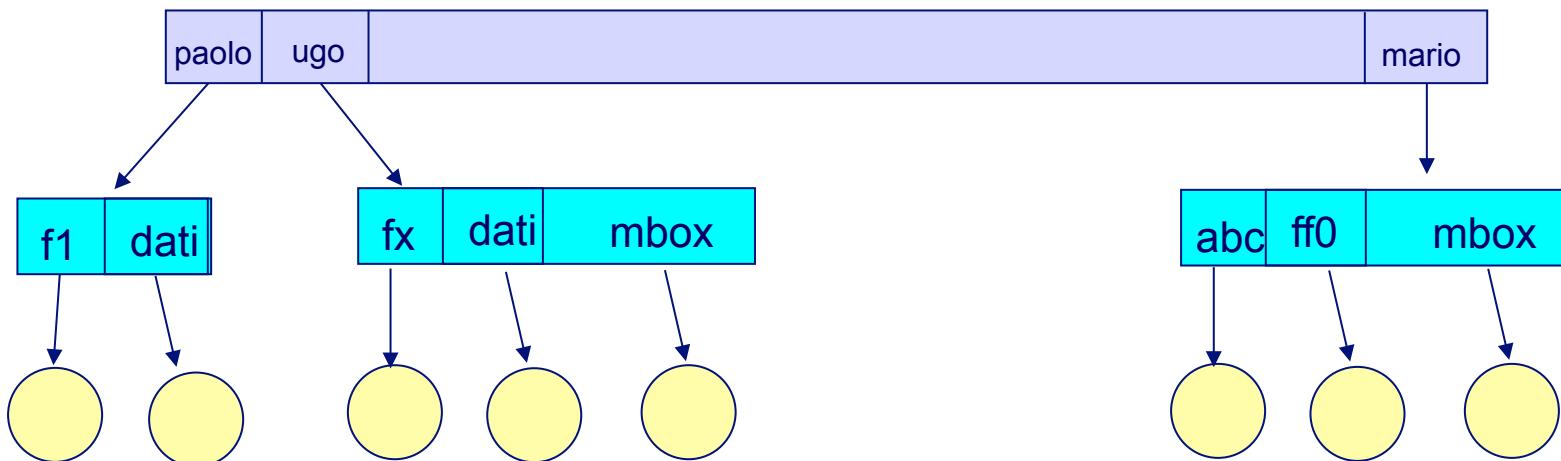
Problemi

- *unicità dei nomi*
- *multiutenza*: come separare i file dei diversi utenti?

Tipi di directory

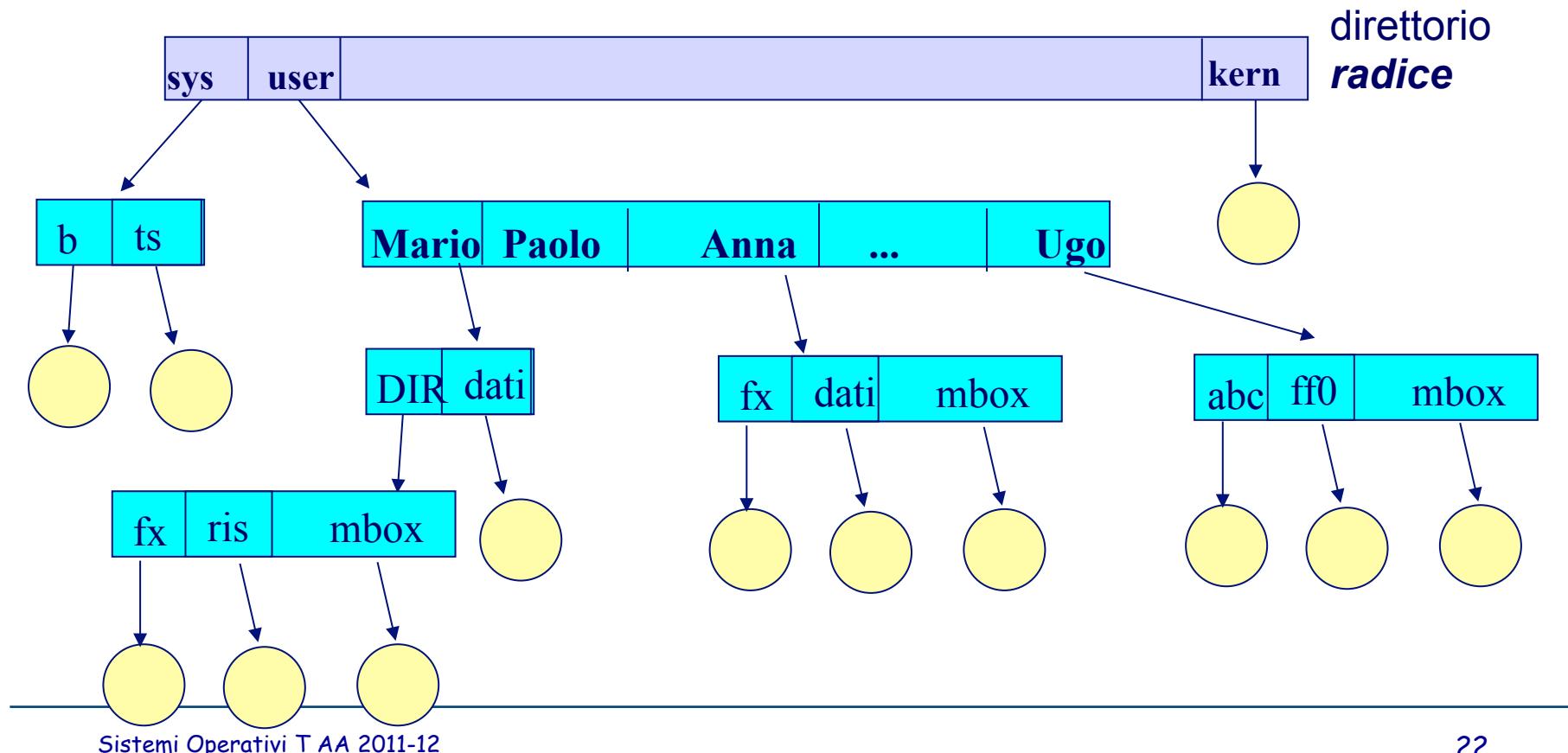
Struttura a due livelli

- primo livello (*directory principale*): contiene una directory per ogni utente del sistema
- secondo livello: *directory utenti* (a un livello)



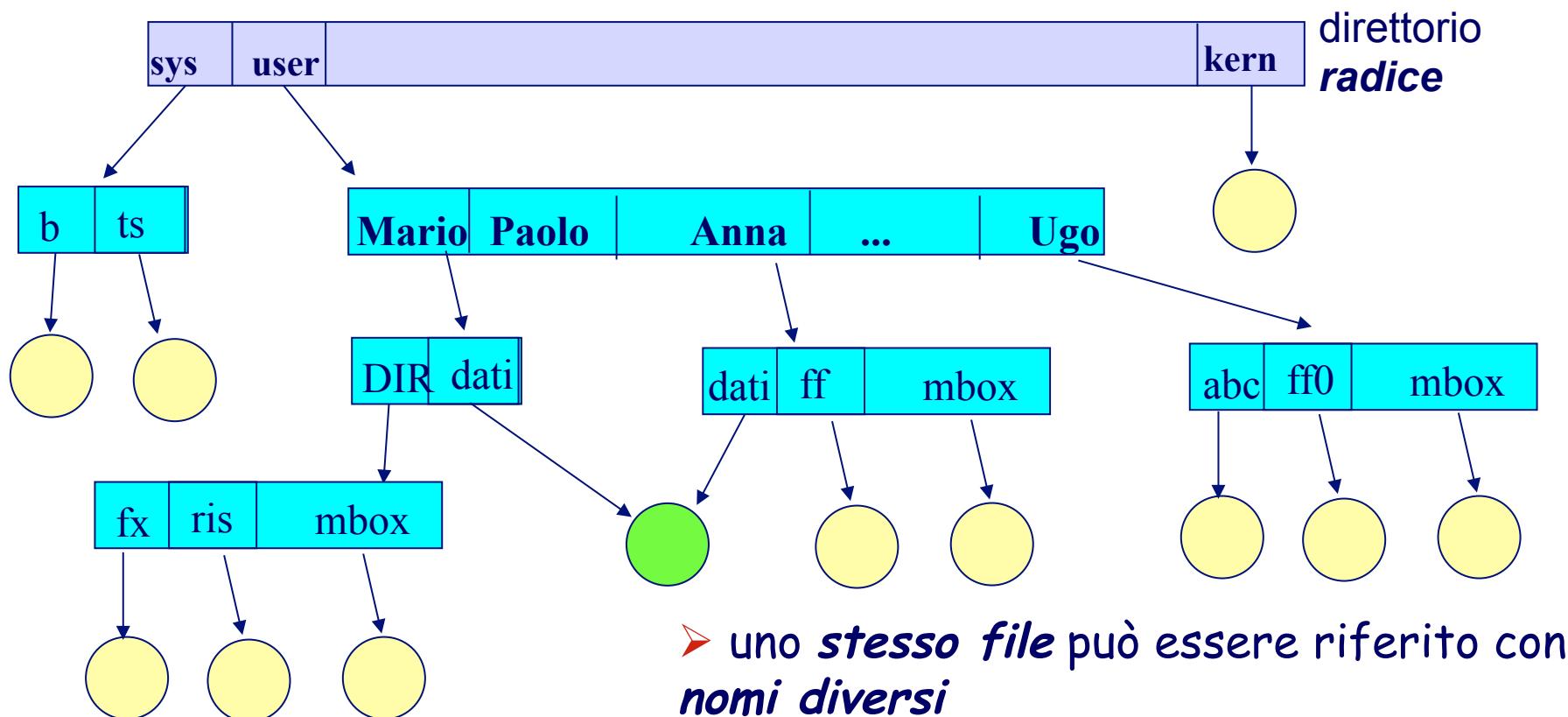
Tipi di directory

Struttura ad albero: organizzazione gerarchica a N livelli. Ogni direttorio può contenere file e altri direttori



Tipi di directory

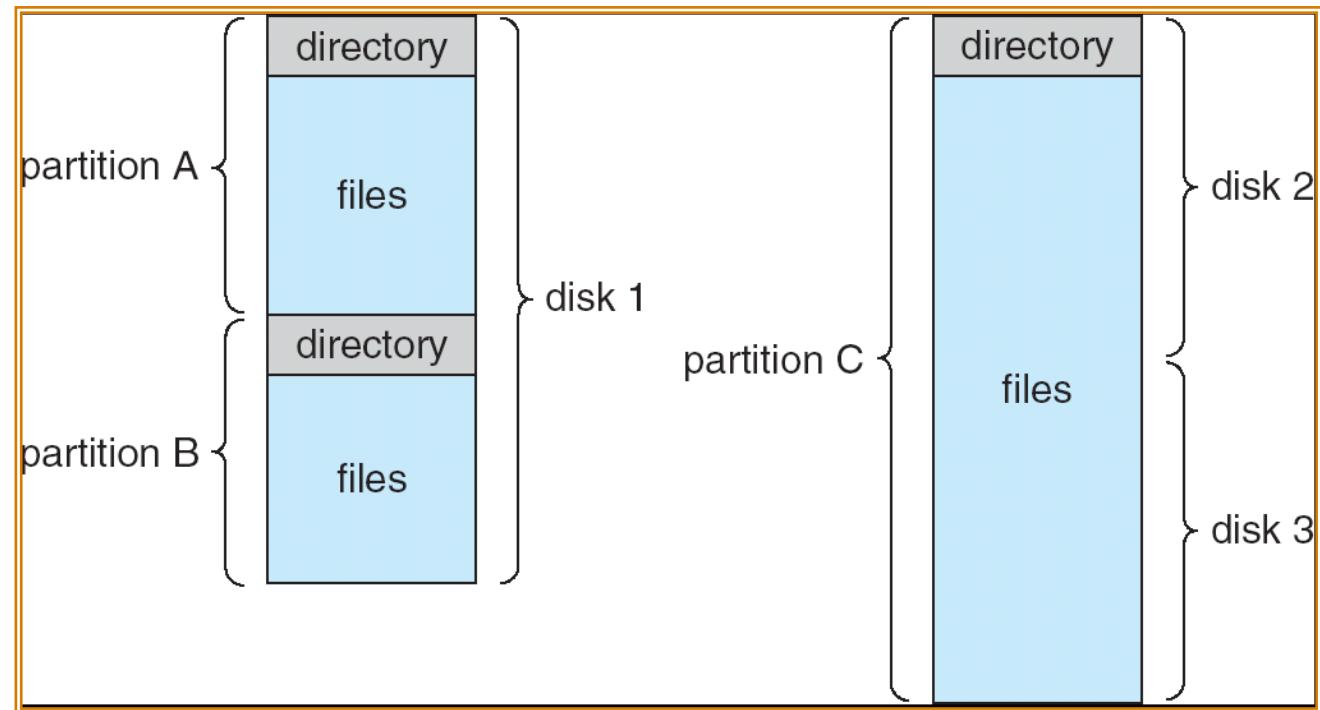
Struttura a grafo aciclico (es. UNIX): estende la struttura ad albero con la possibilità di *inserire link differenti allo stesso file*



Directory e partizioni

Una singola unità disco può contenere più partizioni

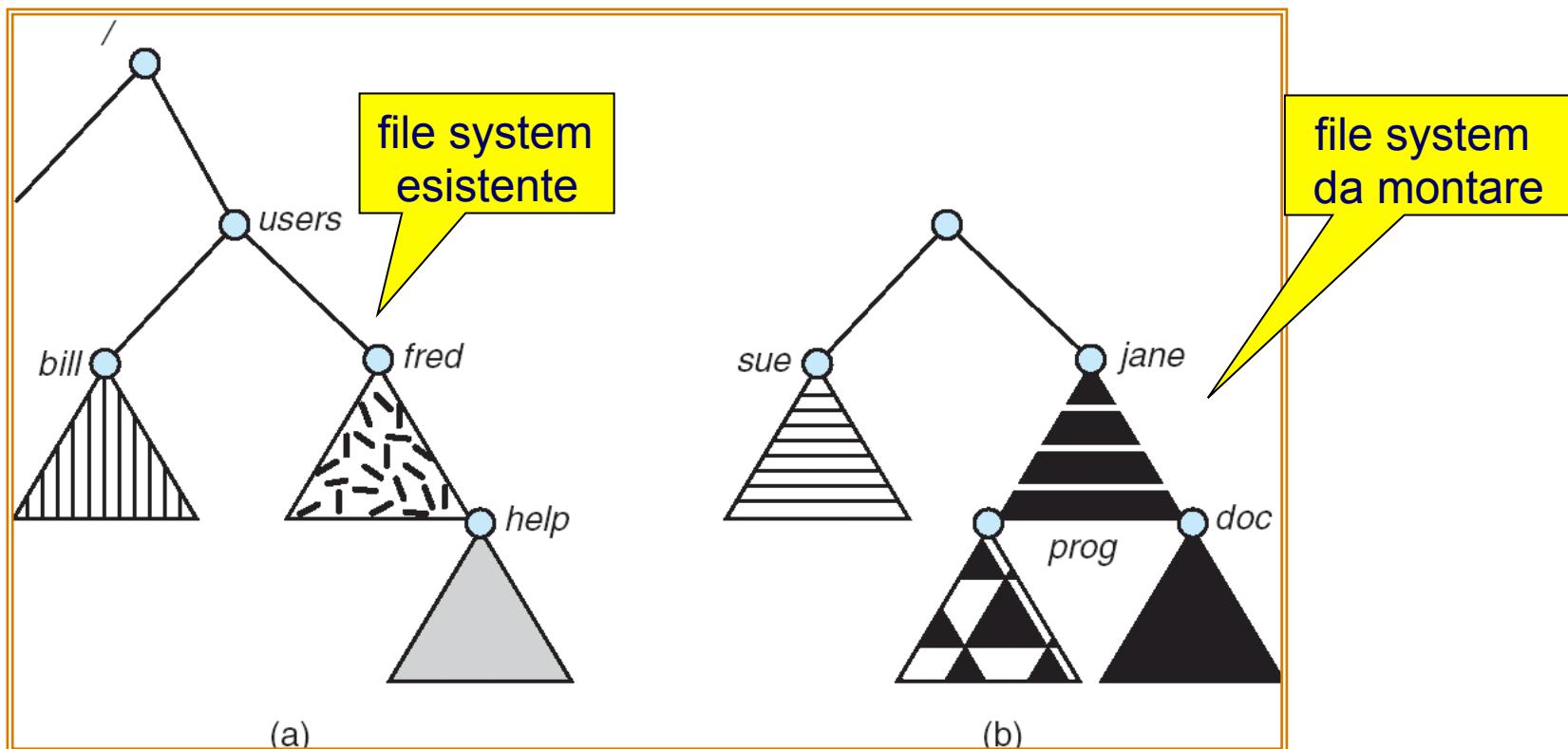
Una singola partizione può utilizzare più di una unità disco



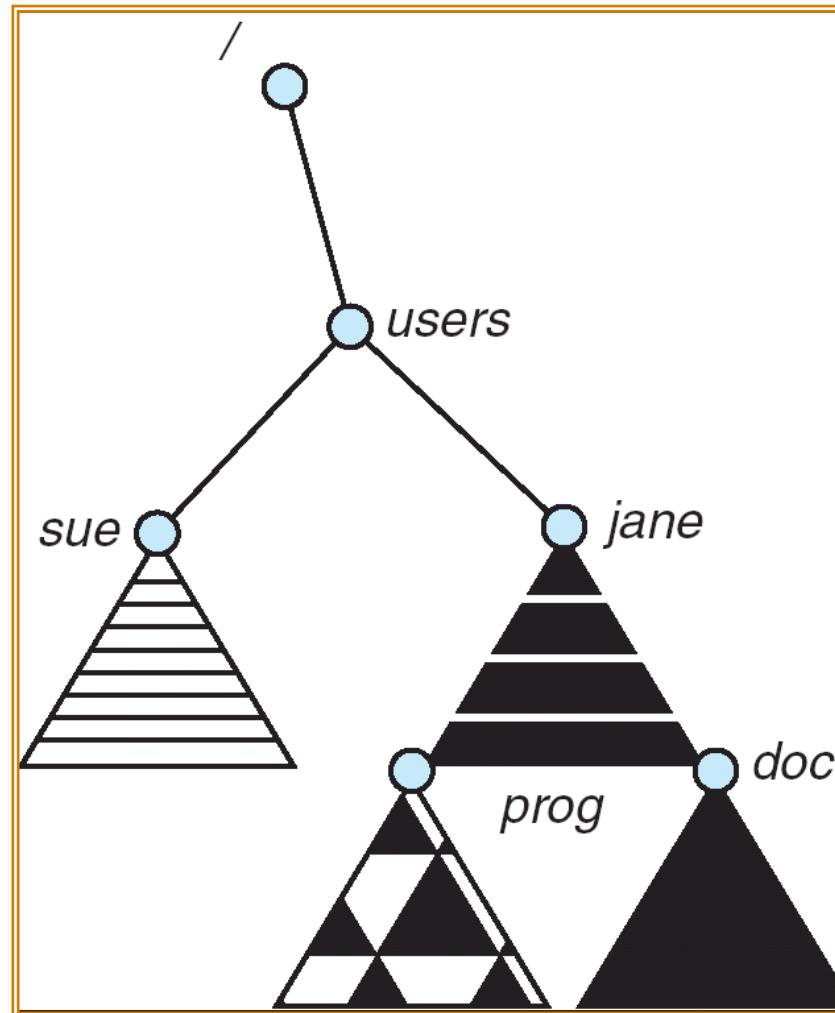
Unità disco e organizzazione/posizione di directory all'interno del file system devono essere correlati?

File System Mounting

Molti SO richiedono il *mounting esplicito* all'interno del file system prima di poter usare una (nuova) **unità disco** o una **partizione**



Dopo il mounting ad un determinato mount point



File system e protezione

Il *proprietario/creatore* di un file dovrebbe avere la possibilità di *controllare*:

- ❑ quali azioni sono consentite sul file
- ❑ da parte di chi

Possibili tipologie di accesso

- Read
- Execute
- Delete
- Write
- Append
- List

Liste di accesso e gruppi (es. UNIX)

Modalità di accesso: read, write, execute

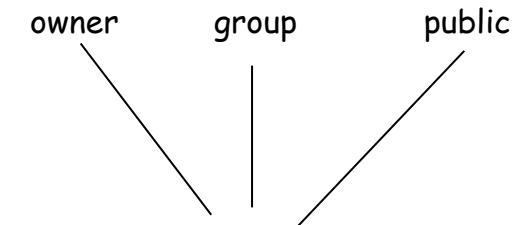
- 3 classi di utenti

1) owner access	7	RWX
		1 1 1
2) group access	6	RWX
		1 1 0
3) public access	1	RWX
		0 0 1

- Amministratore può creare gruppi (con nomi unici) e inserire/eliminare utenti in/da quel gruppo
- Dato un file o una directory, si devono definire le regole di accesso desiderate

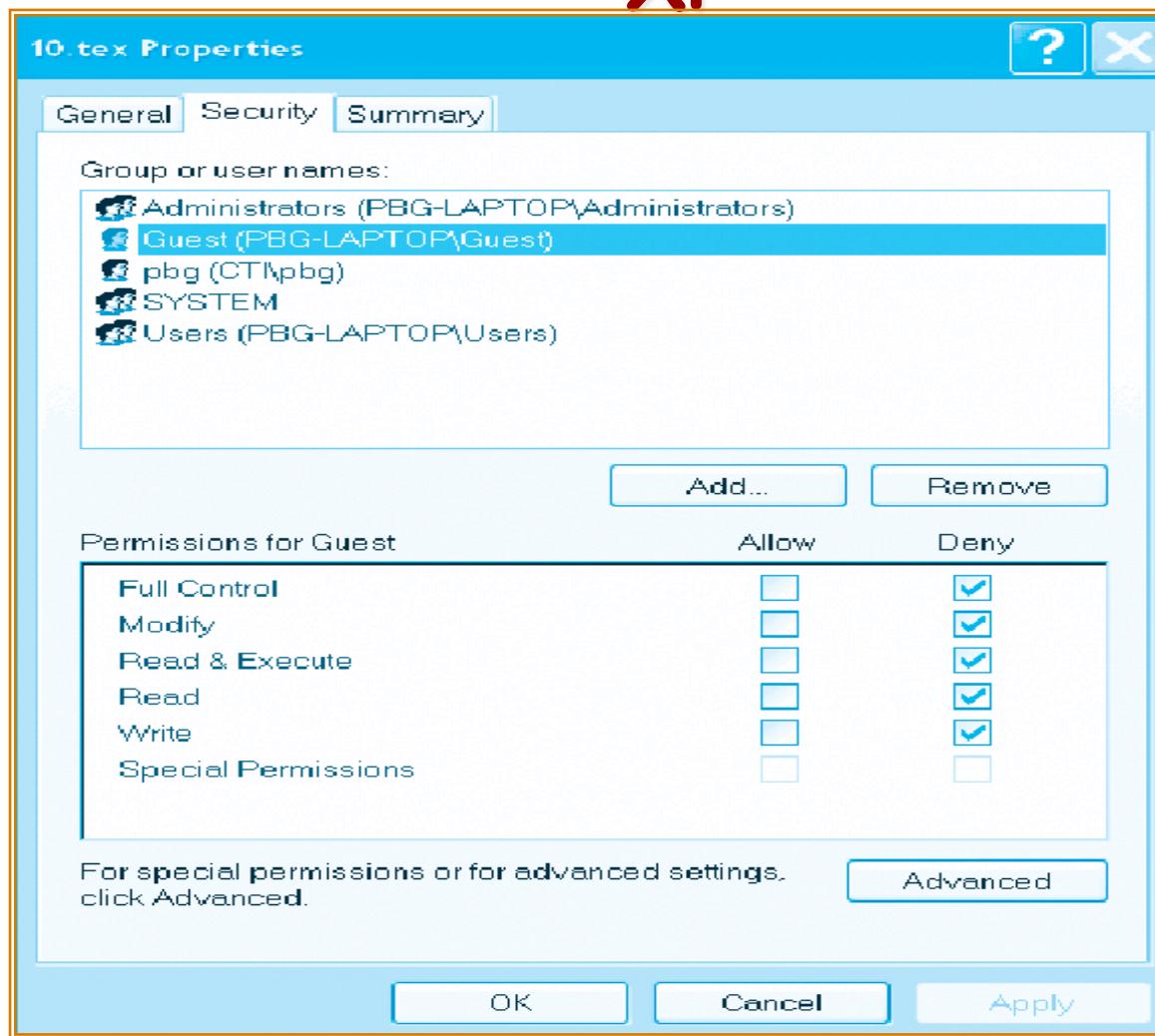
Cambiamento di gruppo: comando

chgrp G game



Cambiamento di diritti di accesso: **chmod 761 game**

Gestione access control list in MS Windows XP



Un esempio di directory listing in UNIX

-rw-rw-r--	1	pbg	staff	31200	Sep 3 08:30	intro.ps
drwx-----	5	pbg	staff	512	Jul 8 09:33	private/
drwxrwxr-x	2	pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2	pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1	pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1	pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4	pbg	faculty	512	Jul 31 10:31	lib/
drwx-----	3	pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3	pbg	staff	512	Jul 8 09:35	test/

Organizzazione fisica del file system

SO si occupa anche della *realizzazione del file system sui dispositivi di memorizzazione secondaria*:

- ❑ *realizzazione dei descrittori e loro organizzazione*
- ❑ *allocazione dei blocchi fisici*
- ❑ *gestione dello spazio libero*

Come può essere realizzato il file system sui dispositivi di memorizzazione secondaria?

Metodi di allocazione

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

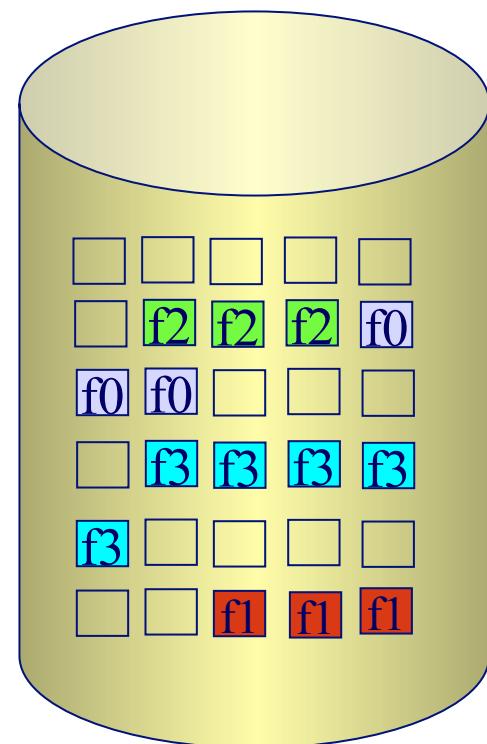
Ogni file è mappato su un insieme di *blocchi fisicamente contigui*

Vantaggi

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

Svantaggi

- individuazione dello *spazio libero* per l'allocazione di un nuovo file
- *frammentazione esterna*: man mano che si riempie il disco, rimangono zone contigue sempre più piccole, a volte inutilizzabili
 - *Necessità di azioni di compattazione*
- *aumento dinamico 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 link danneggiato
- *maggior occupazione* (spazio occupato dai puntatori)
- difficoltà di realizzazione dell'accesso diretto
- *costo della ricerca* di un blocco

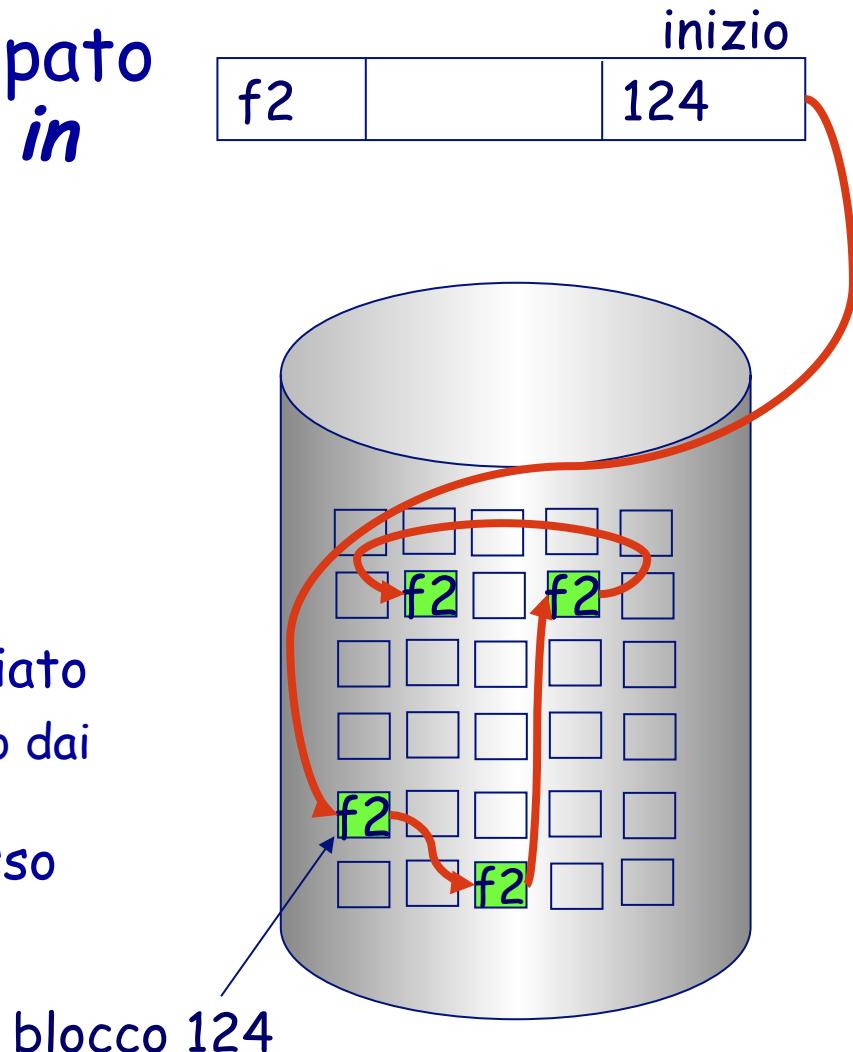
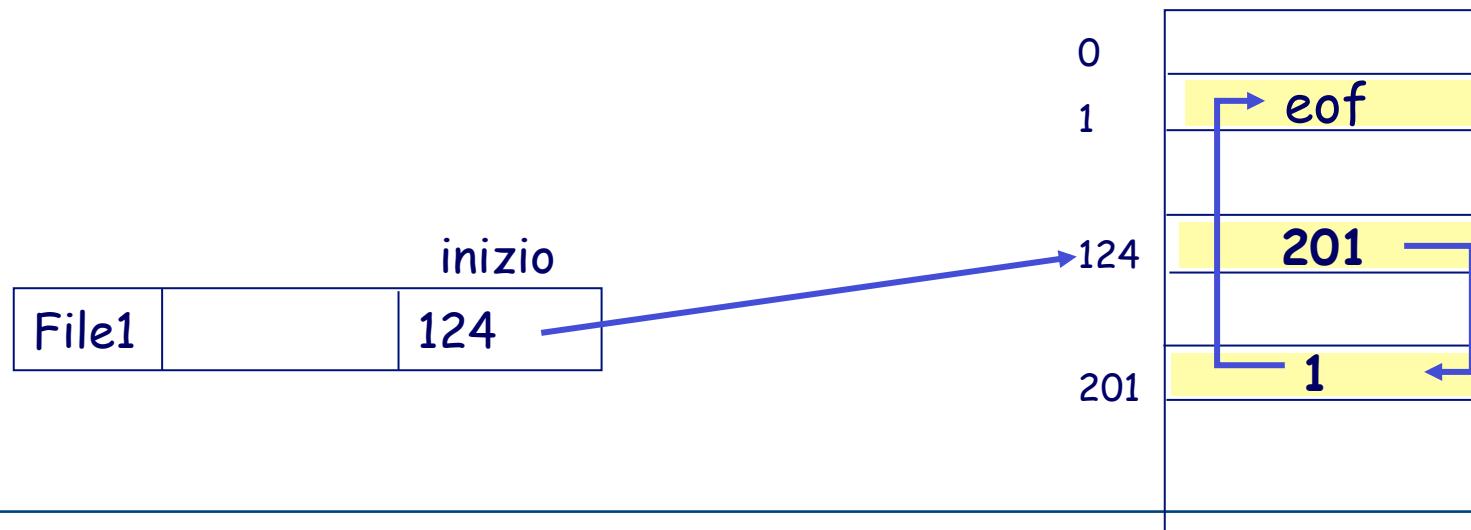


Tabella di allocazione dei file (FAT)

Alcuni SO (ad es. DOS e OS/2) realizzano *l'allocazione a lista* in modo più efficiente e robusto:

- *per ogni partizione, viene mantenuta una tabella (FAT) in cui ogni elemento rappresenta un blocco fisico*
- concatenamento dei blocchi sui quali è allocato un file è rappresentato nella FAT



Allocazione a indice

Allocazione a lista: i puntatori ai blocchi sono *distribuiti sul disco*

- elevato tempo medio di accesso a un blocco
- complessità della realizzazione del metodo di accesso diretto

Allocazione a indice: tutti i *puntatori ai blocchi* utilizzati per l'allocazione di un determinato file sono *concentrati in un unico blocco per quel file (blocco indice)*

Allocazione a indice

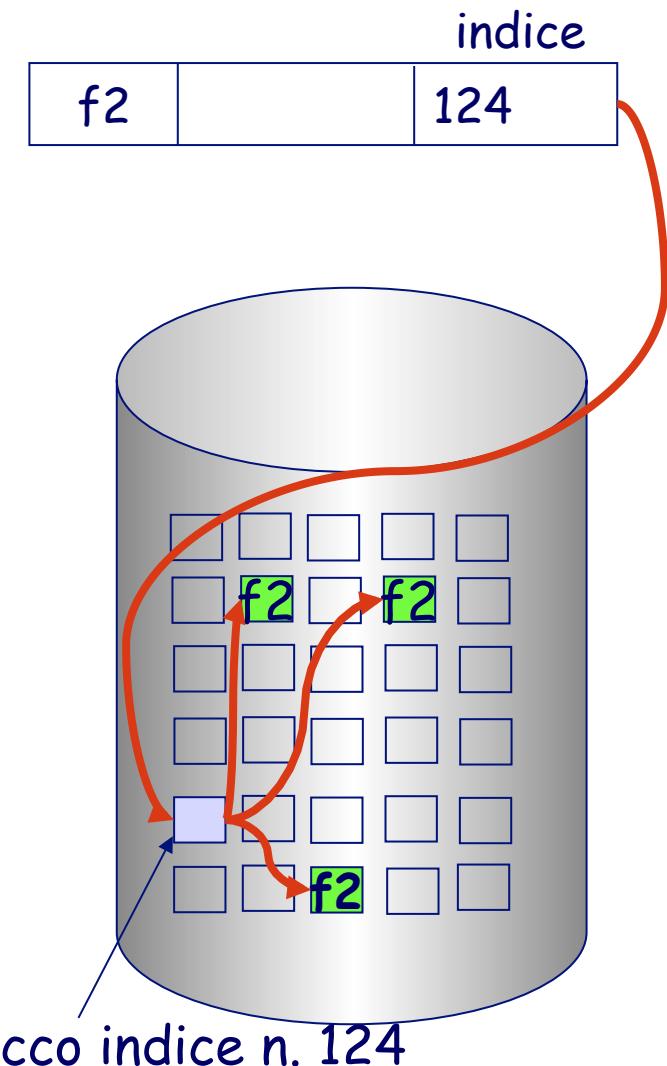
A ogni file è associato un *blocco (indice)* in cui sono contenuti *tutti gli indirizzi dei blocchi* su cui è allocato il file

Vantaggi

- stessi dell'allocazione a lista, più
 - possibilità di accesso diretto
 - maggiore velocità di accesso (rispetto a liste)

Svantaggi

- possibile scarso utilizzo dei blocchi indice



Metodi di allocazione

Riassumendo, gli aspetti caratterizzanti sono:

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

Esistono SO che adottano *più di un metodo di allocazione*; spesso:

- file **piccoli** → allocazione contigua
- file **grandi** → allocazione a indice