IL FILE SYSTEM

Alcune necessità dei processi:

- **memorizzare e trattare grandi quantità di informazioni** (maggiori della quantità di memoria principale),
- **più processi** devono avere la possibilità di **accedere alle informazioni in modo concorrente e coerente**, nello spazio e nel tempo,
- si deve garantire **integrità, indipendenza, persistenza e protezione dei dati**.

I FILE

La soluzione sono i **file** (archivi):

file = insieme di informazioni correlate a cui è stato assegnato un nome.

Un file è la più piccola porzione unitaria di memoria logica secondaria allocabile dall'utente o dai processi di sistema.

La parte del s.o. che realizza questa astrazione, nascondendo i dettagli implementativi legati ai dispositivi sottostanti, è il **file system**.

Esternamente, il file system è spesso l'aspetto più visibile di un s.o.: come si denominano, manipolano, accedono, quali sono le loro strutture, i loro attributi, ecc.

Internamente, il file system si appoggia alla gestione dell'i/o per implementare ulteriori funzionalità.

THE FILE CONCEPT

….. contiguous logical address space

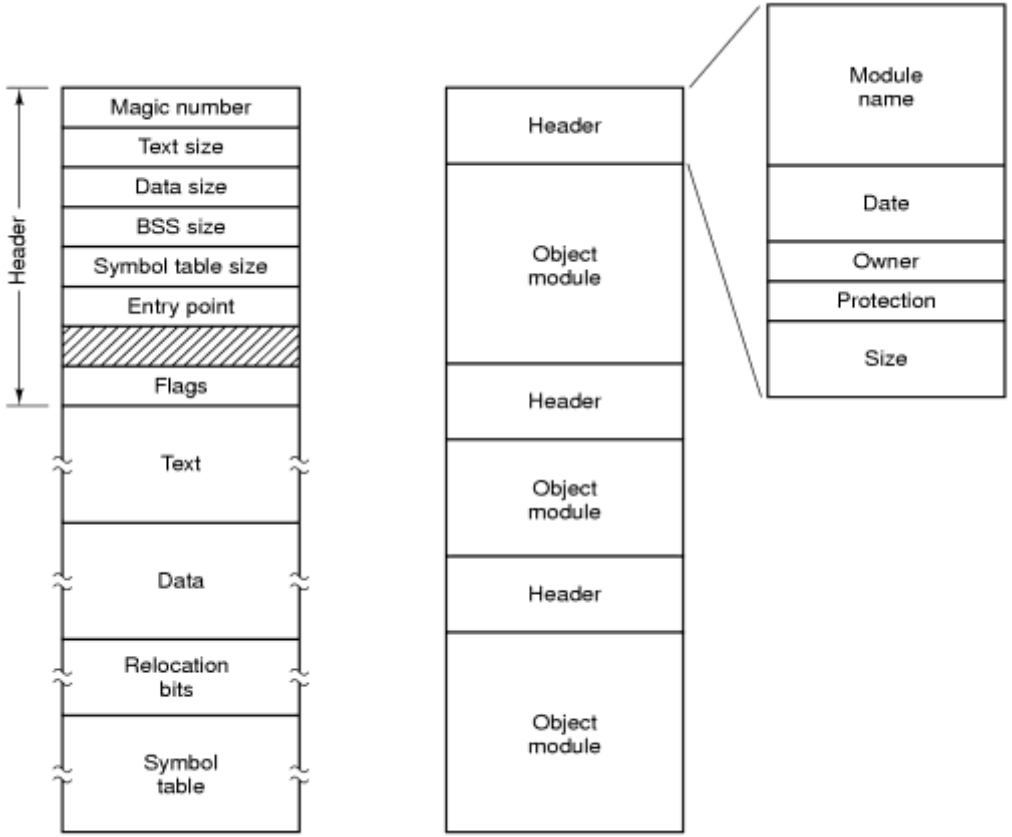
→ Must store large amounts of data

→ Information stored must survive the termination of the process using it

→ Multiple processes must be able to access the information concurrently

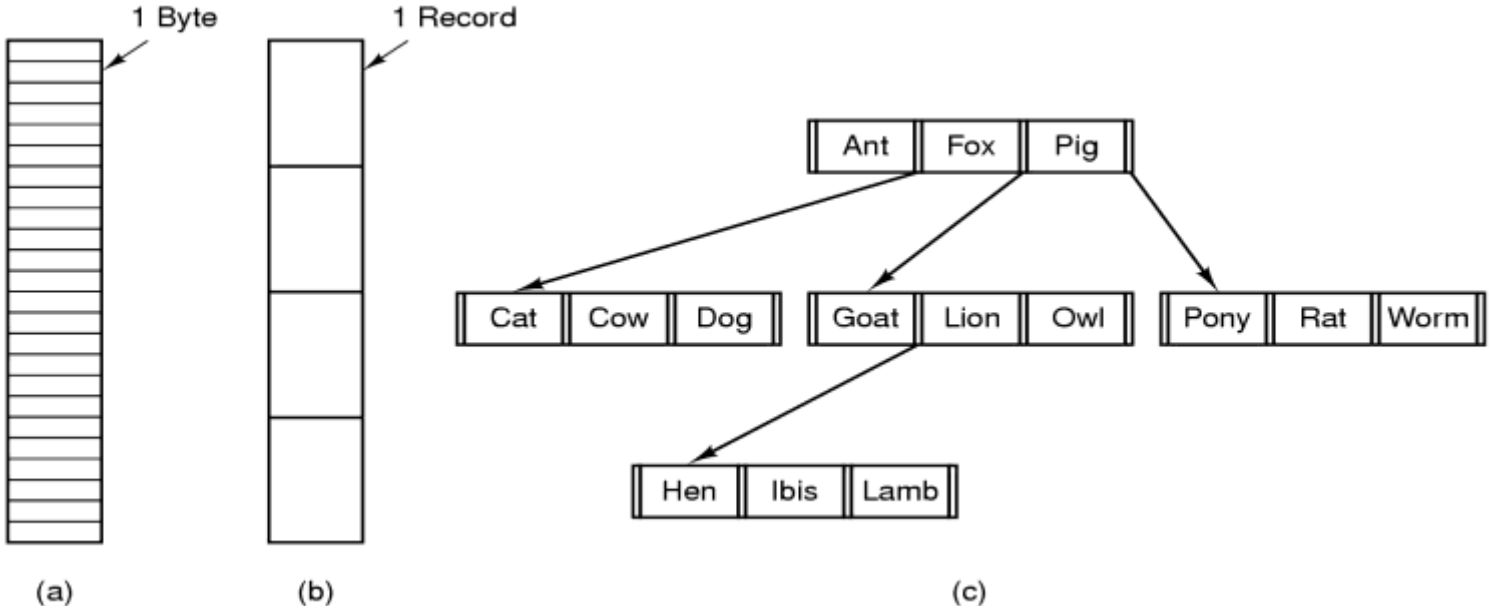
FILE CONTENT

- Data
  - numeric
  - character
  - binary
- Program



FILE STRUCTURES

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file



FILE STRUCTURES

Chi impone la struttura? due possibilità:

- Il sistema operativo: specificato il tipo, viene imposta la struttura e modalità di accesso. Più astratto.
- L'utente: tipo e struttura sono delegati al programma, il sistema operativo implementa solo file non strutturati. Più flessibile.

COMMON FILE ATTRIBUTES

- ✓ **Nome** identificatore del file. L'unica informazione umanamente leggibile.
- ✓ **Tipo** nei sistemi che supportano più tipi di file. Può far parte del nome.
- ✓ **Locazione** puntatore alla posizione del file sui dispositivi di memorizzazione.
- ✓ **Dimensioni** attuale, ed eventualmente massima consentita.
- ✓ **Protezioni** controllano chi può leggere, modificare, creare, eseguire il file.
- ✓ **Identificatori dell'utente** che ha creato/possiede il file.
- ✓ **Varie date e timestamp** di creazione, modifica, aggiornamento info. . .

Queste informazioni (metadati: dati sui dati) sono solitamente mantenute in apposite strutture (directory) residenti in memoria secondaria.

COMMON FILE ATTRIBUTES

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

Tipi dei file – **FAT WINDOWS**: name.extension

Tipo	Estensione	Funzione
Eseguibile	exe, com, bin o nessuno	programma pronto da eseguire, in liguaggio macchina
Oggetto	obj, o	compilato, in linguaggio macchina, non linkato
Codice sorgente	c, p, pas, f77, asm, java	codice sorgente in diversi linguaggi
Batch	bat, sh	script per l'interprete comandi
Testo	txt, doc	documenti, testo
Word processor	wp, tex, doc	svariati formati
Librerie	lib, a, so, dll	librerie di routine
Grafica	ps, dvi, gif	FILE ASCII o binari
Archivi	arc, zip, tar	file correlati, raggruppati in un file, a volte compressi

Tipi dei file — **Unix-Linux**: nessuna assunzione

Unix non forza nessun tipo di file a livello di sistema operativo: non ci sono metadati che mantengono questa informazione.

Tipo e contenuto di un file slegati dal nome o dai permessi.

Sono le applicazioni a sapere di cosa fare per ogni file (ad esempio, i client di posta usano i MIME-TYPES).

FILE OPERATIONS (FILE SYSTEM CALLS)

- **Create** due passaggi: allocazione dello spazio sul dispositivo,e collegamento di tale spazio al file system.
- **Delete** staccare il file dal file system e deallocare lo spazio assegnato al file.
- **Read** dato un file e un puntatore di posizione, i dati da leggere vengono trasferiti dal media in un buffer in memoria.
- **WRITE** dato un file e un puntatore di posizione, i dati da scrivere vengono trasferiti sul media.
- **Get attributes** leggere le informazioni come nome, timestamp, ecc.
- **Set attributes** modificare informazioni come nome, timestamp, protezione, ecc.
- **Seek** Riposizionamento; non comporta operazioni di I/O.

TABELLA DEI FILE APERTI

Queste operazioni richiedono la conoscenza delle informazioni contenute nelle directory.

Per evitare di accedere continuamente alle directory, si mantiene in memoria una tabella dei file aperti.

Due nuove operazioni sui file:
Apertura: allocazione di una struttura in memoria (file descriptor o file control block) contenente le informazioni riguardanti un file.

- `Open` caricare alcuni metadati dal disco nella memoria pricipale, per velocizzare le chiamate seguenti.

Chiusura: trasferimento di ogni dato in memoria al dispositivo, e deallocazione del file descriptor.

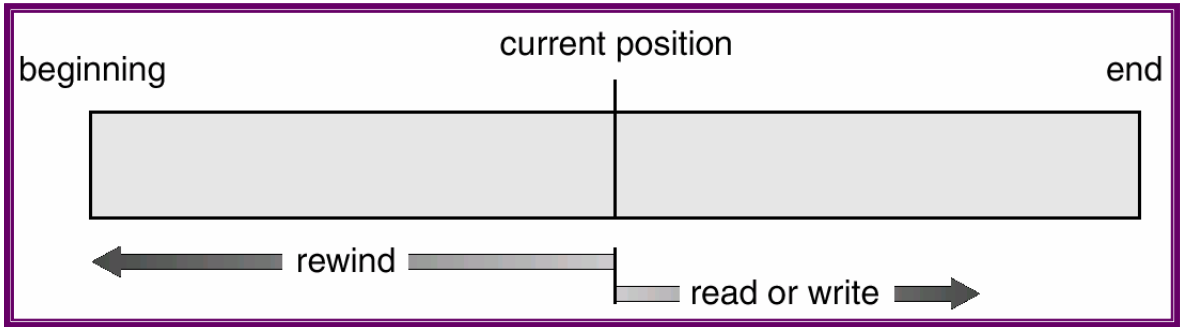
- `Close` deallocare le strutture allocate nell'apertura.

A ciascun file aperto si associa:
Puntatore al file: posizione raggiunta durante la lettura/scrittura.
Contatore dei file aperti: quanti processi stanno utilizzando il file.
Posizione sul disco.

ACCESS METHODS

→ Sequential Access

Un puntatore mantiene la posizione corrente di lettura/scrittura.
Si può accedere solo progressivamente, o riportare il puntatore all'inizio del file.
Adatto a dispositivi intrinsecamente sequenziali (p.e., nastri magnetici)



Operazioni *read next*
write next
reset
no *read after last write*
(rewrite)

ACCESS METHODS

→ Direct Access

Il puntatore può essere spostato in qualunque punto del file.
L'accesso sequenziale viene simulato con l'accesso diretto.
Usuale per i file residenti su device a blocchi (p.e., dischi).

Operazioni

read n
write n
seek to n
read next
write next
rewrite n
n = relative block number

ACCESS METHODS

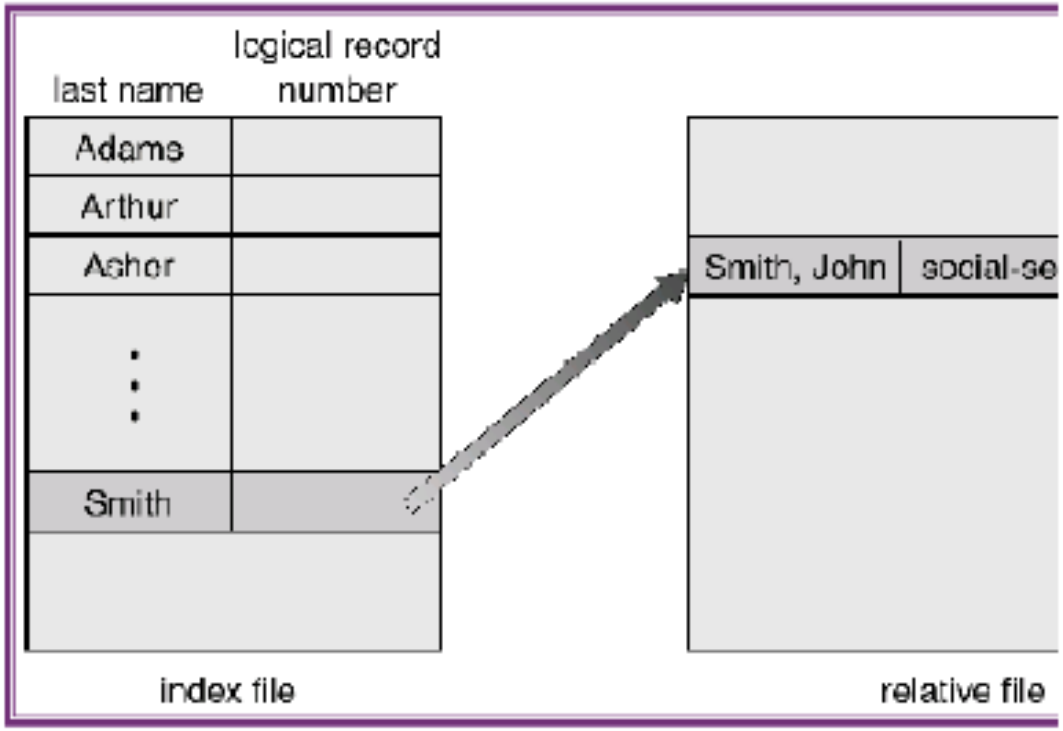
→ Indexed Access

Un secondo file contiene solo parte dei dati, e puntatori ai blocchi (record) del vero file

La ricerca avviene prima sull'indice (corto), e da qui si risale al blocco

Implementabile a livello applicazione in termini di file ad accesso diretto

Usuale su mainframe (IBM, VMS), database. . .

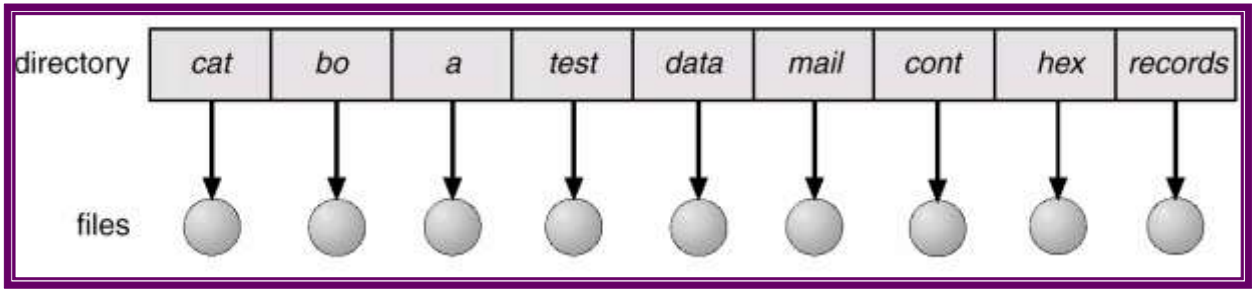


DIRECTORY FILE

Directory ad un livello

A single directory for all users.

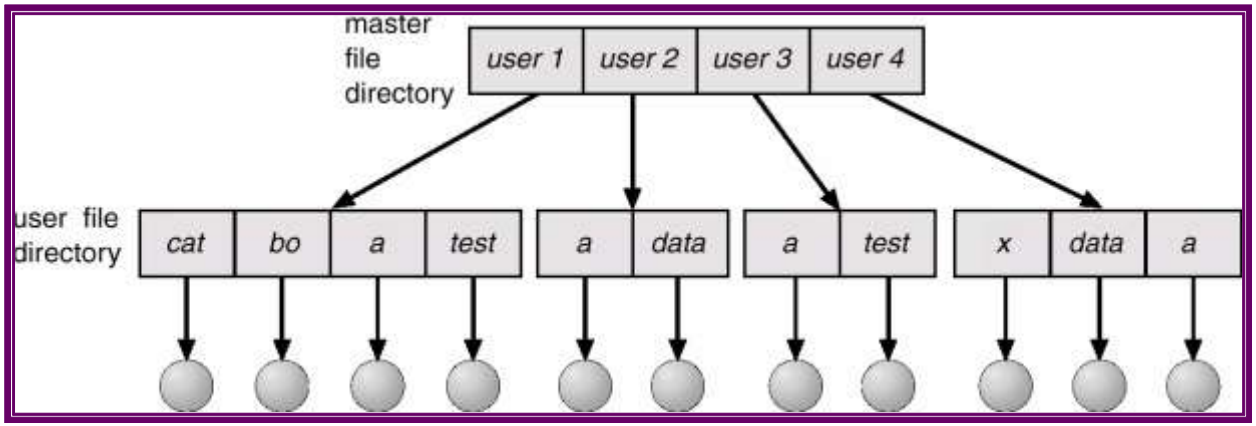
- Φ Naming problem
- Φ Grouping problem



Directory a due livelli

Separate directory for each user.

- Φ Path name
- Φ Can have the same file name for different user
- Φ Efficient searching
- Φ No grouping capability

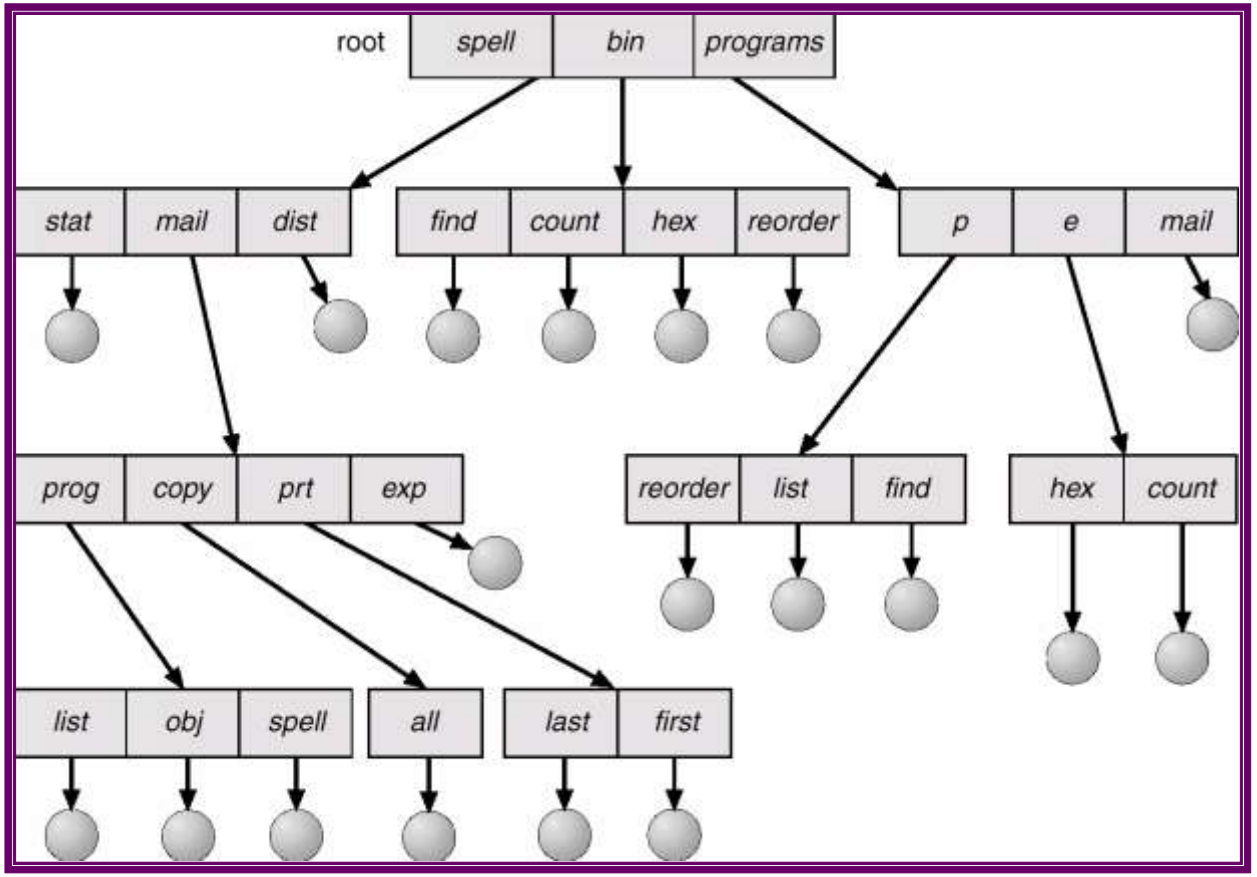


DIRECTORY FILE

Directory Hierarchy

- Φ Efficient searching
- Φ Grouping Capability
- Φ Current directory (working directory)
  - ↳ cd /spell/mail/prog
  - ↳ type list
- Φ Absolute or relative path name
- Φ Creating a new file is done in current directory.
- Φ Delete a file
  - ↳ rm <file-name>
- Φ Creating a new subdirectory is done in current directory.

  mkdir <dir-name>

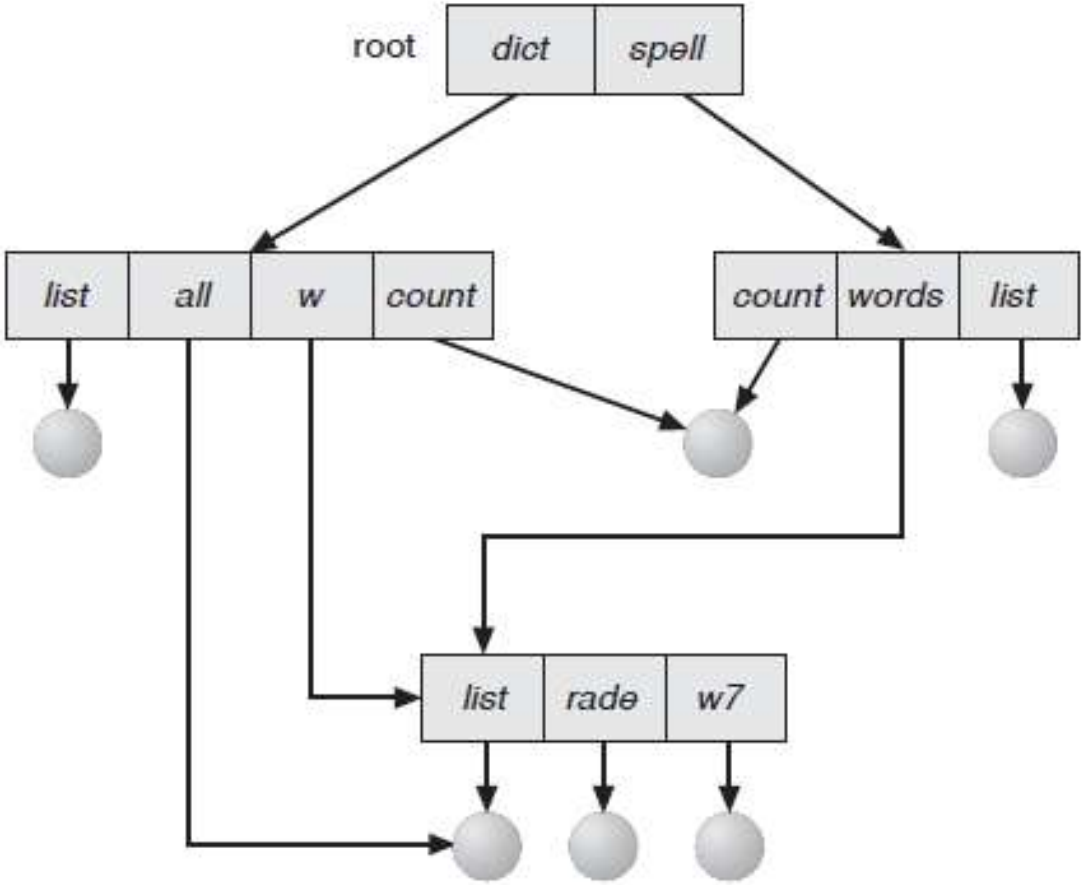


DIRECTORY A GRAFO ACICLICO (DAG)

File e sottodirectory possono essere condivise da più directory.

Due nomi differenti per lo stesso file (aliasing).

Possibilità di puntatori "dangling".



DIRECTORY OPERATIONS

Una directory è una collezione di nodi contenente informazioni sui file (metadati).

Sia la directory che i file risiedono su disco.

- **Create**
- **Delete**
- **Opendir**
- **Closedir**
- **Readdir**
- **Rename**
- **Link**
- **Unlink**

PROTEZIONE

Importante in ambienti multiuser dove si vuole ***condividere file.***

Il creatore/possessore (non sempre coincidono) deve messere in grado di controllare ***cosa può essere fatto e da chi*** (in un sistema multiutente).

MODI DI ACCESSO E GRUPPI IN UNIX

Tre modi di accesso: **r**ead, **w**rite, **e**xecute

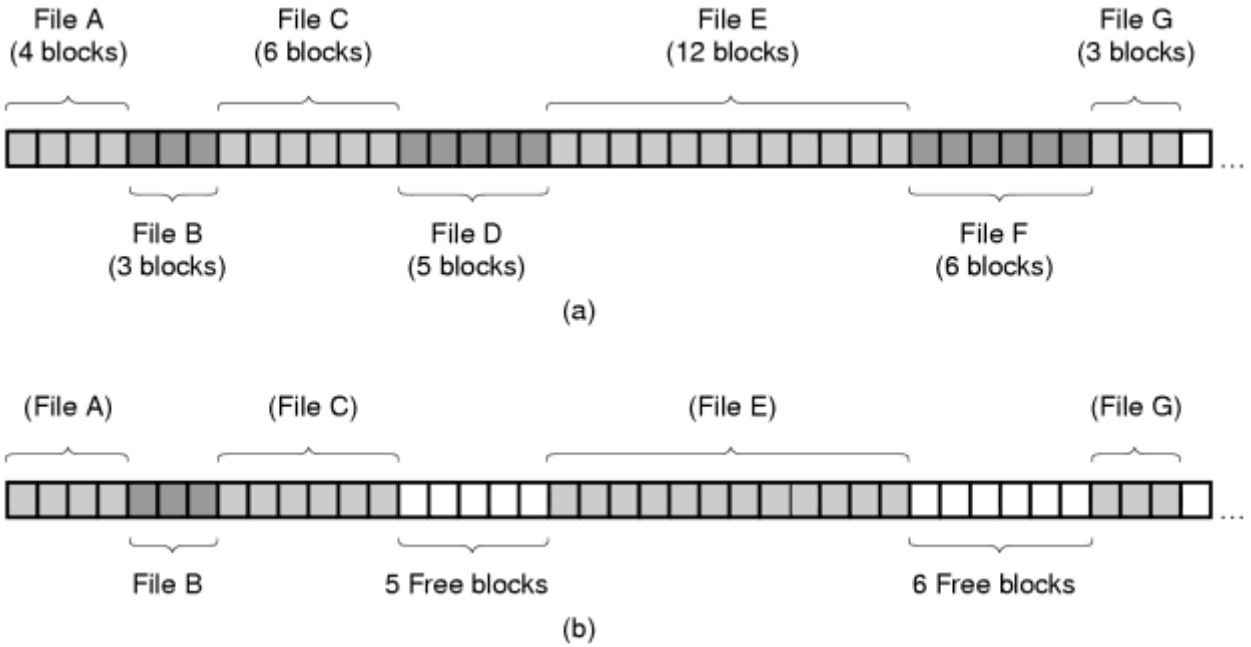
Tre classi di utenti, per ogni file

			R	W	X
a) owner access	7	→	1	1	1
b) groups access	6	→	1	1	0
c) public access	1	→	0	0	1

Ogni processo possiede UID e GID, con i quali si verifica l'accesso

PHYSICAL ALLOCATION OF FILES

Contiguous Allocation



Contiguous allocation of disk space for 7 files
State of the disk after files D and E have been removed

- Simple – only starting location (block #) and length (number of blocks) are required.
- Random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

PHYSICAL ALLOCATION OF FILES

Disk blocks concatenation

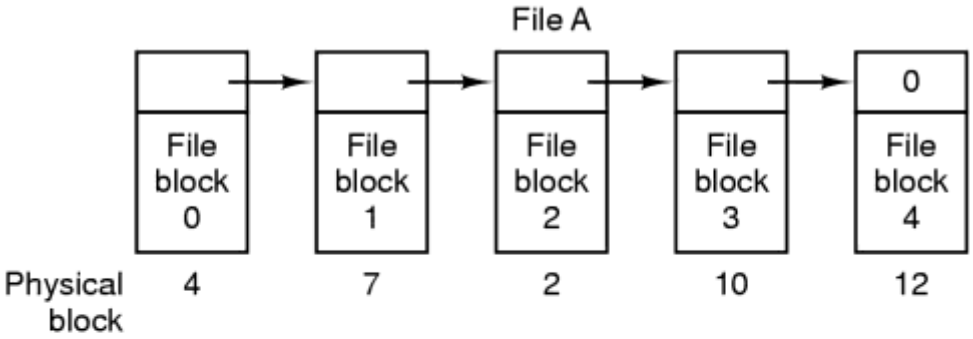
Physical concatenation (**pointers**)

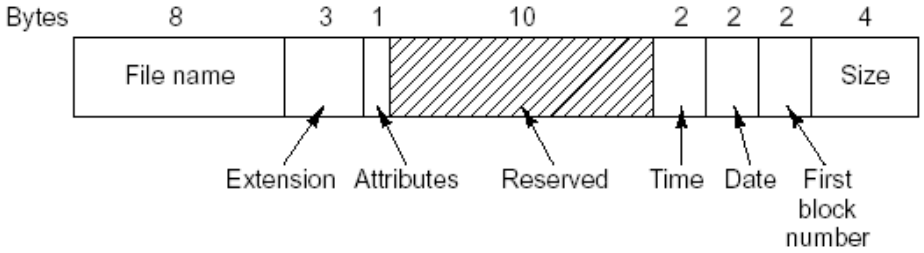
Each disk block of a file points, i.e. contains the physical coordinates of the next block.

Blocks may be scattered anywhere on the disk.

- Simple – need only starting address
- Free-space management system
- waste of space and time
- No random access

File-allocation table (**FAT**) – disk-space allocation used by MS-DOS – must contain physical coordinates of the first block.





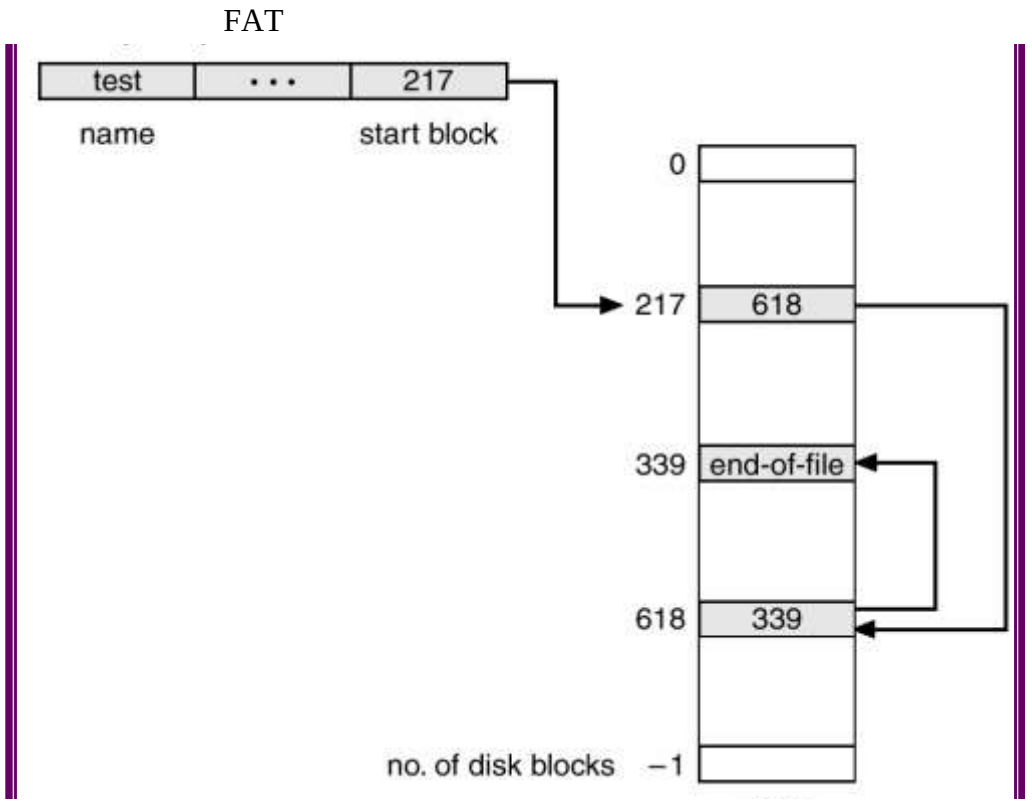
PHYSICAL ALLOCATION OF FILES

Disk blocks concatenation

Logical concatenation (linked list)

I dischi sono indirizzati come dei grandi array monodimensionali di blocchi logici, dove il blocco logico è la più piccola unità di trasferimento con il controller. L'array monodimensionale è mappato sui settori del disco in modo sequenziale.

- **settore 0 = primo settore della prima traccia del cilindro più esterno**
- **la mappatura procede in ordine sulla traccia, poi sulle rimanenti tracce dello stesso cilindro, poi attraverso i rimanenti cilindri dal più esterno verso il più interno.**



Linked List

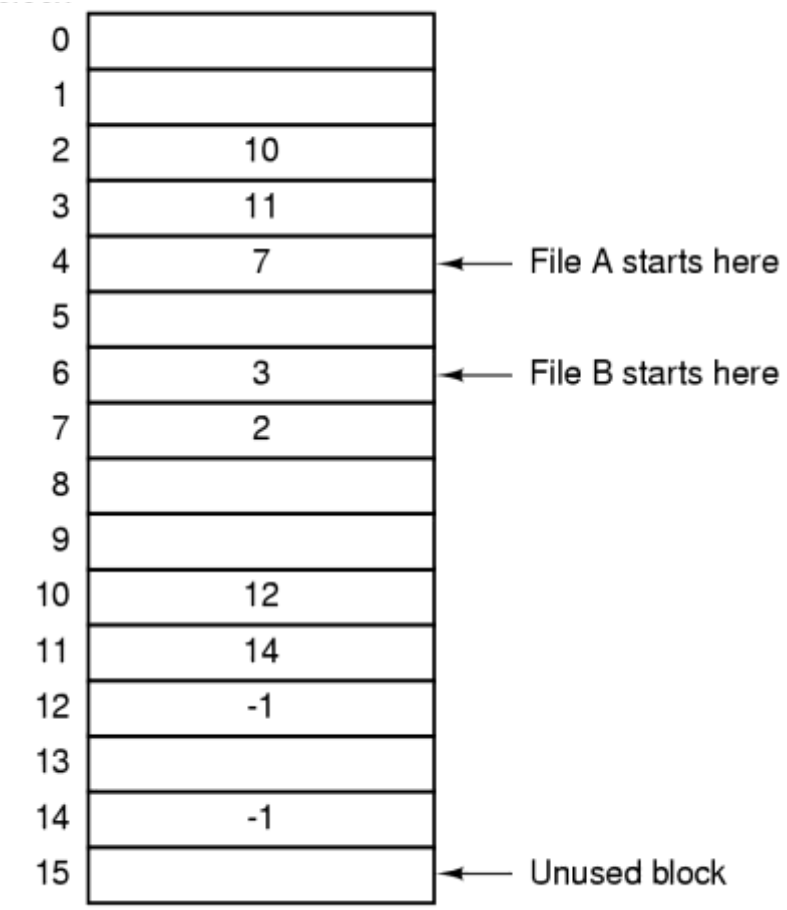
PHYSICAL ALLOCATION OF FILES

Linked List is a memory table

Ad ogni file corrisponde, nella linked list, ha una struttura concatenata di blocchi, che possono essere sparpagliati ovunque sul disco.

Il numero di blocco logico iniziale di ciascun file è contenuto nella FAT.

Si considerino gli stessi file A e B della physical concatenation tramite pointers.



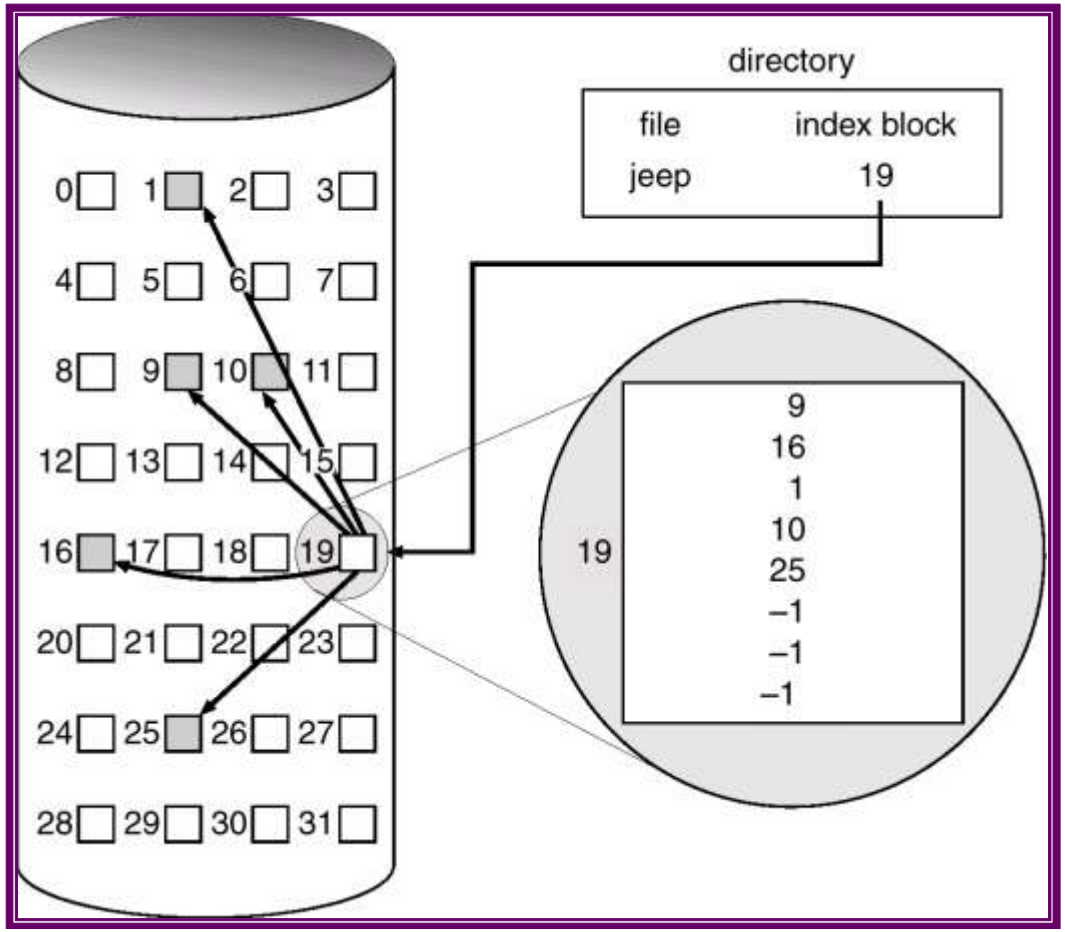
PHYSICAL ALLOCATION OF FILES

Indexed allocation

Brings all pointers together into the *index block.*

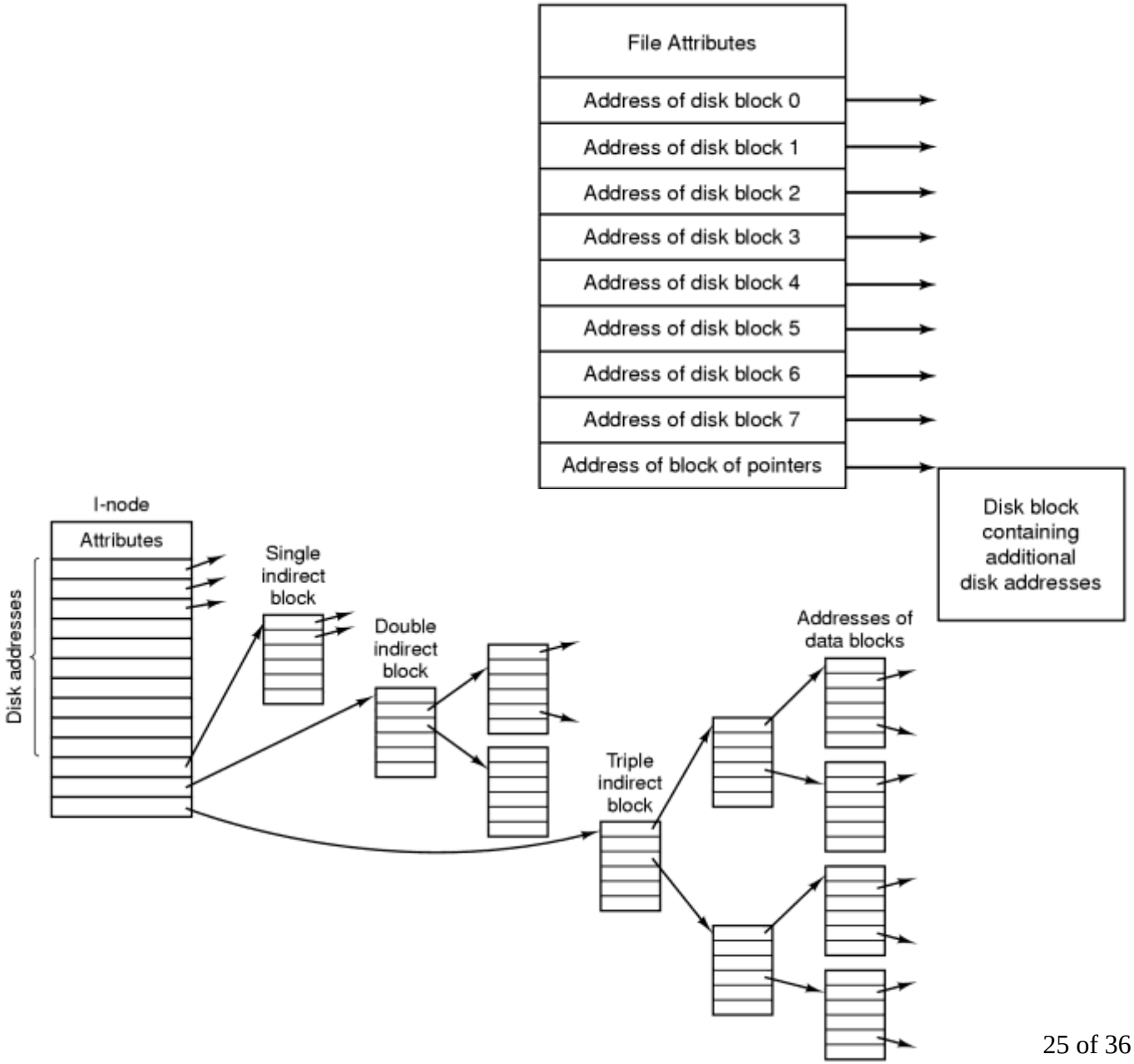
Logical view.

- Φ Need index table
- Φ Random access
- Φ Dynamic access without external fragmentation, but have overhead of index block.



PHYSICAL ALLOCATION OF FILES

Indexed allocation (*i-node*)



Triple indexed allocation

i-node

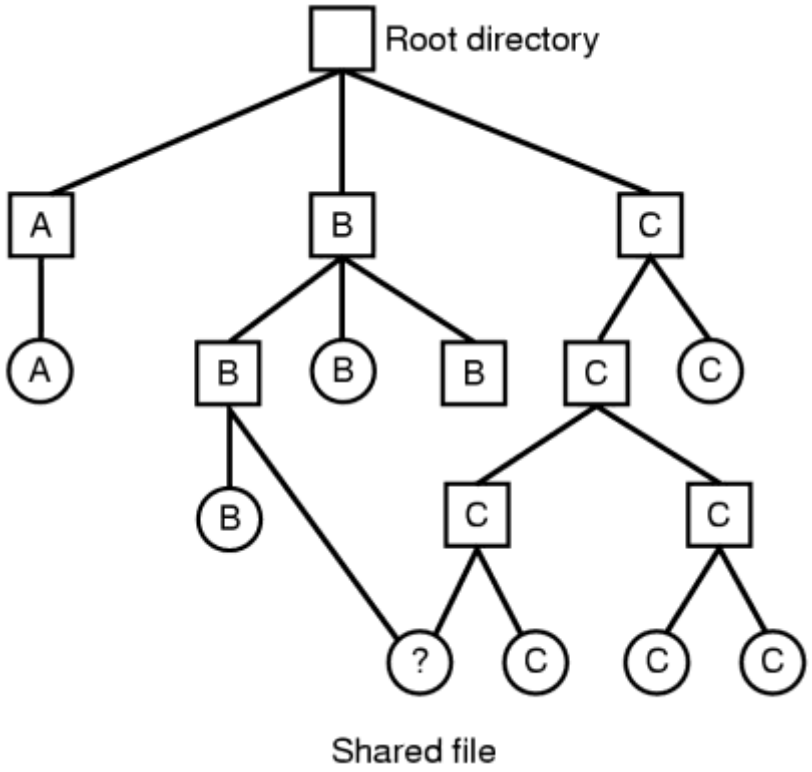
Gli indici indiretti vengono allocati su richiesta.

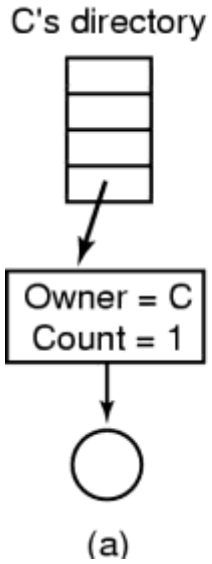
Accesso più veloce per file piccoli.

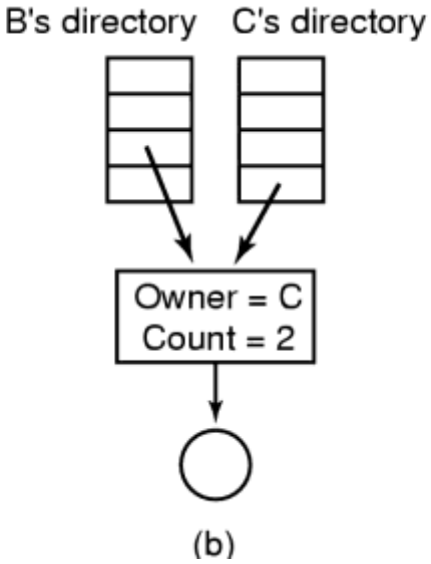
N. massimo di blocchi indirizzabile: con blocchi da 4K e puntatori da 4byte
$L_{max} = 12 + 1024 + 1024^2 + 1024^3 > 1024^3 = 2^{30}$blk $= 2^{42}$byte = 4TB

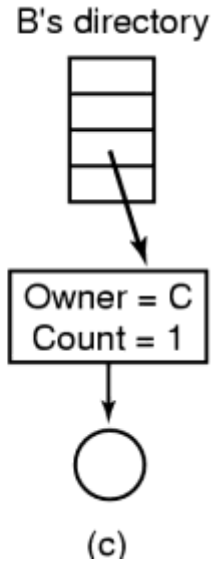
MOLTO OLTRE LE CAPACITÀ DEI SISTEMI A 32 BIT.

SHARED FILES









(a) Situation prior to linking
(b) After the link is created
(c) After the original owner removes the file

UTILIZZO DELLO SPAZIO FISICO E CAPACITÀ DEL SINGOLO BLOCCO

Nelle memorie secondarie soggette a *formattazione preliminare*, lo spazio viene suddiviso in blocchi di capacità fissa (dell'ordine di 2kb o multipli).

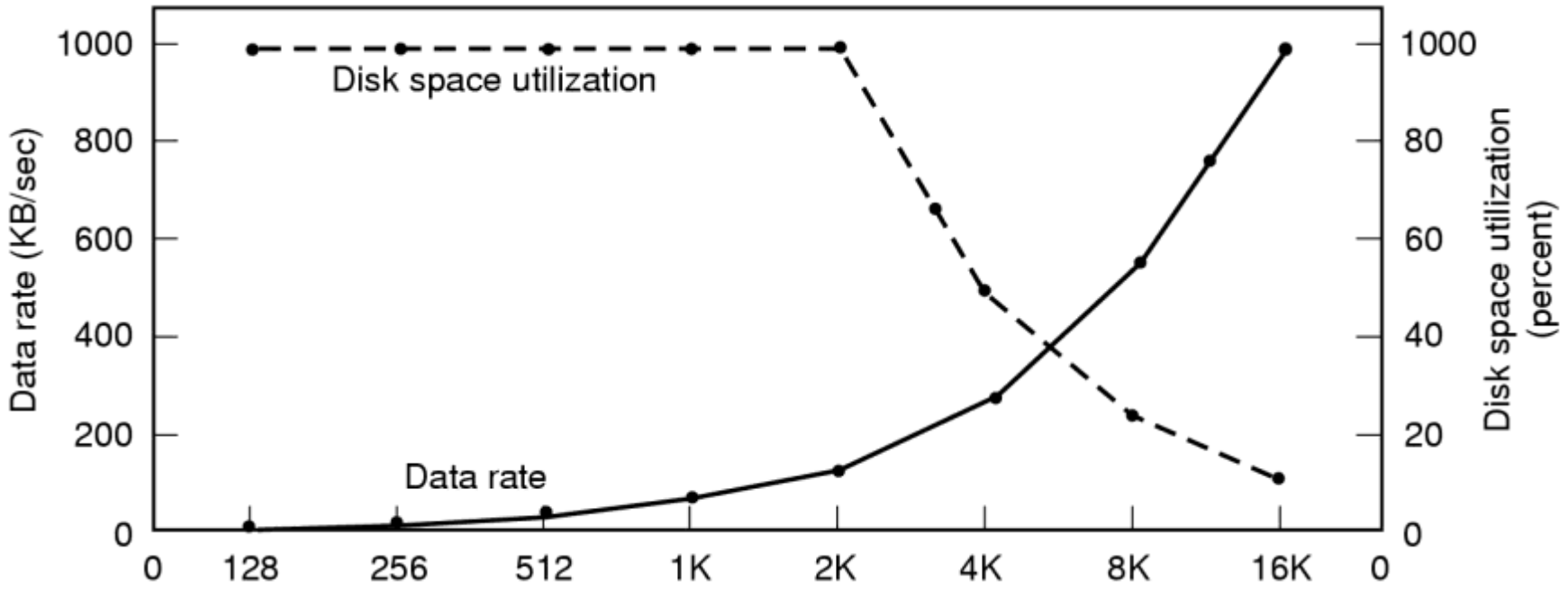
Un "blocco" o ***record fisico*** è l'unità minima soggetta a scrittura e a lettura.

Questo può dare origine ad un cattivo utilizzo del disco, quando la dimensione del "***record logico***" (insieme dei dati che descrivono un singolo elemento dell'archivio) fosse sostanzialmente inferiore rispetto a quella del blocco (molto meno frequentemente un record logico richiede di espandere su più di un record fisico).

Per evitare tale spreco un ***record fisico*** può essere costituito al proprio interno da più "record logici". Si dice allora che *più record logici sono "bloccati" in un record fisico*. Il bloccaggio *consente anche di ridurre i tempi di lettura e scrittura*, ricorrendo all'uso di buffer di memoria: in un buffer di lettura viene alloggiato l'ultimo record fisico letto ed in un altro buffer viene alloggiato il record fisico da scrivere appena completato.

UTILIZZO DELLO SPAZIO FISICO E CAPACITÀ DEL SINGOLO BLOCCO

- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- All files 2KB



TIPI DI ALLOCAZIONE DEI FILE

Le possibili politiche di gestione delle memorie di massa ad accesso diretto sono:

- preallocazione;
- allocazione dinamica.

Nel primo caso, il SO prealloca lo spazio occupato da un file.

Nel secondo caso, lo spazio viene allocato un blocco alla volta quando viene richiesta una scrittura nel file.

I sistemi UNIX-like usano una politica intermedia, preallocando dinamicamente un certo numero di blocchi.

TIPI DI ALLOCAZIONE DEI FILE

Preallocazione vs Allocazione dinamica

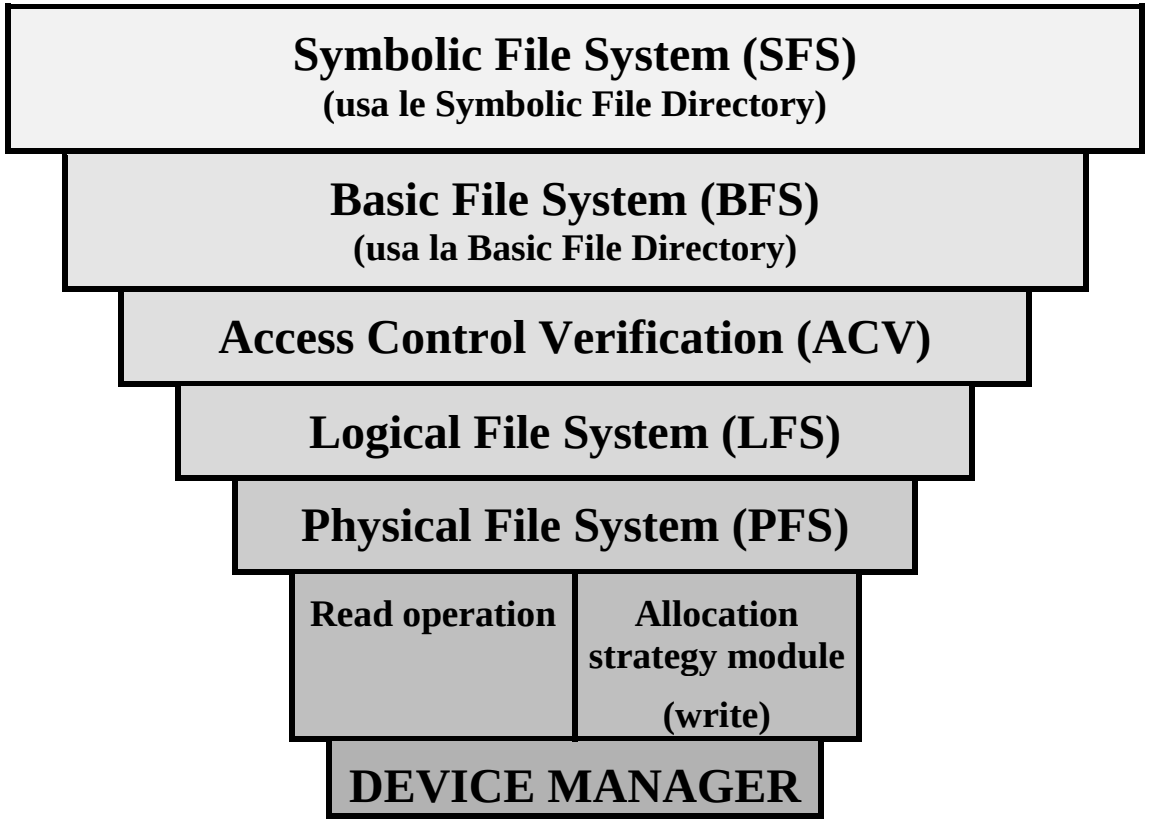
La politica di **preallocazione** impone che il programma, all'inizio della sua esecuzione, specifichi di quanto spazio di memoria di massa ha bisogno. Questo è un modo d'operare alquanto scomodo, poiché è frequente il caso in cui non si abbia la minima idea di quanto spazio realmente serva. Esso però ha il vantaggio di una preliminare verifica della disponibilità dello spazio richiesto.

Invece, nella politica d'**allocazione dinamica**, all'utente non sono richieste informazioni sullo spazio da allocare su disco, poiché questo viene allocato solo quando serve. Però, anche questo modo di procedere presenta degli inconvenienti, poiché si può incorrere in una imprevista indisponibilità di spazio durante l'esecuzione.

Nei sistemi UNIX, quando un programma chiede di scrivere su disco, si alloca un certo numero di blocchi (16). Questi vengono scritti in sequenza fino al penultimo. Nel caso in cui siano necessari altri record, il SO ne alloca altri 16, e così via. Questo è un metodo più veloce rispetto a quello dell'allocazione dinamica, poiché la ricerca dei record liberi (nella tabella) avviene una volta ogni 16 record allocati. Invece, nell'allocazione dinamica, la ricerca dei record liberi avviene ad ogni operazione di scrittura.

L'ARCHITETTURA DI UN FILE SYSTEM

Il modello generale



L'ARCHITETTURA DI UN FILE SYSTEM

Il modello generale

Il modello prevede un file system gerarchico, formato da una directory radice e più sottodirectory.

Ogni directory o cartella (detta Symbolic File Directory o **SFD**) è un file costituito da record contenenti ciascuno:

- ☞ il nome simbolico di un file o sotto-directory;
- ☞ il record della Basic File Directory (**BFD**) in cui sono riportate le informazioni relative al file o sotto-directory.

La BFD, una per l'intero volume, è un file costituito da tanti record quanti sono i file e directory contenuti nel volume in questione.

L'ARCHITETTURA DI UN FILE SYSTEM

Il modello generale (1/2)

La funzione svolta da ciascun livello può essere compresa analizzando le operazioni svolte per l'esecuzione, da parte di un processo-utente, del seguente comando, che qui si suppone riferito, senza perdita di generalità, ad un file sequenziale:

READ nome_volume, path, nome_file, lista_variabili

Tale comando implicitamente prevede la lettura dell'i-esimo record del file, dal quale ricavare i valori delle variabili specificate nella lista.

Il modulo **SFS**, identificato il volume (*nome_volume*) ed il percorso per raggiungere la **SFD** contenente il file richiesto (*path*), provvede al controllo dell'esistenza in essa di un file avente il *nome_file* specificato.

Il SFS provvede quindi a chiamare il BFS fornendogli il record della **BFD** associato al nome specificato.

Il **BFS** accede a tale record della BFD, copiandolo in memoria perché il suo contenuto possa essere utilizzato, a tempo debito, dai moduli sottostanti della gerarchia.

In particolare l'**ACV** provvede a verificare che il processo-utente che ha originato il comando possieda i "diritti di accesso" richiesti per eseguire l'operazione richiesta. L'ACV chiama quindi il LFS.

L'ARCHITETTURA DI UN FILE SYSTEM

Il modello generale (2/2)

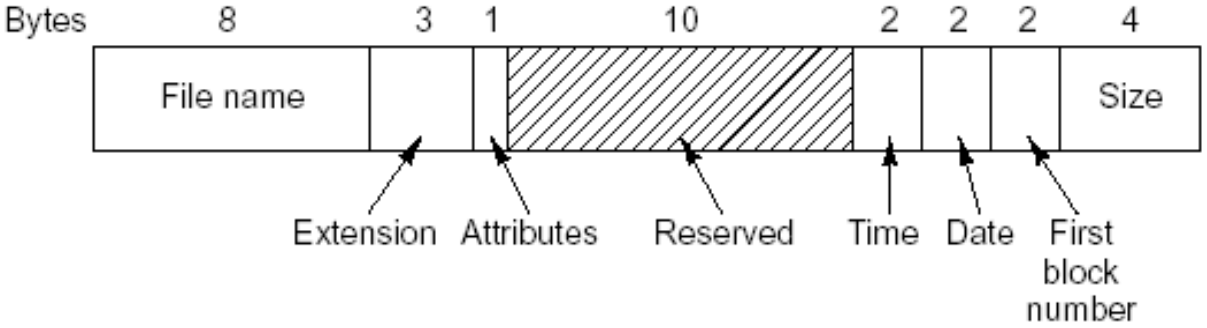
Il **LFS** determina il record logico al quale si sta facendo riferimento. Dal numero del record logico e dalla struttura del file (lunghezza del record logico e numero di record logici costituenti il singolo record fisico), si ricava il record fisico al quale si deve accedere. Quest'ultima informazione, viene passata al modulo PFS.

Il **PFS** può quindi procedere alla individuazione delle coordinate del record fisico sul disco ed avviare la successiva operazione di lettura.

Se il comando richiedesse un'operazione di scrittura, invece che una di lettura, l'Allocation Strategy Module provvederebbe a ricavare il blocco libero su cui effettuare l'operazione di scrittura.

MODELLO GENERALE E SUA REALIZZAZIONE

Il DOS non distingue tra symbolic e basic file directory; infatti adotta un unico tipo di directory: la File Allocation Table (FAT).
I singoli record della FAT indicano se si riferiscono a file o directory. La FAT contiene l'indirizzo del primo record del file.



The MS-DOS directory entry.

Invece, UNIX adotta il modello distinguendo tra directory (equivalente alla symbolic file directory del modello generale) e i-list (equivalente alla basic file directory del modello). I suoi record, uno per ciascun file, si chiamano I-NODE. I file possono essere di tipo ordinario, di tipo directory o di tipo speciale (drivers). Tali files speciali sono contenuti nella directory /DEV.
Il proprietario del file definisce i diritti di lettura, di scrittura e d'esecuzione. Tali diritti vengono definiti per sé stesso (cioè il proprietario), per il suo gruppo e per gli "altri". In totale, le informazioni sui permessi occupano nove bit.