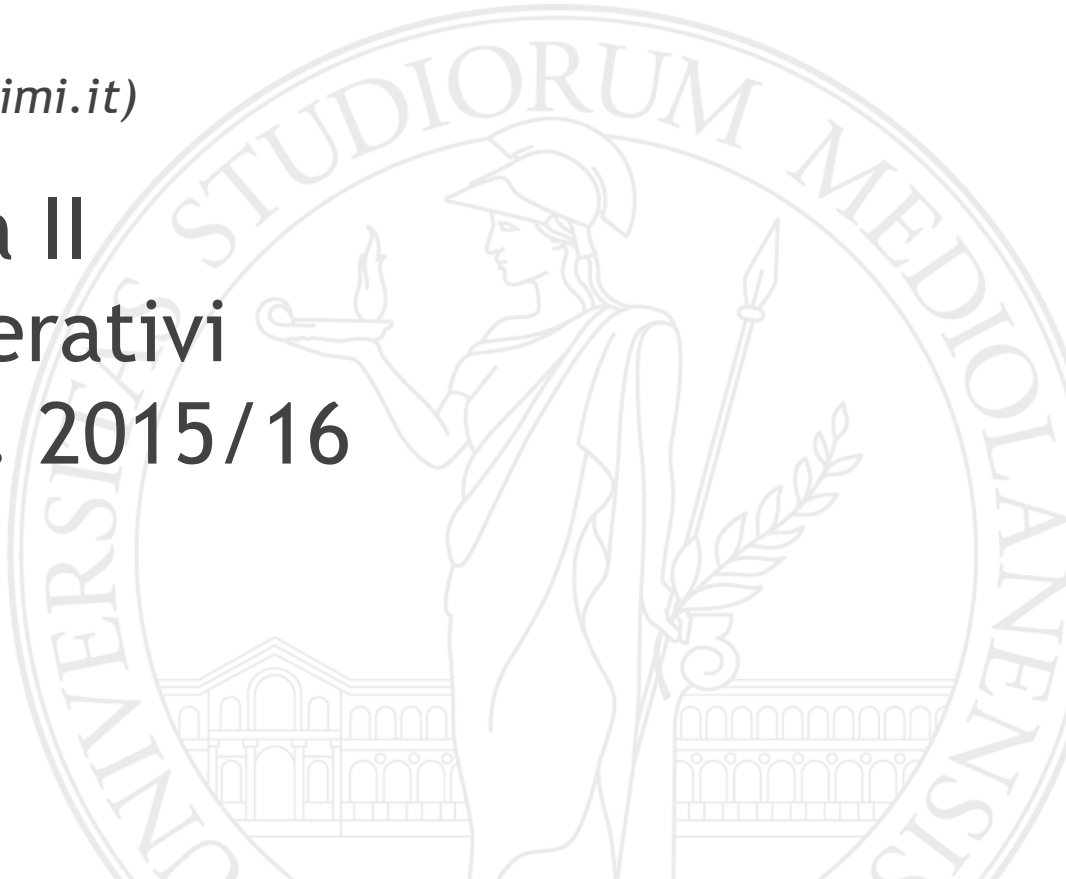
UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Alberto Ceselli
(alberto.ceselli@unimi.it)

Informatica II
Sistemi Operativi
DIGIP - a.a. 2015/16



Sistemi Operativi

(modulo di Informatica II)

L'interfaccia del file system

Patrizia Scandurra

Università degli Studi di Bergamo

a.a. 2014-15

Sommario

- Il concetto di file
- Metodi di accesso
- Struttura delle directory
- Montaggio del file system
- Condivisione dei file
- Protezione

Il concetto di file

- Unità logica di memorizzazione
  (visione utente!)
- Non dipende dal tipo di dispositivo fisico
- Tipi:
  - Dati
    - numerici,alfabetici, binari
  - Programmi
  - Riferimenti (shortcut .lnk)

Gestione dei file

Il SO gestisce i file attraverso due componenti:

- Il **file system**: fornisce gli strumenti per la creazione e la manipolazione dei file, per garantire l'affidabilità dei file quando si verificano guasti, permettere la protezione dei file e la condivisione tra gli utenti (per l'accesso concorrente)
- Il **sotto-sistema di I/O**: fornisce accesso efficiente ai dati memorizzati nei dispositivi di I/O, gestione efficiente dei dispositivi di I/O

Attributi del file

- Un file consiste di due tipi di dati: i *dati* contenuti nei file e i *meta-dati* (o dati di controllo o attributi) usati per accedere al file
- I meta-dati sono mantenuti nel **descrittore del file** (struttura dati *anch'essa mantenuta in memoria di massa*)
  - **Nome** – unica informazione mantenuta in una forma leggibile dagli esseri umani
  - **Tipo** – necessario per supportare differenti tipi di file
  - **Locazione** – locazione fisica del file nel dispositivo
  - **Dimensione** – dimensione corrente del file (byte o blocchi)
  - **Protezione** – determina chi può leggere, scrivere, eseguire
  - **Ora e data e identificativo del proprietario** – al momento della creazione, modifica ed ultimo accesso; dati che possono essere utili per motivi di protezione, sicurezza e controllo d'uso

Tipi di file – Nome, estensione

Tipo di file	Estensione usuale	Funzione
Eseguibile	`.exe`, `.com`, `.bin` o nessuna	Programma in linguaggio macchina pronto per essere eseguito
Oggetto	`.obj`, `.o`	Compilato, linguaggio macchina, non linkato
Codice sorgente	`.c`, `.cc`, `.java`, `.pas`, `.asm`, `.a`	Codice sorgente in vari linguaggi
Batch	`.bat`, `.sh`	Comandi per l'interprete dei comandi
Testo	`.txt`, `.doc`	Dati testuali, documenti
Programma di elaborazione testi	`.wp`, `.tex`, `.rtf`, `.doc`	Vari formati di programmi di elaborazione testi
Libreria	`.lib`, `.a`, `.so`, `.dll`	Librerie di procedure per programmatori
Stampa o visualizzazione	`.ps`, `.pdf`, `.jpg`	File in formato ASCII o binario per la stampa o la visualizzazione
Archivio	`.arc`, `.zip`, `.tar`	File collegati, raggruppati in un unico file, talvolta compressi, a scopo di archiviazione o di memorizzazione
Multimedia	`.mpeg`, `.mov`, `.rm`	File binari contenenti informazioni audio o audio/video

Operazioni di base sui file

- Tutte le operazioni richiedono al SO l'accesso al file sul dispositivo
- Per rendere più efficiente la ricerca, il SO mantiene una **tabella dei file in uso (o aperti)**
  - In sistemi multi-utente (ad es. Unix-like):
    - **Una per il SO**, per info non dipendenti dal singolo processo + "contatore aperture" + lock di blocchi in lettura/scrittura
      - Info: posizione nel dispositivo, date di accesso, dimensione del file, ecc..
    - **Una per processo** che punta a quella del SO
- Operazioni di base fornite dal sistema operativo:
  - **Open(f)** – prende il nome del file e cerca nel dispositivo, copiando il descrittore del file da questo alla tabella dei file aperti
  - **Close(*f*)** – il descrittore del file viene rimosso dalla tabella dei file aperti

Operazioni comuni sui file

- **Scrittura:** aggiunge dati ad un file
  - L'utente specifica il file e i dati da scrivere: write(f,dati)
  - Il SO mantiene un puntatore (di scrittura) alla posizione corrente

- **Lettura:** preleva dati da un file
  - L'utente specifica il file e un puntatore alla zona di memoria destinazione dei dati: read(f,&dati)
  - Il SO mantiene un puntatore (di lettura) alla posizione corrente del file

- **Riposizionamento** all'interno di un file – ricerca del file
  - sposta la posizione del puntatore di lettura/scrittura

Operazioni comuni sui file (Cont.)

- **Creazione**: allocazione, creazione del nuovo descrittore del file, aggiunta del descrittore al file system
- **Cancellazione**: elimina un file - il SO dealloca lo spazio sul dispositivo fisico e lo aggiunge alla lista dello spazio disponibile sul disco, rimuove il descrittore del file dal file system
- **Troncamento**: cancellazione del contenuto, ma non degli attributi (eccetto la lunghezza)!
- Altre operazioni: rinomina, appending, ecc..

Struttura dei file

Un file ha:

- *Struttura logica*
  - I dati sono organizzati in unità logiche di lunghezza fissa, ma arbitraria detti *blocchi logici (o record)*
  - Ad es. nei sistemi UNIX-like un record è un byte
- *Struttura fisica*
  - I dati sono organizzati in unità fisiche di lunghezza fissa e dipendente dal dispositivo dette *blocchi fisici*
  - **Dimensioni tipiche dei blocchi di unità a disco rigido** variano da 32 a 4096 byte, tipicamente 512 byte
- **La struttura logica e fisica sono differenti!**
  - I dati vengono "impaccati" prima di essere memorizzati in modo da sfruttare al meglio il dispositivo
    - Possibilità di **"frammentazione interna"**

Metodi di accesso

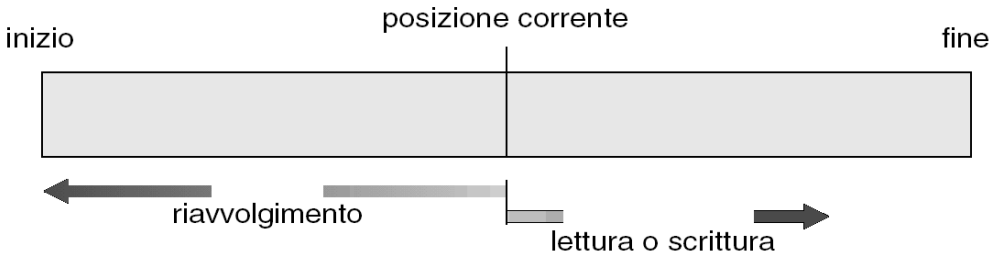
- **Accesso sequenziale**

  lettura

  scrittura

  posizionamento al record precedente o successivo

  reset (o riavvolgimento) all'inizio o alla fine del file



- **Accesso diretto o casuale (presuppone un ordinamento logico dei record)**

  lettura n

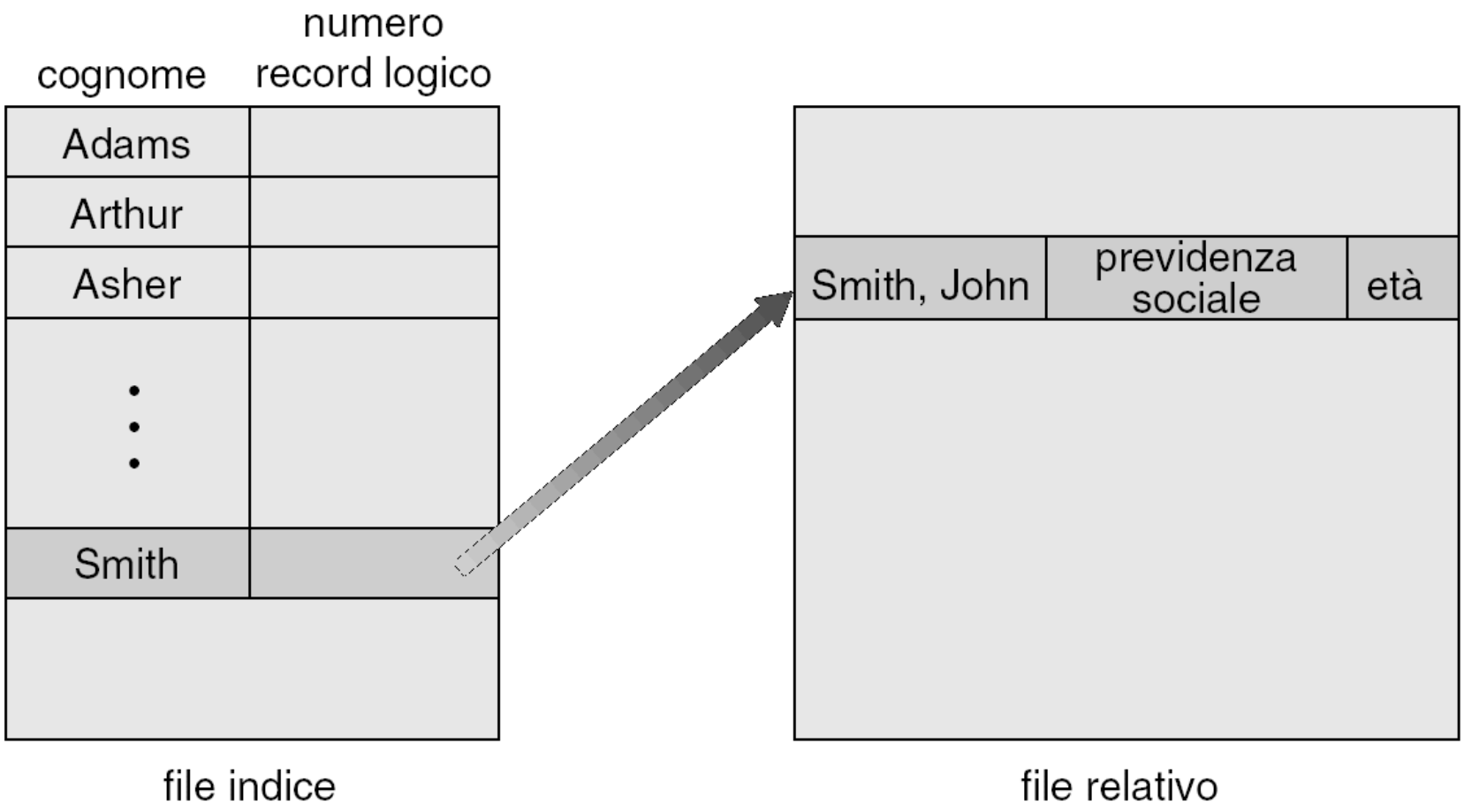
  scrittura n

  posizione a n

  riscrittura n

  n = numero blocco (record) del file rispetto all'inizio del file

Metodo di accesso tramite file INDICE



La ricerca di un blocco nel file prevede prima di ricercare nell'indice (attraverso una chiave) e poi usare il puntatore per accedere al blocco effettivo

Organizzazione del file system

Un file system è organizzato in:

File

- Contengono effettivamente i dati o i programmi

Directory

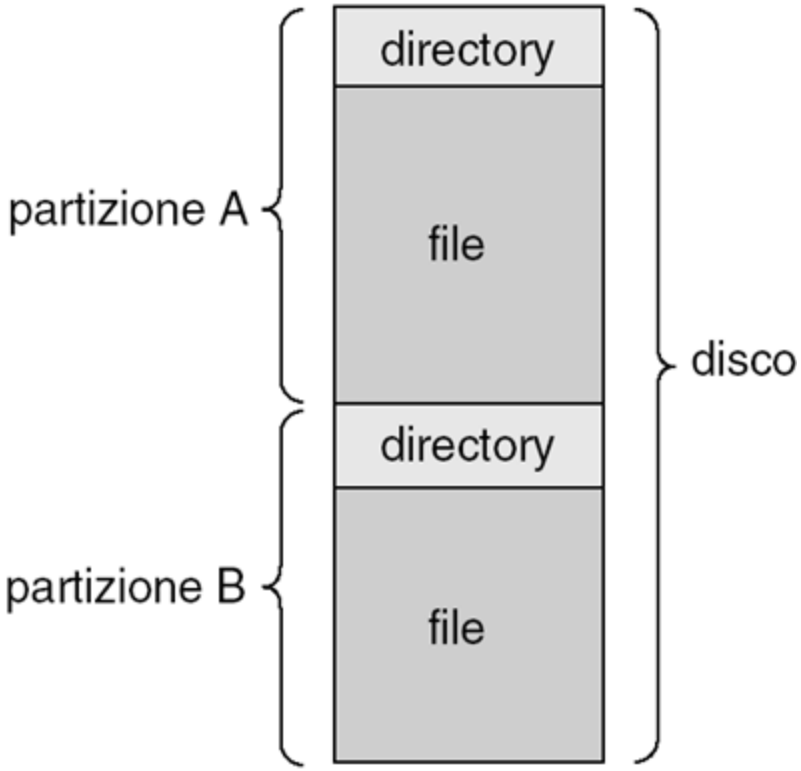
- Contengono informazioni su un gruppo di file e fungono da **indice**
- Contiene gli attributi di file e altre sotto-directory

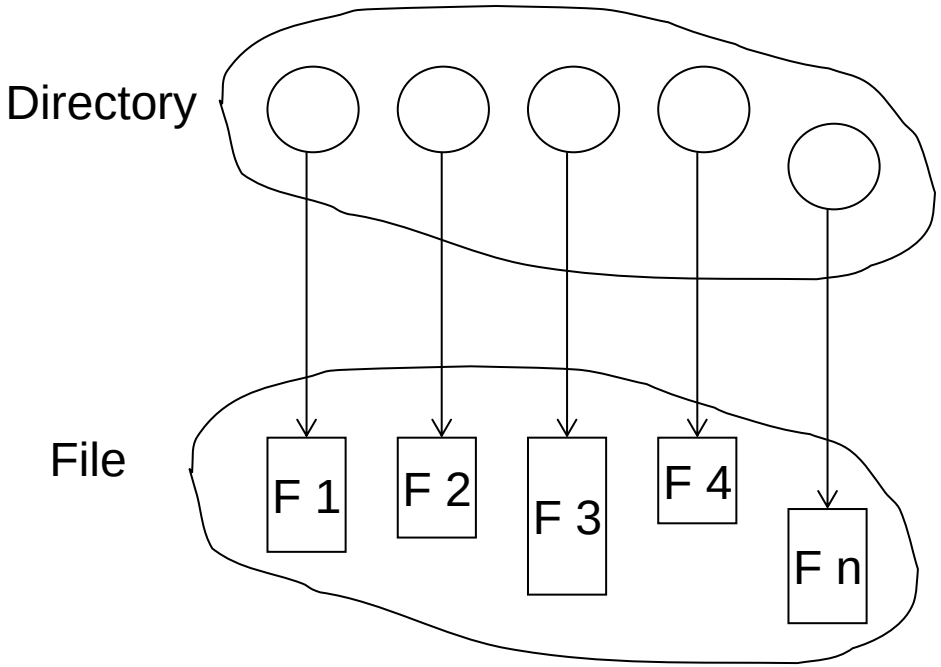
Partizioni (o volumi)

- Contengono insiemi di file correlati
- Considerate come **"dischi virtuali" separati**
- Una partizione ha una **directory del dispositivo** (o **indice del volume**) per tutti i file della partizione

Struttura della directory

- Un insieme di nodi che contengono le informazioni su tutti i file





Operazioni eseguibili su una directory

- Ricerca di un file
- Creazione di un file
- Cancellazione di un file
- Elencare una directory
- Rinominare un file
- Attraversamento del file system

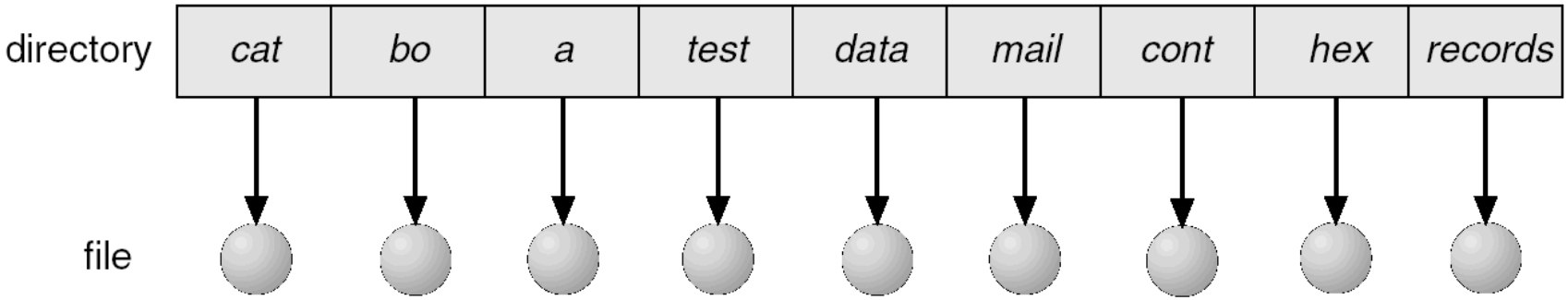
ecc..

Directory a singolo livello

Il file system organizza le directory in una *struttura* al fine di fornire libertà nella scelta dei nomi e la condivisione dei file

Una semplice struttura di directory:

- Una directory a singolo livello **per tutti gli utenti**

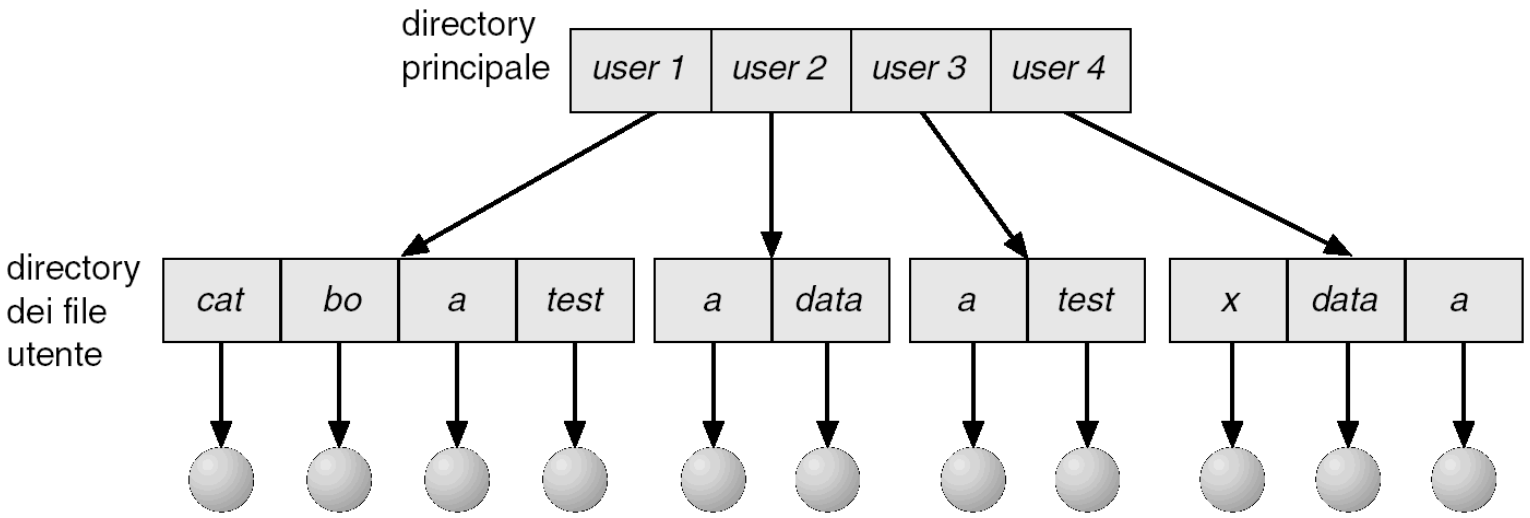


Limiti:

- Problema di rinomina dei file
- Problema di raggruppamento dei file

Directory a due livelli

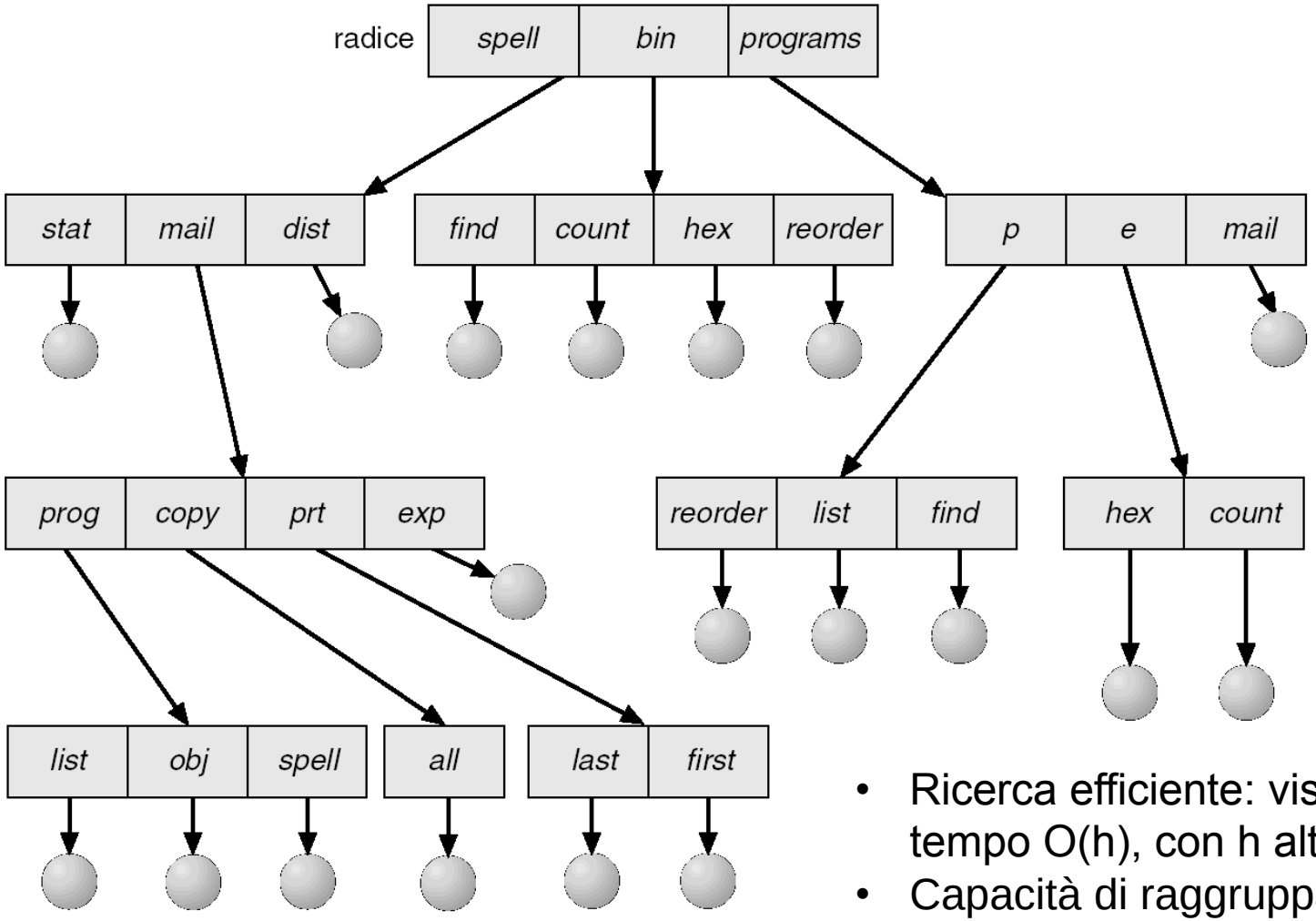
- **Directory separata per ogni utente (*home directory*)**



- Nome del percorso
- E' possibile avere lo stesso nome del file per utenti diversi
- Ricerca efficiente nella home directory
- Ancora poca capacità di raggruppamento

Directory strutturata ad albero (più di 2 livelli)

Ogni utente ha una **directory radice** (**root**) ed una **directory corrente**



- Ricerca efficiente: visita dell'albero in tempo O(h), con h altezza dell'albero
- Capacità di raggruppamento

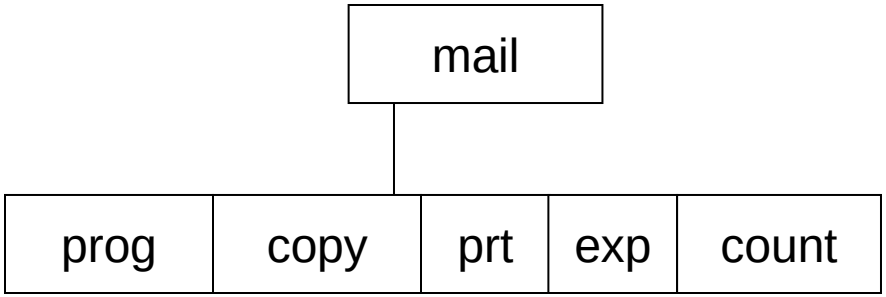
Directory strutturata ad albero (Cont.)

- Percorso **assoluto** e **relativo**
- Ad ogni istante un utente si trova in una directory, la directory corrente. Ogni operazione avviene in essa:
  - La creazione di un nuovo file è realizzata nella directory corrente
  - Cancellazione di un file: **rm** <nome-file>
  - La creazione di una nuova sottodirectory è realizzata nella directory corrente

    mkdir <dir-name>

  ecc..

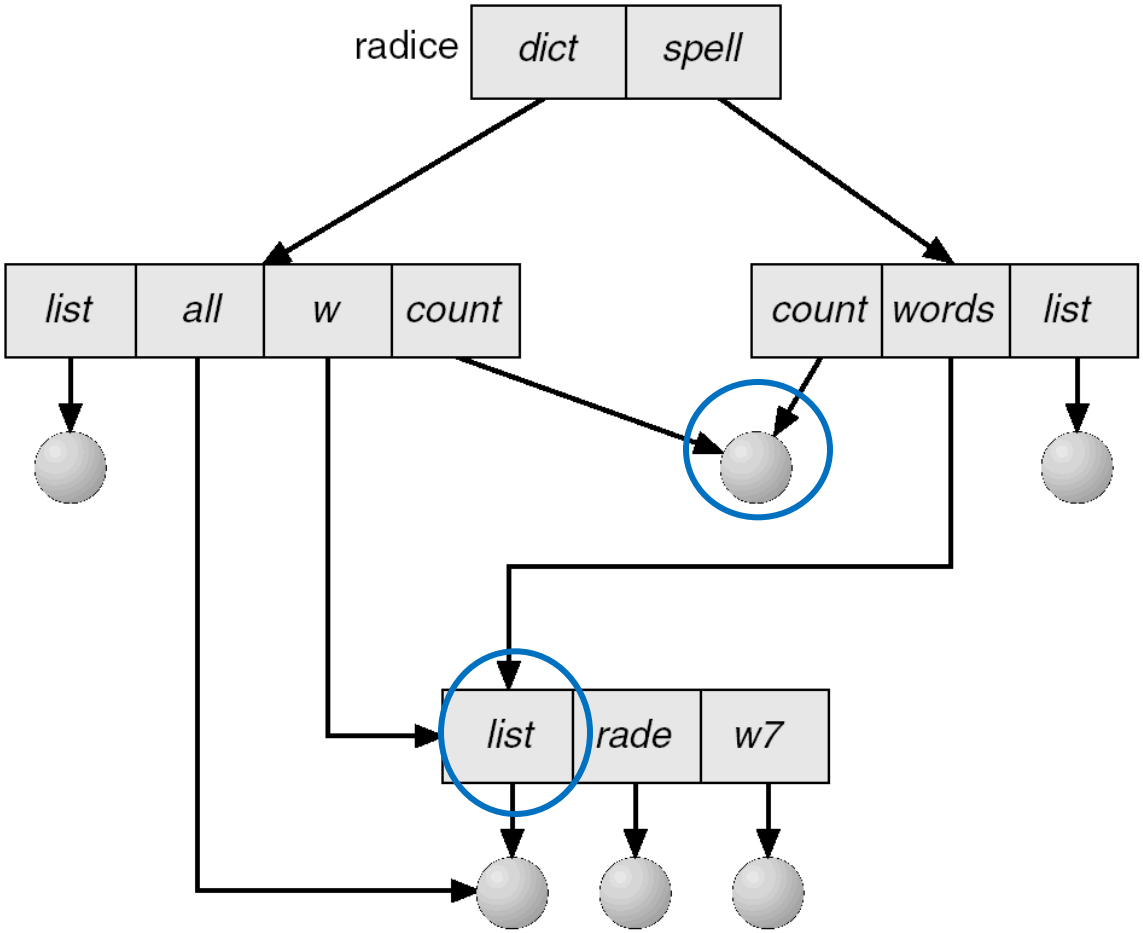
Esempio: **mkdir** *count* nella directory corrente **/mail**



Cancellazione "mail" ⇒ cancellazione dell'intero sottoalbero fondato da "mail"

Directory a grafo aciclico

- **La struttura ad albero non facilita la condivisione**
- **La struttura a grafo aciclico permette sottodirectory e file condivisi mediante link**



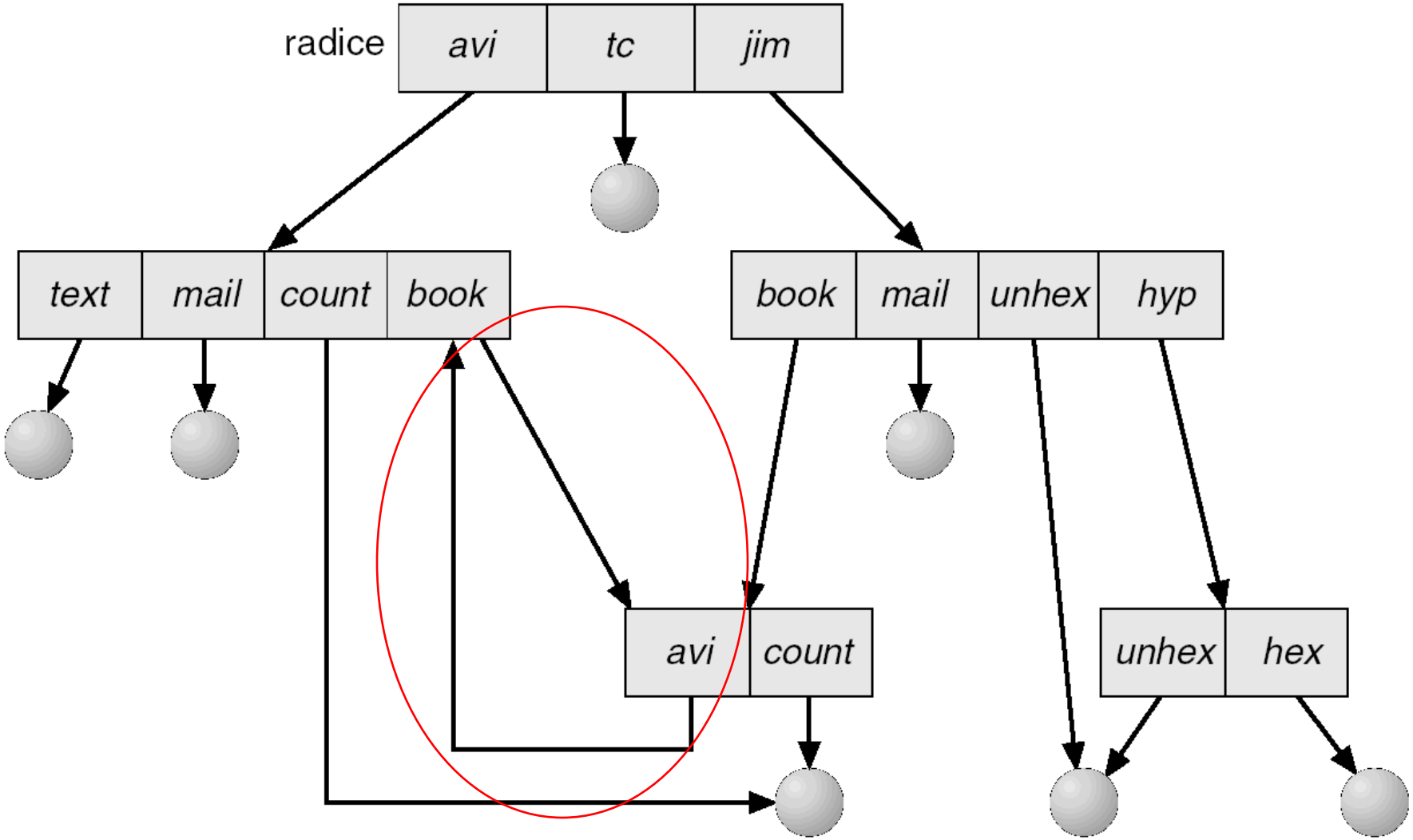
Directory a grafo aciclico (Cont.)

- Due differenti nomi (omonimia – aliasing)
- Se *dict* cancella *list* ⇒ problema puntatore pendente

  Soluzioni:

  1. **Cancellare il file e i puntatori pendenti restano tali**; ma il SO lo segnala quando un utente prova ad accedervi -- in Unix- Windows –like
  2. **Cancellare il file solo dopo la cancellazione dell'ultimo link** – richiede lista di tutti i riferimenti al file o anche solo un "contatore dei riferimenti"
  3. **Non permettere le directory condivise o i link** – ad es, MS-DOS (struttura ad albero)

Directory a grafo generale



Directory a grafo generale (Cont.)

- **Come possiamo garantire che non ci siano cicli?**
  - Schema di **garbage collection**: "raccolta di spazzatura" attraversando l'intero file system alla ricerca di link "morti"
    - contatore dei riferimenti non nullo per via del ciclo, ma i link potrebbero non esistere più
  - Ogni volta che viene aggiunto un nuovo link usare un algoritmo per rilevare i cicli nei grafi
  - Permettere solo link ai file, non sotto-directory

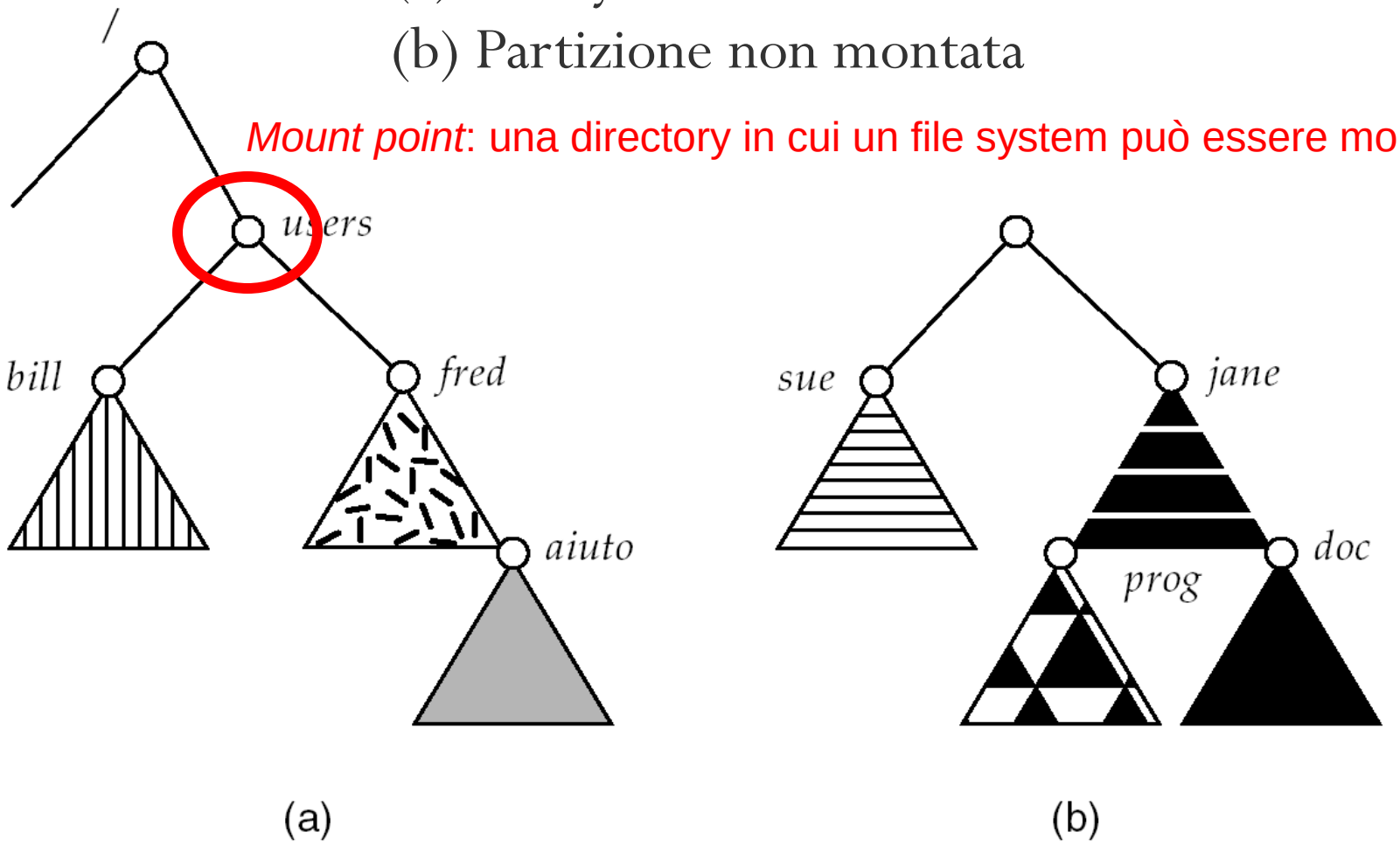
Montaggio del file system

Connette il file system alla struttura delle directory del sistema

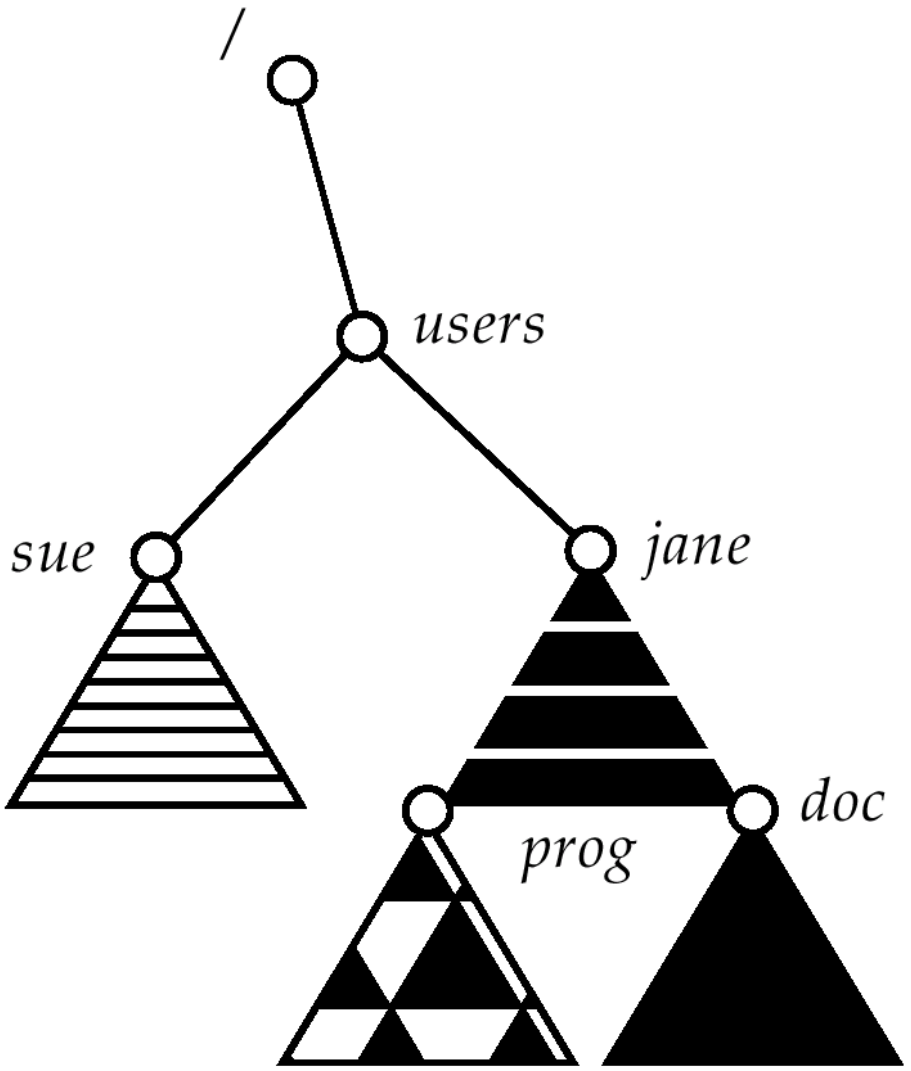
(a) File system esistente

(b) Partizione non montata

Mount point: una directory in cui un file system può essere montato



Punto di montaggio



Condivisione dei file

- La condivisione di file su sistemi multiutente è preferibile
- La condivisione può essere realizzata attraverso uno schema di condivisione
- **Sui sistemi distribuiti**, i file possono essere condivisi **attraverso una rete**
- Il **Network File System (NFS)** è un comune metodo di condivisione dei file tra macchine in remoto tramite il modello client-server

Protezione

- Il proprietario/creatore del file dovrebbe essere in grado di controllare:
  - che cosa può essere fatto
  - da chi

- Tipi di accesso:
  - Lettura
  - Scrittura
  - Esecuzione
  - Accodamento (append)
  - Cancellazione
  - Elenco (list) – elenco del nome e degli attributi del file

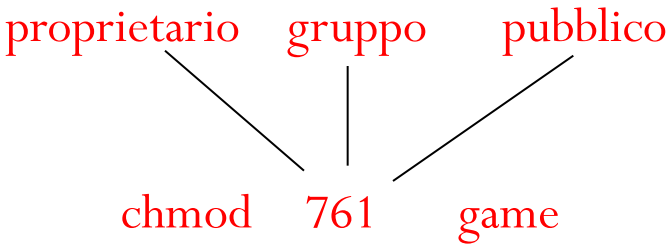
Lista degli accessi (ACL) e gruppi

- **Modo di accesso**: lettura, scrittura, esecuzione RWX
- **Tre classi di utenti**:

Unix-like

			RWX
a) accesso proprietario	7	⇒	1 1 1
b) accesso gruppo	6	⇒	1 1 0
c) accesso pubblico	1	⇒	0 0 1

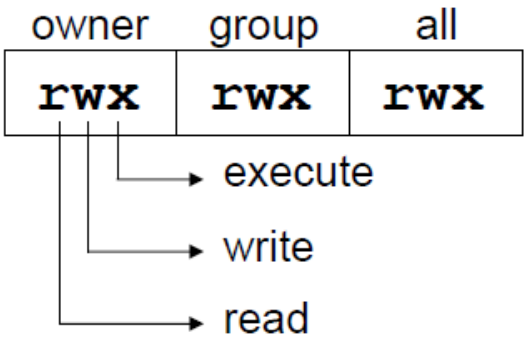
- Chiedere al gestore di creare un gruppo (nome unico), per esempio G, ed aggiungere alcuni utenti al gruppo
- Per un file particolare (ad esempio *game*) o una sottodirectory, occorre definire un appropriato accesso



Attaccare un gruppo al file: chgrp G game

Protezione: Unix-like

- Ancora un esempio:



rwx	rwx	rwx
111	101	101
7	5	5

- Es. di output facendo il list ("ls") di una directory:

-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/