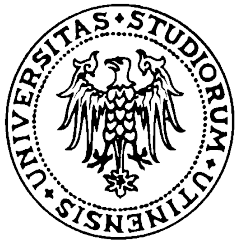
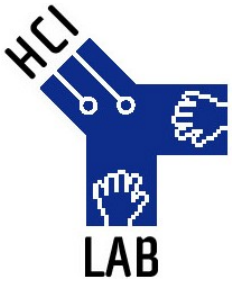
Interfaccia del file system



Fabio Buttussi

HCI Lab
Dipart. di Matematica ed Informatica
Università degli Studi di Udine

www.dimi.uniud.it/buttussi

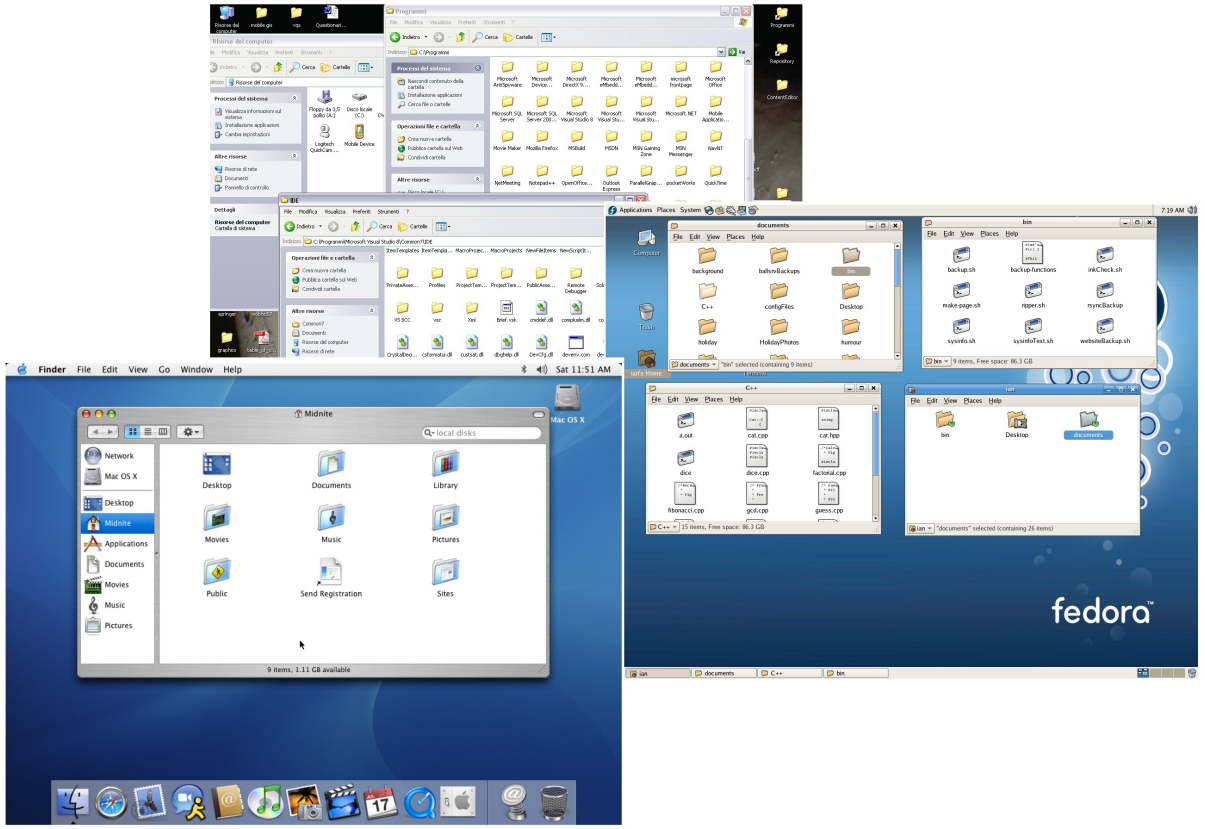


Cos'è il file system?

- Consiste in:
  - Insieme di **file**
  - **Strutture dati** per organizzare i file nel sistema e fornire informazioni su di essi
- Permette l'**accesso** a **dati** e **programmi** del **sistema operativo** e degli **utenti**

Come lo vediamo?

- E’ indipendente dall’aspetto grafico con cui vengono rappresentati file e directory



Cos'è un file?

- **Spazio d'indirizzamento logico contiguo** per la memorizzazione di informazioni in **memoria secondaria**
- Permette un'**astrazione** dalle caratteristiche fisiche dei dispositivi di memoria
- Le informazioni si possono scrivere in memoria secondaria **soltanto** all'interno di file
- I file rappresentano **programmi** (sorgente, oggetto) e **dati** (numerici, alfabetici, alfanumerici, binari)

Attributi dei file

- **Attributi leggibili**: nome, tipo, dimensione, protezione, ora, data, identificativo utente
- **Altri attributi**: identificatore, locazione
- L'informazione sui file viene mantenuta in specifiche **strutture dati** su disco

Tipi di file

- **Estensione** dei file = tecnica comune per includere il tipo nel nome del file
- L'estensione viene usata dal sistema per stabilire il **tipo del file** e le **operazioni eseguibili** su tale file
- In MacOSX ciascun file possiede un **attributo** (impostato alla creazione) contenente il **nome del programma** che lo ha generato
- Unix memorizza un **codice** (***magic number***) all'inizio di alcuni tipi di file per indicarne in modo generico il tipo

Struttura dei file (I)

- **Nessuna** - sequenza di parole, byte
- Struttura a **record**
  - Righe
  - Lunghezza fissa
  - Lunghezza variabile
- Struttura **complessa**
  - Documento formattato
- Tipicamente, un file ha una struttura **definita secondo il tipo**

Struttura dei file (II)

- La struttura può essere decisa da:
  - **Sistema operativo**
  - **Programma applicativo**

- Tutti i sistemi operativi devono prevedere almeno un tipo di struttura, i **file eseguibili**

- Alcuni SO prevedono la **gestione diretta** di diverse strutture di file e mettono a disposizione operazioni specifiche

Struttura interna dei file

- I **dischi** sono suddivisi in **record fisici** (**blocchi**) di **dimensione fissa**
- I **record logici** di cui è costituito un file devono essere immagazzinati nei blocchi fisici
- Ciò avviene mediante **impaccamento** di un certo numero di record logici in blocchi fisici
- UNIX
  - file = sequenza di byte
  - record logico = byte
- Si può avere **frammentazione interna**

Operazioni sui file

- Un file è un tipo di dato astratto su cui, tramite chiamate di sistema, si possono effettuare diverse operazioni
  - **creazione (*create*)**: si deve allocare il file e creare un elemento che lo descriva nelle strutture dati del file system
  - **lettura (*read*)**: puntatore di lettura; si può combinare con il precedente
  - **scrittura (*write*)**: puntatore di scrittura alla prossima posizione
  - **riposizionamento (*reposition*)**: modifica del puntatore di posizione corrente
  - **cancellazione (*delete*)**: rilascio spazio e cancellazione elemento dalle strutture dati del file system

Operazioni sui file

- La maggior parte delle operazioni su file richiede la ricerca nelle strutture dati del file system (in particolare, la struttura della **directory**) dell'elemento associato al file
- Per evitare una ricerca continua, si utilizza la chiamata di sistema ***open()*** che inserisce un riferimento al file in una ***tabella dei file aperti***, velocizzando le operazioni successive
- La chiamata di sistema ***close()*** rimuove l'elemento relativo ad un file dalla tabella dei file aperti
- Negli ambienti multiutente si utilizzano una tabella per ciascun processo ed una *tabella di sistema*

Gestione file aperti

- Diversi dati sono necessari per gestire i file aperti:
  - ***Puntatore al file***: puntatore all'ultima posizione di lettura e scrittura, unico per ogni processo che opera sul file
  - ***Contatore dei file aperti***: tiene traccia del numero di open() e close() per rimuovere l'elemento dalla tabella dei file dopo l'ultima chiusura
  - ***Posizione del file nel disco***: mantenuta in memoria per evitarne il prelevamento ad ogni operazione
  - ***Diritti d'accesso***: modalità d'accesso con cui ogni processo ha aperto un file

Lock sui file aperti

- Funzionalità fornita da alcuni sistemi operativi e file system
- Permette di **gestire l'accesso** a file condivisi da diversi processi
- Può essere **condiviso** o **esclusivo**
- Può essere **obbligatorio** o **consigliato**:
  - *obbligatorio* – una volta acquisito da un processo, il SO garantisce che nessun altro processo possa accedere al file (Windows)
  - *consigliato* – è compito dei programmatori gestire la corretta acquisizione e cessione dei lock (Unix)

Lock sui file in Java

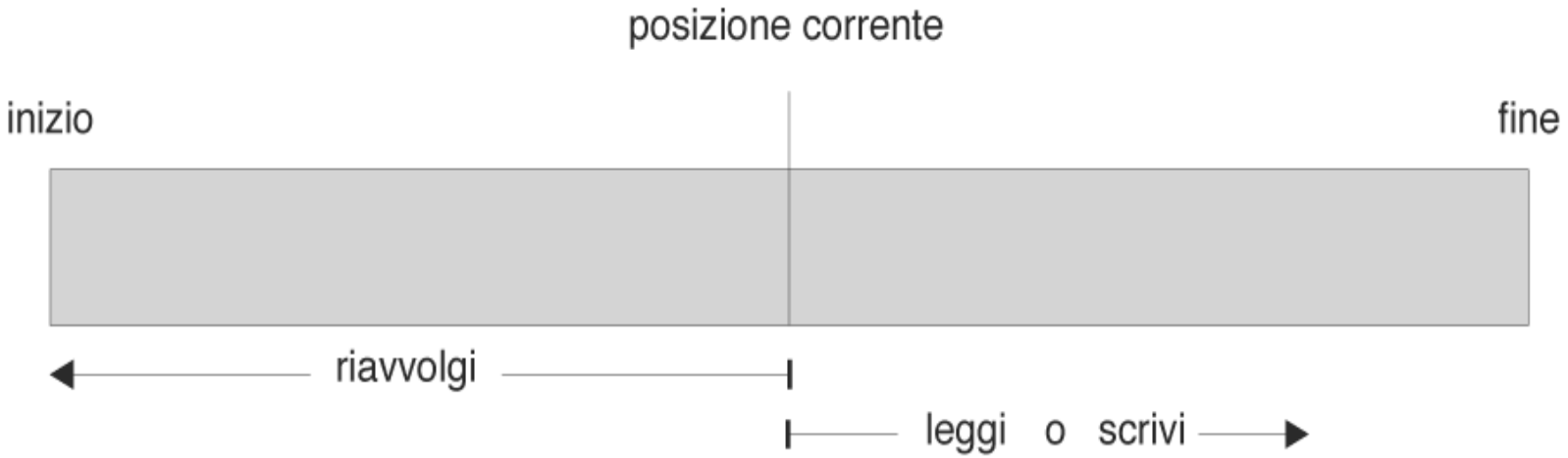
```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
            RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
            // acquisisce il canale per il file
            FileChannel ch = raf.getChannel();
            // acquisisce lock esclusivo per la prima metà del file
            exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
            /** Modifica i  dati . . . */
            // rilascia il lock
            exclusiveLock.release();
```

Lock sui file in Java

```
                    // acquisisce lock condiviso per la seconda metà del file
                    sharedLock = ch.lock(raf.length()/2+1, raf.length(),
                    SHARED);
                    /**Legge i dati . . . */
                    // rilascia il lock
                    sharedLock.release();
            } catch (java.io.IOException ioe) {
                    System.err.println(ioe);
            }finally {
                    if (exclusiveLock != null)
                    exclusiveLock.release();
                    if (sharedLock != null)
                    sharedLock.release();
            }
    }
}
```

Metodi d'accesso ai file

- **Accesso sequenziale**
  - Basato su modello a nastro
  - Le informazioni si elaborano un record dopo l'altro utilizzando chiamate *read next* e *write next*



Metodi d'accesso ai file

- **Accesso diretto**
  - Permette di accedere arbitrariamente ad ogni blocco di cui è costituito un file
  - Metodo utile per accedere a grandi quantità di dati (es: basi di dati)
  - Il numero di blocco è un parametro delle operazioni read() e write()
    - In alternativa, si possono usare *read next* e *write next* come nell'accesso sequenziale e aggiungere un'operazione *position to n*
    - Per semplificare l'allocazione dei file, il numero di blocco è di tipo relativo

Simulazione dell'accesso sequenziale ad un file ad accesso diretto

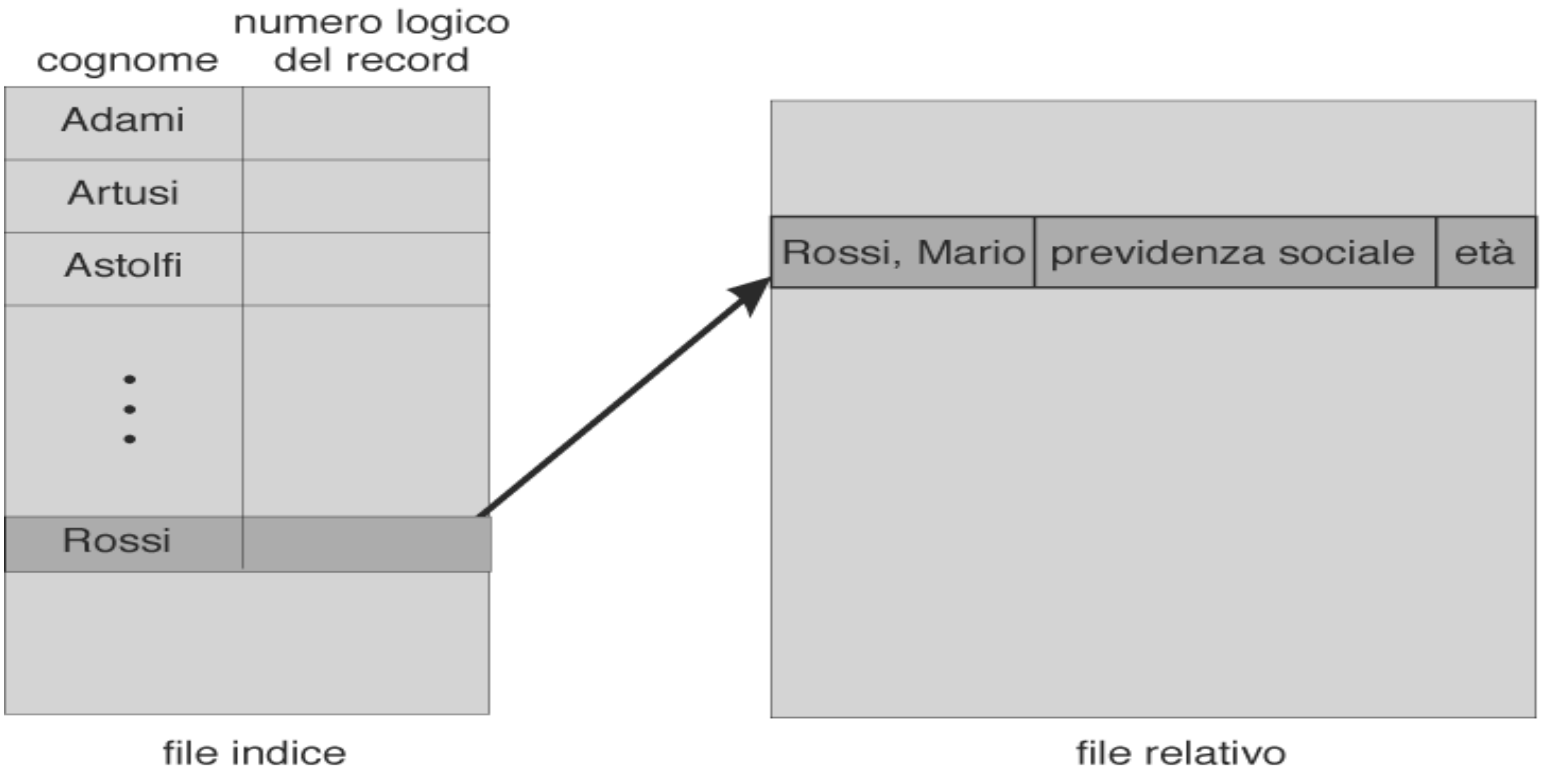
- Non tutti i SO gestiscono entrambi i tipi di accesso

Accesso sequenziale	Realizzazione nel caso di accesso diretto
`reset`	`cp = 0;`
`read next`	`read cp;` `cp = cp + 1;`
`write next`	`write cp;` `cp = cp + 1;`

Altri metodi d'accesso

- Si possono costruire altri metodi d'accesso usando l'accesso diretto come base
- Tali metodi sono tipicamente basati sull'utilizzo di un ***indice*** per i file, contenente puntatori ai vari blocchi
- Per trovare un elemento di un file, si controlla prima l'indice per identificare il blocco desiderato
- Questo meccanismo permette di **limitare il numero di operazioni** di I/O necessarie alla ricerca
- Esempio: metodo ISAM di IBM (il file indice è a sua volta indicizzato)

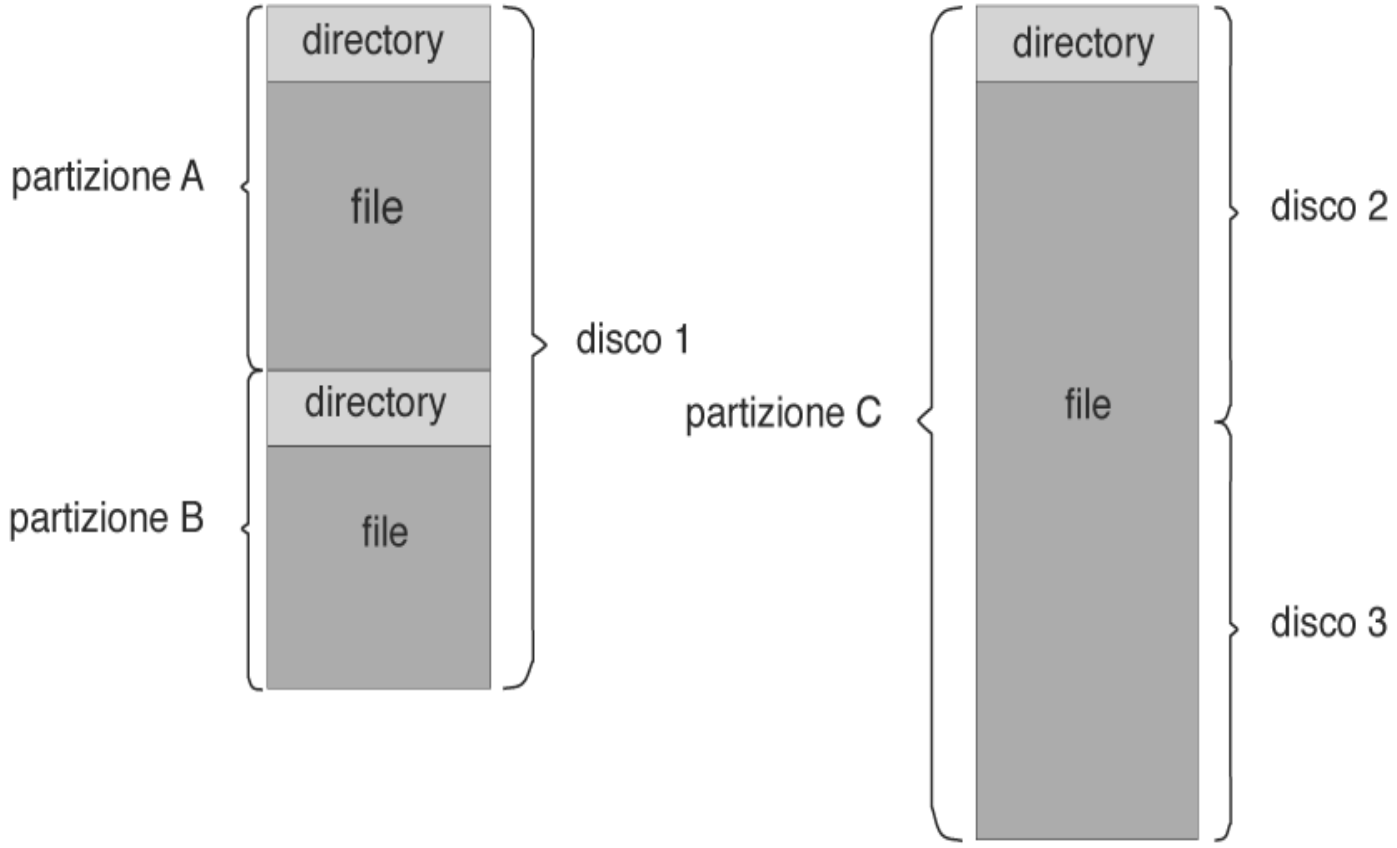
Esempio di indice e relativo file



Organizzazione di un file system

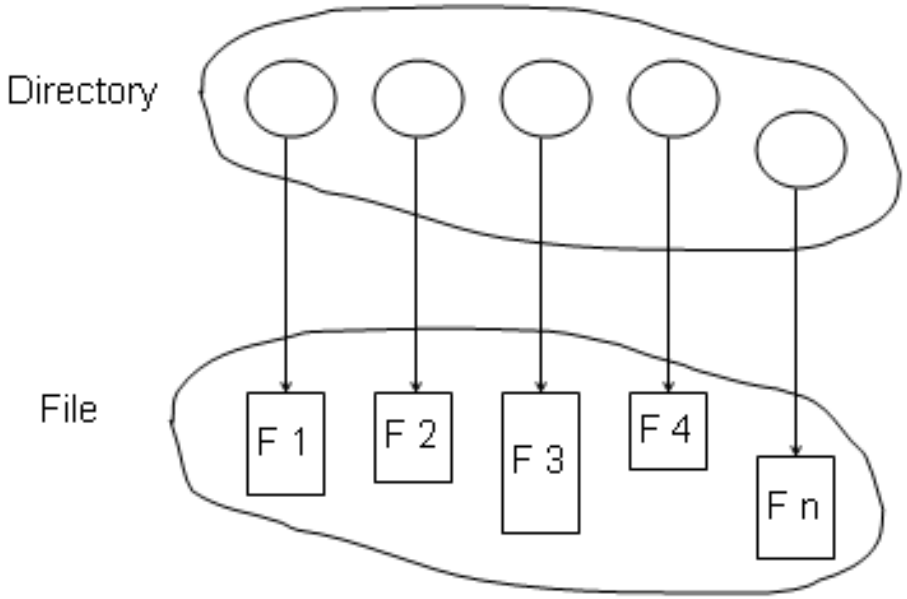
- Su un **unico disco** possono essere presenti **più file system** ed altre aree come lo spazio di swap
- Ogni parte in cui si può suddividere logicamente un disco viene chiamata **partizione**
- Non è detto che ogni partizione contenga un file system
- In generale, si indica con ***volume*** ogni **blocco di memoria secondaria contenente un file system**
- Ogni volume contenente un file system deve contenere le informazioni su tutti i file presenti
- Tali informazioni risiedono in una struttura dati chiamata **struttura della directory** che risiede su disco

Tipica organizzazione di un file system



Struttura della directory

- La struttura della directory può essere vista come una **tabella** che associa **nomi di file** ad **elementi della tabella** stessa
- Ogni elemento della tabella contiene (o permette di accedere a) informazioni su un particolare file



Operazioni relative alla struttura della directory

- Ricerca di un file (o file multipli)
- Creazione di un file
- Cancellazione di un file
- Elenco dei file
- Rinomina di un file
- Attraversamento del file system

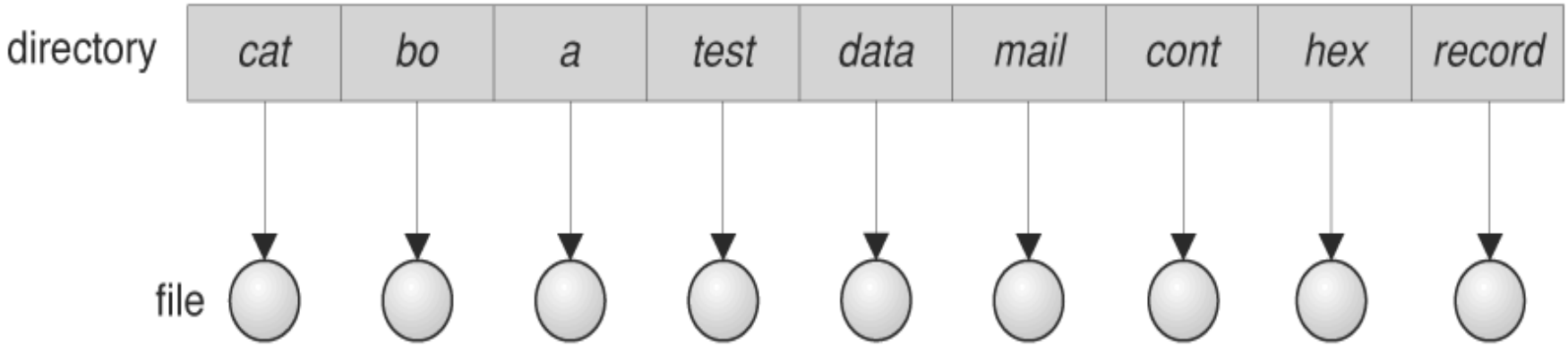
Organizzazione logica della struttura della directory

- Si possono utilizzare vari schemi per definire la struttura logica della directory

- Vantaggi
  - **Efficienza** – ricerca rapida di un file
  - **Naming** – conveniente per gli utenti
    - Due utenti possono utilizzare lo stesso nome per file differenti
    - Lo stesso file può avere nomi differenti
  - **Raggruppamento** – raggruppamento logico dei file in base alle proprietà (e.g., programmi Java, giochi, …)

Directory a livello singolo

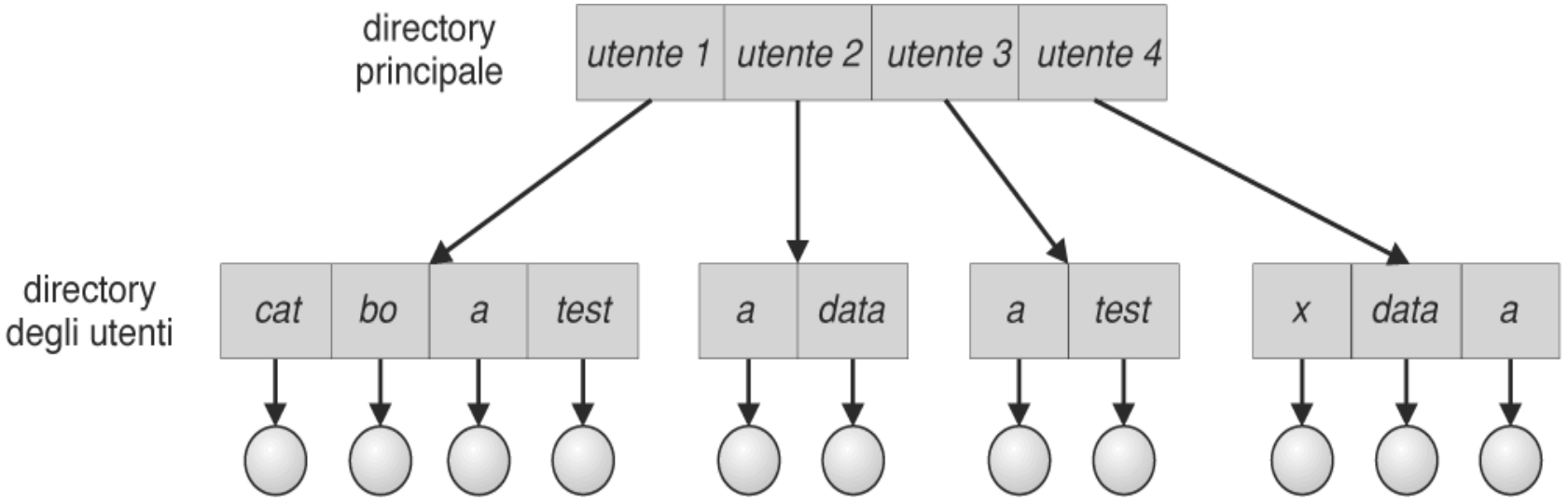
- Directory **unica** per tutti i file e tutti gli utenti



- Porta a problemi di **naming** (i file devono avere nomi unici) e di **raggruppamento**

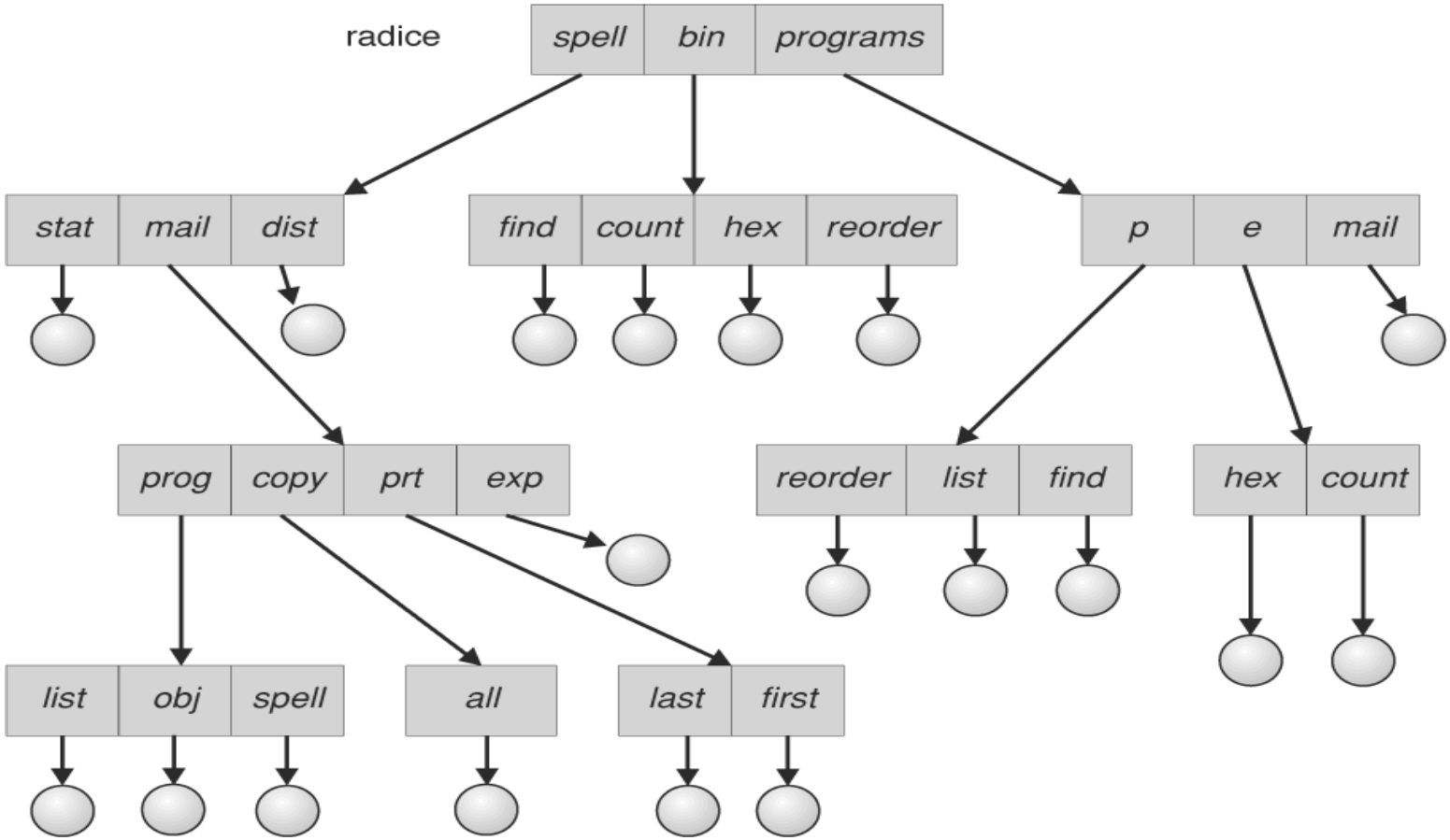
Struttura della directory a due livelli

- Directory separate per ogni **utente**



- Ricerca efficiente
- Utenti diversi possono avere file con lo stesso nome
- Nome del percorso (per accedere a file di altri utenti)
- Nessuna funzionalità di raggruppamento
- Ricerca dei file di sistema (percorso di ricerca)

Struttura della directory ad albero



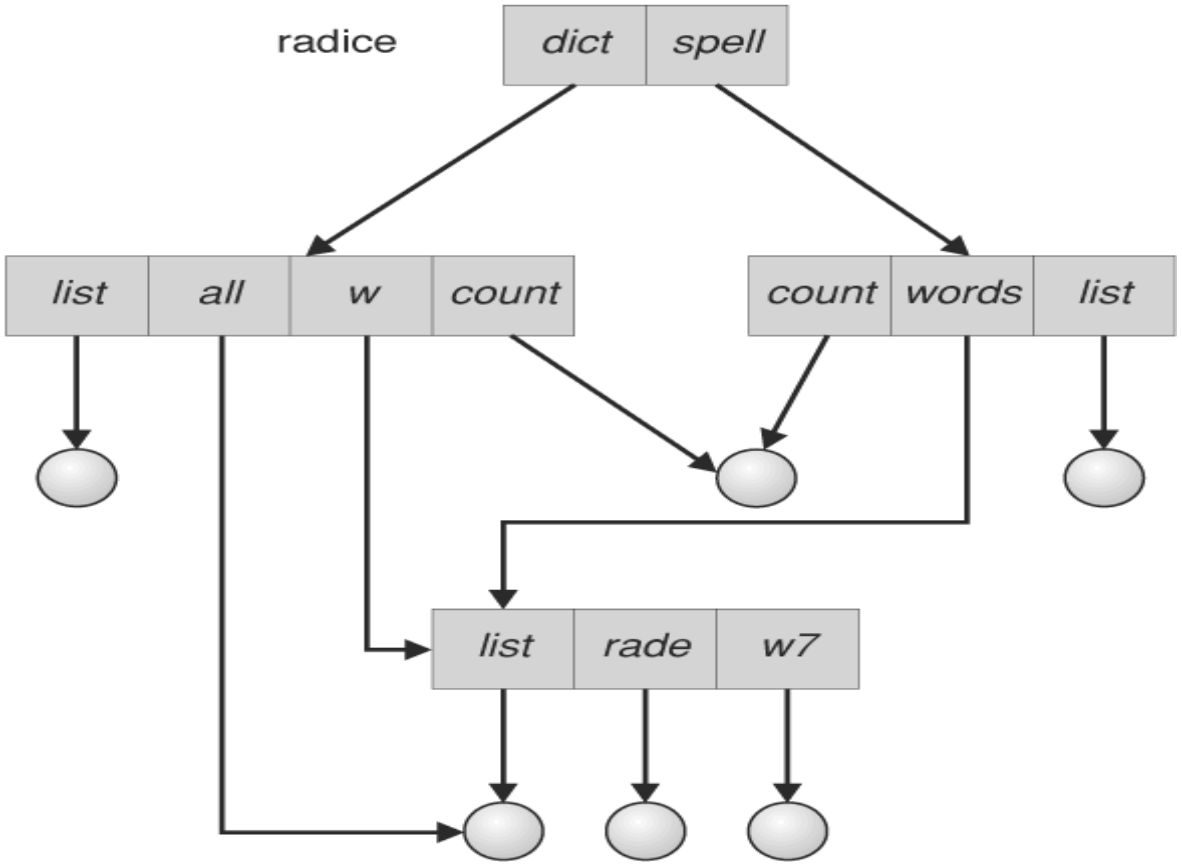
- Ricerca efficiente e possibilità di raggruppamento

Struttura della directory ad albero

- Le directory sono **file** trattati in modo **speciale**
- Ogni utente dispone di una **directory corrente** (working directory), modificabile tramite chiamata di sistema
- I nomi di percorso possono essere **assoluti** (partono dalla radice) o **relativi** (partono dalla working directory)
- Creazione e cancellazione di file o directory vengono fatte nella **working directory** tramite opportune chiamate di sistema
- Cancellazione di una directory
  - DOS, impossibile se la directory contiene file
  - Unix, porta alla cancellazione dell'intero sottoalbero la cui radice è rappresentata dalla directory

Directory con struttura a grafo aciclico

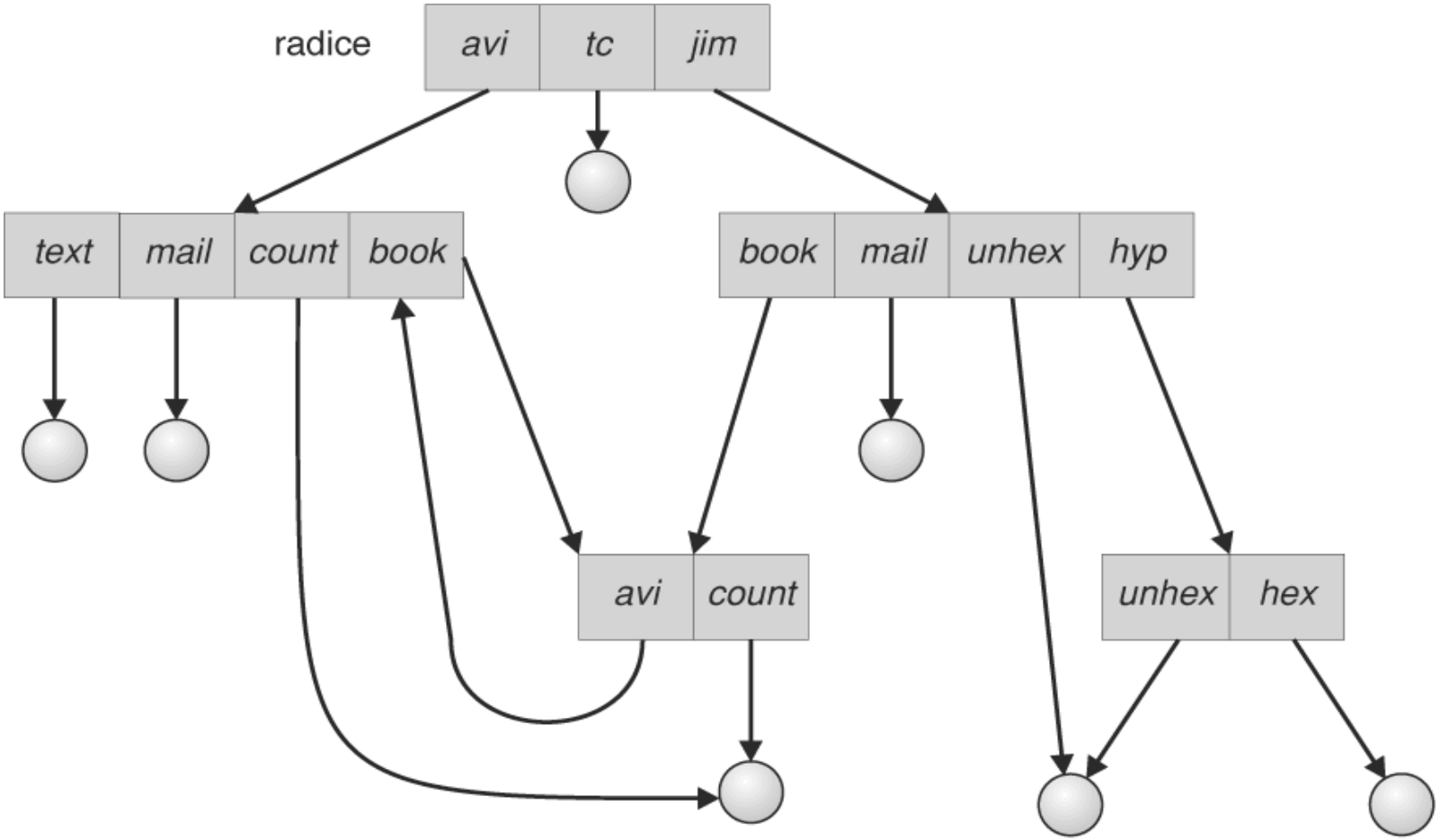
- Esistono **sottodirectory** e file **condivisi**



Directory con struttura a grafo aciclico

- Si può realizzare prevedendo un nuovo tipo di elemento
  - **Collegamento** – puntatore ad altro file o directory
  - **Risoluzione di un collegamento** – seguire il puntatore per localizzare il file
- In alternativa, si possono duplicare le informazioni relative ai file nelle directory di condivisione (problemi di coerenza)
- Problemi
  - Nomi diversi possono riferirsi allo stesso file (**aliasing**)
  - Puntatori orfani a file cancellati. Soluzioni:
    - Puntatori all'indietro, permettono la cancellazione di tutti i puntatori
    - Gestione manuale
    - Conservazione di file fino alla cancellazione di tutti i puntatori (lista o contatore)

Directory a grafo generale

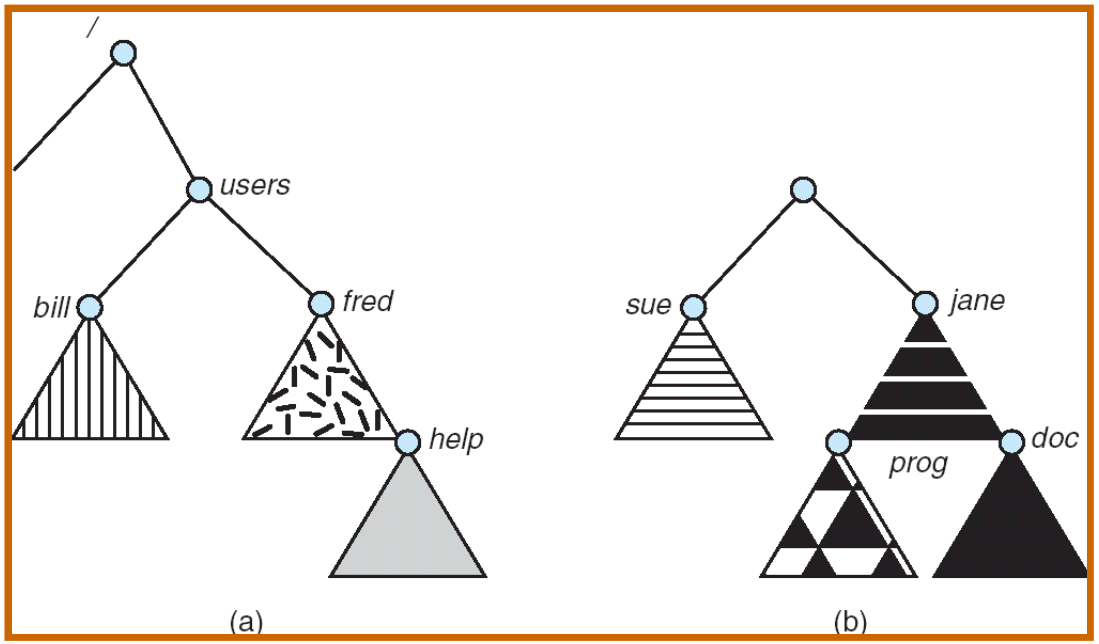


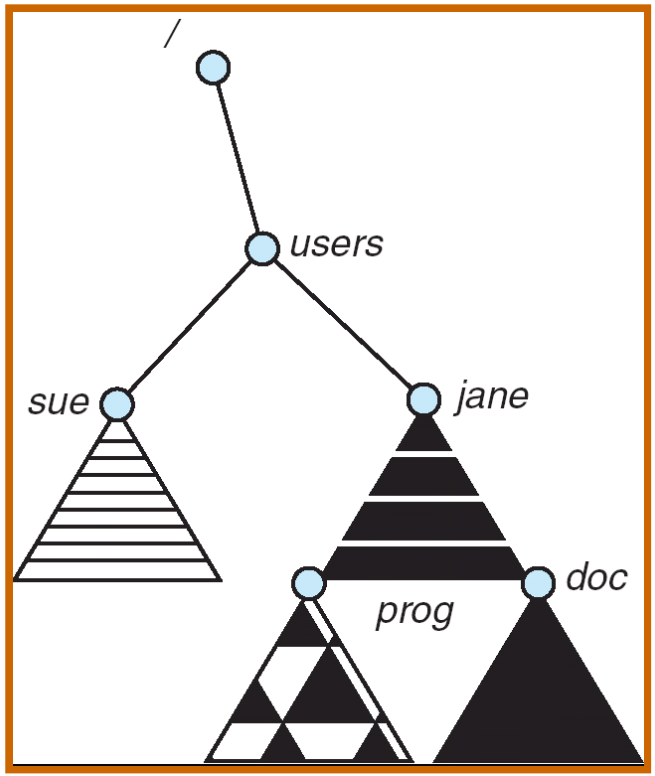
Directory a grafo generale

- I **cicli** all'interno di una struttura di directory possono portare a problemi di prestazioni e di correttezza
- Come evitare la generazione di cicli?
  - Permettere solo collegamenti a file e non a directory
  - Al momento dell'aggiunta di un nuovo collegamento, utilizzare un algoritmo di identificazione di cicli per determinare se l'aggiunta è possibile
- In caso di cicli, il contatore dei riferimenti ad un file o directory può essere > 0 anche se il file non è più raggiungibile
  - Serve garbage collection

Montaggio di un file system

- Un file system deve essere **montato** prima del suo utilizzo
- Il montaggio avviene ad un **punto di montaggio**
- Il contenuto precedente, di solito, diventa inaccessibile





Montaggio sotto Unix

- In Unix, il montaggio dei file system avviene utilizzando il comando ***mount***
- Mount effettua l'associazione della directory che costituisce il punto di mount con la radice del nuovo file system da collegare
- Il comando

  mount /dev/sd1c /users

  installa il file system memorizzato sulla partizione di disco rappresentata da ***/dev/sd1c*** sotto il path ***/users***
- Un elenco dei file system collegati a un particolare sistema è memorizzato nel file ***/etc/fstab***
- La disconnessione dei file system (non utilizzati) avviene mediante il comando ***unmount***

Esempio di fstab

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>      <options>                          <dump>  <pass>
proc            /proc           proc        defaults                             0       0
/dev/hda6       /               reiserfs    notail                               0       1
/dev/hda1       /media/hda1     vfat        defaults,utf8,umask=007,gid=46       0       0
/dev/hda5       /media/hda5     vfat        defaults,utf8,umask=007,gid=46       0       1
/dev/sda1       /media/sda1     ntfs        defaults,nls=utf8,umask=007,gid=46 0       1
/dev/hda7       none            swap        sw                                   0       0
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto                          0       0
```

- Ogni linea è composta da 6 campi:
  - il **dispositivo** che contiene un file system (tra cui il file system virtuale *proc* che permette di riassumere le informazioni generali del sistema)
  - la **directory**, o mount point, in cui sarà possibile accedere al contenuto dei dispositivi (da notare che per la swap non è richiesto il mount point)
  - il **tipo** di file system (i file system supportati si possono vedere nella man page di fstab)
  - le **opzioni** che regolano l'accesso al dispositivo (le opzioni si possono vedere nella man page di mount)
  - indica se il dispositivo deve essere usato dal comando **dump** per farne dei backup (0 disattivato, 1 attivato), l'opzione è obsoleta
  - indica se il dispositivo deve essere controllato dal comando **fsck**, nell'ordine da 1 in poi (0 indica non controllare)

Condivisione e protezione file

- Nei sistemi multiutente, la **condivisione** di file è una funzionalità desiderabile
- Il proprietario/creatore del file dovrebbe poter impedire accessi impropri ai file, controllando:
  - Cosa può essere fatto
  - Da chi
- **Tipi di accesso**
  - Lettura
  - Scrittura
  - Esecuzione
  - Aggiunta
  - Cancellazione
  - Lista

Controllo degli accessi ai file

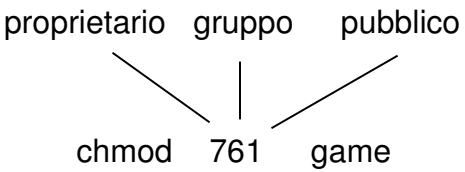
- Tramite **lista di controllo degli accessi** che elenca tutti gli utenti che possono accedere ad un file e come
  - permette controlli complessi
  - dimensioni elevate
- Tramite **liste condensate** basate su suddivisione degli utenti in proprietario, gruppo, pubblico
  - il proprietario è l'utente che ha il maggior controllo su un file o directory
  - i membri del gruppo possono effettuare un sottoinsieme delle operazioni del proprietario
  - gli identificatori del gruppo e del proprietario di un certo file o directory (**ID utente e ID di gruppo**) sono memorizzati insieme con gli altri attributi del file
  - si può combinare con il metodo precedente
- Tramite password

Controllo degli accessi in Unix

- Modalità d'accesso: lettura, scrittura, esecuzione
- Tre classi di utenti

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

- Tipicamente si crea un gruppo (es. MioGruppo) e si aggiungono utenti a tale gruppo
- Per ogni file (e.g., *game*) o sottodirectory, viene definito un accesso appropriato

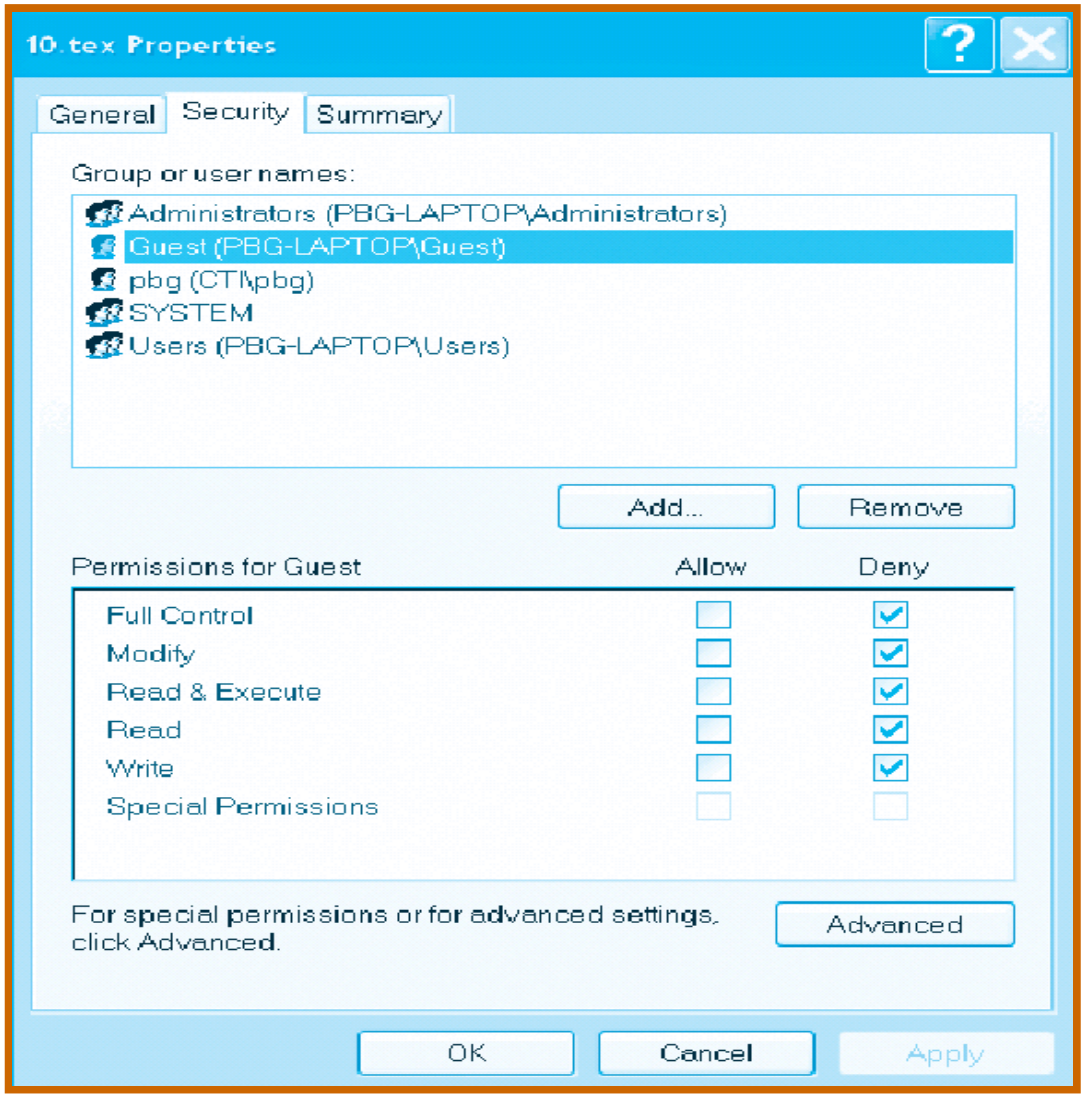


- Associazione di un gruppo ad un file: **chgrp MioGruppo game**

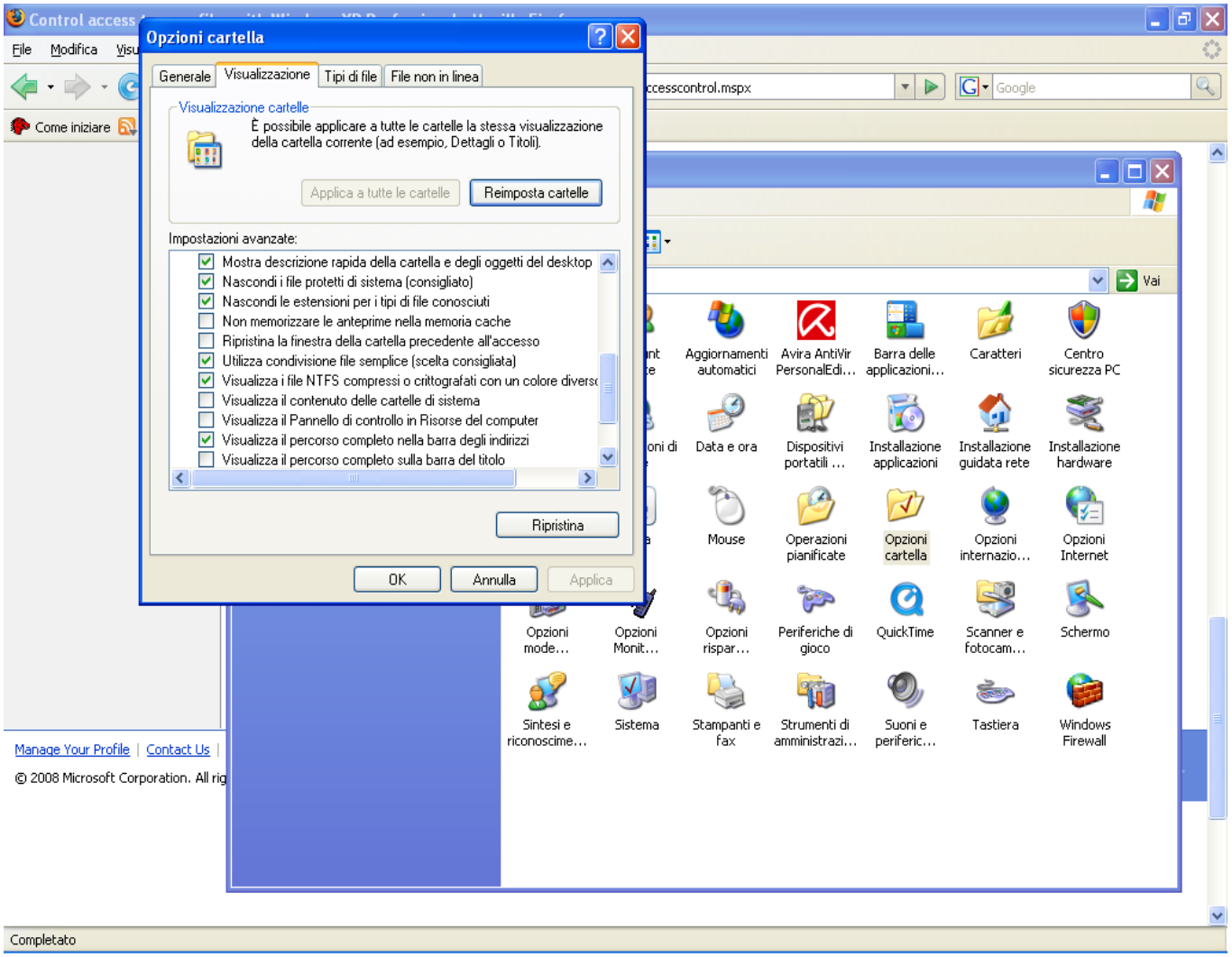
Controllo degli accessi in Unix

```
-rw-rw-r--  1 pbg  staff      31200  set   3 08:30  intro.ps
drwx------  5 pbg  staff        512  lug   8 09:33  privato/
drwxrwxr-x  2 pbg  staff        512  lug   8 09:35  doc/
drwxrwx--   2 pbg  studente     512  ago   3 14:13  studente-prog/
-rw-r--r--  1 pbg  staff       9423  feb  24  1993  program.c
-rwxr-xr-x  1 pbg  staff      20471  feb  24  1993  program
drwx--x--x  4 pbg  facoltà      512  lug  31 10:31  lib/
drwx------  3 pbg  staff       1024  ago  29 06:52  mail/
drwxrwxrwx  3 pbg  staff        512  lug   8 09:35  test/
```

Controllo degli accessi in Windows XP



Controllo degli accessi in Windows XP



Condivisione di file - File system remoti

- Utilizzo di funzionalità di rete per accedere ai file system tra sistemi diversi
  - Manualmente attraverso programmi quali FTP
  - Automaticamente, sfruttando **file system distribuiti**
  - In modo semiautomatico attraverso il **world wide web**
- Il modello **client-server** permette ai client di montare file system remoti attraverso i server
  - Un server può servire client multipli
  - L'identificazione dei client e degli utenti è insicura e complicata
  - **NFS** è un protocollo standard UNIX per la condivisione di file
  - **CIFS** è un protocollo standard Windows
  - Le chiamate di sistema standard vengono tradotte in chiamate remote
- Sistemi Informativi Distribuiti **(distributed naming services)** come LDAP, DNS, NIS, Active Directory implementano un accesso unificato ad informazioni necessarie per il calcolo remoto

Condivisione di file - Malfunzionamenti

- I file system remoti aggiungono nuove modalità di fallimento a causa di possibili **malfunzionamenti** di **rete** o dei server
- Il client tipicamente termina le operazioni in corso sul file system remoto oppure le posticipa
- Il recupero da un malfunzionamento può coinvolgere informazione sullo stato di ogni richiesta remota
- Protocolli senza stato come NFS includono tutta l'informazione in ogni richiesta, permettendo un facile recupero, ma minore sicurezza