File System

Sommario

- Interfaccia del file system
- Concetto di File
- Metodi di accesso
- Struttura delle Directory
- Montaggio del File System
- Protezione

File system

- Per la maggior parte degli utenti il **file system** è l'aspetto più visibile del SO.
- Il file system rappresenta una astrazione del modo con cui i dati sono allocati e organizzati su un dispositivo memoria di massa.
- Attraverso il file system, Il SO offre una visione **logica** uniforme della memorizzazione delle informazioni sui diversi supporti (dischi, nastri, CD, ecc.).
- Elemento di base nella gestione a livello logico della memoria di massa è il **file**.

File

- Un **file** è un insieme di informazioni correlate e registrate nella memoria secondaria con un nome.
- Il file è la più piccola unità di memoria secondaria assegnabile all'utente che può scrivere sulla memoria secondaria solo registrando un file.
- Il file può avere una sua struttura interna, a seconda del formato o del tipo. Per esempio un file eseguibile è una sequenza di sezioni di codice che possono essere caricate ed eseguite.

Attributi

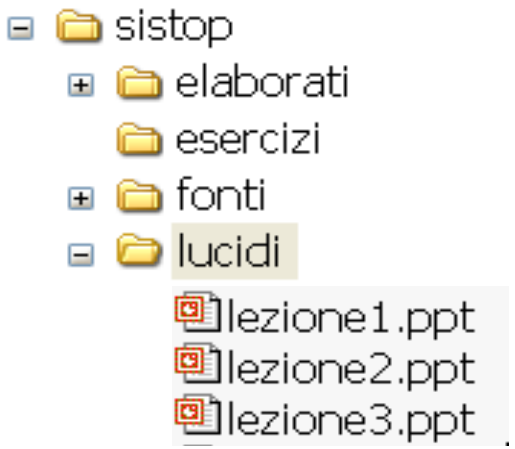
- Principali attributi del file sono:
  - **Nome:** è l'unica informazione mantenuta in un formato simbolico leggibile da una persona.
  - **Tipo**: usato nei sistemi che supportano più tipi diversi di file.
  - **Locazione**: puntatore al dispositivo e alla locazione di quel file sul dispositivo.
  - **Dimensione**.

Attributi

- Altri importanti attributi del file sono:
  - **Protezione**: informazioni di controllo che specificano chi può leggere, scrivere o eseguire il file.
  - **Ora, data e identificazione dell'utente**:informazioni usate per funzioni di protezione dei dati e di monitoring. Possono essere riferite a:
    - Creazione
    - Ultimo utilizzo
    - Ultima lettura

Directory

- Il file system può essere molto ampio (contenere molte migliaia di file) e tipicamente è organizzato in una struttura gerarchica che mantiene gli elementi terminali (i file) all'interno di contenitori detti **directory**.



Operazioni sui file

- La componente del SO che si occupa di gestire i file è detta **file system**
- Le operazioni messe a disposizione (mediante **system call**) dal file system per gestire i file, sono:
  - Create
  - Open
  - Close
  - Write
  - Read
  - Delete
  - Seek (Riposizionemento all'interno del file)
  - Truncate

Open e close

- **Open**: la maggior parte dei SO richiede che venga fatta una esplicita apertura del file prima delle operazioni di lettura e scrittura.
- **Close**: Analogamente viene richiesta una operazione di chiusura del file che indichi il termine del processo di lettura scrittura

Scrittura del file

- La **scrittura** su un file avviene per blocchi attraverso la chiamata ad una system call che specifica il nome del file e le informazioni che si vogliono inserire.
- Il SO:
  - Cerca il file nella directory
  - Mantiene un puntatore al punto in cui dovrà essere effettuata la prossima scrittura.

Lettura da file

- La **lettura** di un file avviene per blocchi attraverso la chiamata ad una system call che specifica il nome del file la locazione di memoria in cui deve essere trasferito il file.
- Il SO:
  - Cerca il file nella directory
  - Mantiene un puntatore al punto in cui dovrà essere effettuata la prossima lettura

Riposizionamento

- Poiché solitamente il file è acceduto in lettura o in scrittura in modo esclusivo, tipicamente viene mantenuto un puntatore alla **posizione corrente sul file**.
- Il **riposizionamento** è in sostanza la modifica del valore del puntatore alla posizione corrente ed è una operazione che non comporta I/O.

Cancellazione

- La **cancellazione** avviene nel modo seguente:
  - Si cerca il giusto file nella directory
  - Si rilascia tutto lo spazio occupato dal file che torna ad essere spazio libero utilizzabile per la memorizzazione di altri file.
  - Si rimuove il riferimento al file nella directory

Troncamento

- Il **troncamento** è una forma di cancellazione che elimina il contenuto del file pur mantenendone i riferimenti (e di conseguenza tutti gli attributi).
- Libera lo spazio su disco ma non viene eliminato il riferimento al file contenuto nella directory

Altre operazioni

- Esistono numerose altre operazioni (alcune delle quali implementate utilizzando le primitive già citate)
  - Accodamento
  - Rinomina
  - Copia
  - Sposta
  - …

File in uso

- Il sistema mantiene in memoria l'elenco dei file in uso in una apposita tabella detta **tabella dei file aperti**
- Quando viene iniziata una operazione sul file, il file viene inserito nella tabella in modo che ogni successiva operazione non comporti fasi di ricerca

Informazioni sui file aperti

- A ciascun file aperto sono associate diverse informazioni:
  - **Puntatore alla posizione corrente nel file**: per il supporto a read e write. Se più processi operano sul file esistono più puntatori alla posizione corrente.
  - **Contatore delle aperture**: se più processi possono aprire il file, il SO deve tenere conto delle aperture avvenute e quando sono tutte chiuse, rimuovere il file dalla tabella dei file aperti.
  - **Posizione su disco del file**: quando il file è aperto viene memorizzata la sua posizione sul dispositivo per evitare di doverlo cercare nuovamente ad ogni operazione.

Struttura dei file

- Tipicamente il SO non si occupa di gestire la **struttura interna** associata ai diversi tipi di file, ma lascia che siano le applicazioni a definirla e utilizzarla.
- Per il SO i file sono sostanzialmente insiemi ordinati di byte e le convenzioni che danno un significato specifico ai byte (cioè il formato dei file) sono gestiti a livello applicazione.
- I vantaggi nel delegare la gestione dei formati alle applicazioni sono diversi:
  - Dimensioni del codice del SO: per gestire tutti i formati il sistema dovrebbe essere enorme. Lasciando il controllo alle applicazioni si limita la dimensione del SO.
  - Flessibilità nella definizione di nuove strutture: quando occorre gestire nuovi formati non è necessario modificare il file system e il SO ma è sufficiente dotarsi delle applicazioni appropriate.
- Tutti i SO supportano almeno il formato dei file eseguibili, per poterli caricare in memoria ed eseguire.

Struttura interna dei file

- Cio' che l'HW vede di un file è una sequenza di accessi effettuati a dei **blocchi** (o **record fisici**) tutti di uguale dimensione.
- Dal punto di vista delle applicazioni, invece, il file è una sequenza di **record logici**, determinati dal formato del file, che hanno dimensioni variabili a seconda dell'esigenza dell'applicazione stessa.
- È improbabile che la dimensione del record logico corrisponda esattamente ad un record fisico e la localizzazione dei record logici all'interno del file può essere complessa.
- esempio per Unix
  - il file è un flusso di byte singoli e il record logico (che viene passato alle applicazioni attraverso system call) è costituito da un byte.
  - Lo scostamento del record logico è rappresentato dalla sua posizione nel file.
  - **Il SO impacchetta i record logici in blocchi fisici (se per esempio il blocco è di 512 byte, mette 512 record logici in ogni blocco).**

Struttura interna del file

- Si noti che :
  - lo spazio fisico è allocato per blocchi, è probabile che l'ultimo blocco di un file sia usato parzialmente, ovvero che una parte dell'ultimo blocco di un file vada sprecata.
  - Maggiore è la dimensione del blocco, maggiore è la frammentazione interna legata a questo fenomeno.

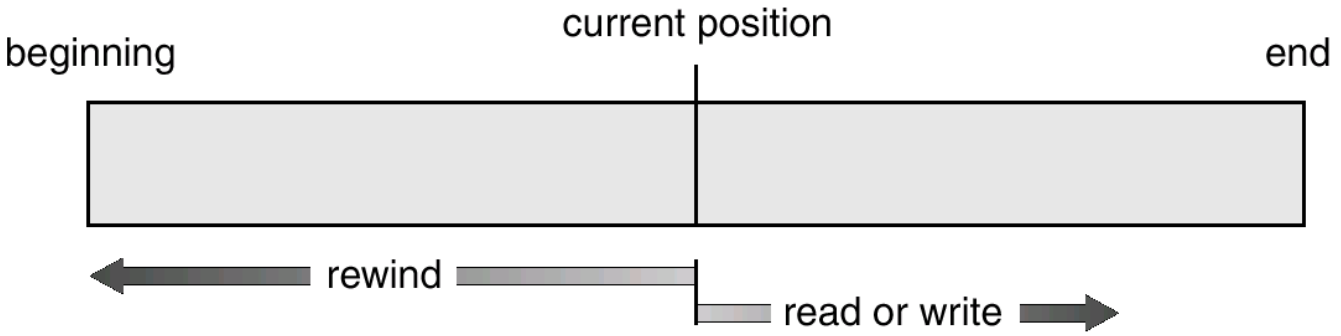
Metodi di accesso

- Il SO può fornire diversi metodi di accesso ai file:
  - **Sequenziale**: le informazioni del file vengono elaborate in ordine, un record dopo l'altro.
    - nastri
  - **Diretto**: il file è costituito da un insieme ordinato di blocchi a cui si accede direttamente.
    - dischi
  - **Attraverso indice**: al file vero e proprio è associato un file indice con lo scopo di velocizzare le ricerche.
    - Database

Accesso sequenziale

- Le informazioni del file vengono elaborate in ordine, un record dopo l'altro. E' il più utilizzato (ES: compilatori).
- Operazioni:
  - **Lettura**: **read**, legge la prossima porzione del file e avanza il puntatore di posizione.
  - **Scrittura**: **write**, scrive i nuovi dati in coda al file e avanza l'end of file.
  - **Reset**: riposiziona il puntatore di posizione a inizio file.

Accesso sequenziale



Accesso diretto

- Il modello i riferimento per l'accesso sequenziale sono i dispositivi come il nastro che consentono solo accessi in sequenza.
- I dispositivi ad accesso casuale (nel senso di non sequenziale) sono modello per un altro metodo di accesso al file: **l'accesso diretto** (o **accesso relativo**) in cui il file è costituito da record logici di lunghezza fissa e viene considerato come un insieme ordinato di blocchi (record) numerati che possono essere acceduti in modo arbitrario.
- Il **numero di blocco** è tipicamente assegnato in modo **relativo** (ciascun file inizia con il suo blocco 0).

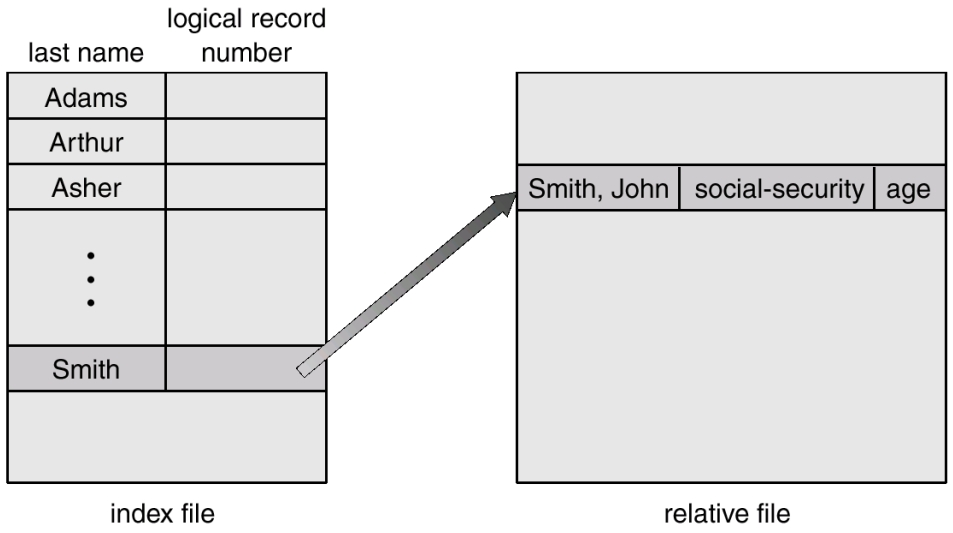
Accesso diretto

- Le istruzioni di lettura e scrittura devono quindi specificare su quale blocco devono essere eseguite:
  - **Lettura**: **read(n)**, legge il blocco con posizione relativa n.
  - **Scrittura: write(n)** scrive il blocco con posizione relativa n.
  - **Riposizionamento**: **seek(n)**, si posiziona sul blocco
- L'accesso sequenziale è facilmente realizzabile se si ha a disposizione l'accesso diretto (cp means current position):

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp*+1;
write next	*write cp*; *cp* = *cp*+1;

Indici

- Un altro metodo di accesso, che offre supporto alla ricerca veloce di occorrenze in un file, è quello che prevede la costruzione di un **indice**.
- L'indice contiene una o più chiavi di ricerca a cui sono associati i puntatori ai diversi blocchi del file.
- Il file contenente l'indice viene quindi associato ad un file di contenuti.

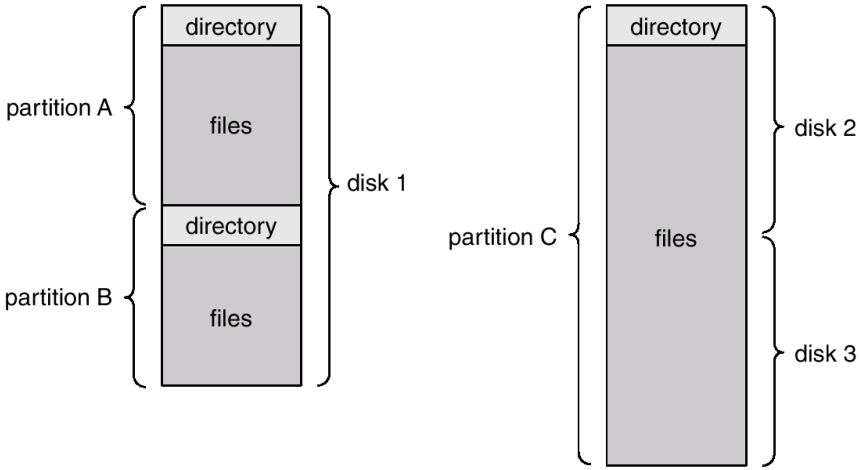


Partizioni

- Il File System può arrivare a gestire migliaia di file su centinaia di gigabyte di disco.
- I file vengono allora organizzati in **partizioni**, dischi virtuali che vengono trattati dal file system come un dispositivo.
- Si possono avere:
  - Due o più partizioni in un solo dispositivo reale
  - Due o più dispositivi reali in una sola partizione
- Ogni disco deve, ovviamente, contenere almeno una partizione.

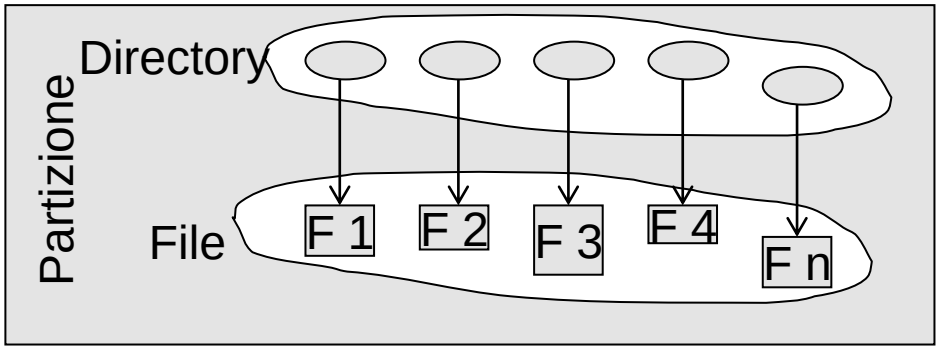
Partizioni

- Ciascuna partizione mantiene le informazioni relative ai file in essa contenuti.
- La struttura dati in cui sono memorizzate queste informazioni è detta **directory del dispositivo** (**device directory** o semplicemente **directory**) o anche **tabella delle partizioni** (**partition table**), e consente di associare ai nomi dei file i corrispondenti archivi.
- Usare, *con circospezione*, il comando **fdisk** per visualizzare il contenuto della tabella delle partizioni.



Directory

- La directory si può considerare come una tabella in cui sono memorizzate le informazioni relative al file e necessarie a:
  - Individuare la posizione del file
  - Gestirlo (copiarlo, cancellarlo, eseguirlo…)
- La directory si può organizzare in molti modi



Informazioni nella directory

- Informazioni tipicamente contenute nella device directory sono:
  - Nome del file
  - Tipo
  - Indirizzo
  - **Lunghezza attuale**
  - Lunghezza massima
  - Data ultimo accesso
  - Data ultimo aggiornamento
  - **Proprietario del file**
  - **Informazioni sulla protezione**

Operazioni sulla directory

- Il file system deve fornire le seguenti operazioni sulla directory:
  - **Ricerca** di un file
  - **Creazione** di una directory
  - **Cancellazione** di una directory
  - **List** della directory
  - **Attraversamento del file system** (visita a tutto il contenuto del file system, particolarmente utile per il back-up). Usato anche dal comando *find*.

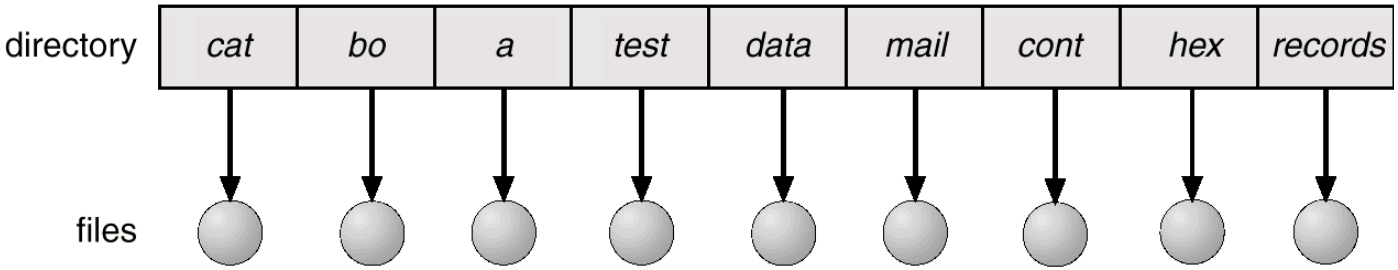
Directory

- **Obiettivi nella realizzazione della directory:**
  - **Efficienza,** nel localizzare un file velocemente.
  - **Gestione dei nomi (Naming)** conveniente per l'utente:
    - Due utenti devono poter dare lo stesso nome a due file diversi
    - Lo stesso file deve poter essere associato a più nomi
  - **Grouping:** i file devono poter essere raggruppati in base alle loro caratteristiche (tutti programmi java, tutti i file word,…) a seconda delle scelte dell'utente, che strutturera' le directory a seconda di come vuole raggruppare i propri files.

Single-Level Directory

- L'organizzazione della device directory più semplice è quella a **livello singolo** (**Single-Level Directory**).
- Tutti i file della partizione sono gestititi mediante una struttura lineare.
- E' particolarmente inefficace quando:
  - Ci sono più utenti (che devono usare nomi diversi per i file!).
  - Ci sono molti file (e anche un solo utente si confonde).

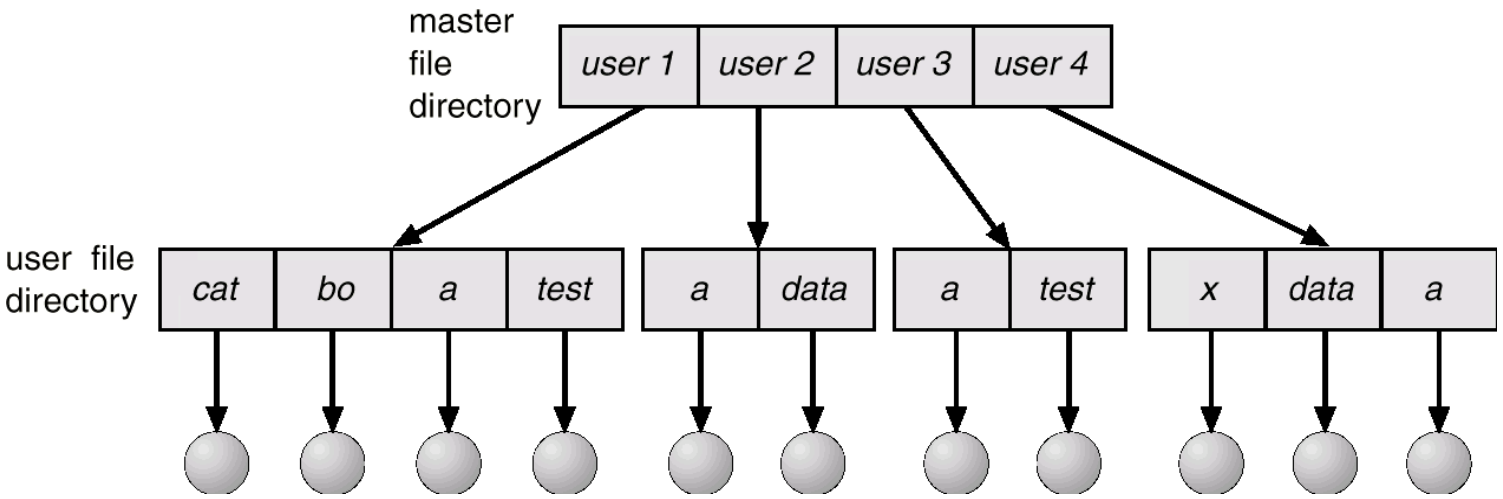
Single-Level Directory



Two-Level Directory

- Il principale problema nella device directory a livello singolo è la gestione dei nomi condivisa tra più utenti.
- Questo problema può essere risolto con una device directory a **doppio livello** (**Two-Level Directory**) in cui ciascun utente ha una sottodirectory separata (detta **UFD**, **user file directory**) e definisce quindi un suo spazio dei nomi.
- Un caso particolare di questo tipo di gestione riguarda gli eseguibili di sistema che l'utente vuole eseguire semplicemente digitando il nome del comando.
  - Il s.o. gestisce una UFD speciale per i comandi e ad ogni richiesta di accesso a un file controlla prima l'UFD dell'utente e poi l'UFD dei comandi (vi ricordate la PATH ?). Il s.o. implementa cioe' dei **percorsi di ricerca** (**search path**).

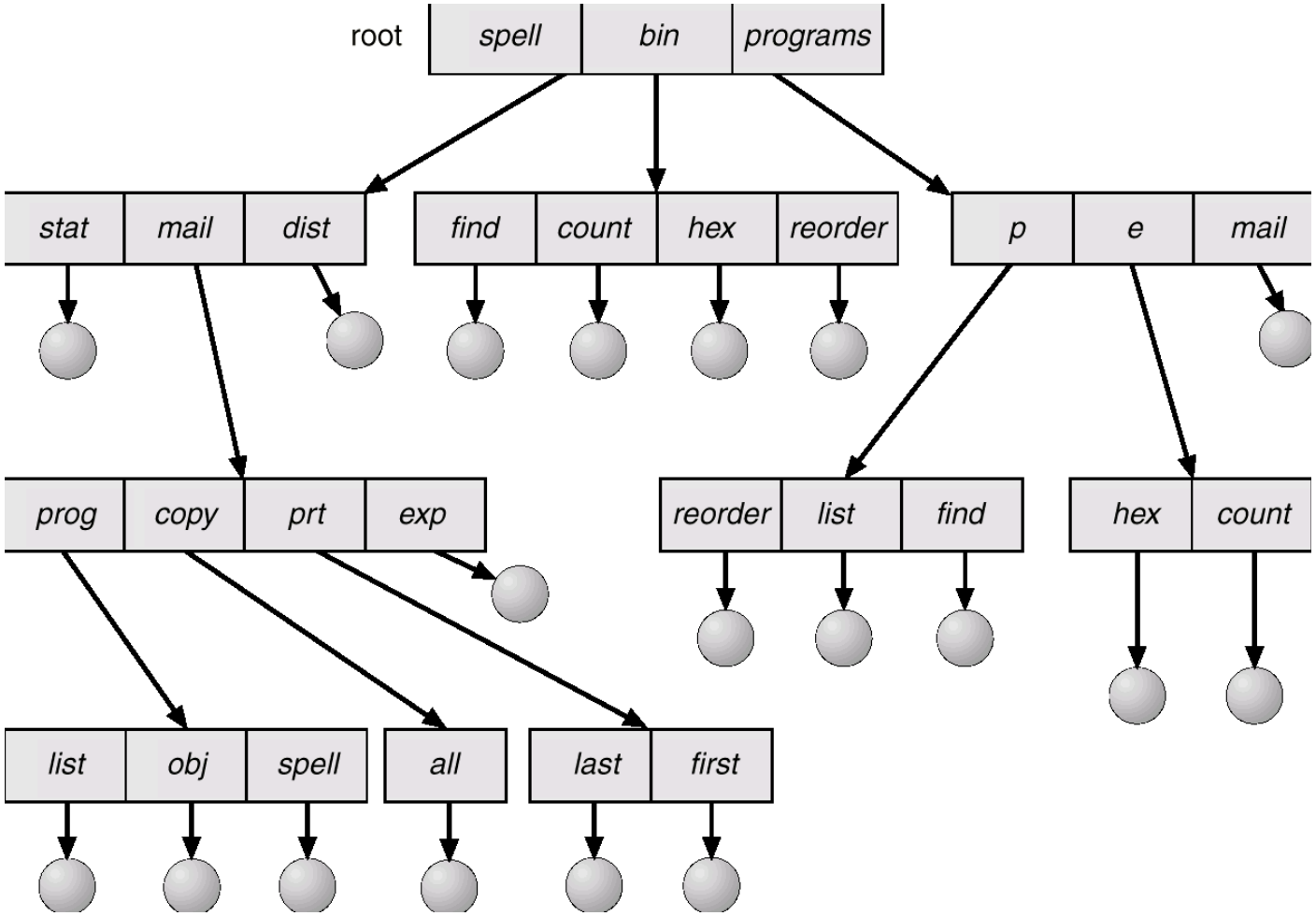
Two-Level Directory



Directory ad albero

- La directory a 2 livelli può essere considerata come un albero a due soli livelli.
- E' naturale estendere il concetto ad alberi in generale, definendo così le **directory ad albero** come strutture ad albero di altezza arbitraria.
- Questa generalizzazione consente agli utenti di creare le proprie sottodirectory e di organizzare a piacimento i file.

Directory ad albero



Directory ad albero

- Quindi:
  - L'albero ha una radice che è detta **root directory**.
  - Una directory (o una sottodirectory) possono contenere file o sottodirectory.
- Un **path name** univoco descrive il percorso che deve essere compiuto dalla root directory, attraverso le sottodirectory, per raggiungere il file.

Directory ad albero

- La directory è un tipo particolare di file
- Per distinguere tra sottodirectory e file viene utilizzato un **bit** per ogni elemento della directory che è settato a:
  - **1** per le sottodirectory
  - **0** per i file
- Il file system fornisce speciali system call per gestire (creare, cancellare, rinominare, spostare, ecc.) le directory.

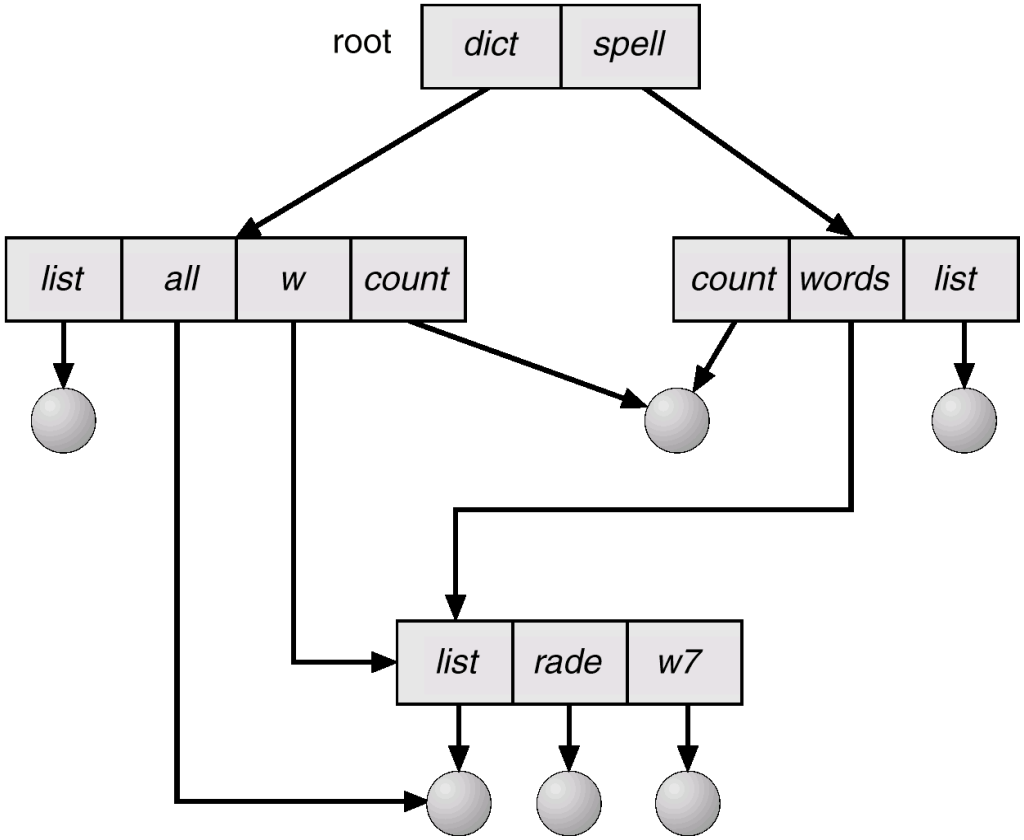
Directory ad albero

- Per evitare di dover usare sempre l'intero path name del file viene definita una directory corrente rispetto alla quale viene effettuata la ricerca dei file.
- La directory corrente può essere cambiata dell'utente (tipicamente con il comando **cd**, **change directory**).
- I **path name** possono essere:
  - **Assoluti**: partono dalla root
  - **Relativi**: partono dalla directory corrente

Directory a grafo aciclico

- Una ulteriore estensione alle directory ad albero prevede di utilizzare strutture a **grafo aciclico**.
- Una struttura di questo tipo è particolarmente adatta alla **condivisione di sottodirectory** tra più utenti: ogni utente ha il suo spazio di directory ma una parte comune è riferita da una certa directory di entrambi.

Directory a grafo aciclico



Directory a grafo aciclico

- Uno dei modi più semplici di implementare le strutture a grafo aciclico è quello di generare un nuovo elemento che è puntatore ad una directory esistente. Questo elemento è chiamato **link simbolico**.
- Un'altra alternativa è di creare due elementi (file o sottodirectory) con contenuto identico e mantenere allineate le due strutture. Questo approccio (**duplicazione**) soffre di diversi problemi legati alla gestione della coerenza

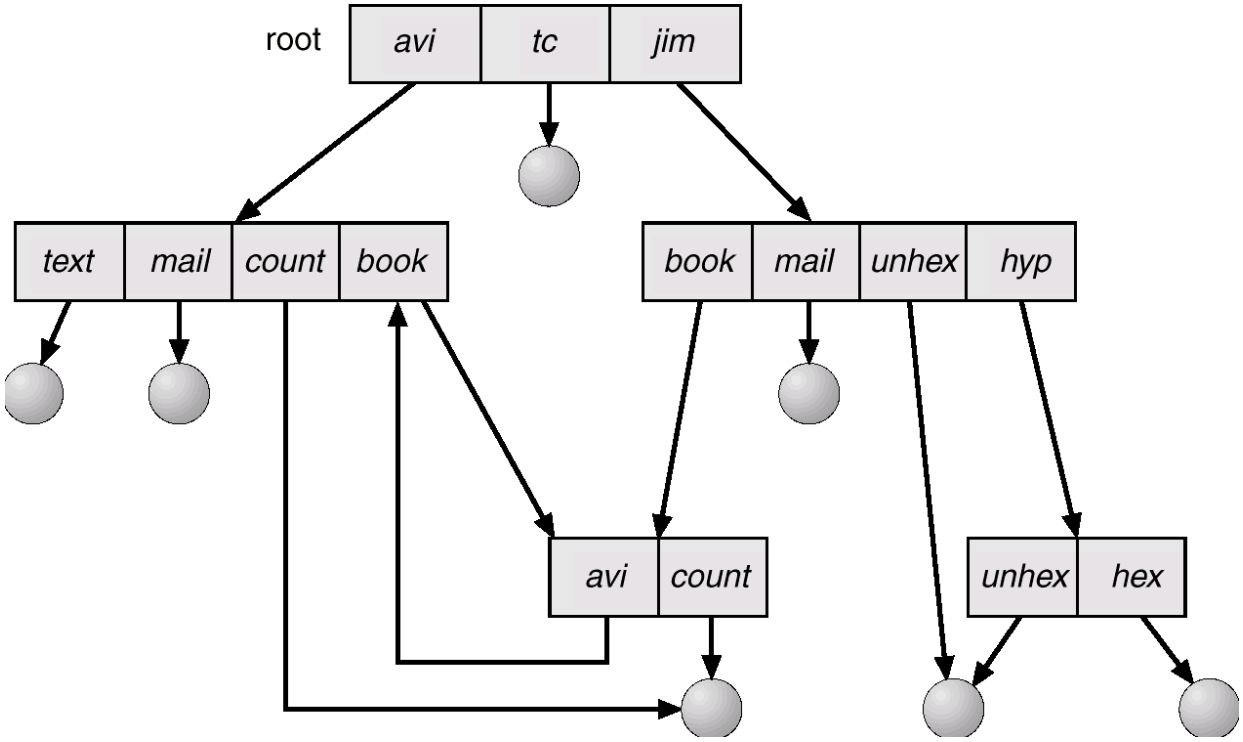
Directory a grafo aciclico

- Rispetto alle directory ad albero, le directory a grafo aciclico presentano diverse difficoltà implementative:
  - Attraversamento: ogni nodo va considerato una volta sola
  - Cancellazione:
    - **Aliasing**: due diversi nomi per lo stesso file
    - i link simbolici a quella locazione vanno eliminati
    - Basta una azione di cancellazione o ne serve una per ogni condivisione? Contatore delle copie.

Directory a grafo generale

- Se si consente ad un utente di definire un link simbolico ad una qualsiasi sottodirectory, nessuno vieta che il link risalga l'albero costruendo un grafo che contiene un ciclo (**grafo generale**).
- Questo tipo di struttura presenta problemi amplificati rispetto al grafo aciclico che abbiamo appena esaminato, dovuti soprattutto alla complessità degli algoritmi di attraversamento.

Directory a grafo



Directory a grafo generale

- Problematiche:
  - **Attraversamento**: il grafo può contenere grafi per cui le procedure di attraversamento devono tenere traccia dei nodi già visitati.
  - **Cancellazione**: il contatore delle copie è ingestibile in caso di autoreferenza, per cui si usa un **garbage collector** che visitando il grafo scopre se una directory è cancellata definitivamente.

Directory a grafo generale

- Poiché le politiche di gestione da adottare rischiano di essere costose alcuni sistemi utilizzano politiche di prevenzione ovvero ogni volta che viene aggiunto un collegamento verificano che non si sia creato un ciclo (**cicle detection**).
- Anche questo meccanismo è però piuttosto gravoso perché le directory sono su disco ovvero su una memoria relativamente lenta.

Linux fornisce 2 tipi di Link a file esistenti

- **Hard link**:
  - viene creata una nuova entry nella directory in cui si vuole l'hard link.
  - le informazioni relative al file vengono copiate dalla vecchia alla nuova directory
  - non è necessario una doppia ricerca nel file system
  - è impossibile distinguere la copia dall'originale.
  - Non può linkare directory (per prevenire la ricorsione nell'attraversamento)
  - Esiste un contatore delle copie del file.
  - Se si **rimuove** un hard link file ed il contatore diventa ZERO, si rimuove il file.
- **Soft Link** (o link simbolico):
  - viene creato un tipo speciale di directory entry, che contiene un riferimento (sotto forma di cammino assoluto) al file in questione
  - quando viene fatto un riferimento al file
    - si cerca nella directory
    - si scopre che si tratta di un link
    - viene *risolto il link (ovvero, viene utilizzato il cammino assoluto registrato nel file*
  - Può linkare directory
  - Se si **rimuove** un link simbolico, il file originale **non** viene rimosso.

Esempio di Hard Link in Linux

echo "File Vecchio" > Vecchio.txt

ls

```
drwxrwxr-x  2 vic vic 4,0K dic 16 22:39 .
drwxr-xr-x 23 vic vic 4,0K dic 16 22:39 ..
-rw-rw-r--  1 vic vic   13 dic 16 22:39 Vecchio.txt
```

Creo l'hard link

ln Vecchio.txt Nuovo

ls -alh

```
-rw-rw-r--  2 vic vic   13 dic 16 22:39 Nuovo
-rw-rw-r--  2 vic vic   13 dic 16 22:39 Vecchio.txt
```

Se si rimuove il file iniziale, l'hard link rimane e mantiene il file

rm   Vecchio.txt

ls -alh

```
-rw-rw-r--  1 vic vic   13 dic 16 22:39 Nuovo
```

cat Nuovo

File Vecchio

Esempio di Soft Link in Linux

echo "File Vecchio" > Vecchio.txt

ls

drwxrwxr-x 2 vic vic 4,0K dic 16 22:39 .

drwxr-xr-x 23 vic vic 4,0K dic 16 22:39 ..

-rw-rw-r-- 1 vic vic 13 dic 16 22:39 Vecchio.txt

Creo il link simbolico

ln -s Vecchio.txt Nuovo

ls -alh

lrwxrwxrwx 1 vic vic 11 dic 16 22:51 Nuovo **->** Vecchio.txt

-rw-rw-r-- 1 vic vic 13 dic 16 22:51 Vecchio.txt

Se si rimuove il file iniziale, il soft link perde il riferimento

rm Vecchio.txt

ls -alh

lrwxrwxrwx 1 vic vic 11 dic 16 22:51 Nuovo -> Vecchio.txt

cat Nuovo

cat: Nuovo: **No such file or directory**

STOP

Basta cosi'

Passiamo direttamente alla

Struttura del File System

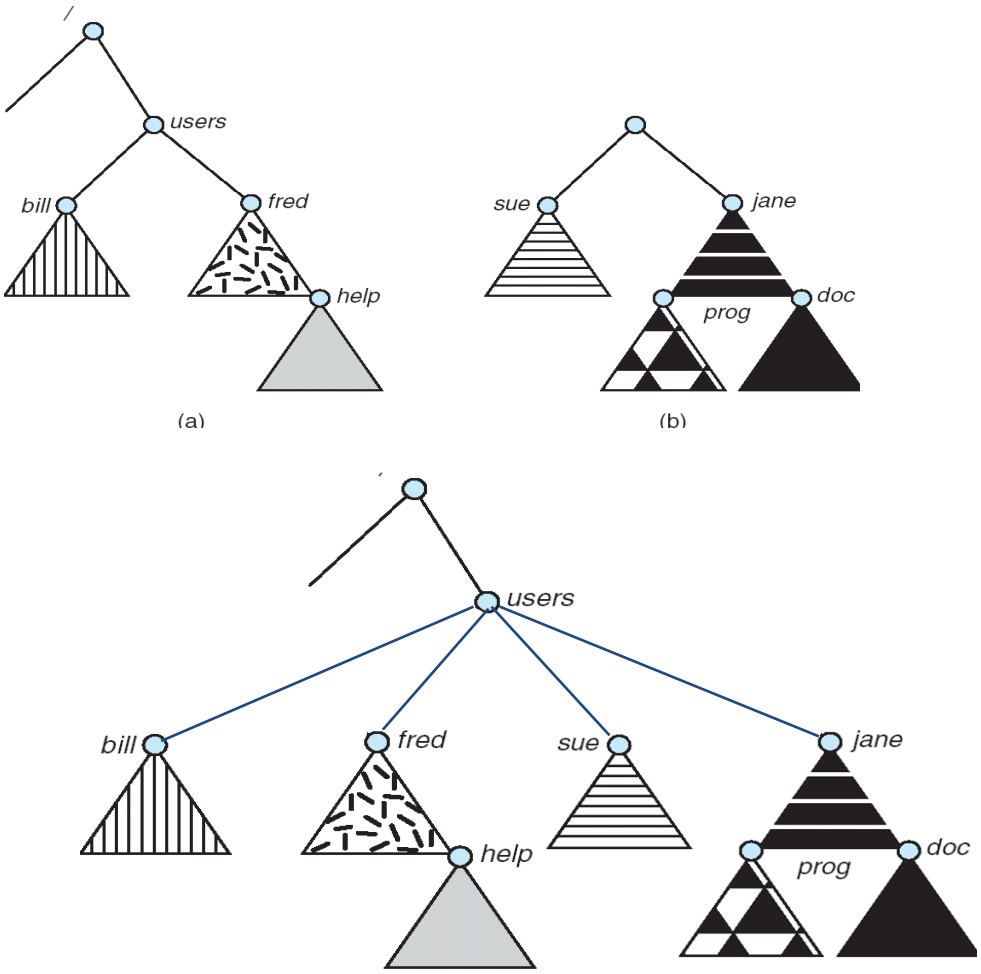
Sicurezza

- La **sicurezza** dei dati gestiti dal file system è determinata da:
  - **Affidabilità**: il file system non deve perdere dati, per cui sono previste funzioni di replicazione o di backup.
  - **Protezione**: il file system deve consentire l'accesso ai dati solo con autorizzazione limitandolo in base:
    - Ai tipi di accesso
    - All'utente che accede.

Mounting

- Con **montaggio** o **mounting** si indica un processo logico che ha lo scopo di preparare un file system (o una sua parte) e renderlo accessibile al sistema operativo.
- A seguito del mounting vengono generati gli indici e le strutture dati indispensabili al SO per utilizzare quella porzione di file system
- La struttura delle directory viene composta facendo il mounting di una serie di volumi.
- La procedura di mounting è molto semplice: si deve fornire al sistema operativo il nome del dispositivo e la sua localizzazione (detta punto di mounting).

Mounting della partizione



Condivisione di file

- La possibilità di condividere le risorse e in particolare i file è una delle caratteristiche fondamentali dei SO moderni
- Il concetto può essere esteso alla condivisione dell'intero file system come nei file system remoti
- La condivisione comporta ovviamente la presenza di meccaniche complesse tra le quali anche:
  - Quelle per l'identificazione dei file attraverso i loro nomi,
  - Quelle di protezione delle azioni di un utente da quelle degli altri

Utenti multipli

- Rispetto ai sistemi che gestiscono un solo utente, un SO che gestisce utenti multipli deve memorizzare più informazioni relative e file e directory che mirano a descrivere chi può fare cosa su quel file/directory.
- In questo contesto sono stati sviluppate diverse soluzioni che, in generale, sono basate sull'idea che sul file possano operare:
  - Un proprietario
  - Un gruppo di lavoro

Tipi di accesso

- Possono essere distinte molte tipologie di accesso:
  - In **lettura**
  - In **scrittura**
  - In **esecuzione**
  - In **aggiunta**
  - In **cancellazione**
  - Con sola **lista** degli attributi del file
- Di solito cancellazione e aggiunta sono trattate come casi di scrittura e l'elenco si riduce a:
  - **lettura (read)**
  - **scrittura (write)**
  - **Esecuzione (execute)**

Accesso

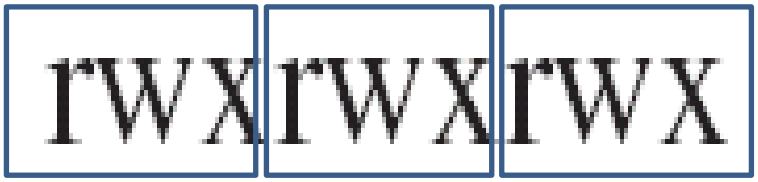
- L'approccio più comune alla protezione dei dati prevede criteri di accesso dipendenti dall'identità dell'utente.
- Lo schema più generale di implementazione consiste nell'associare una lista di utenti aventi diritto all'accesso a ciascun file (**lista d'accesso**).
- Questo meccanismo appesantisce la gestione delle directory e ad ogni file viene aggiunta l'informazione necessaria a memorizzare la lista di accesso

Linux

- Nei sistemi Unix like soluzione consiste dunque nel condensare gli elementi della lista descrivendo tre tipi di utente:
  - Il **proprietario** del file
  - Il **gruppo** di lavoro che condivide il file
  - L'**universo**, ovvero l'insieme di tutti gli utenti di sistema che non sono nel gruppo di lavoro.
- La lista di accesso specifica poi solo quali tipi di accesso (tra read, write e execute) può fare ogni tipologia di utenti
- Sono usati complessivamente 9 bit (3 per ognuno dei 3 tipi di utente)

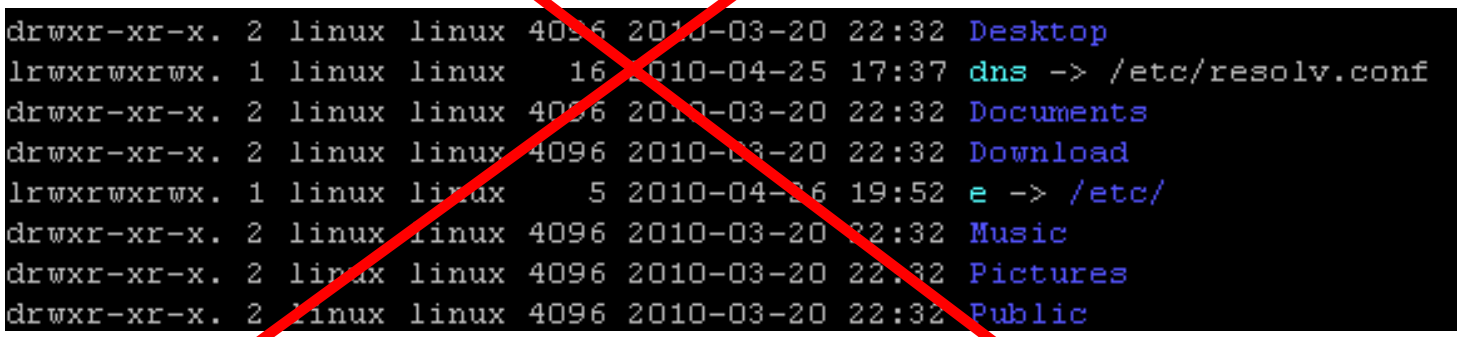
UNIX Directory Listing

-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/



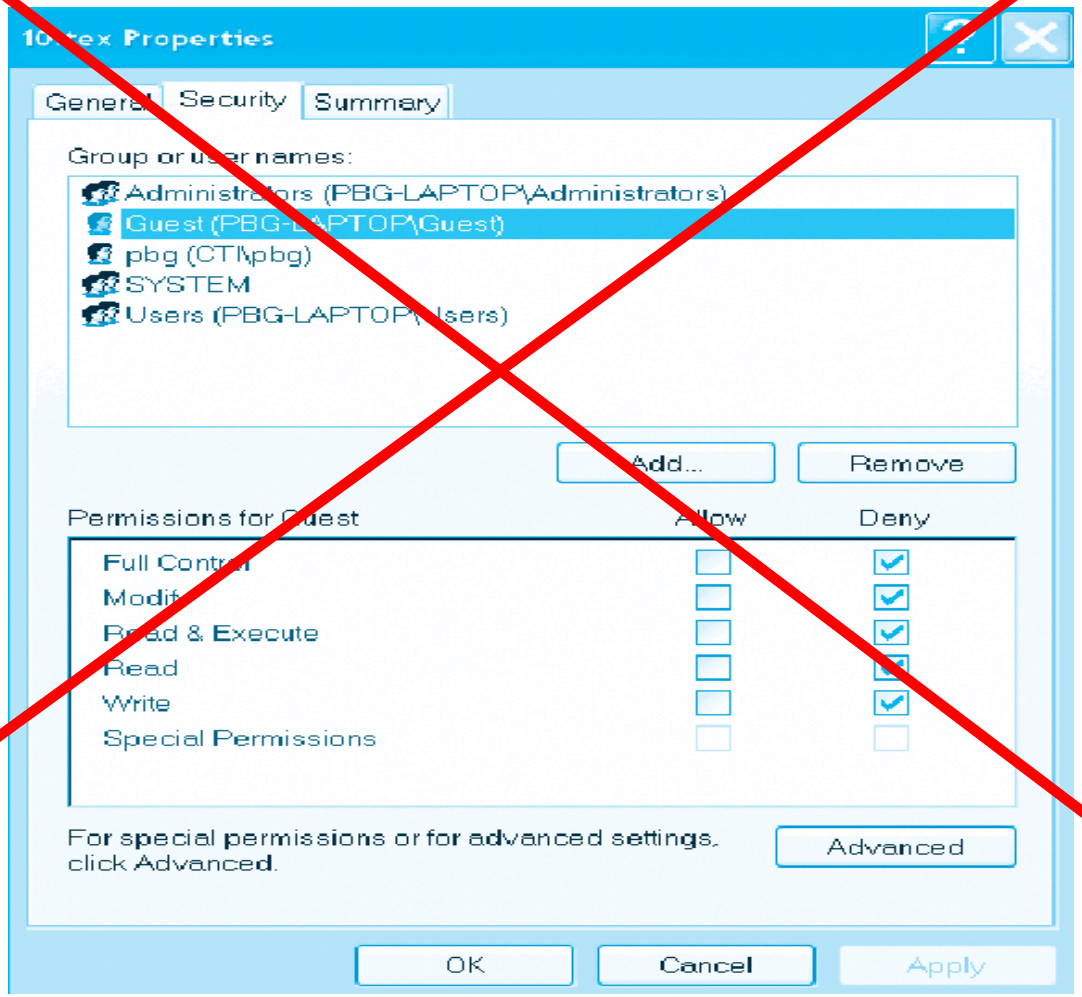
Unix

- Da questa schermata si nota che anche la differenza tra directory e file è realizzata tramite un bit.
- Lo stesso vale per la differenza tra file e link simbolico



- Vedremo assieme altri bit che vengono usati per dare indicazioni sui file e le loro caratteristiche (per esempio setuid)

Windows XP Access-control List Management



Accesso

- Esistono altri metodi per implementare la protezione ma non sono molto utilizzati.
- Per esempio si può fare protezione con password associando una **password** a ciascun file. Questa tecnica presenta numerosi problemi, a partire dal fatto che ogni utente deve ricordare le password di tutti i suoi file e che la condivisione può risultare complicata.
- Vedremo meglio la protezione in generale e come questa possa essere implementata nei SO in una prossima lezione su protezione e sicurezza.

Stop

Basta cosi'

Passiamo alla Struttura del File System

- In questa parte della lezione abbiamo solo introdotto gli aspetti logici del file system ovvero abbiamo descritto il file system come è visto dalle applicazioni (sys call)
- Ora dobbiamo procedere verso il basso per comprendere come queste funzioni sono implementate fino ad arrivare a fare considerazioni sugli aspetti fisici dei dispositivi di memorizzazione
- Nella prossima parte della lezione cominciamo a scendere nella struttura del file system