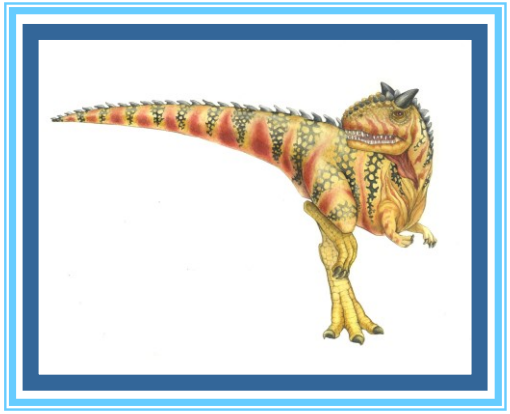
L’interfaccia del file system



Obiettivi

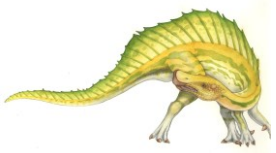
- Spiegare la funzione del file system
- Descrivere le interfacce dei file system
- Presentare i tradeoff di progettazione dei file system, in relazione a metodi di accesso, condivisione dei file, uso dei lock e strutture delle directory
- Analizzare le tecniche di protezione del file system

Sommario

- Concetto di file
- Modalità di accesso
- Struttura delle directory
- Protezione
- File mappati in memoria
- Montaggio di un file system
- Condivisione di file
- UNIX come caso di studio

Generalità – 1

- Durante la loro esecuzione, i processi devono…
  - memorizzare e trattare grandi quantità di informazione (molto maggiori della quantità di memoria principale)
  - avere la possibilità di accedere alle informazioni in modo concorrente e coerente, nello spazio e nel tempo
  - con garanzia di integrità, indipendenza, persistenza e protezione dei dati
- Tali garanzie vengono offerte dai file, che il sistema operativo organizza e gestisce nel file system



Generalità – 2

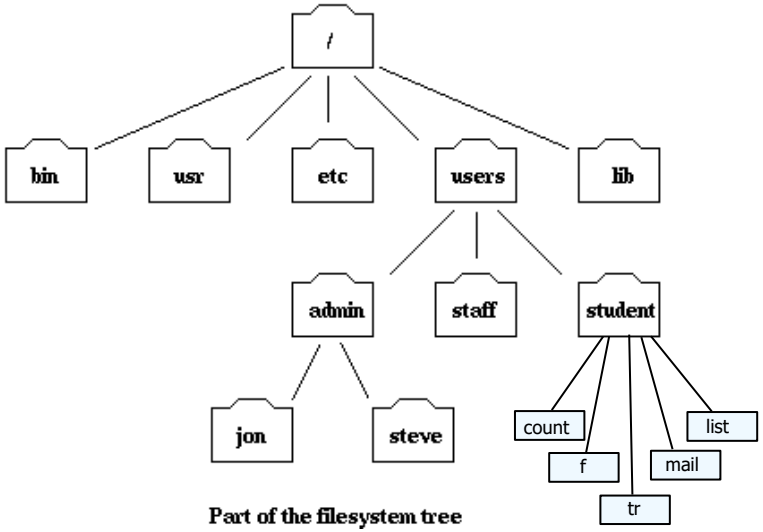
- Il sottosistema per la gestione dei file è la parte più visibile di un sistema operativo (*SO documento–centrici*)
  - Fornisce meccanismi per la registrazione, l'accesso e la protezione di dati e programmi del SO e degli utenti
  - Come i sottosistemi per la gestione dei processi e della memoria "virtualizzano" rispettivamente la CPU e la memoria centrale, così il file system "virtualizza" i dispositivi di memorizzazione permanente, fornendone una visione logica uniforme
- Due livelli:
  - Visione utente – interfaccia, ciò che offre la macchina virtualizzata dal SO
  - Implementazione – principi realizzativi architetturali

Generalità – 3

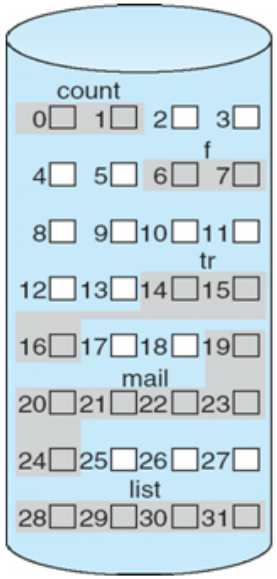
- Un sistema di calcolo può utilizzare diversi media per registrare stabilmente le informazioni
  - Disco rigido, unità a stato solido, memorie flash…
  - Ogni media ha caratteristiche fisiche diverse
- Compito del SO è quello di astrarre la complessità di utilizzo dei diversi mezzi di memorizzazione secondaria fornendone una visione logica e metodi di accesso uniformi
- Interfaccia di accesso ai file
  - Comune
  - Efficiente

File system – 1

- La struttura logica con cui il sistema operativo gestisce i file prende il nome di file system
- Tutti i moderni SO gestiscono l'archiviazione di file in modo gerarchico (*HFS – Hierarchical File System*)
- L'organizzazione gerarchica non ha alcun legame con la posizione fisica effettiva dei file in memoria di massa



Part of the filesystem tree



File system – 2

- Più in dettaglio: il file system è responsabile della gestione dei file contenuti nella memoria di massa
  - struttura i dati in file…
  - …che organizza in directory (o cartelle)
  - Fornisce all'utente un insieme di funzioni di alto livello per operare su di essi, mascherando le operazioni che vengono realmente effettuate per allocare la memoria e per accedervi in lettura/scrittura
- Il file system garantisce una gestione dei file indipendente dalle caratteristiche fisiche dei dispositivi che costituiscono la memoria di massa: astrazione utile sia per l'utente sia per i programmi

File system – 3

- L'utilizzo dei file system è stato esteso oltre i confini originali
- Per esempio, UNIX e Linux forniscono file system temporanei che utilizzano le strutture e l'interfaccia del file system per fornire accesso alle informazioni di sistema
  - Il file system **`/proc`** contiene i dettagli relativi a tutti i processi in esecuzione
  - All'interno di **`/proc`** si trovano anche numerose informazioni sull'hardware del sistema
  - Esempio: con **`cat /proc/cpuinfo`** si ottiene:

```
processor  : 0
vendor_id  : AuthenticAMD
cpu family : 5
model      : 9
model name : AMD-K6(tm) 3D+ Processor
Stepping   : 1
cpu MHz    : 400.919
cache size : 256 KB
...
```

File

- Spazio di indirizzi logici contigui; è un insieme di informazioni correlate e registrate nella memoria secondaria, a cui è stato assegnato un nome
- Dal punto di vista dell'utente:
  - è la più piccola porzione di memoria secondaria indirizzabile logicamente
  - i dati possono essere scritti nella memoria secondaria soltanto all'interno di un file
- Dal punto di vista del SO:
  - i file vengono mappati su dispositivi fisici di memorizzazione non volatili
- Tipi:
  - Dati
    - Numerici, alfanumerici, binari, multimediali
  - Programmi
    - Codice sorgente, oggetto, eseguibile

Attributi dei file – 1

- Nome: identificativo del file per l'utente (unico attributo "in chiaro")
- Identificativo: etichetta unica (numero progressivo) che identifica il file all'interno del file system
- Tipo: necessario per sistemi che supportano tipi diversi
- Locazione: puntatore al dispositivo ed alla posizione del file nel dispositivo
- Dimensione: dimensione attuale del file
- Protezione: parametri di controllo per l'accesso in lettura, scrittura ed esecuzione del file
- Ora, data e identificazione dell'utente: dati necessari alla sicurezza del sistema e per il controllo d'uso
- Agli attributi dei file, spesso, ci si riferisce con il termine di metadati

Attributi dei file – 2

- Alcuni file system più recenti supportano anche gli attributi estesi dei file, tra cui la codifica dei caratteri del file e funzioni di sicurezza come la checksum
- Le informazioni sui file sono conservate nella struttura di directory, che risiede sulla memoria secondaria
  - Un elemento di directory consiste di un nome di file e di un identificatore unico, che a sua volta individua gli altri attributi
  - Un elemento di directory può avere una dimensione > 1KB
  - Dimensione della directory dell'ordine di MB o GB

La finestra di informazioni di un file su MAC OS

Operazioni sui file – 1

- Un file è un *tipo di dato astratto* su cui sono definite le operazioni di:
  - Creazione
  - Scrittura
  - Lettura
  - Posizionamento nel file – *file seek*
  - Cancellazione
  - Troncamento
  - Lettura/Impostazione degli attributi

Operazioni sui file – 2

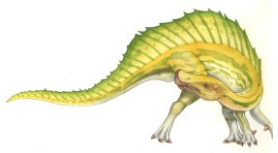
- Creazione
  - Reperire lo spazio per memorizzare il file all'interno del file system
  - Creare un nuovo elemento nella directory in cui registrare nome del file, posizione nel file system, altre informazioni
- Scrittura
  - Chiamata al sistema con nome del file e (puntatore ai) dati da scrivere come parametri
  - Reperimento del file nel file system
  - Scrittura dei dati nella posizione indicata dal puntatore di scrittura e aggiornamento del puntatore
- Lettura
  - Chiamata al sistema con nome del file e indirizzo di memoria dove trascrivere i dati letti come parametri
  - Reperimento del file nel file system
  - Lettura dei dati nella posizione indicata dal puntatore di lettura e aggiornamento del puntatore

Operazioni sui file – 3

- Di solito si mantiene un solo puntatore alla posizione corrente nel file, che serve sia per effettuare operazioni di lettura che di scrittura
- Posizionamento nel file
  - Reperimento del file nel file system
  - Aggiornamento del puntatore alla posizione corrente
  - Nessuna operazione di I/O
- Cancellazione
  - Reperimento del file nel file system
  - Si rilascia lo spazio allocato al file e si elimina il corrispondente elemento della directory
- Troncamento
  - Cancellazione del contenuto del file, che mantiene immutati tutti gli attributi (esclusa la dimensione)
  - Si rilascia lo spazio allocato al file
- Lettura/Impostazione degli attributi
  - Reperimento/aggiornamento del relativo elemento di directory

Operazioni sui file – 4

- Altre operazioni sui file si ottengono mediante opportune combinazioni delle operazioni di base
- Esempio: operazione di copia
  - Creazione di un nuovo file
  - Lettura dal file da copiare
  - Scrittura nel nuovo file
- Quasi tutte le operazioni su file richiedono una ricerca dell'elemento associato al file all'interno della struttura delle directory (e, eventualmente, un aggiornamento dell'elemento stesso)
  - ⇨Occorre "aprire" il file prima di qualsiasi accesso
- Il SO mantiene in memoria centrale una tabella contenente informazioni su tutti i file aperti: la tabella dei file aperti



Operazioni sui file – 5

- Quando si richiede un'operazione su file, si ricercano le informazioni relative reperendole, tramite un indice, nella tabella dei file aperti
- Inoltre, quando il file non è più in uso attivo, deve essere chiuso ed il SO rimuove l'elemento relativo nella tabella dei file aperti
- Le system call per aprire e chiudere i file sono:
  - **`open(`F_i`)`** – ricerca nella struttura di directory sul disco l'elemento F_i, e ne copia il contenuto nella tabella dei file aperti (in memoria centrale); riporta un puntatore all'elemento nella tabella
  - **`close(`F_i`)`** – copia il contenuto dell'elemento F_i, attualmente residente in memoria principale, nella struttura di directory sul disco e lo rimuove
- Le system call che "lavorano" su file chiusi sono esclusivamente **`create()`** e **`delete()`**

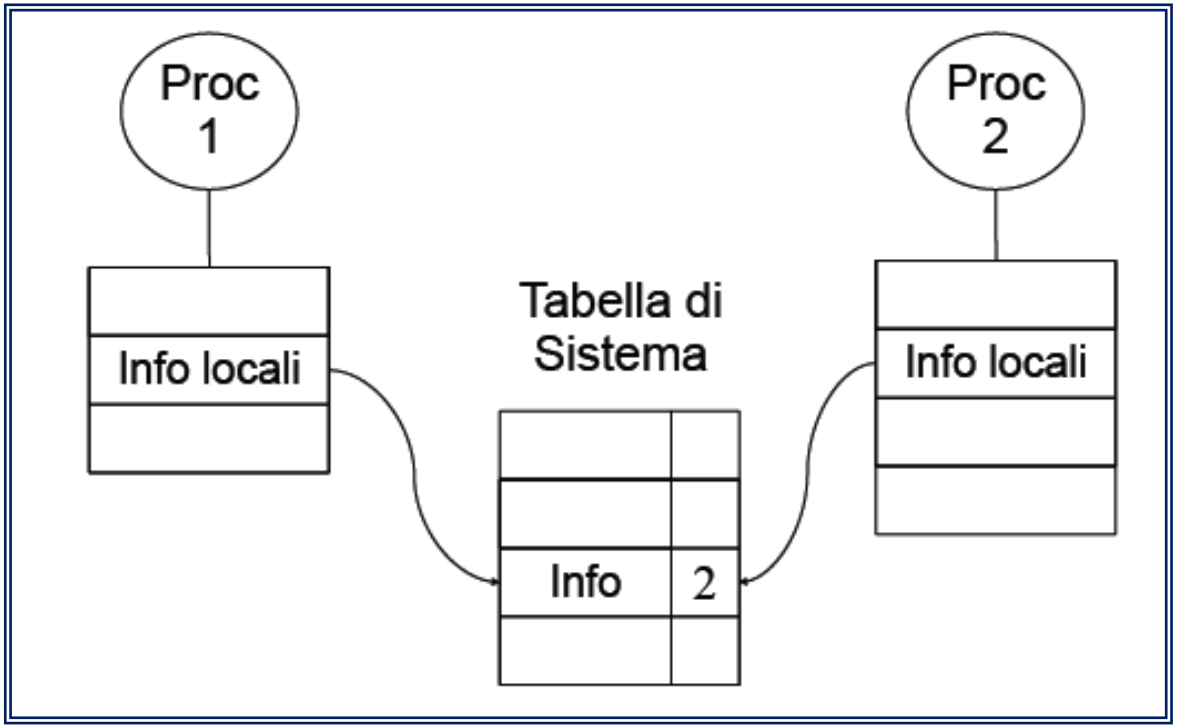
File aperti – 1

- Nei sistemi multiprocessing, due livelli di tabelle:
  - Tabella di sistema: riferimenti a tutti i file aperti nel sistema
    - Posizione del file nel dispositivo di memoria secondaria
    - Dimensione del file
    - Date di ultimo accesso/ultima modifica
    - Contatore di aperture
  - Tabella associata al processo: riferimenti a tutti i file aperti dal processo
    - Puntatore alla posizione corrente nel file
    - Diritti di accesso
    - Puntatore al relativo elemento contenuto nella tabella di sistema

File aperti – 2

- In particolare…
  - Contatore di aperture – conta il numero di processi che hanno aperto il file, per rimuovere opportunamente i dati dalla tabella dei file aperti alla chiusura del file da parte dell'ultimo processo (contenuto nella tabella di sistema)
  - Locazione del file su disco – cache delle informazioni di accesso ai dati permanenti (contenuto nella tabella di sistema)
  - Puntatore alla posizione corrente nel file – puntatore all'ultima locazione dove è stata realizzata un'operazione di lettura/scrittura per ogni processo che ha aperto il file (contenuto nella tabella dei file aperti associata al processo)
  - Diritti di accesso – controllati dal SO per permettere o negare le operazioni di I/O richieste (contenuti nella tabella dei file aperti associata al processo)

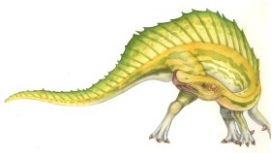
File aperti – 3



Una entry della tabella di sistema può essere rimossa quando il contatore vale 0

Lock sui file aperti

- Garantito da alcuni sistemi operativi e realizzazioni del file system
- Offre una mediazione per l'accesso condiviso a file
- Può essere shared (più processi possono acquisirlo in contemporanea) o exclusive
- Lock esclusivo:
  - Obbligatorio – l'accesso a file viene negato se il lock è già stato acquisito da altro processo (Windows)
  - Consigliato – i processi trovano che lo stato di un dato file è "bloccato" e decidono sul da farsi (UNIX)
- Se il lock è obbligatorio, il SO assicura l'integrità dei dati soggetti a lock; se il lock è consigliato, è compito del programmatore garantire la corretta acquisizione e cessione del lock



Nomi dei file

- Il nome viene associato al file dall'utente che lo crea ed è solitamente necessario (ma non sufficiente) per accedere ai dati del file
- Dovrebbe essere "descrittivo" del contenuto
- Le regole per denominare i file sono fissate dal file system, e sono molto variabili
  - Lunghezza (numero di caratteri): fino a 8 (DOS), 255 (UNIX)
  - Tipo di caratteri: alfanumerici o speciali
  - Case sensitive, insensitive
  - Inclusione di metadati

Tipi di file

- Il tipo è un attributo di un file che ne indica la struttura logica interna e permette di interpretarne correttamente il contenuto
- Alcuni SO "gestiscono" diversi tipi di file
  - Conoscendo il tipo del file, il SO può evitare alcuni errori comuni, ad esempio, stampare un file eseguibile
- Ovvero, un SO che riconosce il tipo di un file può manipolare il file in modo "ragionevole"
- Esistono tre tecniche principali per identificare i tipi di file
  - Meccanismo delle estensioni: il tipo è indicato da un suffisso del nome (DOS/Windows)
  - Attributo "tipo" associato al file nella directory (MAC OS)
  - Magic number/magic cookie: il tipo è rappresentato da un valore posto all'inizio del file (UNIX)

Tipi di file – nome, estensione

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, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

ESEMPI

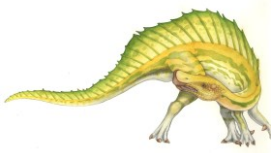
- Possono essere eseguiti file con estensione .com, .exe, .sh
- Nei sistemi Apple e nelle nuove versioni di Windows e Linux ciascun file possiede un attributo di creazione contenente il nome del programma che lo ha creato
- UNIX memorizza un *magic number* per indicare il tipo di file binario e un *magic cookie* per i file di testo (solo alcuni, in entrambi i casi); usa le estensioni solo come suggerimento, non vengono imposte né dipendono dal SO

Struttura dei file

- Nessuna — sequenza di parole o byte
- Struttura a record semplice
  - Linee
  - Record a lunghezza fissa
  - Record a lunghezza variabile
- Struttura complessa
  - Documento formattato
  - File eseguibile rilocabile
- Chi decide:
  - Il sistema operativo
  - L'applicativo che crea il file
- In genere, un file è formato da una sequenza di bit, byte, righe o record, il cui significato è definito dal creatore e dall'utente del file stesso

Supporto alla struttura dei file

- Il tipo di un file e la corrispondente struttura logica possono essere riconosciuti e gestiti in modi diversi nei diversi SO
- Se il SO gestisce molti formati
  - Codice di sistema più ingombrante
  - Incompatibilità di programmi con file di formato differente
  - Gestione efficiente per i formati supportati
- Viceversa...
  - Codice di sistema più snello
  - Formati gestiti dal programmatore
- Esempio: UNIX e Windows attuano una scelta minimale
  - i file sono considerati semplici stringhe di byte
  - solo i file eseguibili hanno un formato predefinito dal SO



Struttura interna dei file

- La dimensione dei blocchi del dispositivo di memoria, detti anche record fisici, è fissata
- Occorre risolvere il problema della corrispondenza fra record logici e record fisici (*packing*)

  - Parte dell'ultimo blocco fisico contenente il file rimane inutilizzata: frammentazione interna
  - Esempio: UNIX definisce tutti i file come un flusso di byte
    - Il record logico è un byte
    - A ciascun byte si può accedere tramite il suo offset (a partire dall'inizio o dalla fine del file)
    - Il SO impacca automaticamente i byte in blocchi fisici della dimensione opportuna

Modalità di accesso a file – 1

read next
write next
reset
no read after last write

- Accesso sequenziale: È un modello di accesso a file che si "ispira" al nastro; utilizzato da compilatori e editor
  - Impossibilità di lettura oltre l'ultima posizione scritta; la scrittura aggiunge informazioni in fondo al file

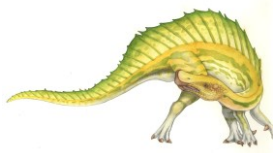
read n
write n
position to n
  read next
  write next
rewrite n

- Accesso diretto: È un modello di accesso a file che si "ispira" al disco

n = numero di record relativo

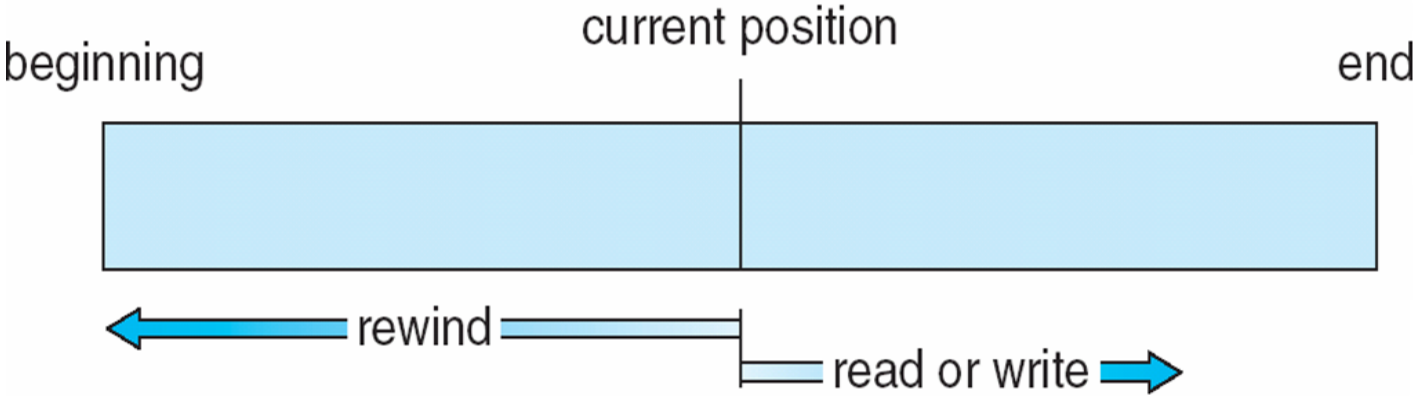
Modalità di accesso a file – 2

- Più in dettaglio…
  - I file ad accesso sequenziale:
    - Possono essere costituiti da record a lunghezza variabile
    - Un puntatore indirizza il record corrente e avanza a ogni lettura o scrittura
    - La lettura può avvenire in qualunque posizione del file, che deve però essere raggiunta sequenzialmente
    - La scrittura può avvenire solo in coda al file (append)
  - I file ad accesso diretto:
    - Hanno record di lunghezza fissa L, decisa dal creatore del file
    - Si opera su record di dati posti in posizione arbitraria nel file: per accedere all'n–esimo record, si calcola $L \times n$ che fornisce la posizione del byte d'inizio record



Accesso sequenziale

File ad accesso sequenziale

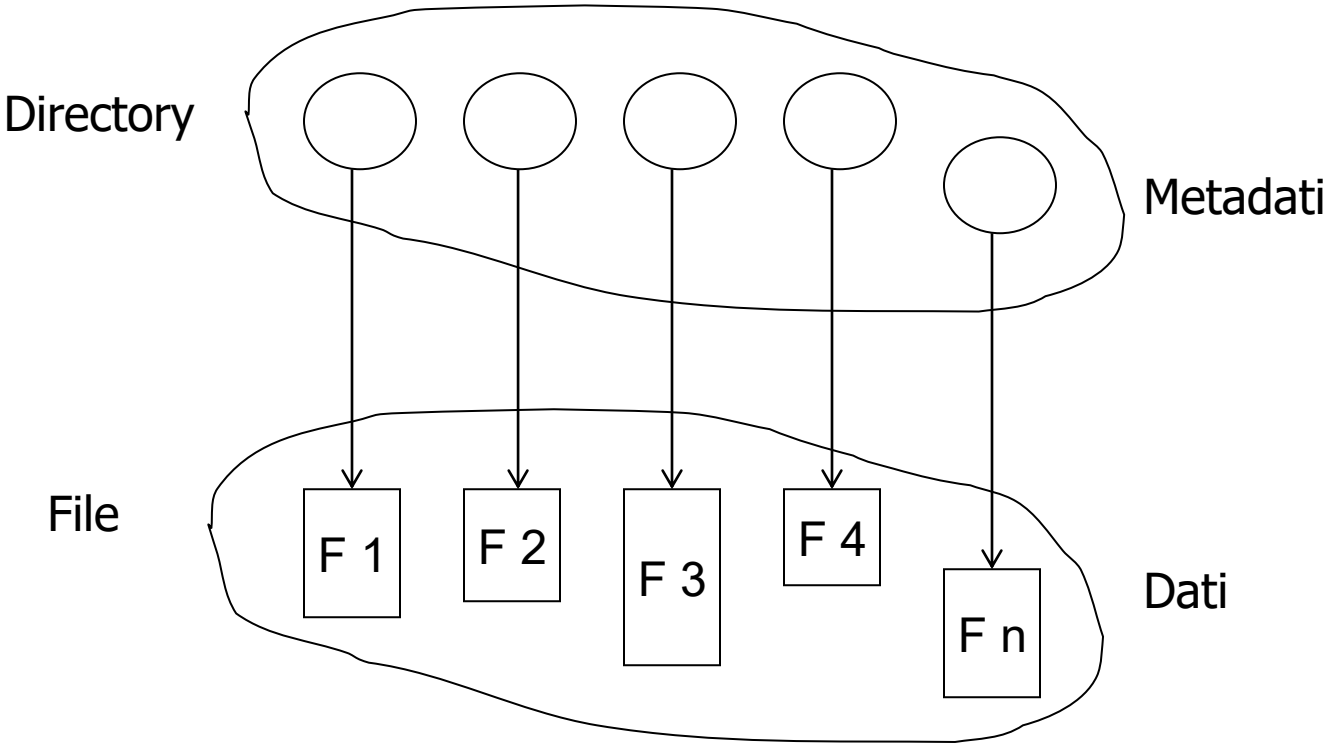


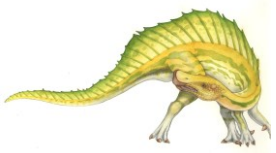
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Simulazione di accesso sequenziale su file ad accesso diretto

Struttura delle directory

- Sia la struttura di directory che i file risiedono su memoria di massa
- La directory può essere considerata come una tabella di simboli che traduce i nomi dei file nelle modalità di accesso alle informazioni in essi contenute





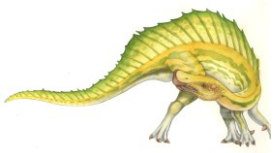
Informazioni contenute nelle directory
Operazioni sulle directory

- Informazioni sui file
  - Nome
  - Tipo
  - Indirizzo
  - Lunghezza attuale
  - Lunghezza massima
  - Data ultimo accesso
  - Data ultima modifica
  - ID del proprietario/gruppo
  - Informazioni di protezione
- Operazioni
  - Ricerca di un file
  - Creazione di un file
  - Cancellazione di un file
  - Elenco dei contenuti
  - Ridenominazione di un file
  - Attraversamento del file system

```
$ ls -lsF prova.exe
1 -rw-r--r-- 1 monica staff 213 May 20 00:12 prova.exe*
```

Operazioni sulle directory – 1

- Ricerca — Possibilità di scorrere la directory per reperire l'elemento associato ad un particolare file
  - I file hanno nomi simbolici
  - Nomi simili possono corrispondere a relazioni logiche fra i contenuti dei file
  - Capacità di reperire tutti i file il cui nome soddisfi una particolare espressione
- Creazione di un file — Aggiunta del record descrittivo del file alla directory
- Cancellazione di un file — Rimozione del record descrittivo del file dalla directory



Operazioni sulle directory – 2

- Elenco dei contenuti di una directory — Possibilità di elencare tutti i file di una directory ed il contenuto degli elementi della directory associati ai file

```
$ ls -l pr*.*
```

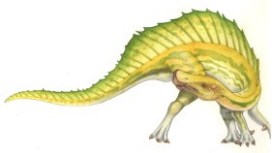
- Ridenominazione di file — Possibilità di modificare il nome di un file – che dovrebbe essere significativo del contenuto – a fronte di cambiamenti del contenuto stesso o di uso del file
  - Può provocare la variazione della posizione del file nella directory
- Attraversamento del file system — Possibilità di accedere ad ogni directory e ad ogni file in essa contenuto, visitandone l'intero "organigramma"

Come organizzare una directory?

- L'organizzazione della struttura di directory deve garantire:
  - Efficienza — capacità di reperire file rapidamente
  - Nominazione — conveniente per gli utenti
    - Due utenti possono utilizzare nomi uguali per file diversi
    - Lo stesso file può avere diversi nomi (ed essere raggiunto tramite percorsi diversi)
  - Grouping — Raggruppamento dei file sulla base di proprietà logiche (ad esempio, tutti i programmi Python, tutti i giochi, etc.)

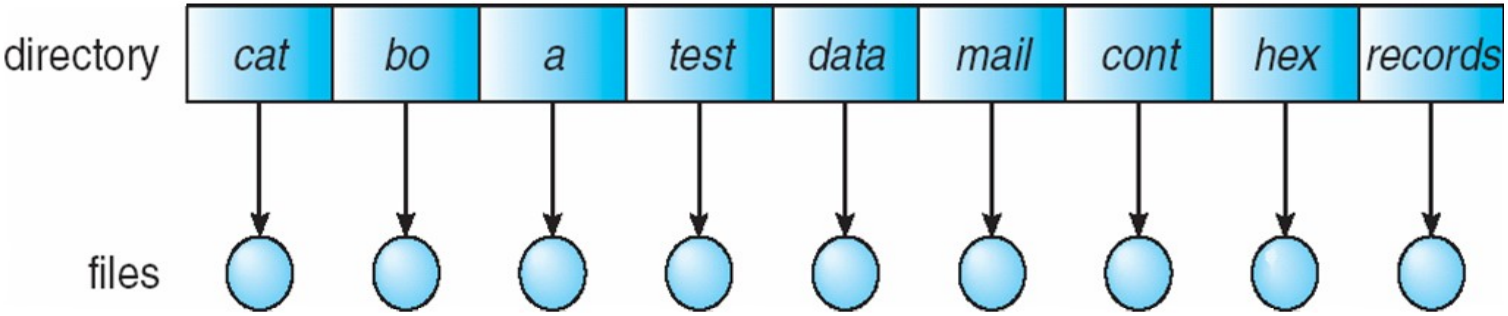
Organizzazione logica delle directory

- È necessario distinguere tra struttura logica e fisica delle informazioni memorizzate:
  - la struttura fisica descrive il modo in cui le informazioni sono scritte fisicamente sulla memoria di massa, così che il sistema possa ritrovarle per poterle leggere e/o modificare
  - la struttura logica è il modo in cui l'organizzazione delle informazioni è presentata all'utente
- Il numero di file memorizzati su un dispositivo di memoria di massa può essere estremamente elevato
  - Necessità di mantenere i file in una forma ordinata
- I SO attuali gestiscono i file in modo gerarchico (alberi, grafi)

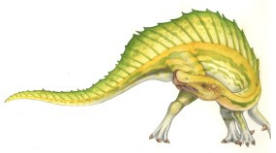


Directory monolivello

- Una directory unica per tutti gli utenti

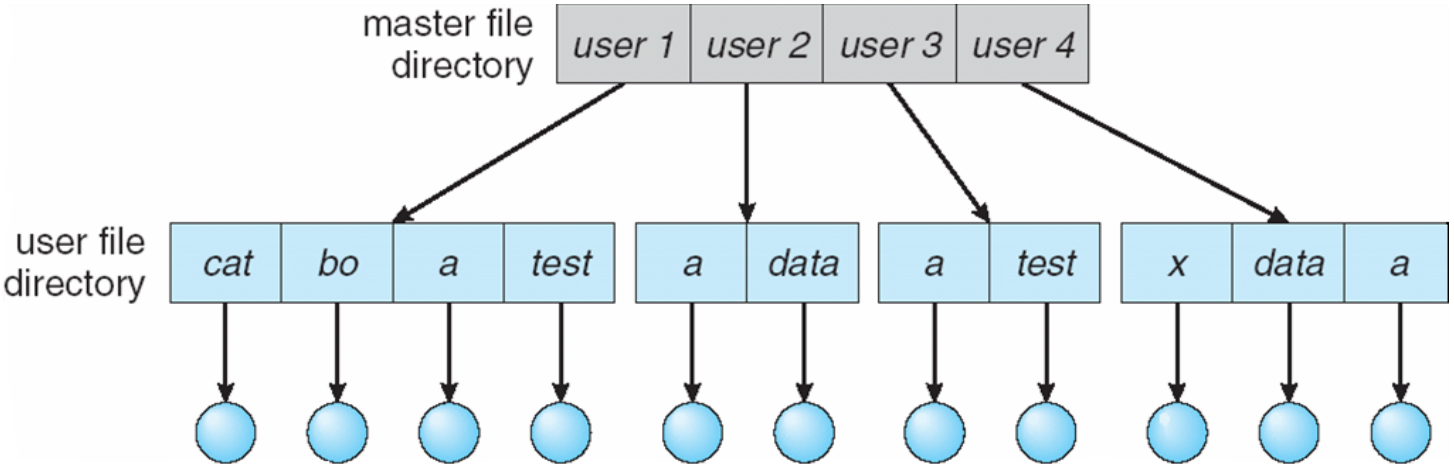


  - Problemi di nominazione: occorre scegliere un nome diverso per ogni file
  - Nessun raggruppamento logico



Directory a due livelli – 1

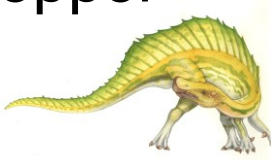
- Directory separate per ciascun utente



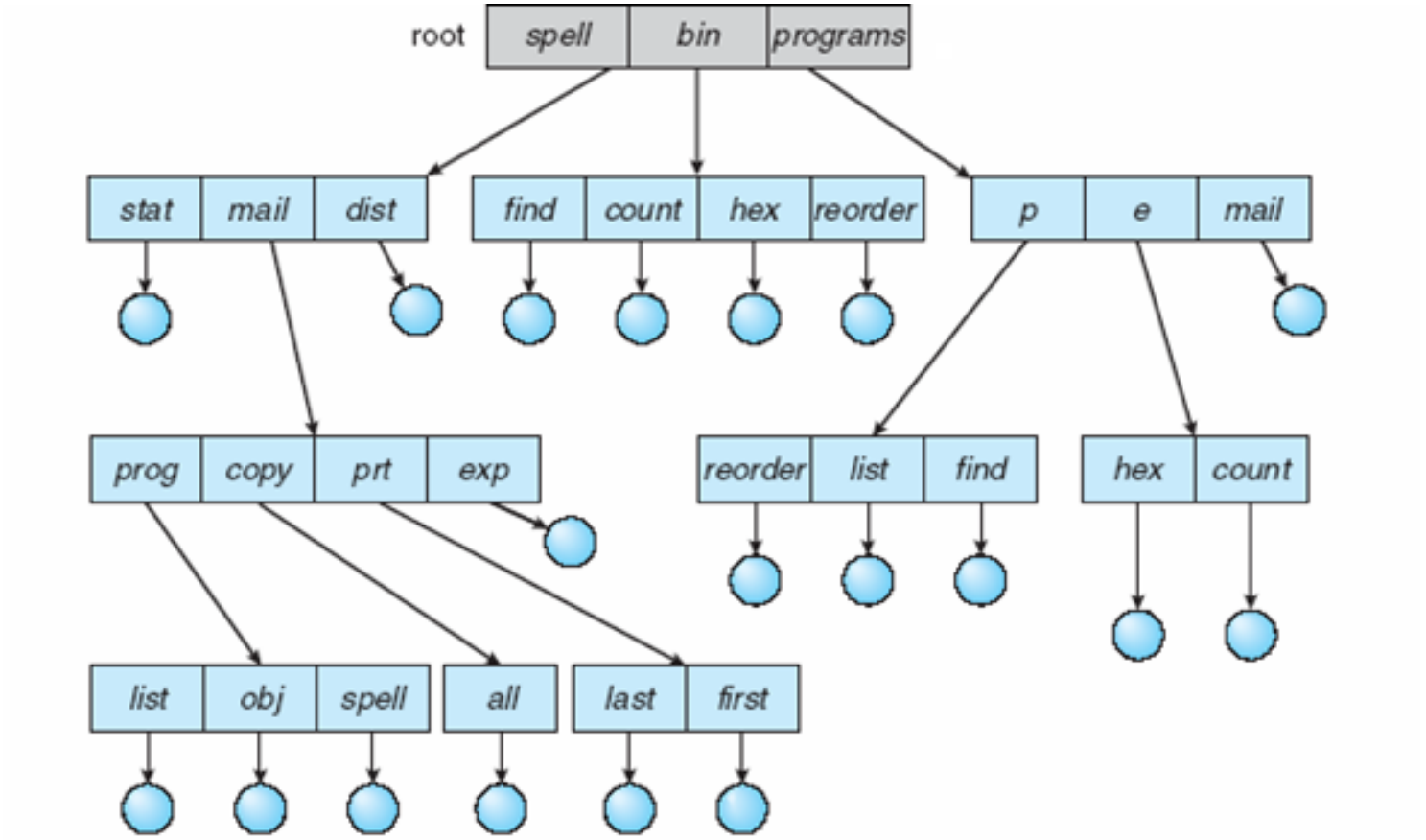
  - Ammessi nomi uguali per file di utenti diversi
  - Ricerca efficiente
  - Nessuna capacità di raggruppamento logico (se non in base ai proprietari)
  - Nome di percorso (*path name*) — permette ad un utente di raggiungere i file degli altri utenti
  - Ricerca dei file di sistema: percorso di ricerca

Directory a due livelli – 2

- Quando si apre la sessione di lavoro si ricerca nella MFD (Master File Directory) l'identificativo dell'utente, che viene ammesso al sistema all'interno della propria UFD (User File Directory)
- Ogni riferimento a file da parte dell'utente viene interpretato dal SO come esclusivamente correlato ai file presenti nella relativa UFD
- Per riferirsi a file di altri utenti, se l'accesso è autorizzato, ogni utente deve utilizzare il pathname completo del file (nome utente, nome file)
- I file di sistema vengono raccolti in opportune directory raggiungibili da tutti gli utenti, seguendo opportuni *percorsi di ricerca* personalizzabili



Directory con struttura ad albero – 1



- Ricerca efficiente
- Capacità di raggruppamento logico
- Directory corrente (o directory di lavoro)

Directory con struttura ad albero – 2

- Pathname (percorso) assoluto o relativo
- La creazione di un nuovo file o la cancellazione di un file esistente viene effettuata nella directory corrente

```
rm <file-name>
```

- La creazione di una nuova directory viene interpretata come la creazione di una sottodirectory della directory corrente

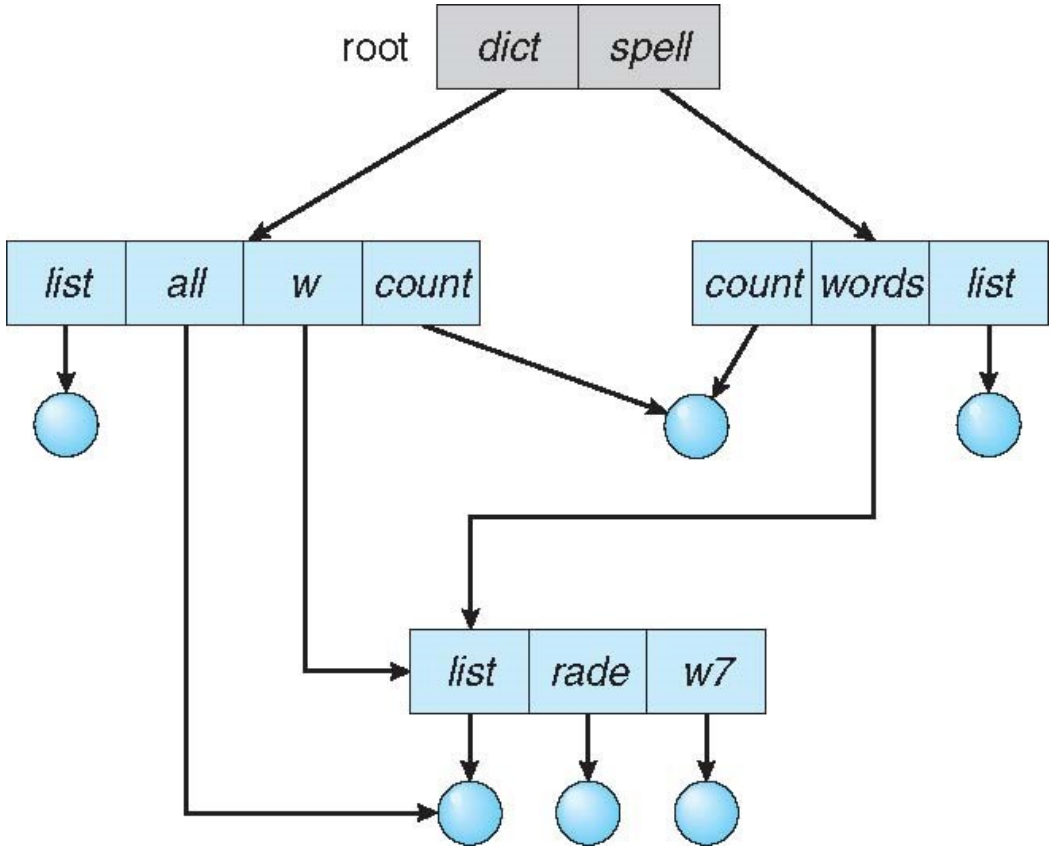
```
mkdir <dir-name>
```

- Cancellazione di una directory
  - Solo se vuota (MS–DOS)
  - Anche se contenente file e sottodirectory (es.: `rm -r`, in UNIX/Linux)

Directory a grafo aciclico – 1

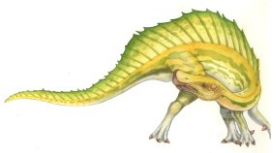
- Presenza di file e sottodirectory condivisi
- Aliasing — due o più nomi diversi possono venire utilizzati per identificare lo stesso oggetto (file o directory)
- La condivisione può essere implementata per mezzo di link (soft link) o per duplicazione dell'elemento di directory (hard link)
- Cancellando **`dict/w/list`**

  ⇨puntatore errato



Directory a grafo aciclico – 2

- Soluzioni
  - Puntatori all'indietro che permettano il reperimento di tutti i link al file cancellato e la loro eliminazione
  - Conservazione del file fino a che non esistano più link
    - Lista dei riferimenti a file: record di lunghezza variabile
    - È sufficiente mantenere il numero di riferimenti: quando il contatore è 0 il file può essere cancellato
- Nuovi "oggetti" contenuti nelle directory e operazioni correlate
  - Link – un nome diverso (un puntatore) per un file già esistente
  - Risolvere un link – seguire il puntatore corrispondente per reperire il file



Directory a grafo aciclico – 3

- Esempio

  L'utente Vivek, nella propria home directory, vuole creare un link al file **`/webroot/home/httpd/index.php`**, identificandolo semplicemente con il nome **`index.php`**

```
$ln -s /webroot/home/httpd/index.php index.php
```

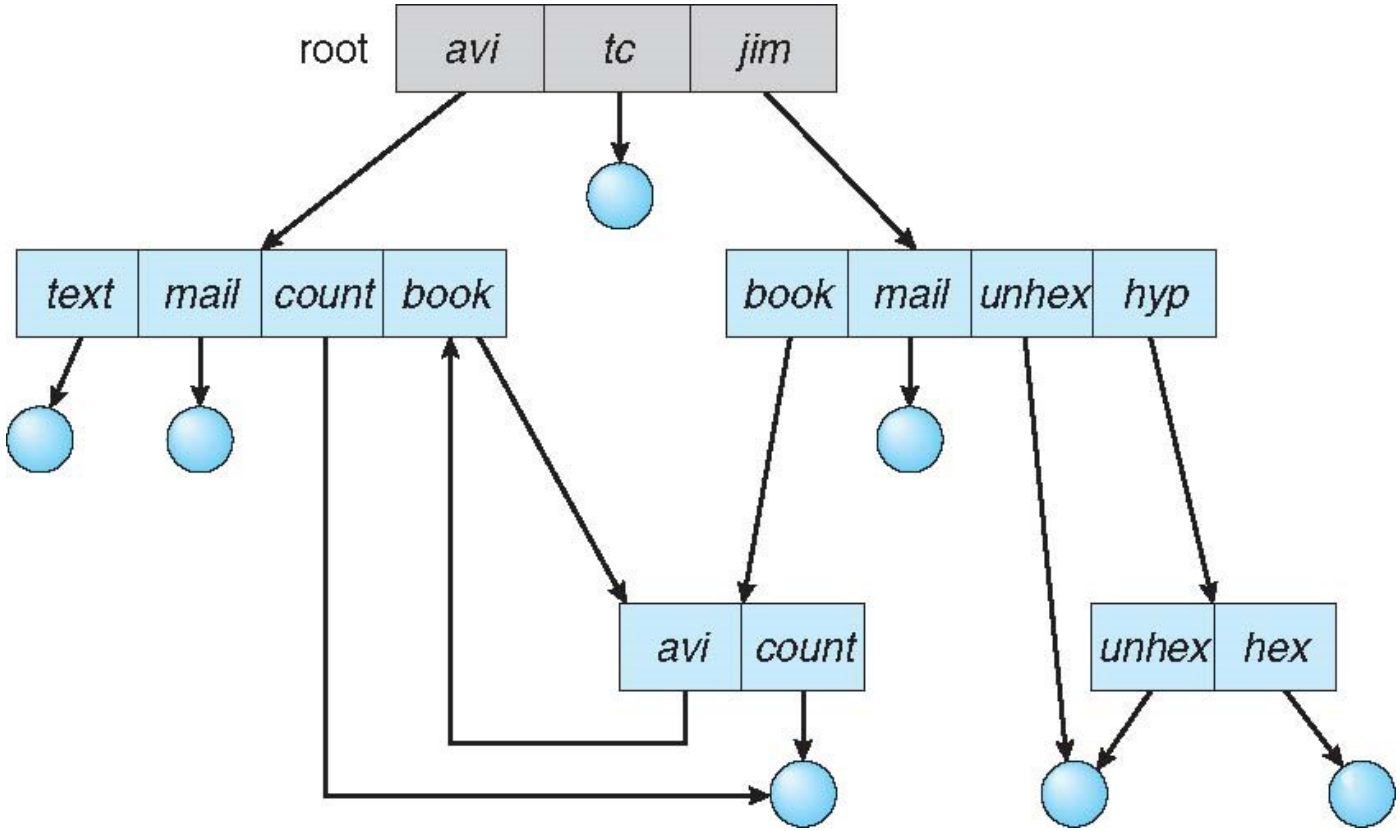
  Invocando successivamente il comando

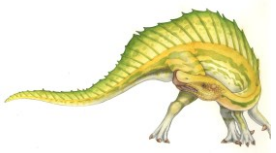
```
$ls -l
```

  otterrebbe per quel file:

```
lrwxrwxrwx 1 vivek vivek 16 2024-05-20 22:53 index.php -> /webroot/home/httpd/index.php
```

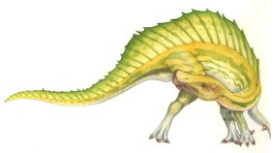
Directory a grafo generale – 1





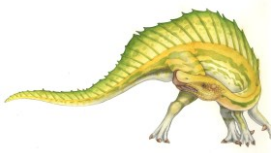
Directory a grafo generale – 2

- Mantenere il grafo aciclico garantisce semplicità degli algoritmi necessari per attraversarlo (per esempio in fase di backup)
- Come assicurare l'assenza di cicli?
  - Sono ammessi link a file, ma non a sottodirectory
  - Ogni volta che viene aggiunto un nuovo link, si verifica la presenza di cicli, mediante l'uso di un algoritmo di rilevamento (molto dispendioso, soprattutto perché effettuato sulla memoria di massa)



Directory a grafo generale – 3

- Se si ammette la presenza di cicli, in fase di scansione del file system, occorre marcare ciò che è raggiungibile ed è già stato visitato, per evitare, per esempio, copie multiple dello stesso elemento durante il backup
  - Nel caso particolare di directory e collegamenti, una tecnica semplice consiste nel non percorrere i collegamenti durante l'attraversamento delle directory
- Occorre inoltre effettuare periodicamente l'operazione di garbage collection — si attraversa il file system, marcando ciò che è accessibile; in un secondo passaggio, si sposta nell'elenco dei blocchi liberi quanto non è contrassegnato



Protezione – 1

- La salvaguardia delle informazioni contenute in un sistema di calcolo dai danni fisici (affidabilità) e da accessi impropri (protezione) è fondamentale per l'integrità e l'usabilità del sistema
  - L'affidabilità è assicurata dalla ridondanza
  - La protezione si ottiene mediante il controllo degli accessi
- Il proprietario/creatore del file deve essere in grado di controllare
  - chi può accedere al file
  - quali tipi di accesso siano leciti

Protezione – 2

- Tipi di accesso a file
  - Lettura
  - Scrittura
  - Esecuzione
  - Append
  - Cancellazione

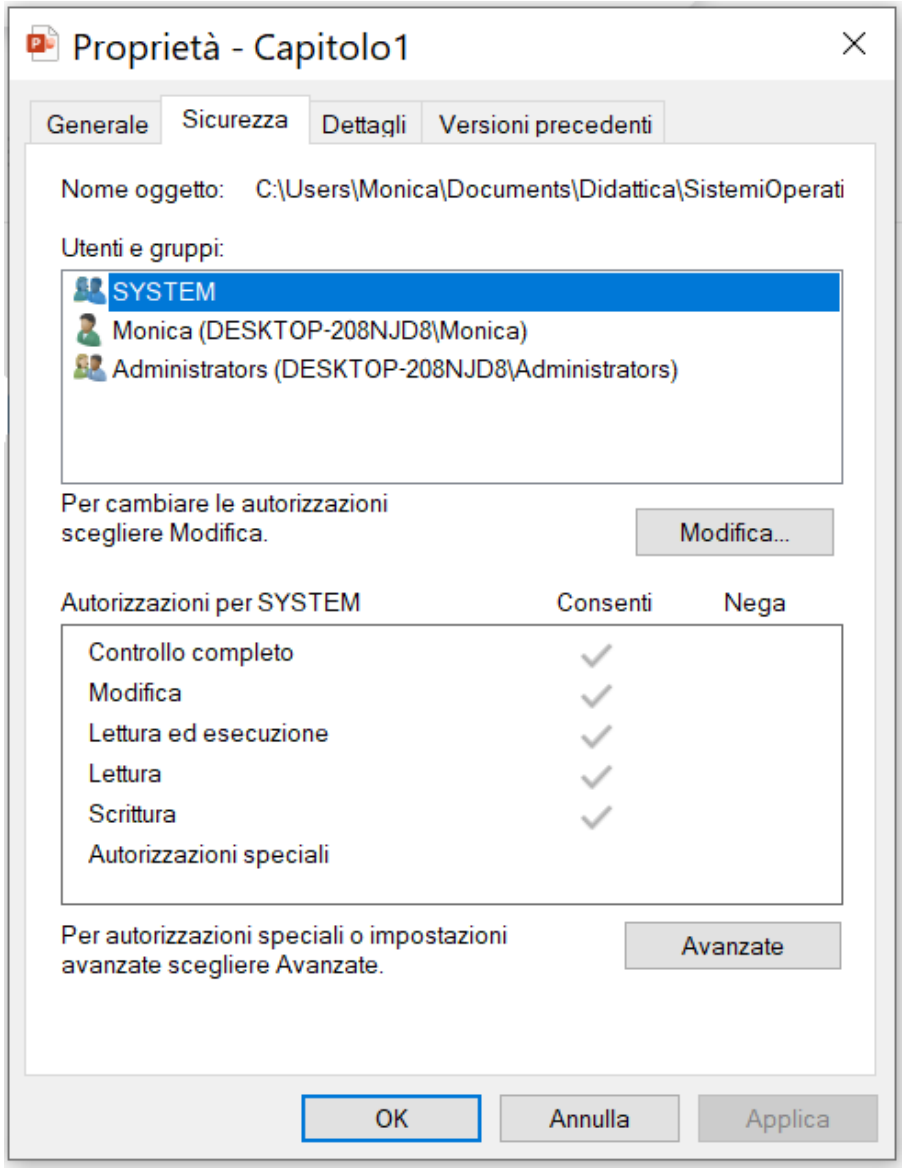
  } Sono «forme di scrittura»
  - Elencazione del nome e degli attributi

⇨ Si rende l'accesso dipendente dall'identità dell'utente

Protezione – 3

- Lista di controllo degli accessi (ACL – *Access Control List*): per ogni file si specificano i nomi di tutti gli utenti che hanno accesso al file e i relativi tipi di accesso
  - L'elemento di directory ha dimensione variabile
- Si raggruppano gli utenti in tre classi distinte – proprietario, gruppo e universo – ognuna con le proprie modalità di accesso
- Si possono combinare gli approcci, "raffinando" il secondo per mezzo del primo, che fornisce una modalità di gestione più selettiva
- Alternativamente: associazione di una password a ciascun file o directory

Esempio: Windows 10, Access Control List

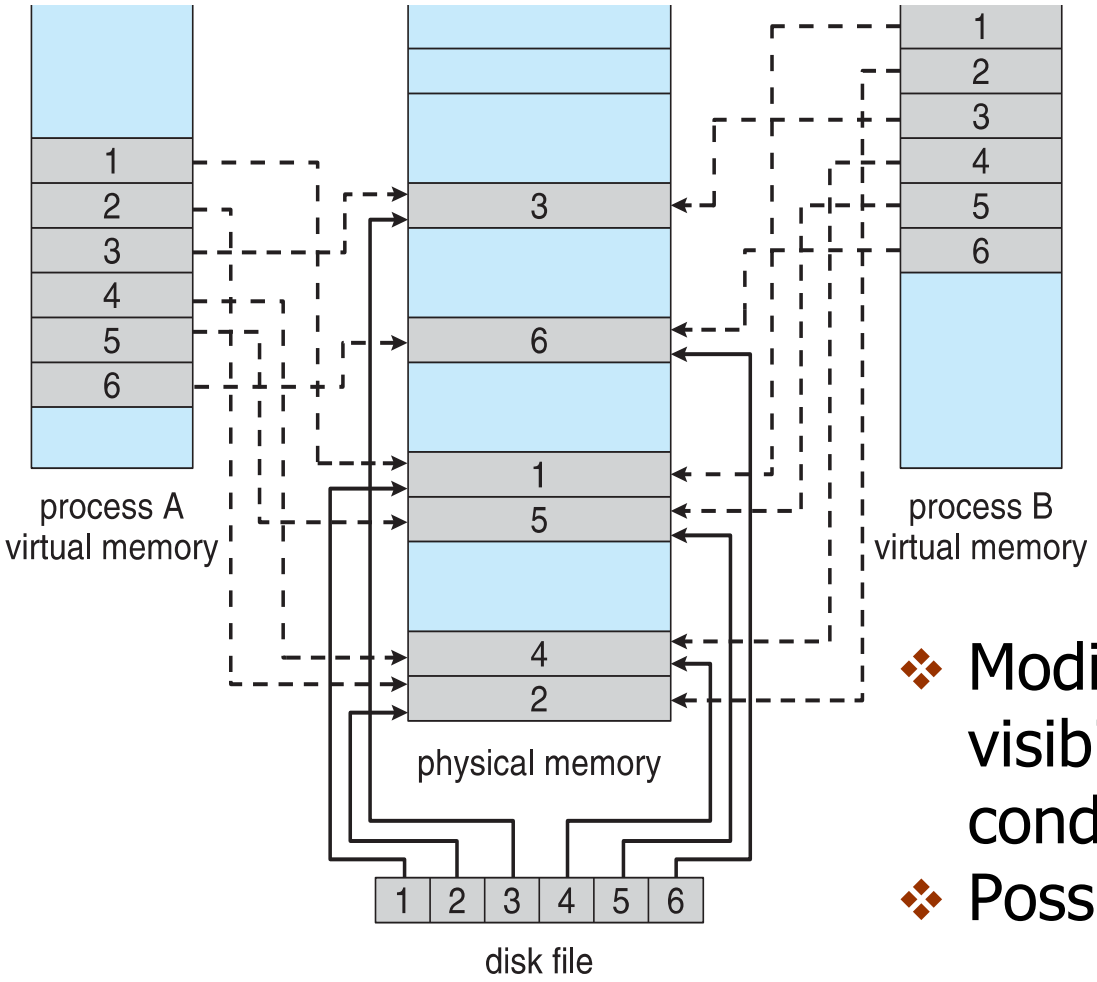


File mappati in memoria – 1

- La mappatura di file in memoria permette il trattamento dell'accesso a file come un normale accesso alla memoria
  - Si realizza associando un blocco del disco ad una o più pagine residenti in memoria
- L'accesso iniziale al file avviene tramite una richiesta di paginazione (page fault)
  - Una porzione del file, pari ad una pagina, viene caricata dal sistema in una pagina fisica
  - Ogni successiva lettura e scrittura del file viene gestita come un accesso ordinario alla memoria
    - ⇨Si alleggerisce il lavoro del sistema di I/O e, più in generale, del sistema operativo (non si effettuano le system call **`read()`**/**`write()`**)
    - ⇨Si ha accesso rapido alle informazioni contenute nel file

File mappati in memoria – 2

- Più processi possono mappare contemporaneamente lo stesso file, garantendo la condivisione di pagine di memoria



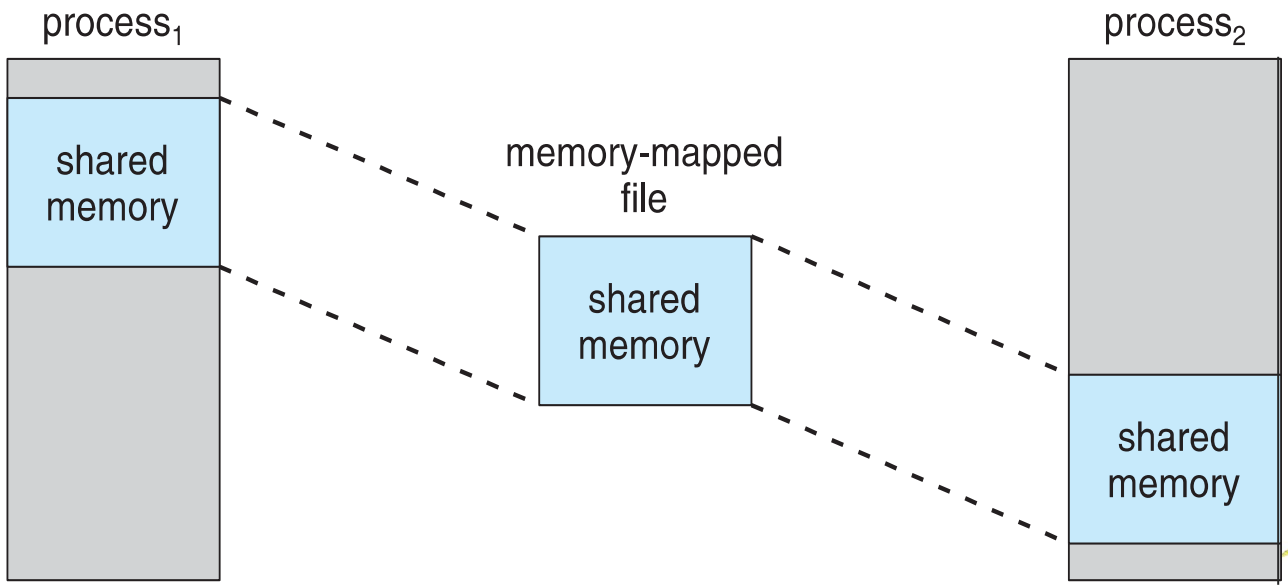
Problema: Quando aggiornare il contenuto del file su disco?

- ⇨ Periodicamente e/o all'atto della chiusura del file con la chiamata a `close()`
- ⇨ Quando il pager scandisce le pagine per controllarne il dirty bit

- Modifiche immediatamente visibili a tutti i processi che condividono il file
- Possibilità di COW

File mappati in memoria – 3

- Alcuni SO prevedono un'apposita chiamata di sistema per la mappatura dei file in memoria
- I processi possono richiedere esplicitamente la mappatura di un file in memoria con la system call **`mmap`**: il file viene mappato nello spazio degli indirizzi del processo
- La condivisione dei file in memoria mostra analogie con la memoria condivisa e, infatti, la memoria condivisa può essere realizzata utilizzando file mappati in memoria



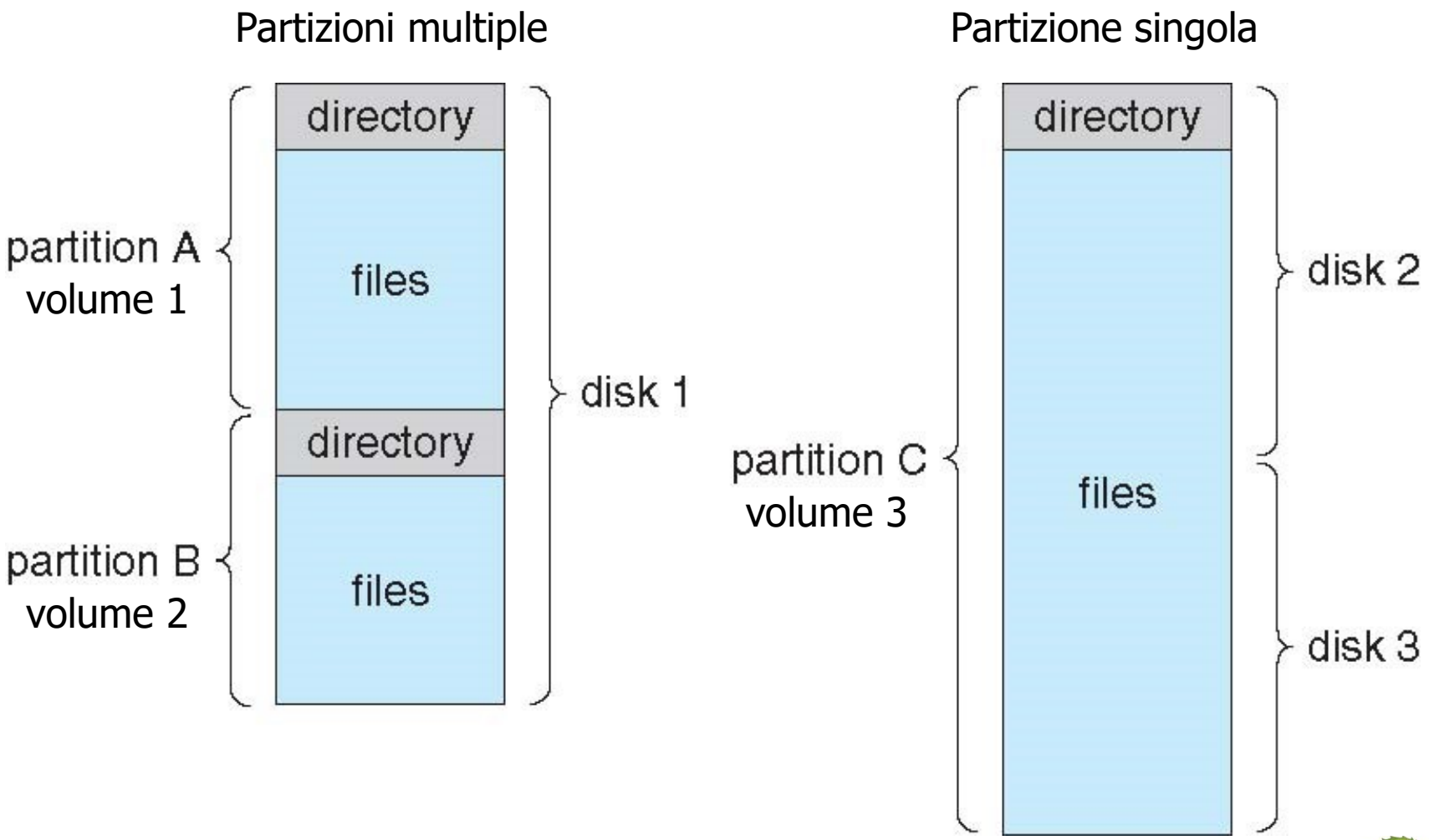
File mappati in memoria – 4

- Esempio: quando il SO carica un processo per l'esecuzione, esegue una mappatura in memoria del file che ne contiene il codice eseguibile
- Problemi: la mappatura dei file in memoria costituisce un modo per ottenere condivisione della memoria fra processi ⇨ l'accesso al contenuto del file mappato in memoria deve essere controllato utilizzando un meccanismo di gestione della concorrenza, per garantire l'integrità dei dati

Struttura del disco

- Un sistema informatico a uso generale dispone di dispositivi di memorizzazione multipli che possono essere suddivisi in partizioni, che contengono volumi, che a loro volta contengono file system
- Una partizione può essere un "pezzetto" di un dispositivo di memorizzazione, un dispositivo intero, dispositivi multipli collegati in RAID, o dispositivi suddivisi e, al contempo, assemblati in RAID
- La suddivisione in partizioni è utile per limitare la dimensione dei file system, per installarne di diverso tipo e per dedicare ad altri scopi (arie di swap e, genericamente, raw) parti del dispositivo
- Ciascuna partizione contenente un file system, detta anche volume, ha una directory di dispositivo (o una tabella dei contenuti del volume), che mantiene informazioni su tutti i file del volume

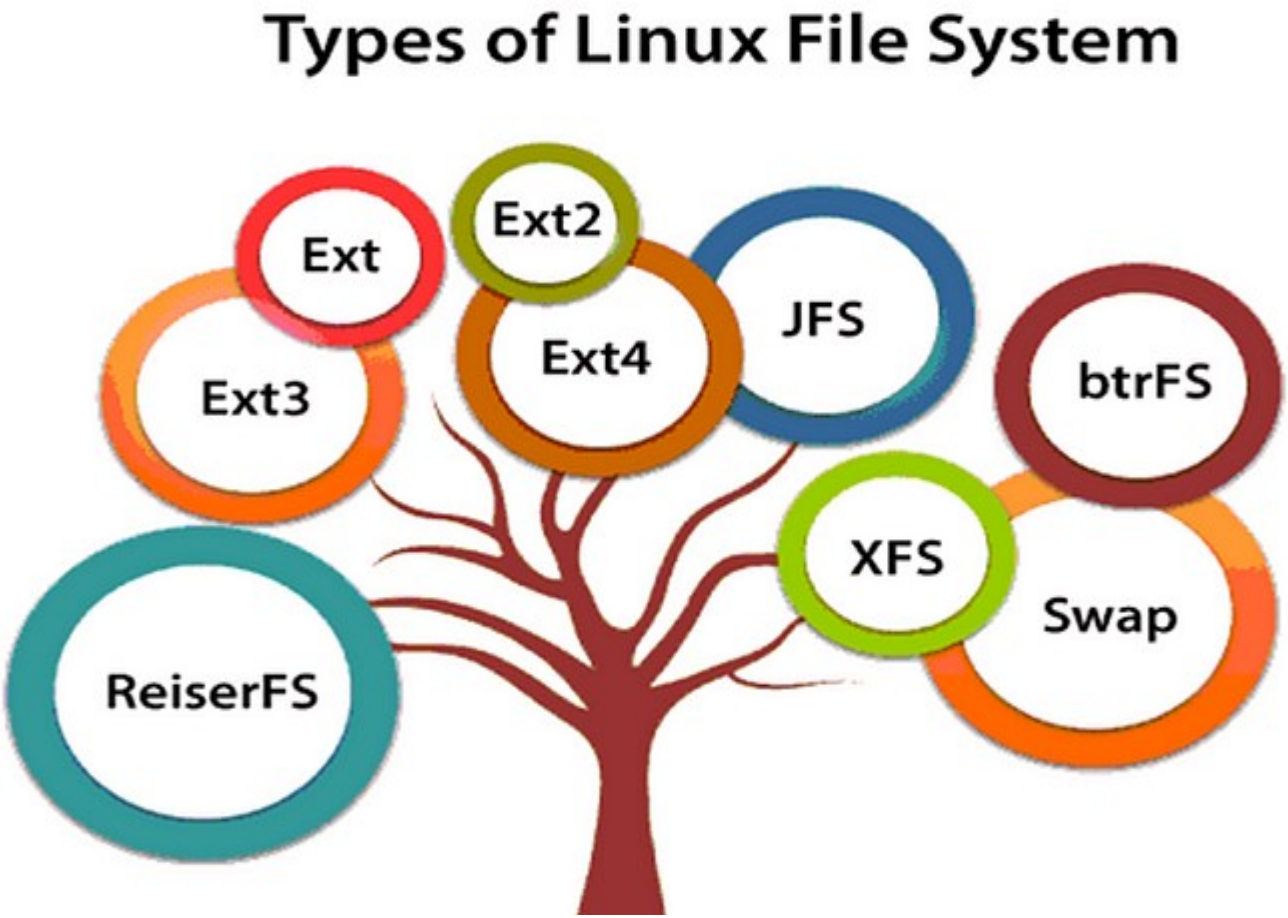
Una tipica organizzazione del file system

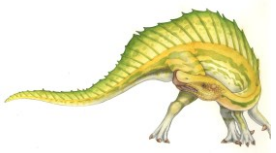


Tipi di file system – 1

- Così come vi sono file system general purpose, esistono anche file system speciali, che spesso sono residenti sullo stesso computer e sono supportati da un unico SO
- L'utilizzo dei file system si è infatti esteso oltre i confini originali
  - UNIX e Linux forniscono un file system **`proc`** (in RAM) che utilizza le interfacce del file system per fornire accesso alle informazioni di sistema (dettagli sui processi)
  - Solaris supporta, fra gli altri, **`tmpfs`** (FS temporaneo, i cui contenuti vengono cancellati se il sistema si riavvia o si blocca), **`ctfs`** (FS virtuale, che gestisce i processi che partono all'avvio del sistema e devono essere eseguiti durante tutto il suo funzionamento), **`lofs`** (FS loopback, che permette di mappare file su dispositivi virtuali), **`procfs`** (FS virtuale, interfaccia del kernel alle strutture dati che descrivono i processi), **`ufs`** e **`zfs`** di uso generale

Tipi di file system – 2





Montaggio di un file system – 1

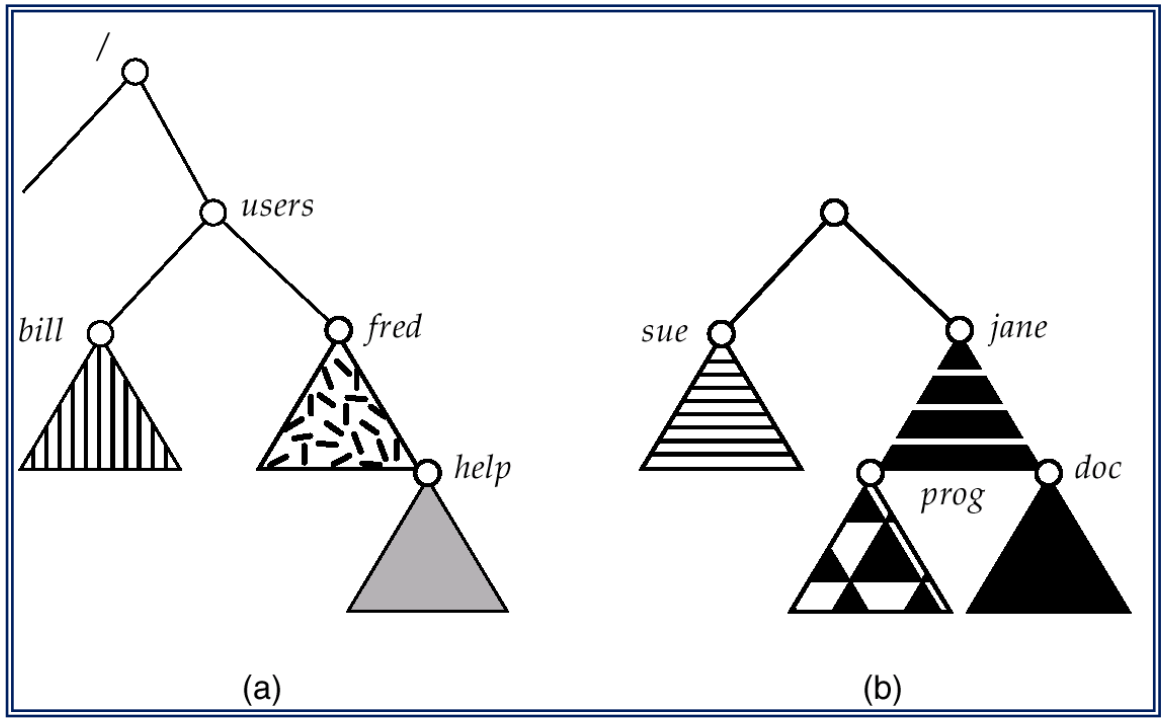
- Un file system deve essere montato prima di poter essere acceduto dai processi di un sistema
- Un file system *unmounted* può essere montato ad un mount point prescelto
- Alcuni SO richiedono un file system prefissato; altri ne supportano diversi e sondano le strutture del dispositivo/i per determinare il tipo del/i file system presente/i (eventualmente montandoli automaticamente in fase di boot)
- Procedura di montaggio
  - Si fornisce al SO il nome del dispositivo da montare (sotto forma di volume e pathname) e la locazione che dovrà assumere nella struttura del file system (punto di montaggio)
  - Di solito, il punto di montaggio è una directory vuota cui sarà agganciato il file system che deve essere montato

Montaggio di un file system – 2

- Dopo il montaggio, il SO verifica la validità del file system, chiedendo al relativo driver di leggere la directory di dispositivo per controllare che abbia il formato previsto
- Una volta montato, il file system risulta accessibile a programmi e utenti in modo trasparente e diventa parte integrante del grafo delle directory
- La directory su cui viene montato un file system può anche non essere vuota, ma nel momento in cui si effettua il montaggio, i dati ivi contenuti non sono più visibili fino all'operazione di unmount

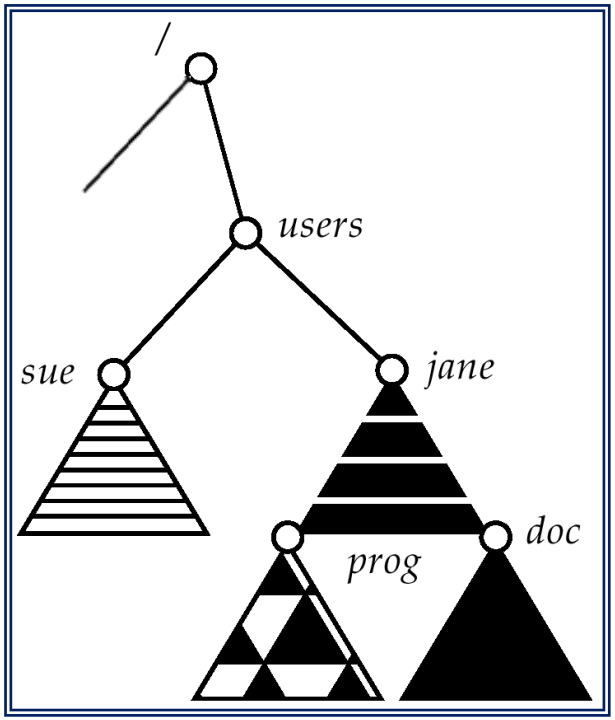
Montaggio di un file system – 3

- Esempio
  - Se si effettua il montaggio su */users* del file system in figura (b) identificato, per esempio, da */device/usr* le home degli utenti *bill* e *fred* (e sottodirectory) diventano inaccessibili



File system esistente

File system unmounted



/users è il mount point

Montaggio di un file system – 4

- I sistemi MAC e Windows rilevano tutti i dispositivi di memoria di massa all'avvio della macchina — e durante l'esecuzione — e montano automaticamente tutti i file system in essi contenuti
  - MAC OS esegue il montaggio nella directory /Volumes, aggiungendo un'icona di cartella sul desktop, etichettata con il nome del file system
  - Windows mantiene una struttura delle directory a due livelli estesa, con una lettera di unità associata a dispositivi e volumi (per esempio montando in E: un dispositivo flash)

Montaggio di un file system – 5

- Storicamente, nei sistemi UNIX–like, i file system dovevano essere montati in modo esplicito
  - Un file di configurazione del sistema contiene una lista di dispositivi/file system e relativi punti di montaggio da utilizzare all'avvio; il mount può essere effettuato anche durante la sessione di lavoro

```
$ mount /dev/usr /users
$ umount /dev/usr
```

  - In Linux: **`/etc/fstab`** è il file di configurazione di sistema per la descrizione statica dei dispositivi di memoria collegati al computer il cui mount deve essere effettuato all'avvio
  - Attualmente, si effettuano anche montaggi automatici

Montaggio di un file system – 6

fstab

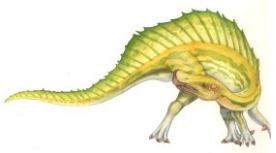
File Edit Search Options Help

```
# Turtle-Kevux Embedded Layout (Dynamic)
#LABEL=turtle_/boot    /boot                  auto       noauto,noatime,rw              0 1
#LABEL=turtle_/home    /home                  auto       defaults,noatime,rw            0 1

# Turtle-Kevux Basic Layout (Static)
#/dev/sda1             /boot                  auto       noauto,noatime,rw              0 1
#/dev/sda3             /home                  auto       defaults,noatime,rw            0 1


## SPECIAL FILESYSTEMS ##
rootfs                 /                      auto       defaults,noatime,rw            0 1
devpts                 /dev/pts               devpts     gid=devpts,mode=0620           0 0
shm                    /dev/shm               tmpfs      size=32m,gid=shm,mode=1770     0 0
proc                   /proc                  proc       gid=proc,mode=0550             0 0
usbfs                  /proc/bus/usb          usbfs      defaults                       0 0
sysfs                  /sys                   sysfs      gid=sys,mode=0550              0 0
tmp                    /tmp                   tmpfs      noauto,size=128m,gid=tmp,mode=1770    0 0
vartmp                 /var/tmp               tmpfs      noauto,size=32m,gid=vartmp,mode=1770  0 0
udev                   /dev                   tmpfs      noauto,size=1m,gid=dev,mode=0750      0 0
run                    /var/run               tmpfs      noauto,size=1m,gid=run,mode=0770      0 0
log                    /var/log               tmpfs      noauto,size=32m,gid=log,mode=2750     0 0
dbus                   /home/services/dbus/system tmpfs  size=4m,gid=dbus,mode=0750    0 0
xkbfs                  /share/X11/xkb/compiled tmpfs     noauto,size=4m,gid=xorg,mode=0750 0 0

## MEDIA DEVICES ##
# The cdrom, cdwriter, and dvd are symbolic links that may need to be updated based on your system
```

Snapshot del contenuto di **`/etc/fstab`** (file system, mount point, tipo, opzioni, dump, fsck)

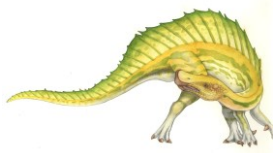
Condivisione di file

- La condivisione di file (file sharing) è particolarmente utile nei sistemi multiutente, per permettere la collaborazione fra utenti e per ridurre le risorse richieste per raggiungere un dato obiettivo di calcolo
- Tuttavia, la condivisione non può prescindere da uno schema di protezione che garantisca un controllo di accesso ai file mediato dal SO
- Nei sistemi distribuiti, i file vengono condivisi attraverso una rete
- Il Network File System (NFS) è un metodo molto diffuso (originariamente implementato in ambiente UNIX) per realizzare la condivisione di file distribuiti



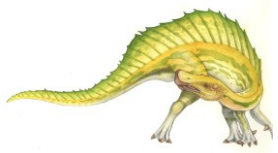
Condivisione di file in ambiente multiutente

- Il modello più diffuso è legato al concetto di proprietario di un file e di gruppo di utenti a cui il proprietario è "affiliato"
  - Il proprietario è l'utente che ha creato il file e che può cambiare gli attributi del file (o della directory)
  - L'attributo di gruppo si usa per definire il sottoinsieme di utenti autorizzati a condividere l'accesso a file
  - Gli identificativi del gruppo (GroupID) e del proprietario (UserID) di un dato file o directory sono memorizzati come attributi del file (nel relativo elemento di directory)



File system remoti

- Uso della rete per ottenere l'accesso a file residenti su sistemi remoti
  - Trasferimento richiesto esplicitamente (anonimo o autenticato) via protocollo FTP
  - Tramite un file system distribuito (DFS), che permette la visibilità e l'accesso (automatico) dal calcolatore locale a directory remote
  - Accesso tramite browser (semi–automatico) attraverso il World Wide Web (l'FTP – anonimo – non è esplicito, ma "nascosto" da un wrapper, nell'operazione di download)
  - Risorse disponibili in Cloud



File system distribuiti – 1

- Nel modello client–server, il server mette a disposizione risorse (sotto forma di directory e file) ai client che ne fanno richiesta
  - Modello molti–a–molti: un server può gestire richieste provenienti da più client, il client può accedere a più server
  - Problemi di autenticazione
    - Protocolli insicuri che possono condurre a spoofing
    - Sicurezza ottenuta mediante autenticazione reciproca di client e server tramite chiavi di cifratura

    ⇨Nuovi problemi: compatibilità fra client e server (relativamente all'algoritmo di cifratura), scambio sicuro delle chiavi

File system distribuiti – 2

- NFS è il protocollo client–server standard nei sistemi UNIX
  - Gli UserID devono coincidere nel client e nel server
  - Dopo il montaggio del file system remoto, le richieste di accesso a file vengono inviate al server attraverso la rete
  - Il server applica i normali controlli di accesso e, qualora producano un risultato positivo, restituisce un file handle al client, che lo usa per eseguire le successive operazioni su file
- CIFS (Common Internet File System) è il protocollo standard per Windows
- In entrambi i casi, le chiamate di sistema locali vengono tradotte in chiamate (per gli stessi servizi) su file system remoti

Malfunzionamenti – 1

- Nei file system locali possono verificarsi malfunzionamenti dovuti a:
  - Problemi hardware dei dischi che li contengono
  - Alterazioni dei metadati (informazioni per il reperimento dei file, contenute nelle directory)
  - Problemi ai controllori dei dischi o agli adattatori
  - Problemi ai cavi di connessione
  - Errori umani

Malfunzionamenti – 2

- Nei file system remoti i malfunzionamenti possono avvenire anche per cadute della rete o dei server remoti sui quali sono residenti i file
- Per il ripristino da malfunzionamenti è necessario mantenere alcune informazioni di stato per ciascuna richiesta remota sia sui client che sui server
- Tuttavia, protocolli molto diffusi, come NFS v3, non mantengono informazioni di stato (presenti invece nella versione v4)

Semantica della coerenza

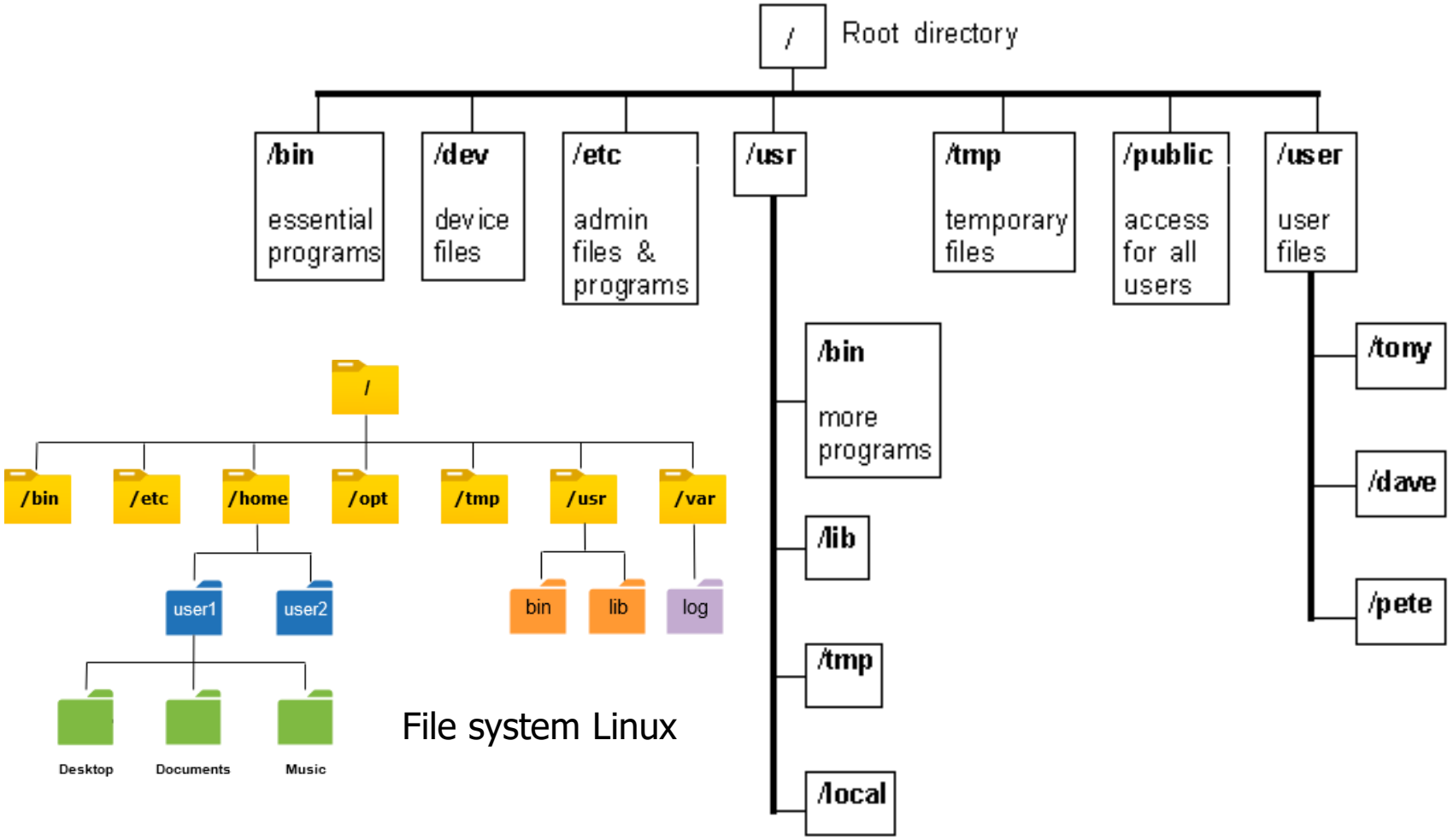
- La semantica della coerenza specifica quando le modifiche apportate da un utente ai dati contenuti in un file condiviso possano essere osservate da altri utenti
- La semantica della coerenza è correlata agli algoritmi di sincronizzazione fra processi, anche se tende ad essere meno critica a causa delle lunghe latenze e delle basse velocità di trasferimento dei dischi e della rete
- La semantica UNIX impone:
  - che le scritture in un file aperto da un utente siano immediatamente visibili agli altri utenti che hanno aperto contemporaneamente lo stesso file
  - la condivisione del puntatore alla posizione corrente nel file – il file ha una singola immagine e tutti gli accessi si alternano, intercalandosi, a prescindere dalla loro origine
  - la contesa per l'immagine fisica del file (che è unica) determina il differimento dei processi utente


LIVE FREE OR DIE
http://www.UNIX-systems.org
UNIX®
UNIX is a Registered Trademark of The Open Group

UNIX – 1

- I file e le directory possono appartenere a più directory, a costituire strutture a grafo generale
- Varie directory ospitano programmi di sistema – per esempio: dev contiene i device driver, bin il codice eseguibile, include le librerie di sistema, etc i file di configurazione etc.
- A ciascun utente è associata una directory, detta home directory
  - Le home directory sono normalmente sottodirectory della user o della home
  - I file creati dall'utente sono contenuti nella sua home directory o in sottodirectory della stessa

UNIX – 2



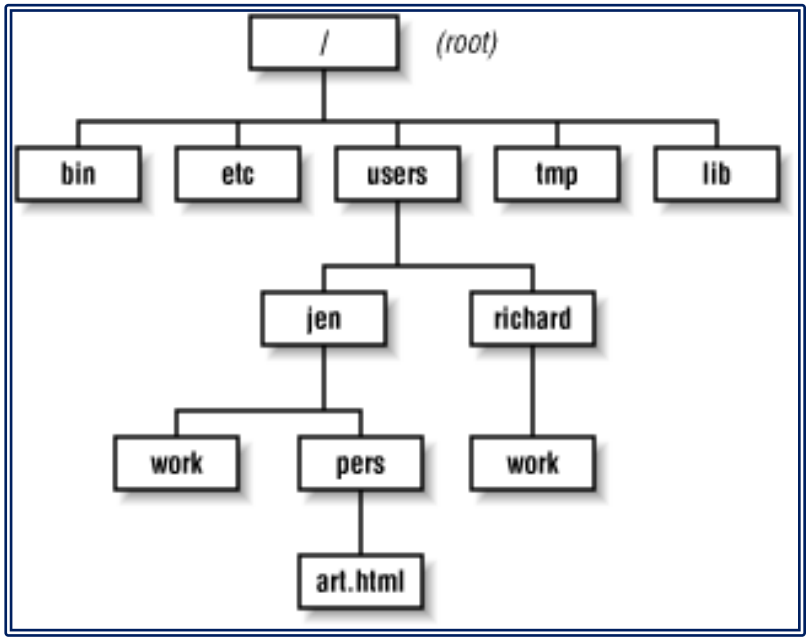
File system Linux

Il file system di UNIX

UNIX – 3

- Ciascun file viene identificato univocamente da un path-name che include l'intero cammino, dalla radice dell'albero (grafo) al file stesso
- Tutti i file e le sottodirectory presenti nella stessa directory devono avere nomi distinti
  ⇨ ciascun pathname è unico
- Il nome completo del file `art.html` è:

`/users/jen/pers/art.html`

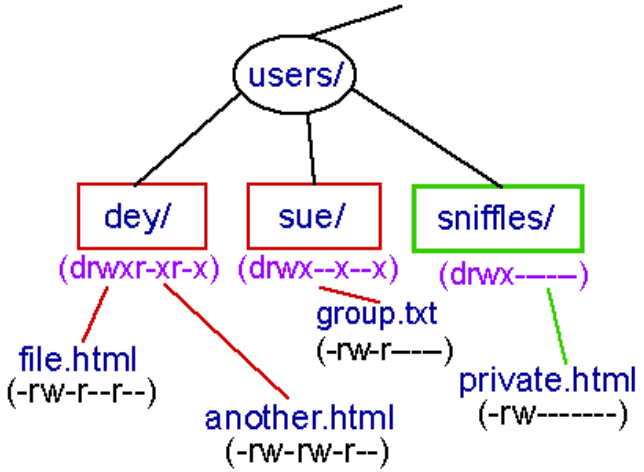


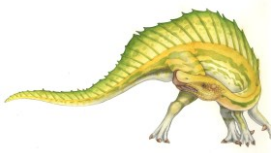
UNIX – 4

- Un utente che interagisce con il file system ha un proprio contesto, cioè una specifica posizione nel file system, corrispondente ad un nodo nel grafo (**`pwd`**)
- Per default, all'atto del collegamento al sistema, il contesto dell'utente è costituito dalla sua home directory
- Il contesto può essere variato, muovendosi ovunque nell'albero delle directory (almeno in quelle accessibili all'utente) con il comando **`cd`**
- Nell'esprimere il nome di un file o di una directory si può omettere di specificare la stringa corrispondente al contesto corrente (utilizzando un path "relativo")

UNIX – 5

- Il simbolo "." fa riferimento al contesto corrente
- Il simbolo ".." fa riferimento alla directory immediatamente superiore al contesto corrente
- Esempio: se il contesto corrente è la directory **`users`**, il file **`[...]/users/sniffles/private.html`** può essere identificato con **`sniffles/private.html`** oppure con **`./sniffles/private.html`**





UNIX – 6

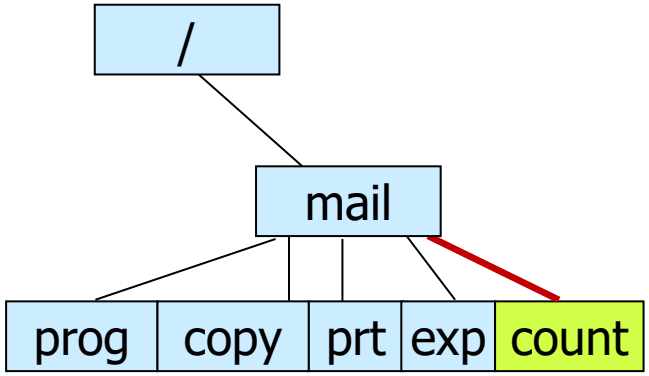
- Comandi per la manipolazione di file:
  - **`cat`** – concatena file e ne mostra il contenuto
  - **`chmod`** – stabilisce i parametri di protezione del file
  - **`chown`** – cambia il proprietario di un file
  - **`cmp`** – confronta byte a byte due file
  - **`cp`** – copia un file in un altro (situato in altra directory, o nella stessa, ma con nome diverso)
  - **`find`** – trova file per nome o per altre caratteristiche
  - **`grep`** – ricerca file attraverso una stringa in essi contenuta (e riporta la linea contenente la stringa)
  - **`ln`** – crea un link
  - **`more`** – visualizza il contenuto di un file (su terminale, per default)
  - **`mv`** – muove un file (serve anche per rinominarlo)
  - **`rm`** – cancella un file
  - **`vi(m)`** – editor di testo

UNIX – 7

- Comandi per la manipolazione di directory:
  - `cd` – cambia la directory corrente
  - `ls` – elenca i file e le sottodirectory contenute in una directory
  - `mkdir` – crea una directory
  - `pwd` – mostra il pathname della directory corrente
  - `rmdir` – rimuove una directory
  - `mount` – monta un file system (`umount` per lo smontaggio)
- Per ottenere aiuto/informazioni sui comandi del SO
  - `help` – per ottenere informazioni sui comandi builtin della shell
  - `info` – sistema di consultazione dei manuali tramite ipertesti
  - `man` – per consultare l'help in linea
- Tutti i comandi possono essere invocati utilizzando particolari opzioni, precedute dal simbolo "–"

UNIX – 8

- Esempio: se la directory corrente è **`/mail`**, il comando **`mkdir count`** produce…



- Cancellando **`/mail`** (con opzione **`-r`**)

  ⇨ viene cancellato l'intero sottoalbero che ha **`/mail`** per radice

UNIX – 9

- Un hardlink è un'etichetta o un nome (alternativo) assegnato ad un file
- Si possono avere nomi distinti che identificano lo stesso file (per esempio in directory diverse)
- I comandi eseguiti relativamente ad uno qualsiasi dei nomi di un file operano sugli stessi "contenuti"

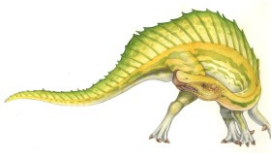
  `ln oldfile newlink`

  crea un nuovo elemento nella directory di lavoro, **`newlink`**, che è comunque correlato ai contenuti del file **`oldfile`** (non rappresenta una copia di **`oldfile`**)
- Qualsiasi cambiamento effettuato su **`oldfile`** si manifesta anche su **`newlink`** (e viceversa)
- Il comando **`rm`** può essere utilizzato anche per rimuovere un hardlink
  - Dopo la rimozione del link il contenuto del file viene comunque mantenuto (fino a che c'è almeno un nome che si riferisce a quel file)

UNIX – 10

- Un link simbolico o softlink è un tipo speciale di file che punta ad un altro file
- Diversamente dagli hardlink, non contiene i metadati relativi al target file, semplicemente punta ad un diverso elemento di directory da qualche parte nel file system
- All'atto della cancellazione del file, i softlink divengono inutilizzabili, mentre gli hardlink preservano (fino a quando ne esiste almeno uno) il contenuto del file
- Per creare un link simbolico:

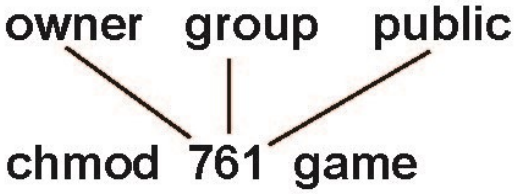
```
ln -s source_file myfile
```



Protezione in UNIX – 1

- Quando crea un file, l'utente specifica la protezione da applicare ad esso, ossia indica quali operazioni possono essere eseguite da ciascun utente sul file
- In UNIX:
  - Tipi di accesso: lettura, scrittura, esecuzione, append (scrittura in coda), cancellazione, nome e lista degli attributi
  - Modo di accesso: read, write, execute
  - Tre classi di utenti: user, group, all (o others)
  - Esempio: il file **`game.c`** con diritti di accesso

			RWX
accesso del proprietario	7	⇨	1 1 1
accesso del gruppo	6	⇨	1 1 0
accesso pubblico	1	⇨	0 0 1



può essere letto, scritto ed eseguito dal legittimo proprietario, letto e scritto dagli utenti del suo gruppo, solo eseguito da tutti gli altri utenti del sistema

Protezione in UNIX – 2

- Uso di **`chmod`** in modalità relativa

  `$ chmod [ugoa] [+−] [rwxX] file(s)`

- Esempi

  `$ chmod u+x script.sh`

  aggiunge il diritto di esecuzione per il proprietario del file **`script.sh`**

  `$ chmod –R ug+rwX src/*`

  aggiunge (ricorsivamente) i diritti di lettura e scrittura per il proprietario ed il gruppo relativamente ai file contenuti nella directory **`src`**; aggiunge inoltre il diritto di esecuzione per le directory

Protezione in UNIX – 3

- Uso di **`chmod`** in modalità assoluta

User			Group			Others		
R	W	X	R	W	X	R	W	X
4	2	1	4	2	1	4	2	1

- Esempi

  `$ chmod 755 pippo.txt`

  assegna diritto di lettura, scrittura ed esecuzione all'utente proprietario, diritto di lettura ed esecuzione al gruppo ed agli altri utenti

  `$ chmod 644 prova.tex`

  assegna diritto di lettura e scrittura all'utente proprietario ed il solo diritto di lettura al gruppo e agli altri utenti

Protezione in UNIX – 4

- Per assegnare il gruppo ad un file, occorre prima creare il gruppo stesso, supponiamo **`G`**, da superuser (**`groupadd`**)
- Occorre quindi "popolare" il gruppo con gli utenti che ne dovranno fare parte
- Per definire il gruppo del file **`game.c`**, uso di **`chgrp`**

```
$ chgrp G game.c
```

Protezione in UNIX – 5

- Come per i file, a ciascuna directory sono associati tre campi (proprietario, gruppo e universo), ciascuno composto dai tre bit rwx
  - Un utente può elencare il contenuto di una directory solo se il bit r è inserito nel campo appropriato, può modificarne il contenuto solo se è impostato il bit w, e può accedervi se è impostato x

-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/

Esempio 1

- In un file system UNIX si consideri il file

  `/usr/tizio/appunti/esercitazione1`

- Quali diritti deve possedere l'utente **`caio`** sulle directory **`usr`**, **`tizio`** e **`appunti`** per poter cancellare il file?

1) directory **`usr`**: x
2) directory **`tizio`**: x
3) directory **`appunti`**: w, x

Esempio 2

- Si consideri il file **`/usr/tizio/appunti/esercitazione`**, creato dall'utente **`tizio`**.
- I diritti associati alle directory **`usr`**, **`tizio`**, **`appunti`** ed al file **`esercitazione`** sono i seguenti:

	user	group	others
usr	r-x	r-x	r-x
tizio	rwx	--x	--x
appunti	rwx	r-x	r--
esercitazione	rw-	rw-	r--

- Quali tra le operazioni di lettura, scrittura e cancellazione possono essere eseguite sul file **`esercitazione`** dall'utente **`caio`** se:
  - **`caio`** e **`tizio`** appartengono allo stesso gruppo;
  - **`caio`** e **`tizio`** appartengono a gruppi diversi.

Esempio 2 (cont.)

- Soluzione

	user	group	others
usr	r-x	r-x	r-x
tizio	rwx	--x	--x
appunti	rwx	r-x	r--
esercitazione	rw-	rw-	r--

- Se **`caio`** e **`tizio`** appartengono allo stesso gruppo, **`caio`** può leggere (e scrivere) il file **`esercitazione`**, ma non lo può cancellare perché non ha diritto di scrittura sulla directory **`appunti`**.
- Se **`caio`** e **`tizio`** appartengono a gruppi diversi, **`caio`** non può eseguire nessuna operazione sul file **`esercitazione`** poiché non può accedere alla directory **`appunti`** (sulla quale non è abilitato ad eseguire il comando **`cd`**).

Esempio 3

- Descrivere e spiegare il significato degli attributi del file **`pluto`**, ottenuti come risultato del comando

```
$ls –la
$-rw-r--r-x 3 giorgio  collab  41139  Feb 22 11:20  pluto
```

- Qual è l'effetto del comando "**`chmod 624 pluto`**"?
- Qual è l'effetto di "**`chmod u+x pluto`**"?

Esempio 3 (cont.)

- Soluzione

```
$ls –la
$-rw-r--r-x 3 giorgio  collab  41139  Feb 22 11:20  pluto
```

- Il file **`pluto`** può essere letto e scritto dall'utente proprietario, solo letto dal gruppo, letto ed eseguito dall'universo. Esistono tre collegamenti (hard) al file. L'utente proprietario è **`giorgio`**, che appartiene al gruppo **`collab`**. La dimensione del file in byte è 41139 ed è stato acceduto l'ultima volta il 22 Febbraio alle 11:20.
- Il comando "**`chmod 624 pluto`**" lascia invariati i diritti del proprietario, concede il diritto di sola scrittura al gruppo e di sola lettura all'universo.
- Il comando "**`chmod u+x pluto`**" aggiunge il diritto di esecuzione al proprietario.

Fine del Capitolo 12