Interfaccia del file system

Pietro Braione

Reti e Sistemi Operativi – Anno accademico 2021-2022

Obiettivi

- Spiegare la funzione del file system
- Descrivere le interfacce del file system
- Presentare i trade-off di progettazione dei file system:
  - Metodi di accesso
  - Condivisione dei file
  - Uso dei lock
  - Strutture delle directory
- Analizzare la protezione dei file system

Il concetto di file e di file system

- Il **file system** è il modo attraverso il quale il sistema operativo memorizza in linea i dati e i programmi
- Il file system è costituito da:
  - Un insieme di file
  - Una struttura delle directory, che organizza i file
- Un **file** è una unità di memorizzazione logica, un insieme di informazioni correlate, registrate in memoria secondaria, alle quali è stato dato un nome
- Un file a sua volta è costituito da una sequenza di record, righe, bit o byte, il cui significato è definito dal creatore del file

Attributi dei file

- Un file possiede un insieme di attributi:
  - **Nome**: è di solito l'unica informazione in forma umanamente leggibile
  - **Identificatore**: un'etichetta unica fornita dal file system per distinguere i file
  - **Tipo**: tipo di dati contenuti nel file (alcuni sistemi operativi non hanno questo attributo)
  - **Locazione**: dispositivo di memoria secondaria e posizione nel dispositivo dove l'informazione del file è memorizzata
  - **Dimensione**: in byte, parole, record…
  - **Protezione**: informazione di controllo accessi
  - **Ora, data e utente** che ha creato, letto o modificato per ultimo il file
  - **Attributi estesi**: checksum, codifica caratteri, applicazioni correlate…
- Le informazioni sul file sono memorizzate nelle **directory**

Operazioni dei processi sui file

- **Creazione**: viene riservato spazio nel filesystem per i dati, e viene aggiunto un elemento nella directory
- **Scrittura**: a partire dalla posizione determinata da un puntatore di scrittura
- **Lettura**: a partire dalla posizione determinata da un puntatore di lettura (di solito coincide con il puntatore di scrittura)
- **Riposizionamento** (**seek**): spostamento del puntatore all'interno del file
- **Cancellazione e troncamento**: il troncamento cancella i dati ma non il file con i suoi attributi
- **Apertura**: effettuata prima dell'utilizzo di un file per copiare nella **tabella dei file aperti** dell'OS le informazioni sul file dalla directory
- **Chiusura**: effettuata alla fine dell'utilizzo di un file per liberare l'entry nella tabella dei file aperti

Tabelle dei file aperti (1)

- Un file può essere contemporaneamente aperto da più processi
- Il sistema operativo pertanto mantiene una tabella dei file aperti per ogni processo, dove ogni entry punta ad una corrispondente tabella dei file aperti di sistema
- La seconda mantiene un **contatore delle aperture**, che conta quante volte un file è stato aperto
- Se un processo apre un file, il contatore è incrementato, e quando lo chiude viene decrementato
- Quando il contatore arriva a zero, viene liberata l'entry nella tabella di sistema

Tabelle dei file aperti (2)

- Ad ogni file aperto è di solito associata la seguente informazione:
  - Puntatore alla posizione corrente: nella tabella del processo
  - Contatore delle aperture: nella tabella di sistema
  - Locazione del file sui dispositivi di memoria secondaria: nella tabella di sistema
  - Diritti di lettura/scrittura: nella tabella di sistema

Lock dei file

- Alcuni sistemi operativi permettono di associare ai file (o a porzioni di esso) dei lock per coordinare più processi che operano in concorrenza
- Due tipi di lock:
  - Lock **condiviso**:
    - Simile al lock di lettura nel problema dei lettori-scrittori
    - Più processi possono acquisirlo
  - Lock **esclusivo**:
    - Simile al lock di scrittura nel problema dei lettori-scrittori
    - Solo un processo alla volta può acquisirlo
- Altre possibilità:
  - Lock **obbligatori** (**mandatory**): il sistema operativo impedisce l'accesso al file ad altri processi al file se il lock è acquisito da qualche processo
  - Lock **consultivi** (**advisory**): il sistema operativo non regola l'accesso al file
  - I sistemi Windows adottano i lock obbligatori, i sistemi Unix-like i lock consultivi

Tipi di file

- Possibili tipi:
  - Dati (numerici, testo, binari)
  - Programmi
- Il sistema operativo può essere più o meno consapevole del tipo di file, ma deve almeno riconoscere il tipo di file eseguibile
- Possibili tecniche per riconoscere il tipo di file:
  - Schema del nome (nome . estensione)
  - Attributi nei file (ad esempio in macOS viene registrato il programma che ha creato il file)
  - «Magic number» all'inizio del file (ad esempio «shebang» magic cookie all'inizio degli script Unix)

Struttura dei file

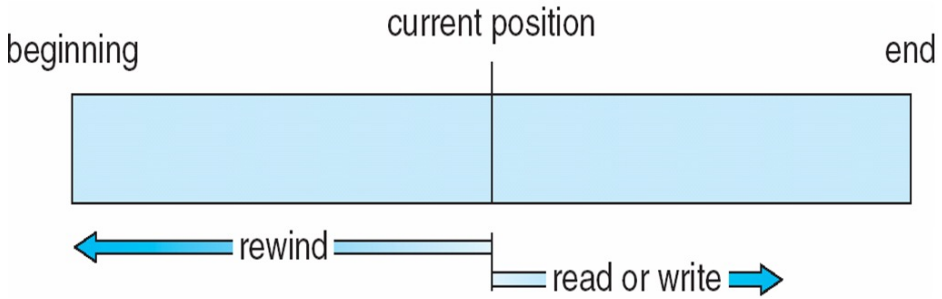
- Possibilità:
  - Nessuna struttura (ad esempio, nei sistemi Unix-like un file è una sequenza di byte)
  - Sequenza di record (righe di testo o record binari, a struttura e lunghezza fissa o variabile)
  - Strutture più complesse e standardizzate, soprattutto per file eseguibili (formato PE in Windows, a.out ed ELF nei sistemi Unix-like, Mach-O in macOS)
- Più il sistema operativo supporta direttamente diverse strutture di file, più diventa complesso
- Inoltre se il sistema operativo è troppo «rigido» sulle possibili strutture, potrebbe non supportare nuovi tipi, o tipi ibridi

Impaccamento

- Un algoritmo di impaccamento stabilisce come un certo numero di record logici di un file è memorizzato in un blocco fisico del dispositivo di memoria secondaria
- La tecnica di impaccamento e le dimensione relative record logico / blocco fisico determinano quanti record è possibile memorizzare in un blocco fisico
- Inoltre la tecnica di impaccamento e la dimensione totale del file determinano il grado di frammentazione interna della memoria secondaria

Metodi di accesso: accesso sequenziale

- Il file è una sequenza di record a lunghezza fissa
- Operazioni:
  - read_next() e write_next() leggono/scrivono il successivo record dalla posizione corrente
  - Operazione di riavvolgimento



Metodi di accesso: accesso diretto

- Operazioni read(n) e write(n) per accedere direttamente all'n-esimo record
- Alternativamente, operazioni read_next(), write_next() e position(n)

Metodi di accesso: accesso indicizzato

- In alcuni sistemi operativi, ad esempio quelli per mainframe IBM, i file possono essere sequenze di record ordinate secondo un determinato campo chiave del record
- In tali sistemi l'accesso può essere basato sulla chiave, e il sistema operativo mantiene un indice per velocizzare l'accesso
- Esempio: accesso ISAM nei sistemi IBM, file system Files-11 prodotto da Digital per il sistema operativo OpenVMS (offre tutti e tre i tipi di accesso su record a lunghezza fissa o variabile)

Directory

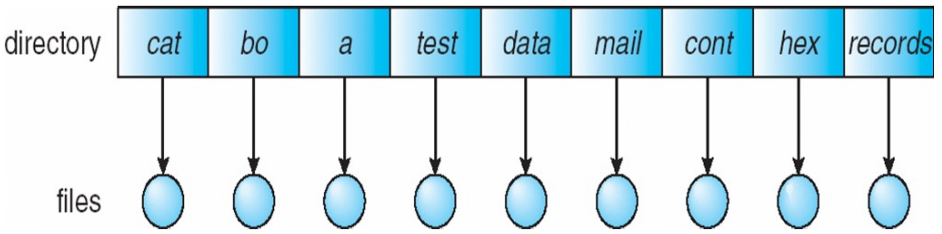
- Una directory è una tabella di simboli che traduce il nome di un file negli elementi in essa contenuti
- Sia i file che le directory risiedono su disco
- Operazioni sulle directory:
  - **Ricerca** di un file: basata sul nome, o su uno schema di possibili nomi
  - **Creazione** di un file
  - **Cancellazione** di un file
  - **Ridenominazione** di un file
  - **Elenco** dei file nella directory
  - **Attraversamento** del file system, ad esempio per backup

Struttura delle directory

- Ad un livello
- A due livelli
- Ad albero
- A grafo aciclico
- A grafo generale

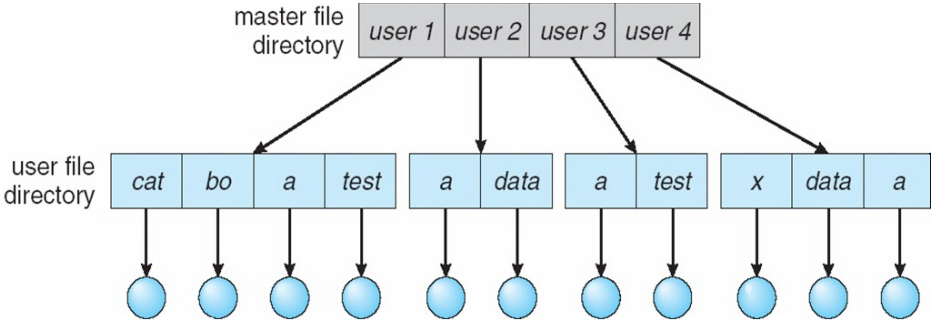
Struttura delle directory ad un livello

- Una sola directory per tutti i file
- Vantaggi: semplicità
- Svantaggi:
  - Difficoltà naming file quando sono molti
  - Difficoltà raggruppamento file di utenti diversi



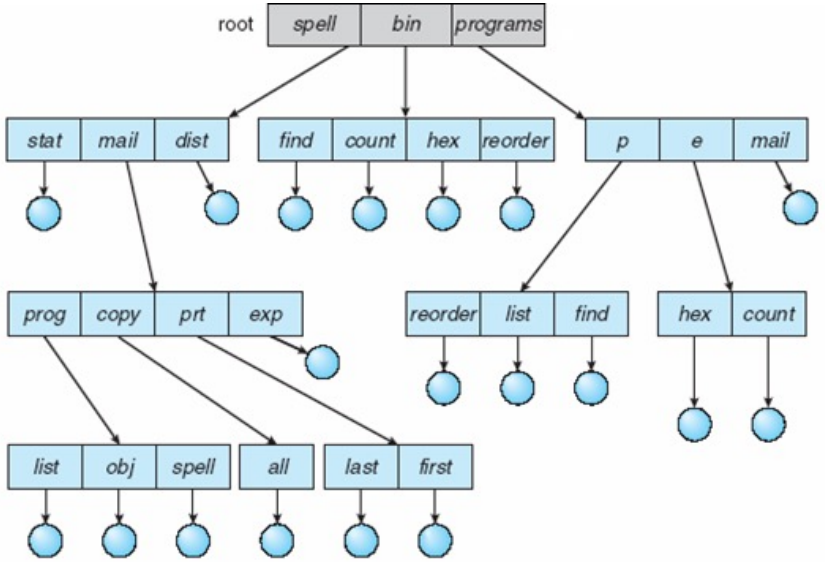
Struttura delle directory a due livelli

- La directory principale contiene delle sottodirectory, una per ogni utente
- La sottodirectory utente contiene i file dell'utente
- Utenti diversi possono dare lo stesso nome a file diversi
- Occorre quindi usare dei nomi di percorso (path name) per identificare un file univocamente
- I file di sistema di solito sono posti in una o più directory speciali, e occorre che il sistema conosca un **percorso di ricerca** per trovarli



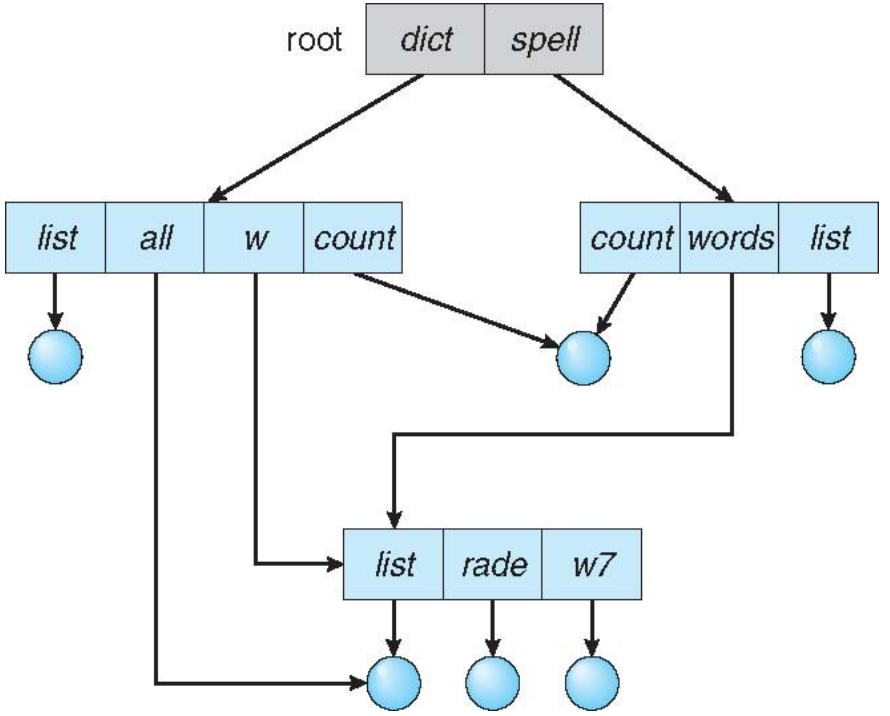
Struttura delle directory ad albero

- Ogni directory contiene ricorsivamente files e altre directory
- Permette agli utente di raggruppare i propri file
- Per semplificare l'accesso ad ogni programma è assegnata una **directory corrente**, dalla quale si possono specificare **path relativi**
- Problema cancellazione directory:
  - Cancello tutto il contenuto
  - Permetto di cancellare una directory solo se è vuota
  - La seconda soluzione è più sicura



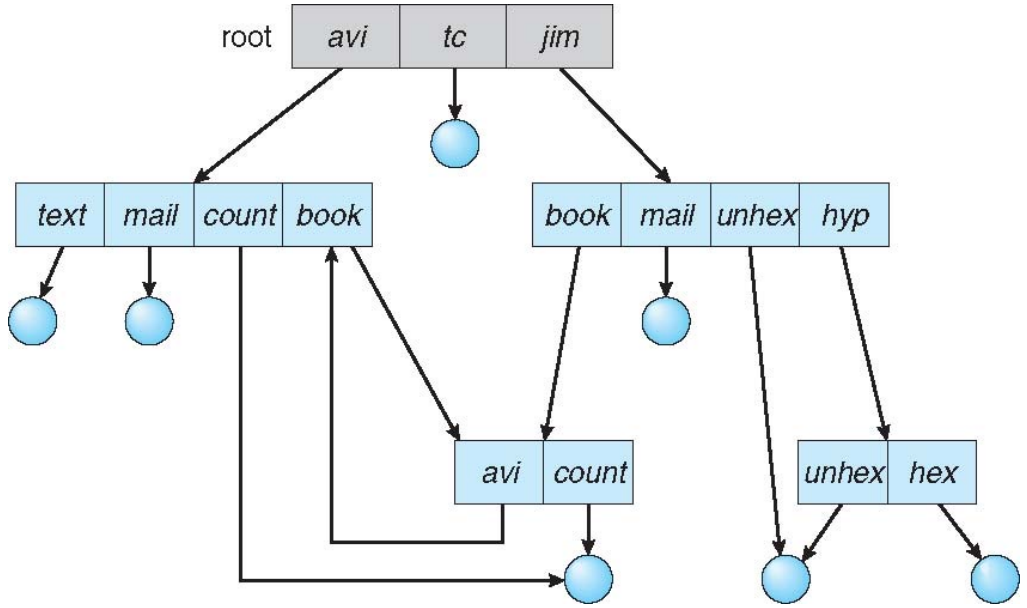
Struttura delle directory a grafo aciclico

- Permette l'aliasing (più di un nome per lo stesso file)
- Cosa succede se cancello un file/directory con un alias?
  - **Hard links**: duplicazione voci di directory; viene introdotto un contatore ai riferimenti, quando è a zero viene cancellato il file
  - **Link simbolici**: riferimenti simbolici al path assoluto del file originale, quando questo è cancellato restano dangling
- Problema attraversamento



Struttura delle directory a grafo generico

- Un contatore al numero dei riferimenti non basta, occorre un'autentica garbage collection
- Attraversamento del file system ancora più complicato



Protezione

- Le informazioni devono essere protette dai danni fisici (affidabilità) e dagli accessi impropri (protezione)
- Un sistema multiutente permette un accesso controllato ai file di un certo utente da parte degli altri utenti
- Operazioni controllabili:
  - Lettura
  - Scrittura
  - Esecuzione
  - Aggiunta (scrittura in coda ad un file)
  - Cancellazione
  - Elencazione (elenco contenuto directory)

Liste di controllo degli accessi (1)

- Idea:
  - Ad ogni file/directory è associata una lista di controllo degli accessi (access control list, ACL)
  - L'ACL specifica gli utenti che possono accedere al file/directory, con i relativi permessi di accesso
  - Il file system controlla l'ACL prima di ogni accesso al file
- Principale svantaggio: le ACL possono diventare molto lunghe
- Approccio usato nei sistemi Unix-like: raggruppare gli utenti in classi distinte:
  - Proprietario: l'utente che possiede il file/directory
  - Gruppo: il gruppo di utenti che condivide il file/directory
  - Pubblico: tutti gli altri utenti

Liste di controllo degli accessi (2)

- Esempio:

			r w x
• Accesso proprietario	7	⇒	1 1 1
• Accesso gruppo	6	⇒	1 1 0
• Accesso pubblico	4	⇒	1 0 0

- Supponiamo di avere un file `game`, di volergli assegnare un gruppo G, e di volergli attribuire i permessi di cui sopra:
  - `chgrp G game  %cambia gruppo al file game`
  - `chmod 764 game %cambia i permessi al file game`