giovedì 22 giugno 2023

ARCHITETTURE DI CALCOLO LEZIONE 23

Memoria virtuale, interfaccia del file system ed implementazione dei File System

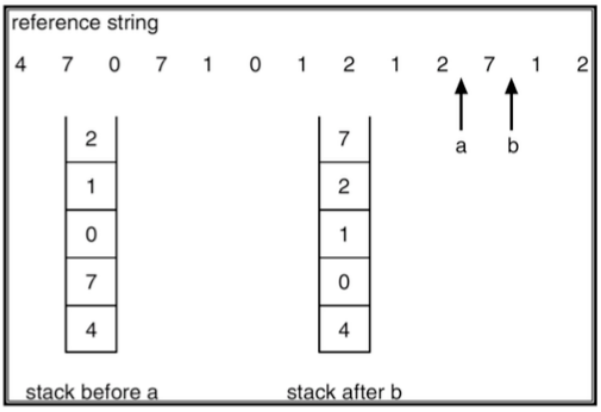
ALGORITMO LEAST RECENTLY USED (LRU)

La scorsa lezione si è già parlato degli algoritmi di sostituzione delle pagine che sono FIFO, l'algoritmo ottimale e l'LRU. L'ottimale serve da termine di paragone poiché, andando a verificare qual è la pagina che sarà utilizzata più in là nel tempo, riesce a ottenere il minor numero possibile di sostituzioni (page fault). L'LRU, invece, lavora al contrario: infatti, tra le pagine in memoria, elimina, quando necessario, quella che è stata usata più lontano nel tempo.
Una problematica fondamentale nell'algoritmo LRU è riuscire a misurare quanto tempo fa è stata acceduta ogni singola pagina in memoria. Questo può essere fatto in due modi principali: il primo modo prevede di implementare l'algoritmo con un contatore (clock); l'altro prevede di implementare l'algoritmo con lo stack.

Implementazione con clock
Nell'implementazione con il contatore, viene associato un registro contatore ad ogni pagina che è in memoria. Questo è un registro in cui viene scritto il clock di quando quella pagina è stata acceduta. Ogni volta che bisogna sostituire una pagina si va a vedere il valore del contatore e viene eliminata quella con l'accesso più lontano e dunque quella con clock più piccolo. Il problema in questo caso è che bisogna scandire il contatore di ogni singola pagina per trovare il clock minimo.

Implementazione con stack
L'implementazione con stack cerca di rendere più veloce individuare quale pagina va eliminata. In questo stack si mantiene il numero di pagine che sono in memoria e funziona come una pila. La pagina che è stata utilizzata più di recente si trova in cima allo stack, mentre quella usata meno di recente si trova in fondo. Una volta che una pagina viene letta o scritta viene spostata in cima. In questo modo automaticamente ci troviamo la pagina letta più in là nel tempo in fondo allo stack.

Esempio



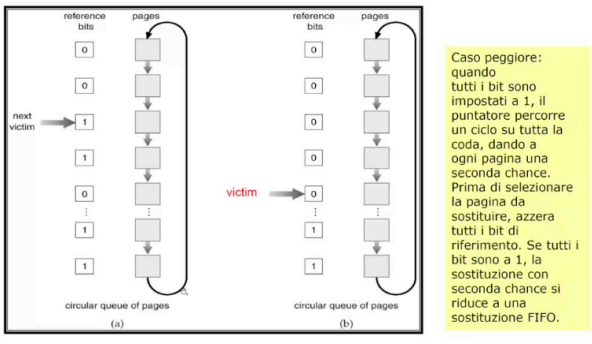
Immaginando una stringa di accessi in memoria, ipotizziamo che all'istante di tempo "a" lo stack contenesse i valori in figura (2,1,0,7,4). In cima c'è la pagina 2, in fondo quella letta più in là nel tempo. Se dopo l'istante "a" vado a leggere la pagina 7, lo stack "b" cambierà spostando la pagina 7 in cima, in quanto è stata l'ultima ad essere stata letta. Se, dopo quest'operazione, dovessi aver bisogno di eliminare una pagina, eliminerei la 4.

L'algoritmo LRU, per funzionare in modo efficiente, ha bisogno di un supporto hardware; tuttavia, quando non si ha a disposizione un hardware dedicato che permetta di seguire in modo efficiente l'algoritmo LRU, si usano degli algoritmi approssimati basati sull'uso di uno o più bit di riferimento, che permettano di capire in modo approssimato qual è la pagina da eliminare.
Tra gli algoritmi approssimati rientra l'algoritmo di seconda chance.

ALGORITMO DI SECONDA CHANCE

Ad ogni pagina viene associato un bit inizializzato a 0. Se una pagina:
È utilizzata→il relativo bit assume valore 1
È indicata dal puntatore:

- Se il suo bit=0→è eliminata
- Se il suo bit=1→ottiene una seconda chance, (prima che il puntatore passi alla successiva pagina, azzera il bit)



L'algoritmo se non esiste nessuna pagina con bit = 0 scandisce tutte le pagine, ad esempio in senso orario. Se trova tutti bit = 1 man mano che li scorre, li sostituisce con 0, senza eliminare la pagina. Prima o poi, dopo la sostituzione, si troverà una pagina con bit = 0 perché cambiando il numero dei bit dopo un primo giro quelli che prima avevano valore 1, in un secondo giro avranno valore 0. Per questo si chiama algoritmo di seconda chance.

Varianti di questo algoritmo:
ALGORITMO DI SECONDA CHANCE MIGLIORATO
Si usano due bit: il bit di riferimento e il bit di modifica che permettono di definire 4 classi:

- Se il bit di riferimento è pari a 0 e il bit di modifica è = 0→la pagina non è stata né usata di recente, né modificata di recente. Questa è la migliore scelta per la sostituzione.
- Se il bit di riferimento è pari a 0 e il bit di modifica è = 0→la pagina non è stata usata di recente ma è stata modificata; è una pagina candidata ad essere eliminata e quindi spostata su disco;
- Se il bit di riferimento è pari a 1 e il bit di modifica è = 0→la pagina è stata usata recentemente, ma non modificata. Ciò significa che probabilmente la pagina sarà riusata a breve e non conviene eliminarla.
- Se il bit di riferimento è pari a 1 e il bit di modifica è = 1→la pagina è stata sia usata che modificata recentemente e quindi non andrà sostituita.

In questo caso si mette unisce il concetto di dirty bit (citato nella scorsa lezione) con bit di riferimento.

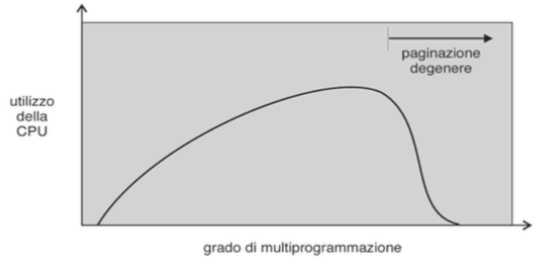
ALGORTIMO CON CONTEGGIO

Si tratta di algoritmi non usati frequentemente, basati sul conteggio dell'uso delle pagine. Ad ogni pagina viene associato un contatore che conta quante volte una pagina è stata riferita. Se si ha a disposizione un contatore per ogni pagina, è possibile utilizzare due algoritmi:
- **Algoritmo LFU (Least Frequently Used)**: sostituisce le pagine con il contatore minimo. L'idea è che una pagina usata attivamente deve avere un conteggio di riferimento alto. Si ha però un problema quando una pagina è usata molto intensamente durante la fase iniziale di un processo, ma poi non viene più usata; poiché è stata usata intensamente il suo conteggio è alto, quindi, rimane in memoria anche se non è più necessaria.
- **Algoritmo MFU (Most Frequently Used)**: sostituisce le pagine con il contatore massimo. È basato sull'idea che la pagina con il contatore minimo è stata inserita da poco e non è stata usata. E in questo caso avrebbe poco senso eliminare una pagina che probabilmente verrà usata successivamente.

THRASHING

Concetto molto importante. È intuitivo che se un processo non ha abbastanza pagine in memoria il tasso di page fault è alto. E questo crea:
- Bassa utilizzazione della CPU, cioè la CPU produce poco lavoro utile.
- Se la CPU lavora ad un basso rate di produttività, lo scheduler della CPU può credere che sia necessario aumentare il grado di multiprogrammazione.
- Un nuovo processo viene inserito in memoria.

Il fenomeno del Thrashing si manifesta quando un processo o un insieme di processi sono costantemente occupati a spostare pagine dal disco alla memoria e viceversa senza eseguire calcoli all'interno della CPU.



- **Modello di località** (set di pagine usate insieme)
  - Un processo di sposta da una località all'altra (da un gruppo di pagine ad un altro gruppo di pagine)
  - Le località si possono sovrapporre.
- Perché si verifica il thrashing ?
  - Σ spazi di località > spazio di memoria locale

Questo grafico mostra il fenomeno del thrashing in funzione del grado della multiprogrammazione.
Sull'asse delle x vi è il numero dei processi attivati dal sistema operativo.
Sull'asse delle y è riportato l'uso della CPU.
Per un certo periodo aumentare il grado della multiprogrammazione fa aumentare anche l'uso della CPU (prima parte della curva). Raggiunto il massimo la curva crolla a picco: quella fase è quella del thrashing (in italiano paginazione del genere). È il momento in cui la CPU passa il tempo a spostare pagine da una parte all'altra senza fare elaborazione.
Modello di località: set di pagine usate insieme per un processo; se un processo utilizzasse una pagina x, poi potrebbe usare la pagina x + 1 e così via. Quello che si verifica è che nell'esecuzione di un processo, il processo si sposta da un gruppo di pagine ad un altro, e questi gruppi si possono anche sovrapporre. Si verifica il thrashing quando la sommatoria delle pagine che fanno parte dello stesso insieme è maggiore dello spazio in memoria che è stato assegnato a quel processo.

Esempio
Se un processo ha a disposizione 4 frame per volta, ma il numero di pagine che fa parte del gruppo di pagine è 10, si genere il thrashing. Se, invece, ho a disposizione 4 frame e solo 2 pagine, il thrashing non si verifica.

SOSTITUZIONE LOCALE E GLOBALE

Per sostituzione globale s'intende che, quando il SO deve selezionare il frame da sostituire, lo va a trovare nell'insieme dei frame di tutti i processi. Quindi se il processo P1 ha bisogno di spazio, il SO può andare a togliere spazio al processo P2 e assegnarlo a P1. Questo significa che il processo non può controllare la propria frequenza di page fault. Il tempo di esecuzione di un processo può variare in modo significativo non solamente in base ai tempi propri, ma anche in base ai tempi del SO.
La sostituzione locale, invece, prevede che se un processo ha bisogno di una pagina il frame necessario per mettere questa pagina si ricava dai frame del processo stesso; non può capitare, dunque, che un processo chieda memoria ad un altro processo.
La sostituzione che viene maggiormente utilizzata è quella globale in quanto garantisce maggior throughput e viene normalmente implementate nei SO più diffusi.

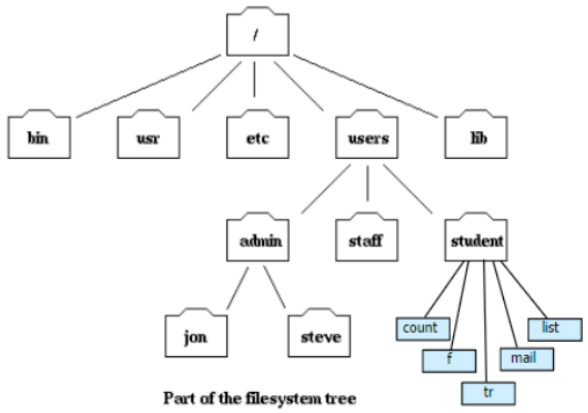
Esempio di Windows di approccio alla gestione della memoria
Ai processi viene assegnata una coppia di valori: un working set minimo e uno massimo.
Working set minimo è il numero di pagine che il processo avrà garantite in memoria.

Working set massimo è il numero di pagine massimo che il processo avrà.
Nel tempo l'ammontare complessivo della memoria del sistema potrebbe diminuire perché magari sono stati avviati troppi processi: i valori stabiliti precedentemente potrebbero cambiare.
Esiste un processo, detto automatic working set trimming che prevede che, quando la memoria libera del sistema scende sotto una certa soglia, questo processo si occupa di aumentare la dimensione della memoria libera rimuovendo dai processi di esecuzione le pagine che sono in eccesso rispetto al working set minimo.

Interfaccia del file system

Per la maggior parte degli utenti, il file system è l'aspetto più visibile di un sistema operativo. Esso fornisce il meccanismo per la memorizzazione di dati e programmi.
Come si può intuire, il file system si basa sul concetto di file, definito come un insieme di informazioni correlate, a cui è stato assegnato un nome, e memorizzate su diversi supporti, quali dischi, nastri magnetici e dischi ottici.
Dal punto di vista dell'utente, un file è la più piccola porzione di memoria logica secondaria; cioè, i dati si possono scrivere in memoria secondaria soltanto all'interno di un file.
I file si possono distinguere in file binari (rappresentano programmi, immagini, ecc.) e testuali (rappresentano testo).
Un file ha una struttura definita dalla tipologia a cui appartiene. Ad esempio:

- Un **file di testo** è formato da una sequenza di caratteri organizzati in righe, ed eventualmente pagine;
- Un **file sorgente** è formato da una sequenza di funzioni, ciascuna delle quali è a sua volta organizzata in dichiarazioni, seguite da istruzioni eseguibili;
- Un **file eseguibile** consiste di una serie di sezioni di codice che il caricatore può caricare in memoria ed eseguire (es. i programmi in cui è presente il risultato della compilazione dei file sorgente e che producono un codice eseguibile o interpretabile direttamente da un computer o macchina virtuale).



Part of the filesystem tree

I file system moderni sono generalmente organizzati in modo gerarchico (ordine che ha valenza unicamente da un punto di vista logico ed è rappresentato tramite grafi), ottenendo i cosiddetti HFS (Hierarchical File System), basati sull'uso di directory.
Nell'immagine a destra in alto si vede la directory principale, che si suddivide in cartelle, le quali possono contenere file od ulteriori cartelle.
È importante sottolineare che l'organizzazione gerarchica non ha alcun legame con la posizione fisica effettiva dei file in memoria di massa (es. non è detto che i file della cartella "student" siano posti nel disco gli uni vicini agli altri).

Il file system è responsabile di organizzare i file in directory (o cartelle) e di fornire all'utente tutte le funzioni di alto livello (es. system call, comandi dell'interfaccia del file system, ecc.) che permettono di operare su tali file.
Ogni file è caratterizzato da una serie di attributi:

- Nome: nome simbolico (es: lettera.doc, libro.pdf, sort.java; l'estensione non è obbligatoria).
- Tipo: identifica il tipo di file (es. file di testo, java, python, ecc.).
- Locazione: indirizzo del file sul dispositivo fisico.
- Dimensione: numero di byte, word o blocchi.
- Protezione: bit di lettura, scrittura, esecuzione.
- Ora, data e id utente: creazione, ultima modifica, ultimo accesso; usate per protezione, sicurezza, e monitoring.

Gli attributi di ciascun file sono contenuti nella directory di appartenenza, quindi, si può definire la directory come un file contenente informazione (ossia gli attributi) di altri file.
Le operazioni che si possono eseguire su un file sono:

- Creazione; il SO deve:
    - Reperire lo spazio per memorizzare il file all'interno del file system
    - Creare un nuovo elemento nella directory in cui registrare i metadati, ovvero nome del file, posizione nel file system, altre informazioni
- Scrittura; sono necessari:
    - 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
    - Di solito si mantiene un solo puntatore alla posizione corrente nel file; quindi, prima di effettuare un'altra operazione in una posizione diversa, bisogna muovere il puntatore
- Lettura; sono necessari:
    - 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
    - Di solito si mantiene un solo puntatore alla posizione corrente nel file; quindi, prima di effettuare un'altra operazione in una posizione diversa, bisogna muovere il puntatore
- Copia; ciò che avviene è la:
    - Creazione di un nuovo file
    - Lettura dal file da copiare
    - Scrittura nel nuovo file
- Ri-posizionamento nel file; sono necessari:
    - Reperimento del file nel file system
    - Aggiornamento del puntatore alla posizione corrente
    - Nessuna operazione di I/O
- Cancellazione; si verifica il:
    - Reperimento del file nel file system

    - Rilascio dello spazio allocato al file e si elimina il corrispondente elemento della directory (il file non è completamente eliminato finché non viene sovrascritto, caratteristica particolarmente vantaggiosa nella computer forensic)
- Troncamento, ossia eliminare una parte del file stesso; consiste nel:
    - Cancellazione del contenuto del file, che mantiene immutati tutti gli attributi (esclusa la dimensione)
    - Rilascio lo spazio allocato al file
- Lettura/impostazione degli attributi; prevede il:
    - Reperimento/aggiornamento dei metadati (attributi) del relativo elemento di directory

Sono dette operazioni di lettura tutte quelle operazioni che permettono l'accesso alla struttura della directory e la modifica di un suo elemento. Il SO mantiene in memoria centrale una tabella contenente informazioni su tutti i file aperti, detta la tabella dei file aperti, che è costantemente aggiornata ed utilizzata come tool per accedere velocemente ai file in uso.
Le system call per aprire e chiudere i file sono:
- **open (Fi)**: ricerca nella struttura della directory sul disco dell'elemento Fi e copia del contenuto nella tabella dei file aperti (in memoria centrale); si riporta un puntatore all'elemento nella tabella
- **close (Fi)**: copia del contenuto dell'elemento Fi, attualmente residente in memoria principale,
  nella struttura di directory sul disco e successiva rimozione.

Le system call che "lavorano" su file chiusi sono esclusivamente create () e delete ().
Nei sistemi multiprocessing, per garantire una miglior efficienza, sono state creati due livelli di tabelle:
- Tabella di sistema: riferimento a tutti i file aperti nel sistema o Posizione del file nel disco
    - Dimensione del file
    - Date di ultimo accesso/ultima modifica
    - Contatore di aperture
- Tabella associata al processo: riferimento a tutti i file aperti per ciascun processo
    - Puntatore alla posizione corrente nel file
    - Diritti di accesso
    - Puntatore al relativo elemento contenuto nella tabella di sistema

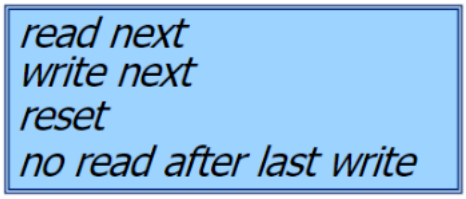
La tabella dei file aperti è dotata di:
- **Contatore di aperture**: conta il numero di processi che hanno aperto il file, per rimuovere opportunamente i dati dalla tabella dei file aperti alla chiusura del file da parte dell'ultimo processo (contenuto nella tabella di sistema)
- **Locazione del file 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)

L’accesso ad un file può essere shared (condiviso tra più processi) od exclusive (si acquisisce il lock).
Il lock esclusivo può essere:

- **Obbligatorio**: l'accesso al file viene negato se il lock è già stato acquisito da altro processo; è il SO a garantire l’esclusività nell’accesso al file (Windows)
- **Consigliato**: i processi trovano che lo stato di un dato file è "bloccato" (il lock viene percepito come un warning) e decidono sul da farsi; è il processo a decidere autonomamente sull’esclusività nell’accesso al file (UNIX)

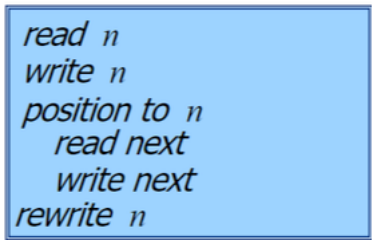
Se il lock è obbligatorio, il SO assicura l'integrità dei dati soggetti a lock; se il lock è consigliato, è compito del programmatore garantire la corretta acquisizione e cessione del lock
Un concetto fondamentale nella gestione dei file è la differenza tra:

- Accesso sequenziale: è un modello di accesso a file che si "ispira" al nastro poiché, per arrivare alla posizione x, si devono leggere prima tutte le posizioni da 0 a x-1; utilizzato da compilatori e editor.

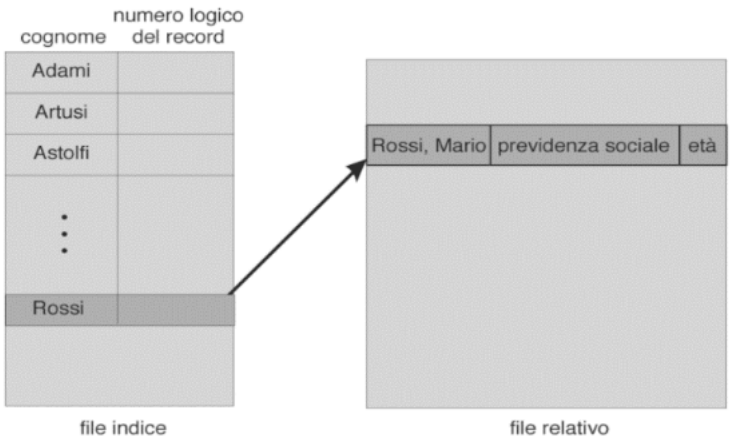


Caratteristiche:
•Impossibilità di lettura oltre l'ultima posizione scritta;
•la scrittura aggiunge informazioni in coda al file (append)
•uso di un puntatore

- Accesso diretto (o casuale): è un modello di accesso a file che si "ispira" al disco per cui si può andare alla posizione x, senza che vi sia bisogno di ulteriori passaggi precedenti.(n indica la posizione n-esima)
  Caratteristiche:



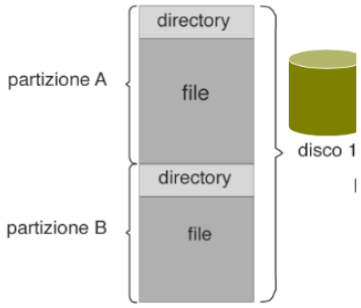
•I record devono essere di lunghezza costante L, decisa dal creatore del file
•Per accedere all’n-esimo record, si calcola L x n che fornisce la posizione del byte d’inizio record



Una sottocategoria dell’accesso diretto è l’accesso diretto con indice, nel quale viene associato ad ogni file un file “indice”, e che utilizza una chiave di ricerca ed un puntatore, il quale indica la posizione del record d’interesse.

Struttura delle directory

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.
I moderni file system si basano sulle partizioni (o volumi).
Le directory contengono le seguenti informazioni:



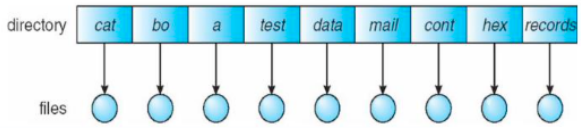
- Nome
- Tipo
- Indirizzo
- Lunghezza attuale
- Lunghezza massima
- Data ultimo accesso
- Data ultima modifica
- ID del proprietario (del gruppo)
- Informazioni di protezione

Inoltre, sulle directory si possono effettuare le seguenti operazioni:

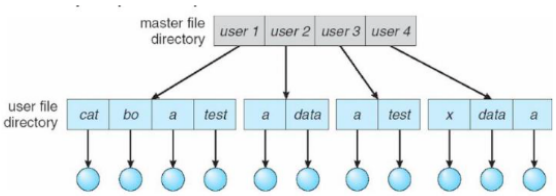
- Ricerca di un file
  - Possibilità di scorrere la directory per reperire l'elemento associato ad un particolare file
  - I file hanno nomi simbolici
  - Nomi simili possono corrispondere a relazioni logiche fra i contenuti dei file
  - Capacità di reperire tutti i file il cui nome soddisfi una particolare espressione (es. il comando “ls – l pr*. *” lista tutti i file di qualsiasi estensione (*. *) che inizino con “pr”)
- Creazione di un file
  - Aggiunta del record descrittivo del file alla directory
- Cancellazione di un file
  - Rimozione del record descrittivo del file dalla directory
- 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
- Ridenominazione di un 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
- Attraversamento del file system (se ci si trova in una directory, attraversarla vuol dire entrare in una sottodirectory contenuta in essa)
  - Possibilità di accedere ad ogni directory e ad ogni file in essa contenuto

La struttura delle directory deve garantire:

- Efficienza: reperire rapidamente i file di interesse
- Nominazione (o naming): permette agli utenti di dare un significato ai file; non vi è conflitto se due file hanno lo stesso nome ma si trovano su directory differenti (possibile se, per esempio, il computer è usato da più utenti)
- Grouping: raggruppare file sulla base della loro tipologia

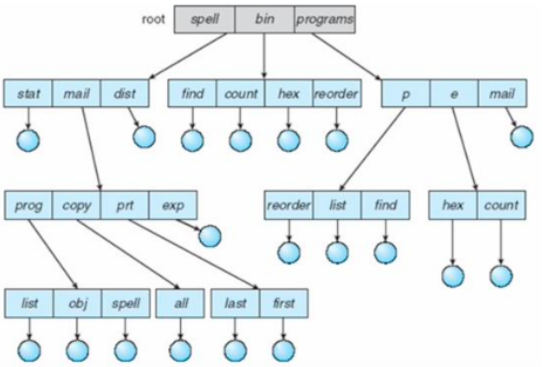


È necessario distinguere tra la struttura fisica e logica di una directory; la prima fa riferimento al modo con cui le informazioni sono messe sul disco mentre la seconda indica il modo con cui l'utente visualizza le informazioni, per cui è possibile avere viste fisiche diverse per una stessa vista logica.
Come già detto, nei sistemi moderni le directory sonorappresentate come dei grafi ma in passato si faceva uso, invece, delle directory monolivello, ovvero cartelle in cui non era possibile creare delle sottodirectory. Questa modello causa problemi di nominazione, in quanto ogni file deve avere un nome diverso dagli altri, e non permette alcun raggruppamento logico.



Un avanzamento rispetto alla directory monolivello è la directory a due livelli, in cui ciascun utente può avere una propria directory; in questomodello sono ammessi nomi uguali per file diutenti diversi e si riesce a ottenere una ricerca più efficiente, grazie all'introduzione del path name, ma ancora non è prevista alcuna capacità di raggruppamento logico (se non in base ai proprietari).
Nella directory di secondo livello, quando si apre una 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).
Per riferirsi a file di altri utenti, se l'accesso è autorizzato, ogni utente deve utilizzare il path name 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.
In ultimo, è stata introdotta la directory con struttura ad albero, in cui è possibile una ricerca efficiente, si ha capacità di raggruppamento logico e viene introdotto il concetto di directory corrente, sui cui l'utente lavora a meno che non specifichi il percorso di un'altra directory.

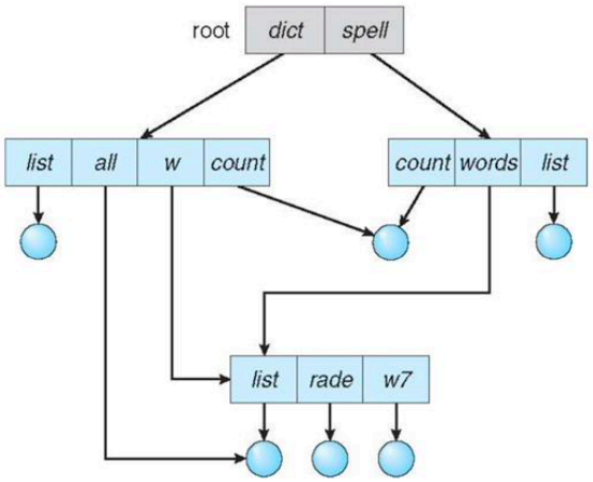
DIRECTORY CON STRUTTURA AD ALBERO



Nella rappresentazione del file system ad albero si ha una radice (in inglese root) da cui partono delle sottodirectory che a loro volta si ramificano in altre sottodirectory che contengono al loro interno ulteriori directory e file (pallini). Questo modo di organizzare i file risulta essere molto più pratico perché permette di organizzare i file secondo raggruppamento logico e supporta anche la ricerca efficiente dei file. Nei file system organizzati ad albero esiste il concetto di percorso (o path name) assoluto o relativo.

PATH NAME ASSOLUTO: indica il nome del file a partire dalla radice del file system, quindi dal percorso iniziale.
PATH NAME RELATIVO: indica il nome del file solamente all'interno della directory di lavoro (quindi quella corrente).
Se si vuole creare una directory su Unix con il comando MK DIR, si crea una directory nella directory corrente e se si vuole crearla in un percorso assoluto andrebbe specifica con MK DIR/ e il nome del percorso.
Le operazioni opposte a quelle di creazione sono quelle di cancellazione. Se si vuole eliminare una directory, bisogna ricordare che in sistemi MS – DOS quest'operazione è possibile solo se la directory è vuota; in UNIX/Linux l'operazione è possibile anche se vi sono file o sottodirectory. Si usa il comando rm (remove) – r.
Esistono anche file system organizzati sotto forma di grafo.

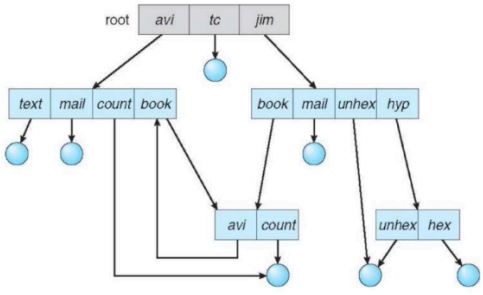
FILE SYSTEM SOTTO FORMA DI GRAFO



Un albero si trasforma in un grafo aciclico nel momento in cui vi sono dei file o sottodirectory condivisi da più directory. Esistono due sottodirectory "count" nell'immagine, una contenuta nella directory spell ed un'altra in quella dict. Il file condiviso dalle due sottodirectory "count" è sempre uno ma può essere raggiunto da due percorsi diversi. Questo vale anche per le sottodirectory list, rade o W7.
Il meccanismo che permette di dare al percorso del file più nomi, è detto link (collegamenti).
I collegamenti possono essere di due tipi:

- **soft link:** si ottiene creando un puntatore alla risorsa che già esiste;
- **hard link:** si ottiene quando la directory si duplica.

Il grafico, tuttavia, è aciclico: si seguono le frecce, ma non si ritorna mai ad un punto di partenza. A volte è possibile avere directory a grafo generale, ovvero con possibile presenza di cicli interni (da directory inferiori ad una progenitrice).



SAPPI CHE: È preferibile utilizzare i grafi aciclici poiché garantiscono una semplicità degli algoritmi necessari per attraversarlo. Sono, infatti, ammessi link solo a file e non a sottodirectory e nel caso in cui si verifichi la presenza di cicli in sottodirectory, si risolve mediante l'uso di un algoritmo di rilevamento (estremamente dispendioso).

PROTEZIONE DEI DATI

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.
La protezione si ottiene mediante il controllo degli accessi, ovvero osservando:
1. chi può accedere al file;
2. quali tipi di accesso siano leciti.

Le modalità di accesso dipendenti dall'identità dell'utente sono: lettura, scrittura, esecuzione, ma anche append e cancellazione (forme di scrittura). Anche l'accesso ai metadati è un'ulteriore modalità che sfrutta elencazione del nome e degli attributi.
Nei sistemi operativi più moderni (UNIX e WINDOWS) ai file sono associate liste 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.
Gli utenti si raggruppano in tre classi distinte [proprietario, gruppo di appartenenza e universo (il resto)], ognuna con le proprie modalità di accesso e con i propri diritti per utilizzare il file stesso. Si possono associare anche delle password ai file di utilizzo.

SAPPI CHE: Cliccando il tasto destro del mouse si denotano le proprietà di un qualsiasi file, i cui diritti possono essere cambiati solo dal proprietario.
Mentre in Windows, il concetto di ACL è recente, in UNIX la prima versione di ACL compare nel 2008 con diversi livelli di diritti (principalmente tre: read, write ed execute) e tre classi di utenti (user, group e all o others).
Il legittimo proprietario, nell'esempio riportato sopra, può leggere, scrivere ed eseguire il file, gli utenti del gruppo possono leggere e scrivere il file, mentre gli altri utenti possono solamente leggere il file game.c. [chmod 761 game]

CONDIVISIONE DI FILE

La condivisione di file (file sharing) è particolarmente utile nei sistemi multiutenti. 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.
Altri modi per accedere a file che si trovano su computer remoti è l'FTP, ovvero il trasferimento richiesto esplicitamente; vi è un server su cui stanno i file e poi dei client che si collegano al server per scaricare file o caricare (modello client-server). Ad oggi poco utilizzato, in quanto si usa il modulo HTTP (del web) molto più server friendly.
Nel modello client-server si rispecchia un modello molti-a-molti e la sicurezza si basa su username e password o su chiave crittografiche rilasciate ad utenti autorizzati.
In generale un file system distribuito (DFS) è un sistema che permette di condividere file tra macchine tra loro remote, in modo da far sembrare all'utente che quei file sono localmente presenti anche se sono su dischi remoti. Sono molto diffusi sistemi di condivisione file basati su Cloud, come Google Drive o DropBox.

MALFUNZIONAMENTI

I file system sono soggetti a malfunzionamenti dovuti a:
- problemi hardware dei dischi che li contengono;
- alterazione dei metadati (di accesso ai dati);
- problemi ai dischi o agli adattatori o ai cavi di connessione; - errori umani.

In un contesto di rete, i malfunzionamenti possono essere dovuti anche a problemi nella modalità di accesso o mancanza di rete.

UNIX

I sistemi della famiglia UNIX hanno struttura a grafo generale (con cicli) in cui i file e le directory possono appartenere a più directory, ospitando programmi di sistema specifici, come per esempio: - dev, contiene i device driver;
- bin, il codice eseguibile;
- include, le librerie di sistema;
  - etc, i file di configurazione etc.

  Vi è anche la cartella home directory, ovvero sottodirectory della user o della home e l'utente proprietario all'interno della home directory può creare ciò che desidera e tutti i file che crea appartengono di default all'utente stesso, che può alterare la proprietà del proprio file facendo accedere utenti esterni alla propria home directory.

IMPLEMENTAZIONE DEI FILE SYSTEM

L'implementazione del file system si occupa dei meccanismi realizzativi, ovvero in che modo la memoria secondaria (il disco) è utilizzata per memorizzare i dati presenti nel file.
Due sono le caratteristiche importanti condivise da tutte le tipologie di dischi, nonostante la differente tecnologia:
- I dischi possono essere riscritti: si può leggere un blocco dal disco, modificarlo e quindi scriverlo nella stessa posizione;
- È possibile accedere direttamente a qualsiasi blocco di informazioni del disco; quindi, risulta semplice accedere a qualsiasi file, sia in modo sequenziale, sia in modo diretto, e passare da un file all'altro.

Tali proprietà escludono, per esempio, alcune tipologie di sistemi di storage come i nastri magnetici che non supportano l'accesso diretto.
Per migliorare l'efficienza dell'I/O, i trasferimenti tra memoria centrale e dischi si eseguono per blocchi di dati, costituiti da uno o più settori di un disco. Ciò dipende dalla tecnologia e, a seconda del tipo di disco, la dimensione dei settori è compresa tra 32 e 4096 byte, ma di solito è pari a 512 byte.
Lo scopo del file system è di memorizzare, individuare e recuperare facilmente i dati presenti su un disco.
L'implementazione si occupa dunque di prendere da un file i blocchi di dati corrispondenti.
Il file system presenta due problemi di progettazione molto diversi:

- Il primo riguarda il fornire l'interfaccia del file system, ovvero la definizione di un file e dei suoi attributi e le funzioni di accesso;
- Il secondo riguarda la creazione di algoritmi e strutture dati che permettano di far corrispondere il file system logico ai dispositivi fisici di memoria secondaria (implementazione del file system).

Il file system è organizzato a livelli funzionali a partire dai programmi applicativi fino ad arrivare ai dispositivi.

Le applicazioni interagiscono con il file system logico, che presenta il file system come un'unica struttura (albero o grafo) con tutte le funzionalità di alto livello. Inoltre, il file system logico mantiene le strutture di file tramite i file control block, FCB, conservando le informazioni e proprietà dei file, gestendo directory, protezione e sicurezza.
Il file system logico si basa a sua volta sul modulo di organizzazione dei file, che controlla l'allocazione dei blocchi fisici sul disco e la loro allocazione con quelli logici. È come se fornisse una traduzione dagli indirizzi logici ai fisici e viceversa. Continente, inoltre, il modulo per la gestione dello spazio libero, tenendo traccia delle parti libere del disco per memorizzare nuovi file.

Al livello più basso abbiamo il file system di base che usa i driver per accedere ai blocchi fisici sul dispositivo e, infine, ci sono i driver dei dispositivi, cioè i programmi di controllo dell'hardware che permettono di effettuare le operazioni di lettura e scrittura e, dunque, si occupano del controllo I/O. A livello fisico vi sono i dispositivi, sostanzialmente dischi e nastri.

CONTROLLLO DELL'I/O- DRIVER DEI DISPOTIVI

Traducono comandi di alto livello ("recupera il blocco fisico 123") in sequenze di bit che guidano l'hardware di I/O a compiere una specifica operazione in una data locazione. I driver scrivono specifiche configurazioni di bit di controllo in specifiche locazioni della memoria del controllore.

FILE SYSTEM DI BASE

Esso invia comandi generici al driver del dispositivo. Ad esempio, un comando in presenza di più unità può essere "leggi i dati dall'unità 1, cilindro 73, traccia 10 ecc.". Il file system vede, legge e scrive blocchi fisici su disco in programmi di più basso livello per il driver. Inoltre, gestisce il buffer del dispositivo, sostanzialmente la memoria locale del dispositivo usata per disaccoppiare la velocità delle richieste dalla velocità dei dati che vengono restituiti. I dati vengono bufferizzati completamente per permettere al disco di rileggerli con più calma.

RICORDA CHE: I buffer servono per disaccoppiare la velocità di produzione dalla velocità di consumo dell'informazione.
Il file system viene organizzato a strati per ridurre la complessità delle funzioni e specializzare tali funzioni su diversi livelli, avendo come aspetto negativo un overhead, ovvero ogni operazione di alto livello prima di essere implementata sul disco deve passare su una serie di strati che impiega tempo per elaborare la richiesta, producendo una somma di costi in msec, comunque accettabile rispetto ai vantaggi che il file system produce.

TIPI DI FILE SYSTEM

Esistono diverse tipologie di file system e non è raro che i sistemi operativi ne prevedano più di uno. Tra questi troviamo:
- CD-ROM;
- UNIX: UFS e FFS;
- Windows: FAT, FAT32 e NTFS;
- Linux: ext (exented file system) e xfs;
- MAC: APFS (Apple file system) e macOS riesce anche a supportare file system della famiglia Windows.

Ad oggi sono stati inventati dei file system distribuiti, ad esempio Google ha inventato i GoogleFS, detto anche BigFiles, che serve per memorizzare file di grandissime dimensioni. Questi sono approcci diversi da quelli convenzionali in cui si cerca di definire tecniche per gestire file estremamente grandi, non trattabili in maniera efficiente dai computer di tutti i giorni.
Anche Oracle SAM ha creato file system ad hoc.