LEZIONE 11

Interfaccia file system

Concetto di file

Un file è un insieme di informazioni coerente, correlate e registrate nella memoria secondaria, cui è stato assegnato un nome.

Il nome del file serve al sistema per stabilire che tipo di file si tratta.

* Alcuni tipi di file : Dati, numerici, alfabetici, alfanumerici, binari, programmi.
* Molti tipi di file sono dei **contenitori,** ad esempio mkw che contiene il filmato, i sottotitoli ecc, oppure word, che non è un file di testo, ma bensì un contenitore, infatti in un file word ad esempio ci si può mettere un’immagine (che sono tipo jpg ad es.)

Struttura dei file

Ci sono diversi tipi di strutture:

* **Nessuna:** sequenza di parole, byte. Bisogna essere però cauti nel dire ciò in quanto potrebbe esserci una struttura che è nascosta.
* **Strutture semplici:**
* Righe: un esempio è **il log file,** ovvero un tipo di file che serve ad accumulare gli eventi che si succedono all’interno di un sistema, perciò è molto importante per il tuning in quanto, nel momento in cui c’è un errore, si può risalire agli eventi che hanno condotto a tale errore.
* Lunghezza fissa, e lunghezza variabile
* **Strutture complesse:**

Formato da eventi che lo caratterizzano, da aggregazioni di informazioni. Possono essere:

* Documento formattato
* HTML
* Relocatable load file ecc.

Attributi dei file

Un file non ha degli attributi assoluti, in quanto dipendono molto dal sistema ospite.

**Le informazioni sui file sono conservate nella struttura di directory (la cartella),** che risiede a sua volta nella memoria secondaria.

* **Nome:** è l’unica informazione in forma umanamente leggibile.
* **Tipo:** informazione necessaria ai sistemi che gestiscono tipi di file diversi.
* **Locazione:** puntatore al dispositivo e alla locazione del file in tale dispositivo.
* **Dimensione:** dimensione corrente del file.
* **Protezione:** controlla chi può leggere, scrivere o far eseguire il file.
* **Ora, data e identificazione dell’utente:** dati utili ai fini della protezione e del controllo di utilizzo.

Operatori sui file

Anche le operazioni dipendono molto dal sistema ospite.

* **Creazione/ cancellazione** di un file
* **Scrittura/ lettura** di un file
* **Riposizionamento** di un file
* **Troncamento** di un file
* **Open (Fi):** ricerca nella directory del disco l’elemento Fi, e ne sposta il contenuto in memoria. Vuol dire che copia il file in memoria centrale, però solo quello che serve, ovvero ciò che deve essere effettivamente eseguito. Si apre tutto il file, ma per il caricamento è diverso, dato che è un oggetto che si può caricare parzialmente, e questo consente di avere solo una parte del codice (quella più attiva).
* **close (Fi):** rimuove il contenuto dell’elemento Fi dalla memoria centrale alla directory del disco.

Metodi d’accesso

Immagine che contiene screenshot

Descrizione generata automaticamente2 tipi di metodi più importanti:

**Accesso sequenziale:** è un tipo di accesso dove si può leggere solo in una determinata posizione. Questo accesso nacque dall’esigenza di leggere sul nastro. Il nastro magnetico era una striscia di plastica su cui c’era una sostanza ferromagnetica. Tramite la corrente di bias (polarizzazione), venivano movimentate tali particelle ferromagnetiche, le quali erano disposte in maniera particolare che consentiva di immagazzinare informazioni. Sequenziale perché si accede ai dati in sequenza mediante operazioni di lettura, scrittura e reset. Infatti **read next e write next** stanno a significare che nel momento in cui ci si trova con la testina su tale nastro (come estrazione), si può solo (a seconda di ciò che si deve fare ovviamente) leggere il prossimo, o scrivere sul prossimo.

**Accesso diretto (o casuale):** a differenza di quello sequenziale ci sta **read n e write n,** che indicano un blocco

* C’è la possibilità di emulare gli accessi sequenziali con accesso diretto e viceversa (ovviamente emulare un accesso diretto con quello sequenziale è molto più complesso)

esempio emulatore accesso sequenziale con accesso diretto. CP è un puntatore e significa current position.

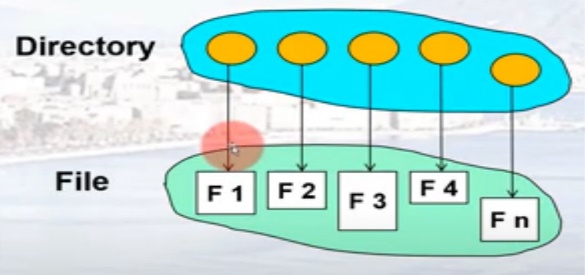
Esempio di indice e relativo file

Immagine che contiene screenshot

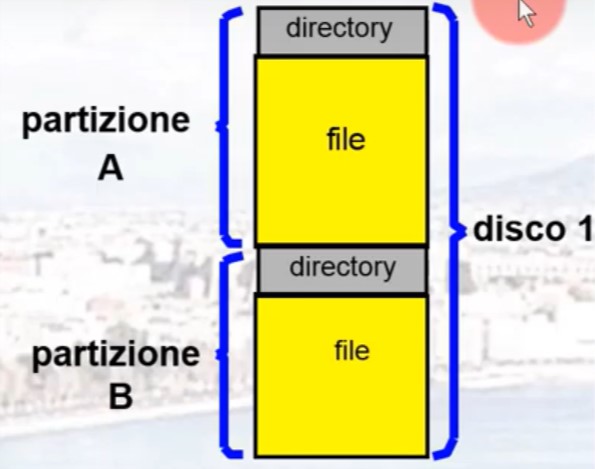
Descrizione generata automaticamente**spiegazione**

bisogna vedere come un file venga memorizzato nella cartella. Si deve immaginare di avere un file indice, che rimanda ai file veri e propri. Quindi da un punto di vista concettuale, si stanno identificando 2 entità: **file indice e file relativo.** Il puntatore deve puntare dalla chiave a quel file relativo (dove sta la freccia). prima di ipotizzare i modi per un accesso del genere, bisogna pensare alla struttura della directory.

**Struttura directory**

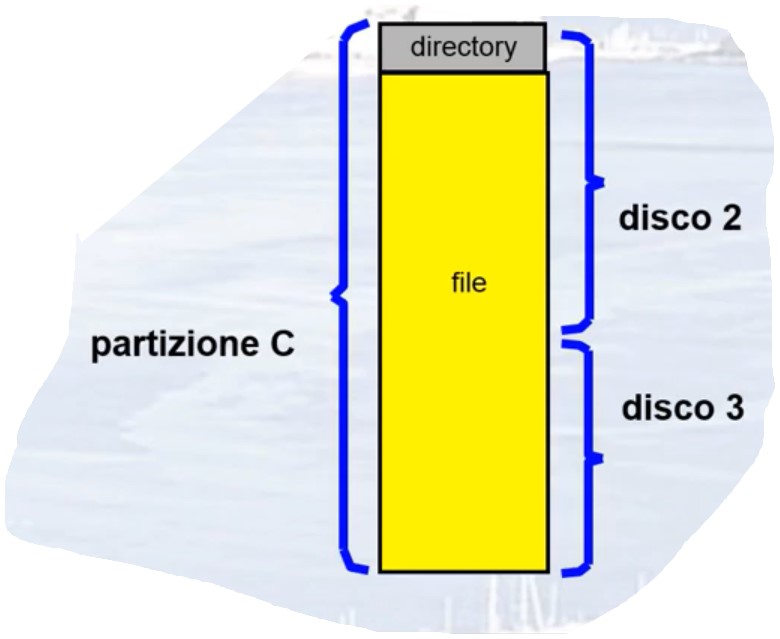
**DIRECTORY:** è un insieme che contiene una serie di elementi, i quali ci permettono di puntare ai file reali. Quindi la directory è solo un elenco di file, ma non contiene i file reali. Tale insieme di file (F1, F2,ecc), non è unico in quanto ci potrebbe essere una copia di backup.

Sia la directory che i file risiedono sul disco. Le copie di backup di queste due strutture vengono generalmente archiviate su nastro

Tipica organizzazione di un file system

Un file system è contenuto all’interno di un’entità detta disco. Il disco può essere diviso in aree molto diverse tra loro, r proprio per questa diversità, si potrebbero chiamare vere e proprie **partizioni,** che costringono ad una visione completamente separata e astratta; cioè non è detto che perché c’è un solo disco, si debba realizzare una raccolta unica, ma si potrebbero creare anche gruppi diversi (tipo partizioni windows e linux), e sarà il SO che permette all’utente di vederle disgiunte anche se sono sullo stesso disco.

Si può fare anche l’inverso, ovvero lo stesso software permette di vedere altri 2 dischi come un pezzo unico così:

**il vantaggio sta nell’avere una visione più uniforme e anche le primitive diventano più semplici da gestire.**

Informazioni sui file nella directory

* Nome
* Tipo
* Indirizzo
* Dimensione corrente
* Dimensione massima
* Data dall’ultimo accesso
* Data dell’ultimo aggiornamento
* ID del proprietario
* Informazioni di protezione

Operazioni eseguite su una directory

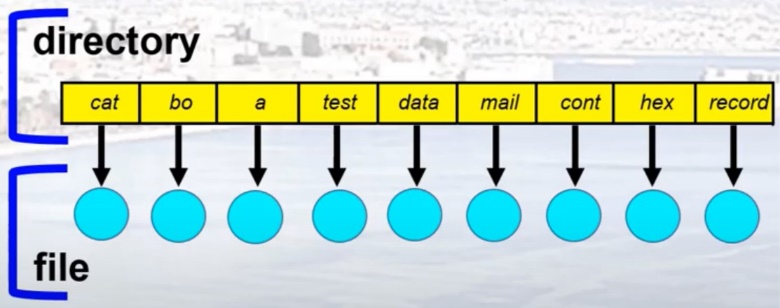
* Ricerca di un file
* Creazione/ cancellazione di un file
* Elencazione di una directory
* Ridenominazione di un file
* **Attraversamento del file system:** così come si fa il browsing del web, bisogna predisporre il SO a fare il browsing delle sue cartelle in maniera agevole. Ovviamente ci sono alcune cartelle alla quale nemmeno il proprietario può accedere, perché il SO data la delicatezza di alcune cartelle, come aree di swap, cartelle di sistema ecc, non consente l’accesso a nessuno.

Organizzare (logicamente) una directory per ottenere:

* **Efficienza:** individuare rapidamente un file
* **Scelta dei nomi:** conveniente per gli utenti
* Due utenti possono usare lo stesso nome per file differenti
* Lo stesso file può avere più nomi diversi tra loro
* **Raggruppamento:** raggruppamento logico dei file in base a specifiche caratteristiche (es. tutti i programmi java, tutti i giochi, ecc)

**Tutto questo lo si può realizzare organizzando le cartelle. Ci sono 3 modi principali per organizzare le cartelle:**

1. **Directory a livello singolo**

**spiegazione**

si immagini una cartella che contenga al suo interno tutti i file. Ognuno di essi avrà un puntatore che punterà ai file dati veri e propri.

**Svantaggi:** avere una singola directory per tutti gli utenti può creare confusione nei nomi dei file e problemi di raggruppamento. Questo svantaggio è però solo per l’utente (dato che l’essere umano non è abituato a gestire troppi file assieme, ma per il sistema è diverso, dato che per lui è indifferente avere una o più cartelle, anzi, dal punto di vista del sistema, con una configurazione del genere è più facile trovare file, essendoci un’unica cartella.

1. **Struttura di directory a due livelli**

**Immagine che contiene erba, orologio

Descrizione generata automaticamenteSpiegazione**

Ogni utente ha la propria cartella, quindi accede ai propri file, quindi ci possono essere pure file con lo stesso nome. In questo modo si avrà una ricerca più efficiente, un management più efficiente.

**Svantaggio:** questo tipo di gestione impedisce il raggruppamento (non ci possono essere sottocartelle).

1. **Struttura di directory ad albero**

**Immagine che contiene disegnando

Descrizione generata automaticamentespiegazione**

ci sta un nodo radice, dal quale si sviluppa tutto il sistema di file. Dalla root si può puntare a delle sottocartelle, che a loro volta possono puntare ad altre sottocartelle, fino ad arrivare alle foglie, ovvero i file veri e propri. Ogni cartella può avere al suo interno o altre cartelle o i file stessi.

* Si usa una struttura ad albero poiché è versatile, facile da implementare, **ma il vero motivo è che si ha la possibilità di navigare dalla root alle foglie usando un numero logaritmico di passi, rispetto al numero complessivo di foglie** (es. ci sono un milione di passi, e si può fare il log del numero dio nodi per arrivare alle foglie dalla root).
* **Questo è la struttura che risulta essere più efficiente proprio perché la discesa dalla root alle foglie è logaritmica,** anche perché è possibile il raggruppamento
* **Directory corrente:** quella che contiene la maggior parte dei file di interesse corrente; ovvero quello nella quale si è posizionati.
* Quando si parla di struttura ad albero, bisogna parlare anche di **percorsi.**
* **Percorsi file:** sono i passi che bisogna svolgere per arrivare al file desiderato.

Essi possono essere:

* **Percorsi assoluti:** si stanno contando i passi partendo dalla radice
* **Percorsi relativi:** si dà il percorso partendo dalla directory corrente.
* Le operazioni su file sono:
* Cancellazione di un file

**rm**<nome\_file>

* La creazione di una nuova sottodirectory avviene nella directory corrente

**mkdir**<nome\_dir>

**cancellando la directory “padre”, si cancella anche tutto ciò che è al suo interno**

Struttura di directory a grafo aciclico

Immagine che contiene testo

Descrizione generata automaticamenteconsente alle directory di avere sottodirectory e file condivisi

**spiegazione**

Il grafo aciclico è un grafo che partendo da un nodo, non si a la possibilità di creare un ciclo che riporti a quel nodo. Infatti in questo caso partendo da words, si riesce a vedere un ciclo, ma non permette di tornare a words (le frecce sono tutte verso il basso). In questo modo è possibile avere cartelle condivise tipo in questo caso list, la quale è puntata sia da w che da words, quindi:

* Nomi diversi possono riferirsi allo stesso file (aliasing)
* C’è però anche un problema, ovvero, se a ogni operazione di cancellazione segue l’immediata rimozione del file, potrebbero rimanere puntatori a file che ormai non esistono più.
* Soluzioni:
* Cercare tutti questi collegamenti e rimuoverli (backpoiters)
* Conservazione dei file fino a che non siano stati cancellati tutti i riferimenti a esso.

Struttura di directory a grafo ciclico

Immagine che contiene segnale

Descrizione generata automaticamente**Spiegazione**

* Come è possibile garantire l’assenza di cicli?
* Consentire collegamenti solo ai file, non alle sottodirectory
* Utilizzare un metodo di “ripulitura” (garbage collection)
* Ogni volta che si aggiunge un nuovo collegamento, utilizzare un semplice algoritmo di individuazione di cicli per verificare che sia tutto ok.

Montaggio di un file system

Per essere reso accessibile un file system deve essere montato.

* La procedura di montaggio è molto semplice: si fornisce al SO il nome del dispositivo e la sua locazione (detta punto di montaggio, mount point) nella struttura di file e directory alla quale agganciare il file system.

Tutto questo lo si fa perché spesso l’obiettivo è **la condivisione di file.**

La condivisione può avvenire attraverso uno **schema di protezione.** Nei sistemi distribuiti, i file sono condivisi attraverso una rete. Il file system di rete (**NFS)** è un metodo comune di condivisione di file. Però con la condivisione di file tramite rete bisogna stare attenti a vedere chi può fare cosa; ecco perché bisogna stabilire schemi di protezione.

Protezione

* Il proprietario/creatore di un file deve essere in grado di controllare:
* Che cosa può essere fatto
* Da chi può essere fatto qualcosa.
* Tipi di accesso
* Lettura/ scrittura
* Esecuzione
* Aggiunta
* Cancellazione
* Elencazione

Controllo degli schemi e gruppi

Modalità di accesso: lettura (R), scrittura (W), esecuzione (X).

Immagine che contiene orologio, segnale

Descrizione generata automaticamenteUNIX permette gli accessi in modo molto semplice, definendo 3 classi di utenti:

ognuna di queste classi ha 3 bit (RWX), che sono la descrizione su chi può fare cosa, 1 consentito , 0 non consentito. Il 7, 6, 1 scritti in figura, rappresentano il sistema ottale (da 0 a 7), quindi per dire che il proprietario ha tutti i permessi, invece di dire 111, si dice 7 (rappresentazione ottale di 111), come 6 al posto di 110, e 1 al posto di 001.

**Quindi in ottale anziché andare a definire i singoli bit che definiscono RWX, si da una cifra unica che rappresenta l’ottale di quella configurazione specifica.**