Sistemi Operativi

Ionut Zbirciog

28 November 2023

File System

I file system sono un modo per organizzare e memorizzare (in modo persistente) le informazioni. Offrono inoltre un'astrazione sui dispositivi di memorizzazione (Disco rigido, SSD, rete, RAM) Le informazioni sono organizzate in file e directory. Esempi di file system: FAT12/FAT16 per MS-DOS, NTFS per Windows, Ext4 per Linux, APFS per macOS/iOS.

File

I file sono un meccanismo di astrazione: forniscono un metodo per salvare informazioni sul disco e leggerle in seguito. Ciò deve avvenire in modo da nascondere all'utente i dettagli di come e dove le informazioni siano memorizzate e di come funzioni effettivamente il disco. I file vengono identificati tramite nomi, che possono variare in base al sistema operativo. Alcuni sistemi operativi limitano la lunghezza dei file (MS-DOS) mentre altri supportano nomi più lunghi. Inoltre, alcuni sistemi come UNIX distinguono tra maiuscole e minuscole (maria != MarIa), mentre sistemi come MS-DOS no (maria == MarIa).

Ciascun file, generalmente, è identificato oltre da un nome anche da un'estensione, indicando generalmente una caratteristica specifica di un file (.jpg per immagini, .c per codice sorgente in linguaggio C, etc.). In alcuni sistemi come UNIX, le estensioni sono puramente convenzionali e non richieste dal sistema operativo. Un compilatore C, invece, potrebbe effettivamente richiedere l'estensione .c per i file da compilare, rifiutandosi di compilarli se non la presentano, ma al sistema operativo poco importa. Altri sistemi come Windows, le estensioni hanno un significato specifico e sono associate a programmi specifici. Gli utenti (o i processi) possono registrare le estensioni nel sistema operativo e specificare per ognuna quale sia il programma che "possiede" quell'estensione. Quando un utente fa doppio clic sul nome di un file, il programma assegnato alla sua estensione viene lanciato con il file come parametro. Per esempio, con un doppio clic su file.docx, si avvia Microsoft Word con file.docx come file iniziale su cui lavorare. Photoshop, invece, non aprirà file con estensione .docx.

I file possono essere strutturati in tanti modi diversi. Tre delle possibilità più comuni sono descritte nella figura sotto.

Sequenza Non Strutturata di Byte

I file sono visti dal sistema operativo come una serie non strutturata di byte. Il significato dei dati è determinato dai programmi a livello utente, non dal sistema operativo. Questo approccio è adottato da sistemi come UNIX, Linux, macOS e Windows, offrendo massima flessibilità.

Estensione	Significato
.bak	File di backup
.c	Programma sorgente in linguaggio C
.gif	Immagine in Compuserve Graphical Interchange Format
.html	Documento HTML (world wide web hypertext markup language)
.jpg	Immagine codificata con lo standard JPEG
.mp3	Musica codificata in formato audio MPEG layer 3
.mpg	Filmato codificato in formato audio MPEG standard
.o	File oggetto (output da compilatore, non ancora linkato)
.pdf	Documento in formato Adobe PDF (portable document format)
.ps	File PostScript
.tex	Input per il programma di formattazione TEX
.txt	File di testo generico
.zip	Archivio compresso

Figure 1: Esempi di estensioni.

Sequenza Di Record Di Lunghezza Fissa

Un file è una sequenza di record con una struttura interna definita e lunghezza fissa. Il modello storico basato su schede perforate a 80 colonne in mainframe. Letture e scritture avvengono a unità di record, meno comune nei sistemi moderni ma era prevalente nei mainframe passati.

File Come Albero di Record

Il file è organizzato come albero di record, con lunghezze variabili e un campo chiave in posizione fissa. L'organizzazione consente ricerche rapide basate su chiavi specifici. Utilizzato principalmente in sistemi mainframe per elaborazioni dati di carattere commerciale (es. DBMS), diverso dalle sequenze non strutturate di UNIX e Windows.

Molti sistemi operativi supportano diversi tipi di file. UNIX (e di nuovo, macOS e Linux) e Windows, per esempio, hanno file e directory normali. UNIX ha anche file speciali a caratteri o a blocchi.

- **File e Directory Normali:** Sono utilizzati in sistemi come UNIX e Windows. I file normali contengono informazioni utente e sono la forma più comune. Le directory sono file di sistema per mantenere la struttura del file system.
- **File Speciali:** A caratteri usati per modellare porte seriale di I/O come terminali e stampanti. A blocchi usati per modellare dischi.
- **File Normali:** File ASCII composti da righe di testo, visualizzabili e stampabili; variano nella terminazione delle righe. File Binari, non leggibili come testo, hanno una struttura interna conosciuta dai programmi che li utilizzano. Per esempio, file eseguibili o archivi.

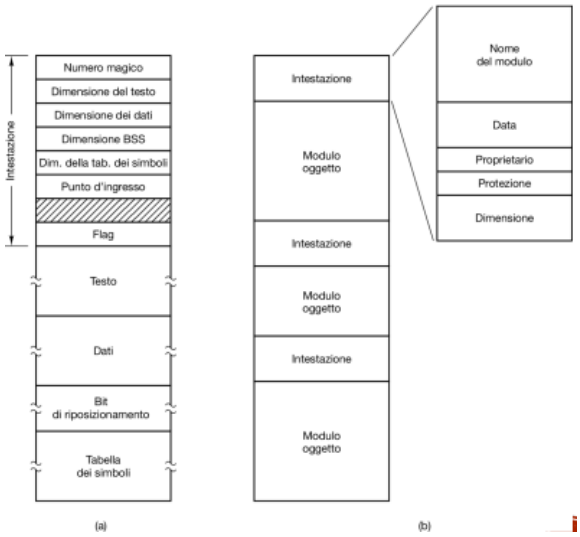

Figure 2: Struttura interna di un file binario.

File Eseguibile - Componenti

- **Intestazione (Header):** Contiene un 'numero magico' per identificare il file come eseguibile (e non eseguire file non "eseguibile"), dimensioni delle parti del file, indirizzo di esecuzione iniziale (punto d'ingresso) e vari flag.
- **Testo e Dati:** Parti effettive del programma, caricare e rilocare in memoria.
- **Tabella dei simboli:** Utilizzata per il debug.

File di Archivio

- **Descrizione:** Raccolta di procedure di libreria (moduli) compilate ma non collegate.
- **Intestazione dei Moduli:** Indicano nome, data di creazione, codice di protezione e dimensione.
- **Carattere Binario:** Stampare questi file produrrebbe caratteri incomprensibili.

Accesso ai File

Nei primi sistemi operativi, c'era un solo metodo di accesso ai file, accesso sequenziale. In questi sistemi, un processo poteva leggere tutti i byte o i record in ordine, a partire dal principio, ma non poteva saltarli né leggerli in ordine sparso. I file sequenziali potevano tuttavia essere riavvolti, in modo da poterli leggere tutte le volte che occorreva.

Successivamente con l'avvento dei dischi è stato introdotto l'accesso causale, che permette la lettura di byte o record in qualsiasi ordine, senza seguire una sequenza. È cruciale per applicazioni come i sistemi di database, dove è necessario accedere rapidamente a record specifici senza attraversare l'intero file. Per specificare dove cominciare a leggere possono essere usati due metodi. Nel primo, ogni operazione read fornisce la posizione del file dalla quale iniziare a leggere. Nel secondo è fornita un'operazione speciale, seek, per impostare la posizione corrente. Dopo una seek il file può essere letto sequenzialmente dalla posizione appena definita come corrente. Quest'ultimo metodo è usato sia in UNIX sia in Windows. Ogni file ha un nome e i propri dati. Tutti i sistemi operativi associano ulteriori informazioni a ciascun file, per esempio la data e l'ora in cui è stato modificato l'ultima volta e la dimensione. Chiameremo attributi (metadati) queste ulteriori voci del file. L'elenco degli attributi cambia in modo considerevole a seconda del sistema.

Gli attributi dei file sono cruciali per:

- la protezione, il controllo dell'accesso
- la gestione efficace dei file nei sistemi operativi

Attributo	Significato	Attributo	Significato
Protezione	Chi può accedere al file e in che modalità	Flag temporaneo	0 per normale; 1 per cancellare il file al termine del processo
Password	Password necessaria per accedere al file	Flag di file bloccato	0 per non bloccato; non zero per bloccato
Creatore	ID della persona che ha creato il file	Lunghezza del record	Numero di byte nel record
Proprietario	Proprietario attuale	Posizione della chiave	Offset della chiave in ciascun record
Flag di sola lettura	0 per lettura/scrittura; 1 per sola lettura	Lunghezza della chiave	Numero di byte del campo chiave
Flag di file nascosto	0 per normale; 1 per non visualizzare negli elenchi	Data e ora di creazione del file	Data e ora di quando il file è stato creato
Flag di file di sistema	0 per file normali; 1 per file di sistema	Data e ora di ultimo accesso al file	Data e ora di quando è avvenuto l'ultimo accesso al file
Flag di file archivio	0 per già sottoposto a backup; 1 per file di cui fare il backup	Data e ora di ultima modifica al file	Data e ora di quando è avvenuta l'ultima modifica al file
Flag ASCII/binario	0 per file ASCII; 1 per file binari	Dimensione attuale	Numero di byte nel file
Flag di accesso casuale	0 per accesso sequenziale; 1 per accesso casuale	Dimensione massima	Numero di byte di cui può aumentare il file

Figure 3: Attributi di un file.

Operazioni sui File

Quali operazioni si possono effettuare sui file?

1. **Create:** Creazione di un file senza dati.
2. **Delete:** Eliminazione di un file per liberare spazio sul disco, attraverso una specifica chiamata di sistema.
3. **Open:** Apertura di un file per consentire al sistema di caricare in memoria gli attributi e gli indirizzi del disco.
4. **Close:** Chiusura del file alla termine degli accessi per liberare spazio nelle tabelle interne.
5. **Read:** Lettura dei dati da un file, generalmente dalla posizione corrente, specificando la quantità di dati richiesti e fornendo un buffer per la loro memorizzazione.
6. **Write:** Scrittura di dati nel file, tipicamente alla posizione corrente, può comportare l'ampliamento del file o la sovrascrittura dei dati esistenti.
7. **Append:** Aggiunta dei dati solo alla fine del file, usata in alcuni sistemi operativi come forma limitata di scrittura.
8. **Seek:** Riposizionamento del puntatore del file su una posizione specifica per file ad accesso casuale, permettendo la lettura o la scrittura da quella posizione.
9. **GetAttributes:** Lettura degli attributi del file.
10. **SetAttributes:** Modifica degli attributi di un file da parte dell'utente, come la modalità di protezione o altri flag, dopo la creazione del file.
11. **Rename:** Ridenominazione di un file, utilizzata come alternativa al processo di copia ed eliminazione del file originale, specialmente utile per file di grandi dimensioni.

Directory

Per tener traccia dei file, i file system normalmente hanno directory o cartelle, che sono anch'esse dei file.

Sistemi di directory a livello singolo

La forma più semplice di sistema di directory è di avere una sola directory contenente tutti i file, talvolta chiamata directory principale (root directory). Il vantaggio di questo schema è la semplicità e la capacità di localizzare i file rapidamente: in fin dei conti c'è solo un posto in cui cercare. Oggi, in era moderna, questo tipo di organizzazione viene spesso usato nei dispositivi embedded, come fotocamere digitali o MP3, in tecnologie RFID, come carte di credito, tessere di trasposto.

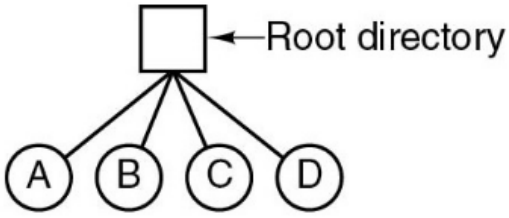

Figure 4: Sistema di directory a livello singolo.

Sistemi a directory gerarchici

Quello che serve è una gerarchia (cioè delle directory ramificate ad albero). Inoltre, se più utenti condividono un file server comune, com'è nel caso di molti network aziendali, ogni utente può avere una directory principale privata per la propria gerarchia. Questo metodo è illustrato nella figura sotto. In questo caso, le directory A, B e C contenute nella directory principale appartengono ciascuna a un utente differente; due utenti hanno creato delle sottodirectory per i progetti su cui stanno lavorando. La capacità di creare un numero arbitrario di sottodirectory fornisce agli utenti un potente strumento di strutturazione per organizzare il lavoro; per questo motivo tutti i file system moderni sono organizzati in questo modo. Vale la pena di ricordare che il file system gerarchico è una delle tante idee sperimentate per la prima volta in Multics negli anni '60.

Quando il file system è organizzato secondo un albero di directory, i nomi dei file vanno specificati in qualche modo. Sono usati comunemente due metodi.

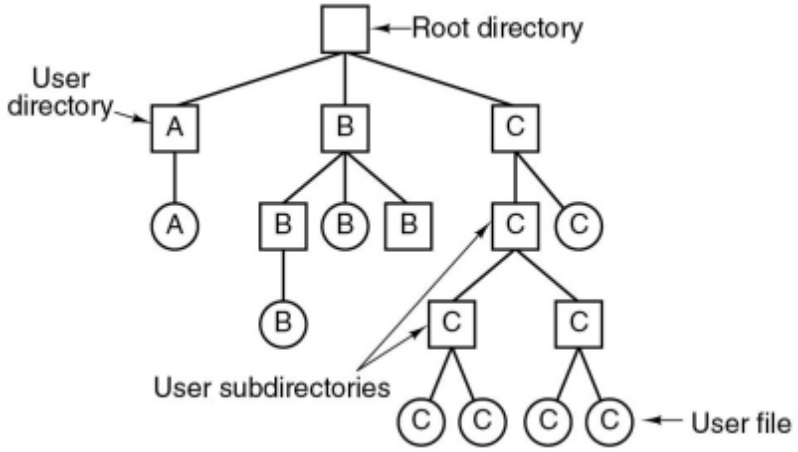

Figure 5: Sistema di directory gerarchico.

Percorso Assoluto

Il primo prevede di assegnare a ogni file un nome di percorso assoluto composto dal percorso che inizia dalla directory principale e arriva al file. Per esempio, il percorso /usr/ast/mailbox significa che la directory principale contiene una sottodirectory usr, che a sua volta contiene una sottodirectory ast, che a sua volta contiene il file mailbox. I nomi di percorso assoluti iniziano dalla directory principale e sono univoci.

Percorso Relativo

L'altro metodo è quello del nome di percorso relativo. È usato congiuntamente al concetto di directory di lavoro, chiamata anche directory corrente. Un utente può designare una directory come directory di lavoro, e in quel caso tutti i nomi di percorso che non cominciano con la directory principale sono considerati relativi alla directory di lavoro. Se l'utente si trova già nella directory /usr/hjb, allora i due comandi sono equivalenti.

Listing 1: Esempio di percorso relativo in UNIX.

```
cp /usr/hjb/mailbox /usr/hjb/mailbox.bak == cp mailbox mailbox.bak
```

In UNIX i componenti del percorso sono divisi tramite /, in Windows il separatore è e in MULTICS era >.

Molti sistemi operativi che supportano un sistema di directory gerarchico hanno due voci speciali in ogni directory, "." e "..", generalmente dette "punto" e "puntopunto" ("dot" e "dotdot"). "Punto" si riferisce alla directory corrente, "puntopunto" alla directory genitore (la directory precedente), con l'esclusione della directory radice che anche nel caso di "puntopunto" fa riferimento a se stessa.

Operazioni sulle Directory

Quali operazioni si possono effettuare sulle directory?

1. **create:** Creazione di una directory vuota con le voci '.' e '..'
2. **delete:** Eliminazione di una directory, possibile solo se la directory è vuota
3. **opendir:** Apertura di una directory per la lettura del suo contenuto
4. **closedir:** Chiusura di una directory dopo la lettura per liberare risorse
5. **readdir:** Restituisce la prossima voce in una directory aperta senza esporre la struttura interna
6. **rename:** Rinomina una cartella, simile al rinomino di un file
7. **link:** Crea un hard link, collegando un file esistente a un nuovo percorso condividendo l'i-node (blocco di dati presente sul disco)
8. **unlink:** Rimuove una voce di una directory, cancellando il file se è l'unico link

Una variante del concetto di collegare i file è il link simbolico (detto anche collegamento o alias). Invece di avere due nomi che puntano alla stessa struttura di dati interna che rappresenta un file, può essere creato un nome che punta a un piccolo file che simboleggia un altro file. Quando viene usato il primo file, per esempio viene aperto, il file system segue il percorso e trova il nome alla fine. Quindi parte con il processo di ricerca usando il nuovo nome. I link simbolici hanno il vantaggio di poter varcare i confini dei dischi e collegare anche file su computer remoti. La loro implementazione talvolta è tuttavia meno efficiente degli hard link.