Introduzione al file system

Tutte le applicazioni devono poter:

- Memorizzare informazioni
- Recuperare informazioni

I processi possono salvare un numero limitato di informazioni nella RAM, che è troppo piccola per memorizzare alcune delle applicazioni.

Un secondo problema è che quando il processo finisce,l'informazione va persa, il che è negativo perché in molte applicazioni dobbiamo poter mantenere informazioni in maniera indefinita, inoltre non possiamo perderle a causa di un crash.

L'ultimo aspetto problematico si verifica quando più processi vogliono accedere alle stesse informazioni nello stesso istante, quindi dobbiamo rendere le informazioni indipendenti dai processi.

Quindi i requisiti essenziali per la memorizzazione a lungo termine sono:

- Capacità di **Salvare grandi quantità di informazioni**
- **Persistenza** delle informazioni oltre la fine del processo.
- **Accessibilità** delle informazioni da più processi.

Per queste memorizzazioni usiamo dischi magetici o SSD che supportano un'interfaccia che sembra una sequenza lineare di blocchi di dimensioni fisse che supporta due operazioni:

- Lettura del blocco k;
- Scrittura del blocco k;

> ⚠ **SSD non sono veri e propri dischi**

Ora sorgono le seguenti domande:

1. Come trovare le informazioni?
2. Come evitare che un utente legga le informazioni di un altro?
3. Come sapere quali blocchi sono liberi?

Possiamo rispondere a queste domande tramite una nuova astrazione: **i File**

Che cosa è un file

Sono unità logiche di informazioni create da processi, un disco ne può contenere milioni e indipendenti.

I processi possono:

- Leggere file
- Creare file

La cosa più importante è che i file non sono influenzati da creazione e morte del processo.
Un file scompare solo quando il proprietario lo rimuove.

Il modo in cui i file sono strutturati, denominati, utilizzati, protetti, implementati sono gestiti dal sistema operativo, più in particolare da quella parte denominata **file system.**

Aspetti importanti per l'utente:

- Cosa costituisce un file
- La denominazione
- La protezione
- Le operazioni che si possono fare

Aspetti NON importanti per l'utente:

- Implementazione tecnica

Ricapitoliamo

Il file system è un modo per organizzare e memorizzare in modo permanente le informazioni, sono un astrazione dei dispositivi di memorizzazione (Disco rigido, SSD), sono tipicamente organizzati in file e directory e alcuni esempi possono essere:

- FAT12/FAT16: MS-DOS
- NTFS: Windows
- Ext4: Linux
- APFS: mac OS/IOS

Nomi dei file

Quindi i file sono delle astrazioni per salvare e leggere informazioni sul disco.
Quando un processo crea un file, gli attribuisce un nome. Tramite il nome altri processi possono accedervi.
Ogni sistema operativo ha delle sue regole per la nomenclatura dei file.

> ⓘ **Esempi di regole per la nomenclatura**
>
> Alcuni SO considerano validi nomi composti solo da stringhe di lettere (MS-DOS), anche la lunghezza dipende dal SO usato (Più moderni supportano fino a 255 lettere per file), inoltre alcuni SO come UNIX distinguono fra lettere maiuscole e minuscole.

Alcuni file system famosi:

- **FAT-16**
- **FAT-32**
- **NTFS**
- **ReFS (Resilient File System)**
- **FAT12** -> Niente caratteri
- **Ext4** -> No 10, o nomi speciali come: "." e ".."

Questi sistemi variano in termini di proprietà come la costruzione dei nomi dei file e il suppporto unicode.

Estensioni

Quasi tutti i SO supportano nomi di file composti da due parti separate da un punto, la parte successiva si chiama estensione e vanno ad indicare una caratteristica del file.

> ⚠ **Nota**
>
> In alcuni sistemi le estensioni sono puramente convenzionali (UNIX), mentre in altri hanno un significato specifico e sono associati a programmi specifici (Windows).

> ⓘ **Gestione estensioni in Windows**
>
> Le estensioni sono registrare nel Sistema Operativo e associate a programmi specifici che si avviano quando l'utente interagisce con il file.

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

Struttura dei file

Possiamo strutturarli in tanti modi, i tre modi più comuni sono:

- **Sequenza non strutturata di byte** --> Al SO non interessa cosa ci sta nei file e spetta ai programmi a livello utente dargli un significato (Windows e UNIX). Questo modello ci da massima flessibilità, poiché il SO non fa nulla, quindi non ci può intralciare.
- **Sequenza di record di lunghezza fissa** --> Ogni record ha una struttura interna e le operazioni di lettura e scrittura avvengono a unità di record veniva principalemente usato da modeli basati su schede perforate.
- **File come Albero di record** --> Il file è organizzato come un albero di record, con lunghezze varaibili e chiave fissa, permette ricerce molto rapide, usato principalmente nei sistemi Mainframe a carattere commerciale.

Tipi di file

I SO supportano molti tipi diversi di file:

- **File normali** --> Contenenti informazioni dell'utente.
- **Directory** --> File di sistema usati per mantenere la struttura del file system.
- **File Speciali** --> Divisi in file speciali a caratteri, usati per modellare dispositivi seriali di I/O come terminali e stampanti e file speciali a blocchi, usati per modellare dischi.

I file normali sono generalmente file ASCII o file binari:

- ASCII -> Composti da righe di testo, visualizzabili e stampabili; variano nella terminazione della riga.

- Binari -> Non leggibili come testo ed hanno una struttura interna conosciuta dai programmi che li utilizzano(exe o file di archivio).

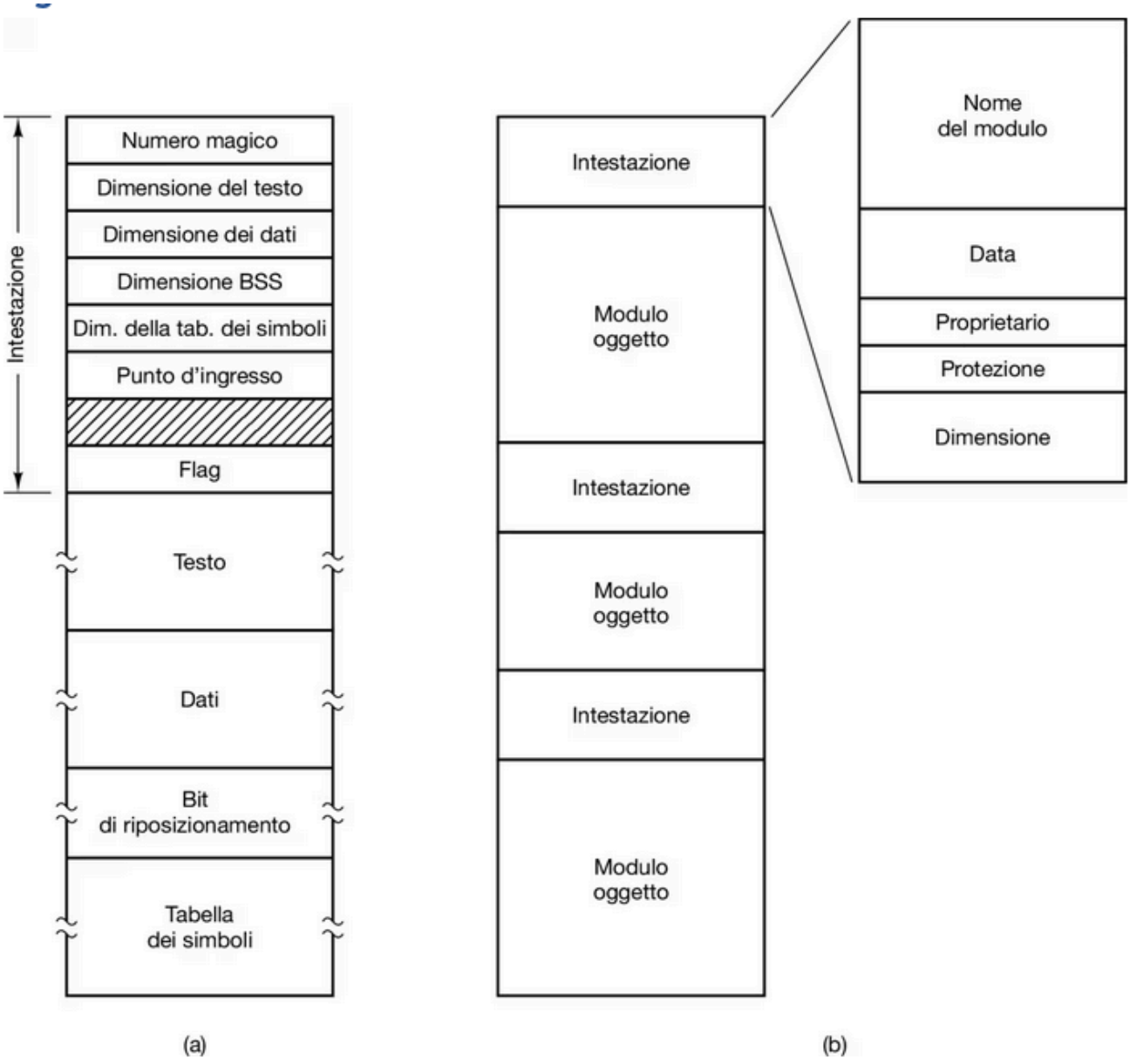
ⓘ Struttura di un file eseguibile

Nonostante il file sia solo una sequenza di byte, il SO lo può eseguiresolo se ha un formato corretto. I paragrafi sono cinque:

1. Header
2. Testo
3. Dati
4. Bit di rilocazione
5. Tabella dei simboli
   L'intestazione parte con un numero magico, previene l'esecuzione accidentale di un file non in questo formato, Quindi ci sono le dimensioni delle varie parti, l'indirizzo da cui parte l'esecuzione a alcuni indicatori (flag).
   Poi ci sta tutto il resto che viene caricato in memoria e rilocati usando i bit di rilocazione, la tabella è invece usata per il debbugging.

ⓘ Struttura di un file di archivio

Composto da un insieme di procedure di libreria (moduli) compilate, ma non collegate fra loro.
Ciascuna preceduta da un intestazione che indica nome, data di creazione, proprietario, codice di protezione e dimensione.



(a) Un file eseguibile. (b) Un archivio.

Ogni SO deve riconoscere almeno il tipo di file eseguibile, alcuni ne possono riconoscere di più, ciò non sempre è una cosa positiva, infatti alcuni sistemi, come il vecchio TOPS-20, avevano meccanismi complessi per il riconoscimento del tipo dei file che potevano portare a limitazioni nell'uso dei file.

In UNIX abbiamo uno strumento che ci serve a capire con quale tipo di file stiamo lavorando, cioè la utility *file*.

Accesso ai file

Primi SO --> Accesso sequenziale ai file, bisognava leggerli tutti, non si potevano saltare, ma si potevano riavvolgere, comodi per i nastri magnetici.

Nascita dei dischi --> Possibilità di leggere file senza un ordine, sono chiamati **file ad accesso casuale.** Questi sono essenziali per molte applicazioni.

Per specificare dove cominciare a leggere possono essere usati due modi:

1. Ogni operazione read fornisce la posizione del file dalla quale cominciare a leggere.
2. Operazione seek usata per impostare la posizione corrente, per poi essere letto in maniera sequenziale (UNIX, WIndows).

Attributi dei file

Oltre a nome e dati, ogni file ha degli attributi (Metadati) che variano a seconda del Sistema Operativo.

Esempi:

- **Protezione ed accesso;**
- **Flag specifici di controllo;**
- **Tipologia** del file -> ASCII o binario, accesso casuale o sequenziale;
- Attributi temporali -> Data e ora di creazione;
- Dimensione -> Attuale e massima;
- **Gestione dei Record** -> Lunghezza del record, posizione e lunghezza della chiave;

Questi metadati sono essenziali per la protezione, il contorllo dell'accesso e per avere una gestione efficace dei file nei SO.

ESEMPI DI ATTRIBUTI

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

Operazioni sui file

Ogni Sistema Operativo fornisce delle funzioni con cui si può operare su file, di seguito sono descritte le chiamate di sistema più comuni utilizzate:

1. **Create** -> Creazione di un file senza dati;
2. **Delete** -> Eliminazione di un file per liberare spazio su disco tramite chiamata di sistema;
3. **Open** -> Apertura di un file per consentire al sistema di caricare in memoia gli attributi e gli indirizzi del disco;
4. **Close** -> Chiusura di un file al termine degli accessi per liberare spazio nelle tabelle interne;
5. **Read** -> Lettura dei dati da un file, generalmente dalla posizione corrente;
6. **Write** -> Scrittura di dati nel file, tipicamente dalla posizione corrente; Potrebbe comportare l'ampiamento del file o la sovrascrittura del dati esistenti.
7. **Append** -> Aggiunta di dati solo alla fine del file;
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. **Get Attribute** -> Lettura degli attributi di un file;
10. **Set Attribute** -> Modifica gli attributi di un file da parte dell'utente; Come la modifica della protezione;
11. **Rename** -> Ridenominazione di un file;