

Modelli e problematiche di *file system* *Parte 1 - Indice*

1. Attributi
2. Struttura logica interna di *file*
3. Operazioni ammesse su *file*
4. Struttura logica interna di *directory*
5. Operazioni ammesse su *directory*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 72

Modelli e problematiche di *file system* *Aspetti generali - 1*

- La maggior parte dell'informazione strutturata (i dati) ha durata, ambito e dimensione più ampi della vita delle applicazioni in memoria principale
 - 3 le esigenze più evidenti
 - Nessun limite di dimensione fissato a priori
 - Persistenza dei dati
 - Condivisione dei dati tra applicazioni distinte
- Il *file system* è il servizio di S/O progettato per soddisfare tali bisogni

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 73

Modelli e problematiche di *file system* *Aspetti generali - 2*

- Il termine *file* designa un insieme di dati correlati, residenti in memoria secondaria e trattati unitariamente
- Il termine *file system* (FS) designa la parte di S/O che si occupa dell'organizzazione, gestione, realizzazione ed accesso ai *file*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 74

Modelli e problematiche di *file system* *Aspetti generali - 3*

- La progettazione di FS affronta 2 problemi chiave
 - Cosa occorre offrire all'utente ed in che modo
 - Modalità di accesso a *file*
 - Struttura logica e fisica di *file*
 - Operazioni ammissibili su *file*
 - Come ciò possa essere convenientemente implementato
 - Massima indipendenza dalla architettura fisica di supporto

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 75

Modelli e problematiche di *file system* *File*

- Il *file* è un meccanismo di astrazione
 - Consente di salvare informazione su memoria secondaria per ritrovarvela in seguito, senza dover conoscere né struttura logica e fisica né funzionamento del supporto utilizzato
 - All'utente non interessa come ciò avviene
 - E' invece necessario per l'utente poter designare le unità di informazione mediante nomi unici e distinti
 - L'utente vede e tratta nomi di *file*
 - Le caratteristiche distintive di un *file* sono
 - Attributi (tra cui il nome)
 - Struttura interna
 - Operazioni ammesse

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 76

Modelli e problematiche di *file system* *Attributi - 1*

- **Nome**
 - Stringa composta da 8 a 255 caratteri, inclusi numeri e caratteri speciali
 - Con ≥ 1 estensioni che possono designare il "tipo" di *file* come visto dall'utente
 - **MS-DOS** (base di Windows 95 e Windows 98)
 - Nomi da 1 a 8 caratteri, con ≤ 1 estensione da 1 a 3 caratteri, designante, senza distinzione tra maiuscolo e minuscolo (*case insensitive*)
 - **UNIX** (base di Linux)
 - Nomi fino a 14 (ora 255) caratteri, *case sensitive*, con estensioni, solo informative, senza limite di numero e di ampiezza
 - In generale, l'utente può configurare presso il S/O l'associazione tra l'ultima estensione del *file* ed il tipo applicativo corrispondente

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 77

Modelli e problematiche di *file system*

Attributi - 2

- Altri attributi significativi
 - **Dimensione corrente**
 - **Data di creazione** (può non essere mostrata)
 - **Data di ultima modifica**
 - Indica la "freschezza" del contenuto
 - **Creatore e possessore** (anche distinti)
 - P.es.: il compilatore crea *file* di proprietà dell'utente
 - **Permessi di accesso**
 - Lettura, scrittura, esecuzione

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 78

Modelli e problematiche di *file system*

Attributi - 3

Protezione	Permesso di accesso al file	Flag
Password	Chiave di accesso al file	
Creatore	Identità del processo che ha creato il file	
Proprietario	Identità del processo utilizzatore del file	
Uso	0 – lettura/scrittura 1 – sola lettura (<i>read-only</i>)	
Visibilità	0 – normale 1 – file non visibile (<i>hidden</i>)	
Livello	0 – normale 1 – file di sistema	
Archiviazione	0 – salvato (<i>backed up</i>) 1 – non salvato	
Tipo di contenuto	0 – ASCII 1 – binario	
Tipo di accesso	0 – sequenziale 1 – casuale (<i>random</i>)	
Permanenza	0 – normale 1 – da eliminare dopo l'uso (<i>temporary</i>)	
Accesso esclusivo	0 – libero ≠ 0 – bloccato (<i>locked</i>)	

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 79

Modelli e problematiche di *file system*

Struttura - 1

- La struttura dei dati all'interno di un *file* può essere vista da 3 livelli di astrazione distinti
 - Livello **utente**
 - Il programma utente associa significato al contenuto del *file*
 - Livello di **struttura logica**
 - I dati grezzi (non interpretati) sono raggruppati dal S/O in strutture logiche per facilitarne il trattamento
 - Livello di **struttura fisica**
 - Il S/O mappa le strutture logiche sulle strutture fisiche della memoria secondaria disponibile (p.es.: blocchi su disco)
- Le possibili strutture logiche di un *file* sono:
 - A sequenza di *byte*
 - A *record* di lunghezza e struttura interna fissa
 - A *record* di lunghezza e struttura interna variabile

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 80

Modelli e problematiche di *file system*

Struttura - 2

- **Sequenza di *byte* (*byte stream*)**
 - La strutturazione logica più rudimentale e flessibile
 - La scelta di UNIX (→ Linux) e Windows
 - Il programma utente sa come dare significato al contenuto informativo del *file*
 - Minimo sforzo per il S/O
 - L'accesso ai dati utilizza un puntatore relativo all'inizio del *file*
 - Lettura e scrittura operano a blocchi di *byte*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 81

Modelli e problematiche di *file system*

Struttura - 3

- **Record di lunghezza e struttura fissa**
 - Gli spazi non utilizzati sono riempiti da caratteri speciali (p.es.: NULL o SPACE)
 - Il S/O conosce la struttura del *file*
 - L'accesso ai dati è sequenziale ed utilizza un puntatore al *record* corrente
 - Lettura e scrittura operano su *record* singoli
 - Scelta obsoleta e legata a specifiche limitazioni dell'architettura di sistema

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 82

Modelli e problematiche di *file system*

Struttura - 4

- **Record di lunghezza e struttura variabile**
 - Struttura interna di ogni *record* descritta ed identificata univocamente da una chiave (*key*) posta in posizione fissa e nota nel *record*
 - Chiavi raccolte in una tabella a se stante, ordinata per chiave, contenente anche i puntatori all'inizio del *record*
 - Accesso ai dati per chiave di *record*
 - Uso abbastanza diffuso in sistemi *mainframe*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 83

Modelli e problematiche di *file system*

Struttura - 5

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 84

Modelli e problematiche di *file system*

Modalità di accesso - 1

- **Accesso sequenziale**
 - Viene trattato un gruppo di *byte* (un *record*) alla volta
 - Un puntatore indirizza il *record* corrente, ed avanza ad ogni lettura o scrittura
 - La lettura può avvenire in qualunque posizione del *file*, la quale però deve essere raggiunta sequenzialmente
 - Come con un nastro
 - La scrittura può avvenire solo in coda al *file* (*Append*)
 - Sul *file* si può operare solo sequenzialmente
 - Ogni nuova operazione fa ripartire il puntatore dall'inizio

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 85

Modelli e problematiche di *file system*

Modalità di accesso - 2

- **Accesso diretto**
 - Consente di operare su gruppi di dati (*record*) posti in posizione qualsiasi nel *file*
 - La posizione nel *file* è determinata rispetto alla sua base (*offset* = 0)
- **Accesso indicizzato**
 - Per ogni *file*, una tabella di chiavi ordinate contenenti gli *offset* dei rispettivi *record* nel *file*
 - Ricerca binaria della chiave e poi accesso diretto
 - Denominato **ISAM** (*indexed sequential access method*) perché consente accesso sia indicizzato che sequenziale

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 86

Modelli e problematiche di *file system*

Classificazione

- Il FS può trattare diversi tipi di *file*
 - Classificazione distinta da quella dell'utente!
 - *File* normali (*regular*), sui quali l'utente può operare la propria classificazione
 - Contenuto ASCII (testo) o binario (eseguibile)
 - *File* catalogo (*directory*), con i quali il FS consente di descrivere l'organizzazione di gruppi di *file*
 - *File* speciali, con i quali il FS modella dispositivi orientati a carattere (p.es.: terminale) od a blocco (p.es.: disco)

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 87

Esempio 1

File binari in UNIX/Linux

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 88

Modelli e problematiche di *file system*

Operazioni ammesse - 1

- **Creazione**
 - Inizialmente vuoto; inizializzazione attributi
- **Apertura**
 - Deve precedere l'uso; permette di predisporre le informazioni utili all'accesso
- **Cerca posizione (*Seek*)**
 - Solo per accesso casuale
- **Cambia nome**
 - *Rename* (può implicare spostamento nella struttura logica del FS)
- **Azioni più complesse** (p.es.: copia) si ottengono tramite combinazione delle operazioni di base
- **Distruzione**
 - Rilascio della memoria occupata
- **Chiusura**
 - Rilascio delle strutture di controllo usate per l'accesso ed il salvataggio dei dati
- **Letture. Scrittura**
 - *Read, Write, Append*
- **Trova attributi**
- **Modifica attributi**

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 89

Modelli e problematiche di *file system* Operazioni ammesse - 2

• Sessione d'uso di un *file*

- Si può accedere in uso solo ad un *file* già aperto
- Mediante l'apertura, il S/O predispone uno specifico strumento, chiamato *handle*, di accesso a quel *file*
- Dopo l'uso, il *file* dovrà essere chiuso
- UNIX/Linux ha una tabella dei *file* aperti a due livelli
 - Nel primo ci sono le informazioni del *file* comuni a tutti i processi
 - Nel secondo i dati specifici del particolare processo

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 90

Modelli e problematiche di *file system* Esempio d'uso con chiamate di sistema

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc,
        char *argv[]){

    FILE *fp;
    char dato;

    if (argc != 2) {
        printf("Nome del file?");
        exit(1);
    }

    // continua ...
    1/2
```

```
if ((fp = fopen(argv[1], "w"))
    == NULL) {
    printf("File non aperto.\n");
    exit(1);
}
do {
    dato = getchar();
    if (EOF == putc(dato, fp)) {
        printf("Errore di lettura.\n");
        break;
    }
} while (dato != 'c');
fclose(fp);
}
2/2
```

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 91

Modelli e problematiche di *file system* File mappati in memoria

- Il S/O può mappare un *file* in memoria virtuale
 - Il *file* continua a risiedere in memoria secondaria
 - All'indirizzo di ogni suo dato corrisponde un indirizzo di memoria virtuale (base + *offset*)
 - Con memoria segmentata si ha { *file* = segmento } potendo così usare lo stesso *offset* per entrambi
 - Le operazioni avvengono in memoria principale
 - Chiamata di indirizzo → *page fault* → caricamento → operazione
 - A fine sessione le modifiche effettuate in memoria primaria vengono riportate in memoria secondaria
- Riduce gli accessi a disco, ma ha problemi con la condivisione e con i *file* di enorme dimensione

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 92

Modelli e problematiche di *file system* Struttura della *directory* - 1

- Ogni FS usa *directory* (catalogo) o *folder* (cartella) per tener traccia dei suoi *file* regolari
- Le *directory* possono essere classificate, rispetto alla loro struttura, come
 - A livello singolo
 - A due livelli
 - Ad albero
 - A grafo aperto
 - A grafo generalizzato

Il file system

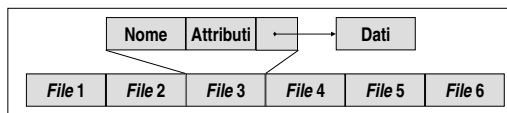
Architettura degli elaboratori 2 - T. Vardanega

Pagina 93

Modelli e problematiche di *file system* Struttura della *directory* - 2

• *Directory* a livello singolo

- Tutti i *file* sono elencati su un'unica lista lineare, ciascuno con il proprio nome
 - I nomi devono pertanto essere unici
- Semplice da capire e da implementare
- Gestione onerosa all'aumentare dei *file*



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 94

Modelli e problematiche di *file system* Struttura della *directory* - 3

• *Directory* a due livelli

- Una *Root Directory* contenente una *User File Directory* (UFD) per utente
- L'utente registrato può vedere solo la propria UFD
 - Le UFD di altri solo se esplicitamente consentito
 - Buona soluzione per isolare utenti in sistemi multiprogrammati
- *File* localizzati indicandone il percorso (*path name*)
- I programmi di sistema possono essere copiati su tutte le UFD, oppure (meglio) posti in una *directory* di sistema condivisa e localizzati mediante cammini di ricerca predefiniti (*search path*)

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 95

Modelli e problematiche di *file system* Struttura della directory - 4

• **Directory ad albero**

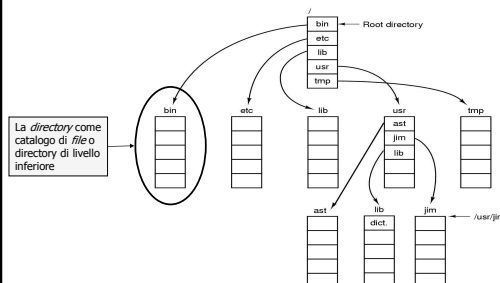
- Consente un numero arbitrario di livelli
- Il livello superiore viene detto radice (*root*)
- Ogni *directory* può contenere *file* regolari o *directory* di livello inferiore
- Ogni utente ha la sua *directory* corrente, che può essere cambiata con comandi di sistema
- Se non si specifica il cammino (*path*), si assume la *directory* corrente
- Il cammino può essere **assoluto** (espresso rispetto alla radice) o **relativo** (rispetto alla posizione corrente)

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 96

Modelli e problematiche di *file system* Esempio di directory ad albero

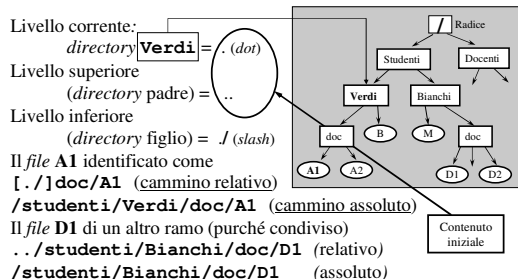


Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 97

Modelli e problematiche di *file system* Esempio (/ per UNIX/Linux, \ per Windows)



Il file system

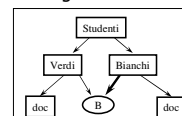
Architettura degli elaboratori 2 - T. Vardanega

Pagina 98

Modelli e problematiche di *file system* Struttura della directory - 5

• **Directory a grafo aperto** (e generalizzato)

- L'albero diventa grafo, consentendo ad un *file* di appartenere simultaneamente a più *directory*
- UNIX/Linux utilizzano collegamenti simbolici (**link**) tra il nome reale del *file* e la sua presenza virtuale
- La forma **generalizzata** consente collegamenti ciclici (riferimenti circolari)
- Un S/O potrebbe duplicare gli identificatori di accesso al *file* (**handle**) → nomi distinti
- Questo però rende più difficile assicurare la coerenza del *file*



Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 99

Modelli e problematiche di *file system* Operazioni su directory (Linux)

Crea <i>directory</i>	mkdir	→ Create
Cancella <i>directory</i>	rmdir	→ Delete
Cambia nome a <i>directory</i>	mv	→ Rename
Apri, chiudi, leggi <i>directory</i>		→ Opendir, Closedir, Readdir
Crea collegamento a <i>file</i>	ln	→ Link
Rimuovi collegamento a <i>file</i>	rm	→ Unlink

Hard link : un puntatore al *file* originario viene inserito nella *directory* remota; questo crea 2 vie d'accesso distinte allo stesso *file*

Symbolic link : *file* speciale il cui contenuto è il cammino del *file* originario; questo mantiene 1 sola via d'accesso al *file*

Il file system

Architettura degli elaboratori 2 - T. Vardanega

Pagina 100