FILE SYSTEM

Il File System è quella parte del Sistema Operativo che si occupa della GESTIONE DEI FILE

Quindi, il FILE SYSTEM è quella parte del S.O. che si occupa della organizzazione generale dei dati che risiedono in MEMORIA SECONDARIA

Il sistema di gestione dei file **DEVE NASCONDERE** all'utente tutti i dettagli relativi ai dispositivi di memorizzazione ( ,  ,  ,) e fornire una ASTRAZIONE di lavoro

⇨ Concetto di **FILE**

Def. 1: un file è un insieme di informazioni (dati) logicamente correlate

Def. 2: un file è una sequenza di bit, byte, linee o record, il cui significato è definito dal creatore o dall'utilizzatore

FUNZIONI TIPICHE DI UN FILE SYSTEM:

1) identifica in modo unico i contenitori di informazione

2) nasconde le caratteristiche fisiche (ASTRAZIONE)

3) fornisce metodi di accesso e di controllo di accesso alle informazioni dei file

4) garantisce la permanenza delle informazioni

(segue File System)

FILE

ESEMPI:
- file che contengono dati
- file che contengono programmi sorgenti
- file che contengono programmi oggetto
- file che contengono programmi eseguibili
- etc.

Un file ha sempre (almeno) un **NOME** che lo identifica

Può avere altre proprietà:
- tipo
- tempo di creazione
- creatore
- lunghezza
- etc.

TIPI DI FILE

Il Sistema Operativo può riconoscere strutture diverse per i file: **FILE** ===> insieme di record con determinate strutture

ESEMPI DI TIPI DI FILE:

text	⇨	sequenza di caratteri organizzati a linee
source	⇨	programma sorgente
object	⇨	codice macchina prodotto da compilazione
image	⇨	codice eseguibile

oppure, non riconoscere nessuna struttura:
FILE ===> insieme di caratteri (stream di byte)
ESEMPI DI S.O.: **DOS e UNIX**

PUNTO DI VISTA DELL'UTENTE

Per gli utenti di un S.O., la parte relativa ai file è il servizio più visibile di un Sistema Operativo

La maggior parte dei COMANDI forniti da un S.O. riguarda i file

COMANDI TIPICI:

- elenco dei file di una directory
  ls (UNIX) o DIR (DOS)

- copia di file
  cp (UNIX) o COPY (DOS)

- cancellazione di file
  rm (UNIX) o DEL (DOS)

- visualizzazione di file
  cat (UNIX) o TYPE (DOS)

- etc.

POSSIBILITÀ di usare anche "ABBREVIAZIONI" nei nomi di file ⇨ ***WILD CARD***

ESEMPIO:

***** e **?** (UNIX e DOS)

DIRECTORY

L'accesso ai file avviene attraverso l'uso di nomi simbolici gestiti dal File System

I FILE sono OGGETTI a LUNGO TEMPO di VITA (*permanenti*)
⇨ bisogna preservare il collegamento fra NOME del file e le informazioni contenute nel file

Ogni nome consente di ritrovare un DESCRITTORE di un FILE
⇨ un DESCRITTORE associa il nome di un file con le sue proprietà e la sua locazione sul dispositivo fisico di memorizzazione

I DESCRITTORI dei file sono mantenuti in strutture dette DIRECTORY

Quindi,
una DIRECTORY è un insieme di DESCRITTORI di file

NOTA BENE: da un punto di vista strettamente formale è scorretto dire che una directory contiene dei file, ma nel parlare comune lo si dice (e lo diremo) MA sapendo che è un modo abbreviato per dire in realtà che una directory contiene dei descrittori di file!

⇨ può essere un FILE di descrittori?

CURIOSITÀ: Il nome directory deriva da *telephone directory* il nome inglese dell'elenco telefonico! Questo può aiutare dato che non diremmo mai che un elenco telefonico contiene gli abbonati, ma diciamo che contiene il modo di trovare le informazioni di un abbonato.

OSSERVAZIONE:
sinonimi di directory ➔ direttorio, cartella o folder

(segue Directory)

OPERAZIONI SU UNA DIRECTORY:

- Ricerca di un file nella directory
- Elenco (totale o parziale) dei descrittori di file
- Creazione di un file
- Cancellazione di un file

Se abbiamo più directory:

- Creazione di una directory
- Cancellazione di una directory

STRUTTURA DELLE DIRECTORY

L'organizzazione delle directory caratterizza un S.O. per l'UTENTE

1) DIRECTORY ad albero

⇨ struttura gerarchica

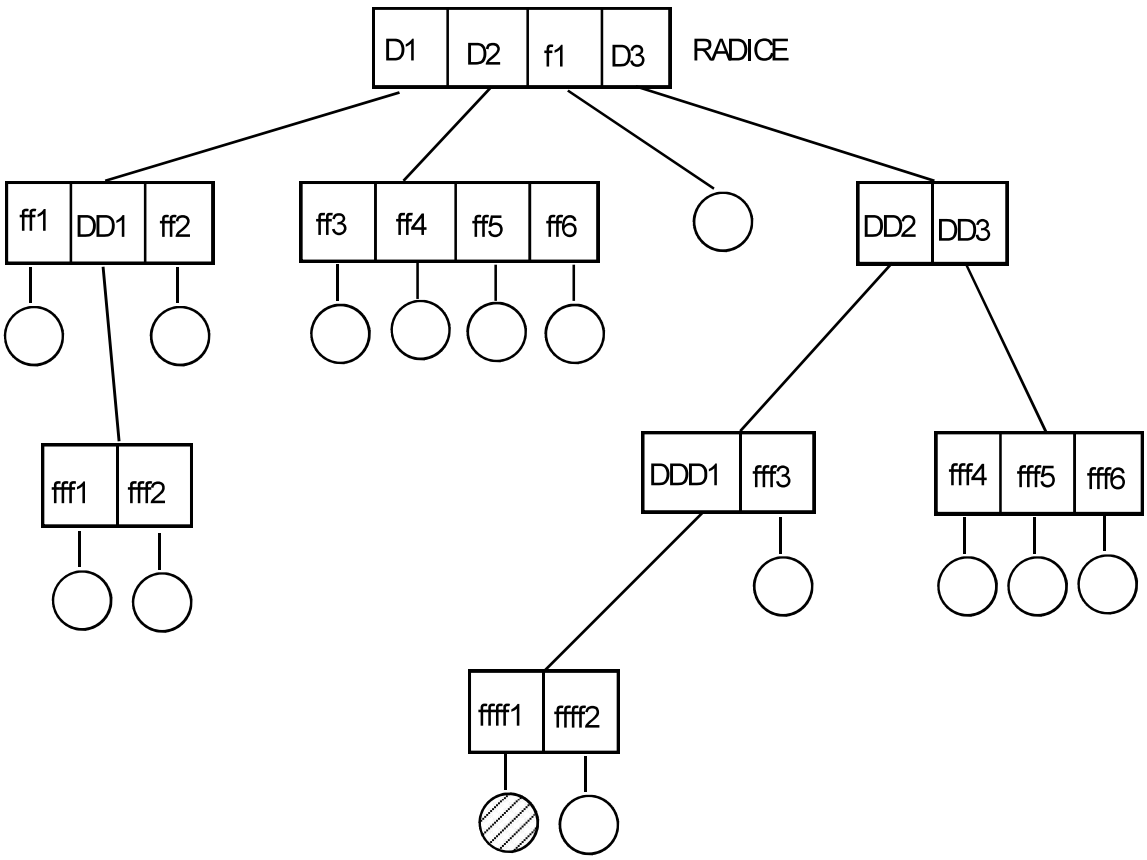
L'utente può CREARE SOTTODIRECTORY

PROPRIETÀ:

a) L'albero di directory ha una UNICA RADICE
b) Ogni file ha un unico PATH NAME, cioè un unico percorso che collega il file alla radice dell'albero
c) Ogni directory può contenere sia file che directory

In ogni istante si ha un DIRECTORY CORRENTE
⇨ possibilità di variare la directory corrente

ESEMPIO DI DIRECTORY AD ALBERO



I NOMI DI UN FILE possono essere di tre tipi:

- **nomi completi (nomi assoluti)**
  - ⇨ cammino dalla radice
    ESEMPIO: RADICE-D3-DD2-DDD1-ffff1

- **nomi relativi alla directory corrente**
  - ⇨ cammino dalla directory corrente
    ESEMPIO: se la directory corrente è RADICE-D3
    il nome relativo è DD2-DDD1-ffff1

- **nomi relativi SEMPLICI**
  - ⇨ nome dato nella propria directory
    ESEMPIO: nella directory DDD1 il nome relativo è ffff1

(segue STRUTTURA DELLE DIRECTORY)

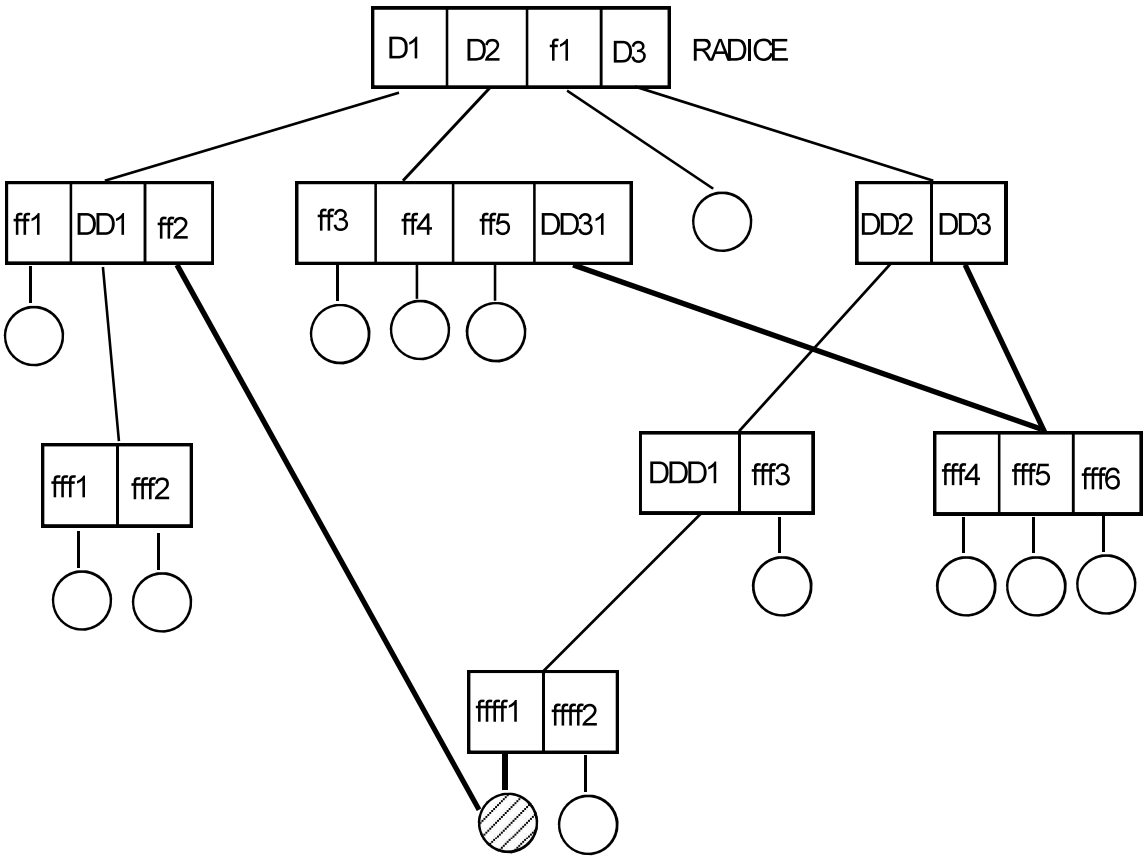
2) DIRECTORY a grafo aciclico

Sostanzialmente analogo alla struttura gerarchica, ma in più:

PROPRIETÀ:

Lo stesso file può essere riferito (eventualmente con nomi relativi diversi) da directory diversi

⇨ consente la condivisione di file e directory



ESEMPIO: Esistono due PATH NAME cioè due nomi assoluti che consentono di reperire lo stesso file:

RADICE-D3-DD2-DDD1-ffff1

RADICE-D1-ff2

Possibilità di avere anche directory organizzate in grafi generali

GENERALIZZAZIONE DEL CONCETTO DI FILE

Il concetto di FILE può essere usato anche come ASTRAZIONE del SISTEMA di INGRESSO/USCITA

Questo vuol dire che tutte le periferiche (ad esempio,  e ) sono trattate come file

L'utente può usare un UNICO ed UNIFORME insieme di servizi per trattare i file e l'I/O
===> I/O INDIPENDENTE DALLE PERIFERICHE

ESEMPIO:
un file di testo può essere stampato con una operazione di COPIA in cui il "file" destinazione è la periferica STAMPANTE

NOTA: Questo è l'approccio di UNIX che è stato poi imitato dal DOS
⇨ CONCETTI DI RIDIREZIONE E PIPING DI COMANDI

OPERAZIONI SUI FILE

Il **PROGRAMMATORE DI SISTEMA** accede ai file tramite CHIAMATE DI SISTEMA (PRIMITIVE)

Non ci interessa come viene implementato, ma quale sono le operazioni possibili su un file ⇨ ASTRAZIONE DI DATO

1. CREAZIONE DI UN FILE
CREA (nome-file, attributi)

Determina l'associazione nome/locazione fisica ⇨ crea un nuovo DESCRITTORE in una directory

2. SCRITTURA SU UN FILE
SCRIVI (nome-file, dato/i)

Se il file ha una struttura, allora i dati devono avere la stessa struttura, altrimenti i dati saranno semplicemente byte

3. LETTURA DA UN FILE
dato = LEGGI (nome-file)

4. CANCELLAZIONE DI UN FILE
CANCELLA (nome-file)

Elimina l'associazione nome/locazione fisica ⇨ cancella (o invalida) il DESCRITTORE nella directory

NOTA BENE: nome-file può essere sia un nome assoluto che un nome relativo alla directory corrente

APERTURA e CHIUSURA

In tutte le operazioni (a parte la creazione) vi è una ricerca del descrittore del file nella directory di appartenenza (mantenuto in memoria secondaria): se si devono effettuare una serie di operazioni di lettura o scrittura questo può risultare ***INEFFICIENTE***

Vengono allora introdotte altre due OPERAZIONI che agiscono su una struttura dati mantenuta in memoria centrale
TABELLA DEI FILE APERTI
⇨ **maggiore EFFICIENZA**

5. APERTURA DI UN FILE
APRI (nome-file)

Crea un elemento nella TABELLA dei FILE APERTI che realizza la corrispondenza con una **copia** del descrittore del file

Le ricerche d'ora in poi avvengono su questa struttura dati di dimensione LIMITATA ⇨ ricerca più efficiente

NOTE SULLA APRI:

1) Una operazione di APERTURA può servire anche per dichiarare quale modalità di accesso si vuole usare: solo lettura, solo scrittura, lettura e scrittura
   La modalità richiesta deve essere AUTORIZZATA dal meccanismo di PROTEZIONE DEI FILE: ad esempio, non si può scrivere su un file READ-ONLY

2) Con questo approccio, l'operazione di creazione implica, in genere, una "apertura in scrittura" del file

VARIANTE:
La OPERAZIONE APRI ritorna un indice nella TABELLA DEI FILE APERTI ⇨ le funzioni LEGGI, SCRIVI, CHIUDI usano come parametro questo indice INVECE che il nome del file

6. CHIUSURA DI UN FILE

CHIUDI (nome-file)

Elimina l'informazione relativa al file dalla TABELLA DEI FILE APERTI

Questa operazione è necessaria data la dimensione limitata della TABELLA dei FILE APERTI

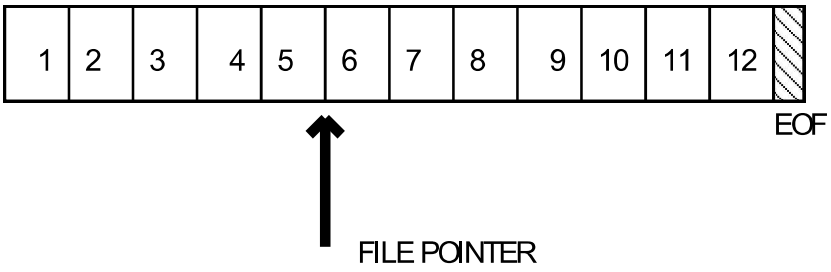
Ma le scritture e le letture su quale parte del file hanno effetto?

⇨ dipende dai **METODI DI ACCESSO**

METODI DI ACCESSO

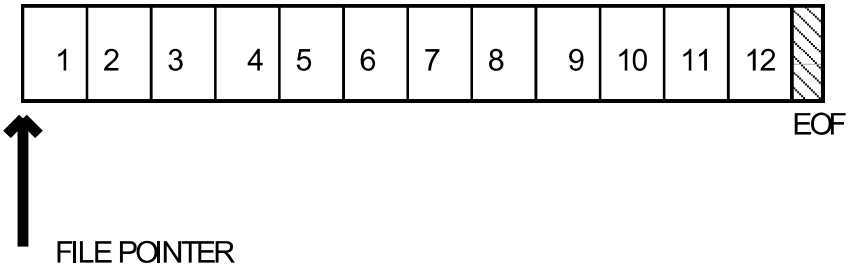
Il più semplice metodo di accesso ai file è quello **SEQUENZIALE**

Col metodo di accesso sequenziale, le OPERAZIONI di LETTURA e SCRITTURA agiscono sul dato riferito dalla POSIZIONE CORRENTE all'interno del file
⇨ **FILE POINTER**

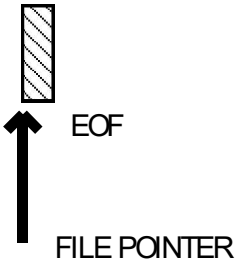


La prossima lettura/scrittura interesserà il dato identificato dal numero 6.

All'atto della APERTURA in lettura, il FILE POINTER è posto all'inizio del file



All'atto della CREAZIONE di un file, il FILE POINTER è posto sulla marca di END-OF-FILE



Il file è vuoto: le successive scritture spostano l'EOF

(segue METODI DI ACCESSO)

Insieme con l'accesso sequenziale può essere fornita una nuova OPERAZIONE che consente di realizzare una sorta di **ACCESSO DIRETTO** ai dati di un file

7) AGGIORNAMENTO FILE POINTER
SPOSTA (nome-file, nuova-posizione)

Consente di aggiornare la posizione corrente del FILE POINTER a nuova-posizione

---

OSSERVAZIONE:

La marca di END-OF-FILE è un concetto astratto: i diversi S.O. la possono implementare in modi differenti