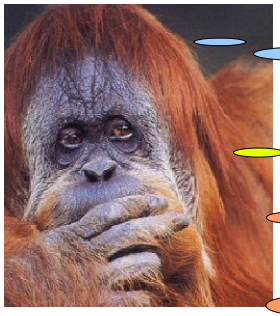
file system

capitolo 10 del libro (VII ed.)

Introduzione

- Programmi e dati sono conservati in memoria secondaria: tipicamente un insieme di dischi. Gli utenti (voi, io) vedono la memoria secondaria come organizzata in **file** e **directory**. Qualcuno ha sentito il termine "**file system**"
- dopo alcuni anni di uso inconsapevole di questi termini un utente potrebbe cominciare a porsi alcune domande …



che cos'è un file ?

cosa vuol dire file system?

cos'è una directory?

File

- Il termine file rappresenta un concetto logico (astratto)
- Un file è un insieme di informazioni correlate, definito da un creatore (a volte tramite l'ausilio di un programma, es. editor), a cui è associato un nome
- Dal punto di vista dell'utente i file sono gli elementi di base in cui è organizzata la memoria, sono indifferenziati
- Però esistono *file* e *file* ...


dvi


testo


shell


crittato

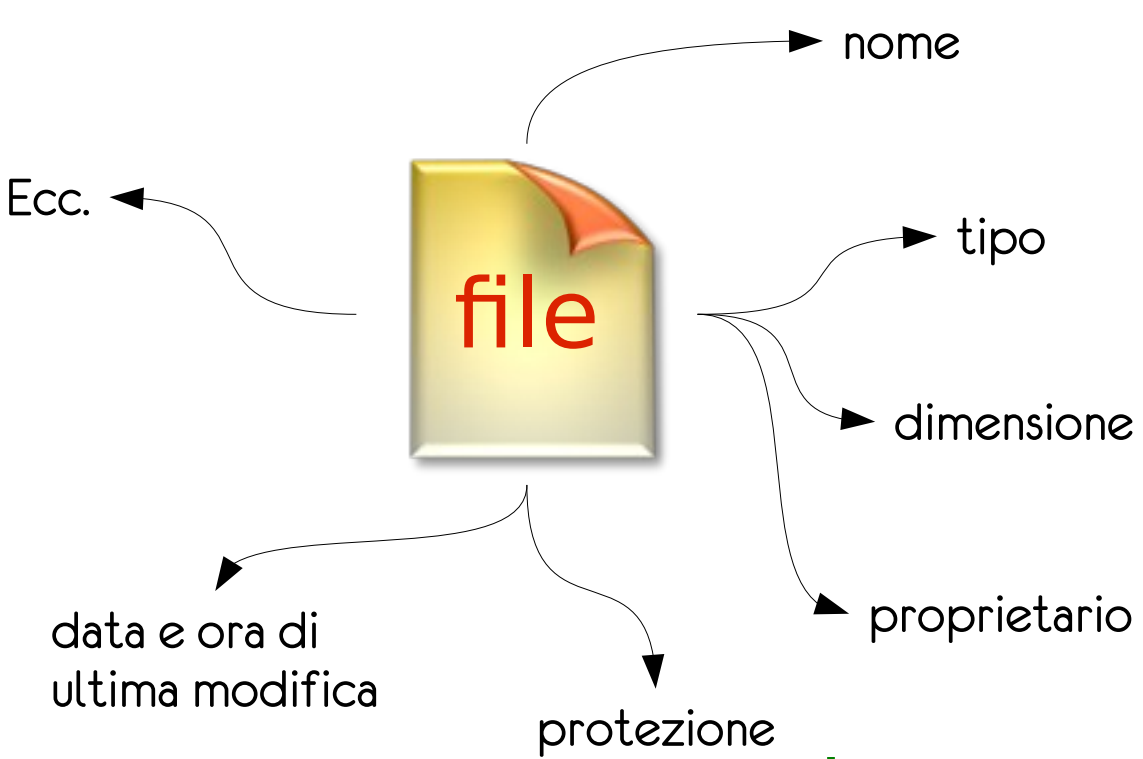

html


binario

Metadati

- il fatto che esitano tipi di file, con estensioni e icone differenti implica che un file non sia soltanto un insieme di dati correlati
- un file è in parte contenuto e in parte oggetto descritto attraverso un insieme di caratteristiche che lo riguardano. Tali caratteristiche sono anche dette attributi o metadati *(dati inerenti i dati)*
- **Esempio**
  - provate ad applicare, in linux, il comando file a un file qualsiasi (es. *clock.png*), otterrete una risposta tipo:
  - clock.png: PNG image data, 128 x 128, 8-bit/color RGBA, non-interlaced
  - provate a rinominare il file in modo da ingannare il comando, per es. rinominiamolo semplicemente clock:
  - clock: PNG image data, 128 x 128, 8-bit/color RGBA, non-interlaced
  - queste informazioni devono essere associate al file stesso indipendentemente dalla sua estensione, non sono dei contenuti, sono metadati

Metadati



Posso associare programmi di gestione a tipi di file, es. acroread o xpdf ai file di tipo pdf: cliccando sull'icona che rappresenta un file pdf viene avviato dal SO il gestore di default.

Basta mantenere una tabella di associazioni <tipo, applicativo>

Fra i diversi modi per associare a un file il suo tipo, particolarmente rilevante l'uso di "magic number"

Altri SO si affidano alle estensioni dei file

Magic Number

MAGIC NUMBER: un codice conservato all'interno del file stesso. Unix adotta questo meccanismo

ESEMPI:

- **Java class file**:
  CAFEBABE (esadecimale)
- **Linux script**:
  "shebang" (#!, 23 21) seguito dal path dell'interprete necessario
- **File postscript**:
  %!
- **File PDF**:
  %PDF
- **File GIF**:
  codice ASCII per "GIF89a", cioè 47 49 46 38 39 61

File di tipo diverso

- perché ho bisogno di diversi tipi di file? Perché tutte queste diverse estensioni?
- Ogni tipo/estensione corrisponde a un particolare **formato** con cui i dati sono organizzati, es.

file immagine

matrice di pixel colorati

però posso avere png, gif, tiff, eps, raw, svg, jpg, xcf, ...

```
%%Page: 1 1
% Translate for offset
14.173228346456694 14.173228346456694 translate
% Translate to begin of first scanline
0 63.97795275590552 translate
63.97795275590552 -63.97795275590552 scale
% Image geometry
64 64 8
% Transformation matrix
[ 64 0 0 64 0 0 ]
% Strings to hold RGB-samples per scanline
/rstr 64 string def
/gstr 64 string def
/bstr 64 string def
{currentfile /ASCII85Decode filter /
                                    str readstring pop}
I85Decode filter /RunLengthDecode filter gstr readstring pop}
I85Decode filter /RunLengthDecode filter bstr readstring pop}

 12861 ASCII Bytes

lMh%en`Srqo\TUu~>
lMh%en`8Qfo\TUu~>
```

FORMATO EPS

ACROREAD
KGHOSTVIEW
si aspettano i dati in questo formato

File di tipo diverso

- perché non posso usare Microsoft Word (o OpenOffice o un editor di pagine HTML) per scrivere un programma C?
- perché se lo faccio il compilatore si arrabbia?

```
void push(lista *p_L, double x)
{
   printf("push %.2f \n",x);
   //stack vuoto
   if(p_L == NULL)
   {
      *p_L = (lista) malloc (sizeof(struct
      if(p_L == NULL) return;
      (*p_L) -> info = x;
   }

   else ...
```

Grassetto, colori, indentazione ecc. sono esplicitamente rappresentati da questi programmi, quindi il contenuto del file è il codice + tutta l'informazione relativa alla sua rappresentazione

File di tipo diverso

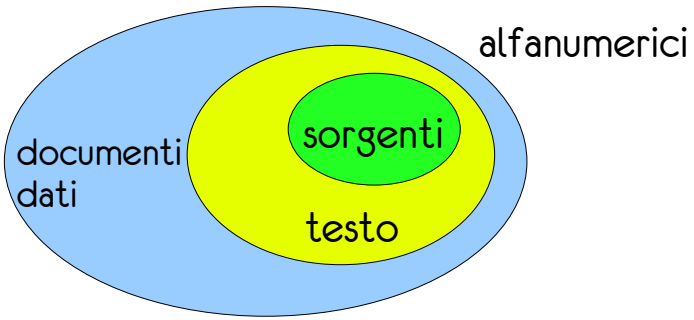
- perché non posso usare Microsoft Word (o OpenOffice o un editor di pagine html) per scrivere un programma C?
- perché se lo faccio il compilatore si arrabbia?

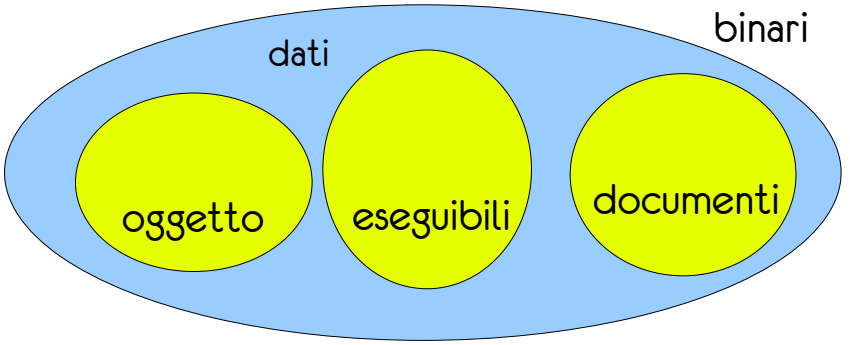
```
<font color="#0000ff">//push sullo stack</font>
<font color="#298a52"><b>void</b></font> push(lista *p_L, <font
color="#298a52"><b>double</b></font> x)
{
   printf(<font color="#ff00ff">&quo
   <font color="#6b59ce">%.2f</fon
   <font color="#ff00ff"> </font><fo
   <font color="#ff00ff">&amp;quot;</font>,x);
   <font color="#0000ff">//stack vuoto</font>
   <font color="#a52829"><b>if</b></font>
   (p_L == <font color="#ff00ff">NULL</font>)
   {
      *p_L = (lista) malloc (<font color="#a52829"><b>sizeof</b>
      </font>(<font color="#2 98a52"><b>struct</b></font> nodo)); ...
```

in HTML lo stesso codice sarebbe rappresentato in questo modo

Tipi di file

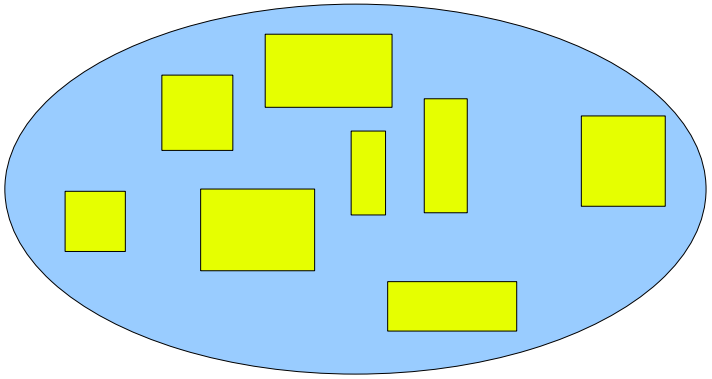
- I file possono essere caratterizzati anche da tipologie più generali rispetto alla specifica applicazione che li gestirà, in particolare:
  - file alfanumerici, contengono sequenze di caratteri. Fra questi:
    - file di testo: sequenze di caratteri divise in righe
    - file sorgenti: sequenze di procedure e funzioni strutturate
  - file binari, contengono byte organizzati secondo una struttura precisa, non sono visualizzabili come testo. Fra questi:
    - file oggetto: sequenze di byte comprensibili per il linker
    - file eseguibili: sequenze di byte comprensibili per il loader
    - altri tipi di documenti: es. file compressi, documenti word, immagini png, ecc.



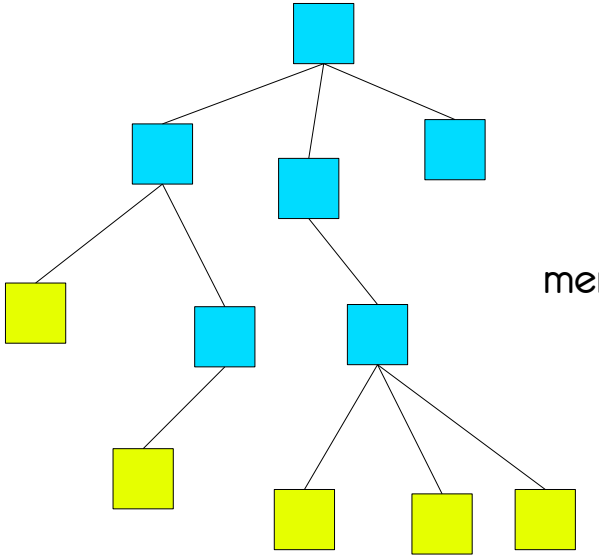


File system

- I file possono essere organizzati in vari modi
- La struttura logica secondo la quale sono organizzati i file è detta file system
- Due visioni classiche:
  - organizzazione piatta: la memoria secondaria è strutturata a livello logico in un insieme di file piatto, non posso avere due file con lo stesso nome
  - organizzazione gerarchica: la memoria secondaria è organizzata ad albero, i nodi intermedi fungono da contenitori e sono detti directory



memoria piatta

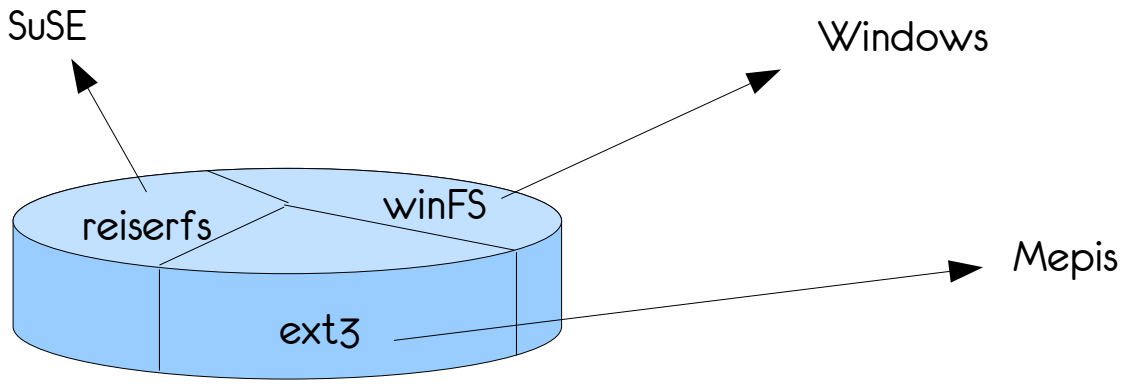


memoria gerarchica

Tipi di file system

- Un file system è mantenuto attraverso l'utilizzo di strutture dati interne, preposte al mantenimento dei metadati (dati inerenti i file e dati inerenti le directory)
- La scelta dei metadati da gestire, la scelta di particolari strutture atte a implementare tali metadati e l'organizzazione in generale della struttura in cui i file sono organizzati porta alla realizzazione di uno specifico file system
- Esistono molti tipi di file system:
  - ext2, ext3 (Tweedie, 1999), ext4 gestisce volumi fino a 1 exabyte (exa: 10^{18}byte, trilioni di byte)
  - ReiserFS, Reiser4 (Namesys 2004)
  - FAT, WinFS
  - Amiga Fast File System
  - ADFS (Acorn computers)
  - file system per flash memory, es: JFFS, YAFFS, smxFFS

Disco e file system

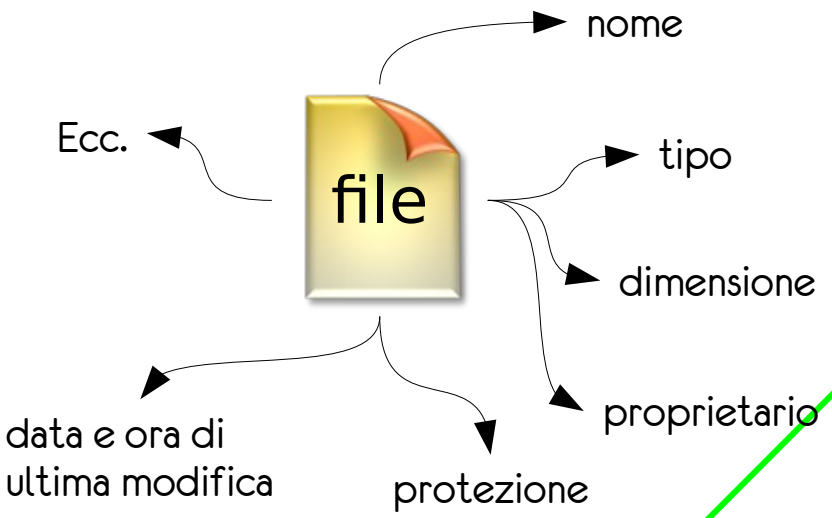


La memoria secondaria può essere suddivisa in partizioni ciascuna delle quali è un diverso file system

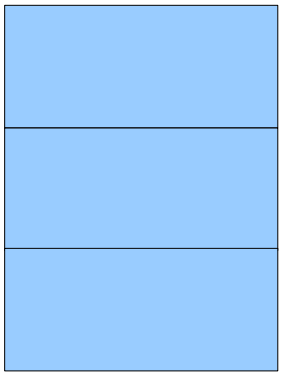
I file system delle varie partizioni possono essere di tipo differente

Questa possibilità è molto utile quando si desidera installare su di uno stesso computer più di un SO

File



File come contenuto



Blocchi che l'utente immagina come sequenziali

File come oggetto gestito dal FS: descrittore

nome
id
tipo
locazione
protezione
data/ora creazione
data/ora ultima modifica
data/ora ultimo accesso
...

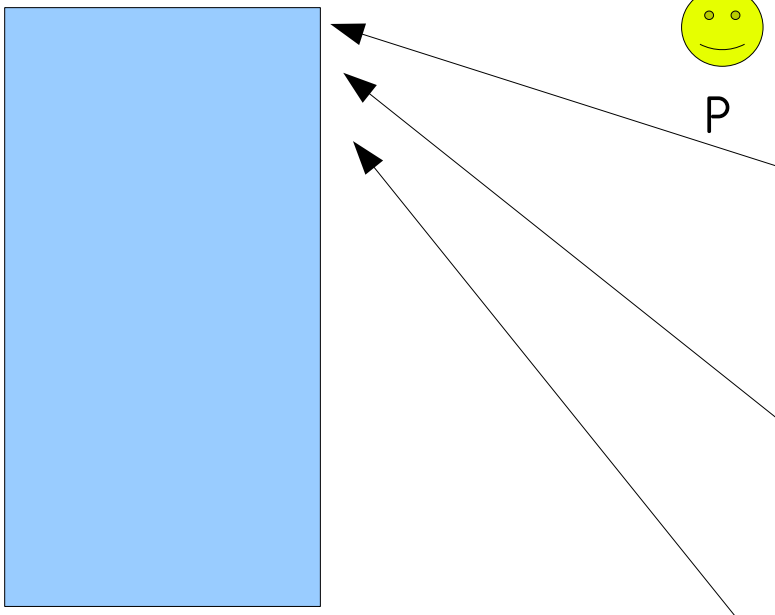
è l'analogo di un PCB (descrittore di processo)

File: operazioni

- Un file è una struttura dati astratta utilizzabile solo attraverso operazioni predefinite
- creazione: occorre trovare spazio sufficiente nel disco, occorre registrare i metadati relativi al file in un'apposita struttura (file system)
- scrittura: occorre identificare la posizione occupata dal file, occorre trovare la posizione in cui scrivere, occorre aggiornare i metadati
- lettura: occorre identificare la posizione occupata dal file, occorre trovare la posizione da cui leggere, occorre aggiornare i metadati
- riposizionamento: si assegna un nuovo valore a un puntatore ai contenuti del file; non è richiesto alcun accesso effettivo in lettura o scrittura, quindi si lavora solo a livello di meta-dati
- cancellazione: occorre individuare il file, aggiornare le strutture di sistema che mantengono informazioni sulla memoria libera, aggiornare le strutture di sistema che mantengono l'organizzazione dei file (file system)
- troncamento: vengono cancellati i contenuti di un file ma vengono mantenuti i metadati

Esempio

- Consideriamo, per es., un processo che debba leggere un intero file ed elaborare i dati così acquisiti



Lettura1:
cerca il file
ottieni puntatore
esegui lettura

Lettura2:
cerca il file
ottieni puntatore
esegui lettura

Lettura3:
cerca il file
ottieni puntatore
esegui lettura

Più efficiente

Apri il file:
cerca il descrittore del file
ottieni puntatore

Lettura1:
esegui lettura

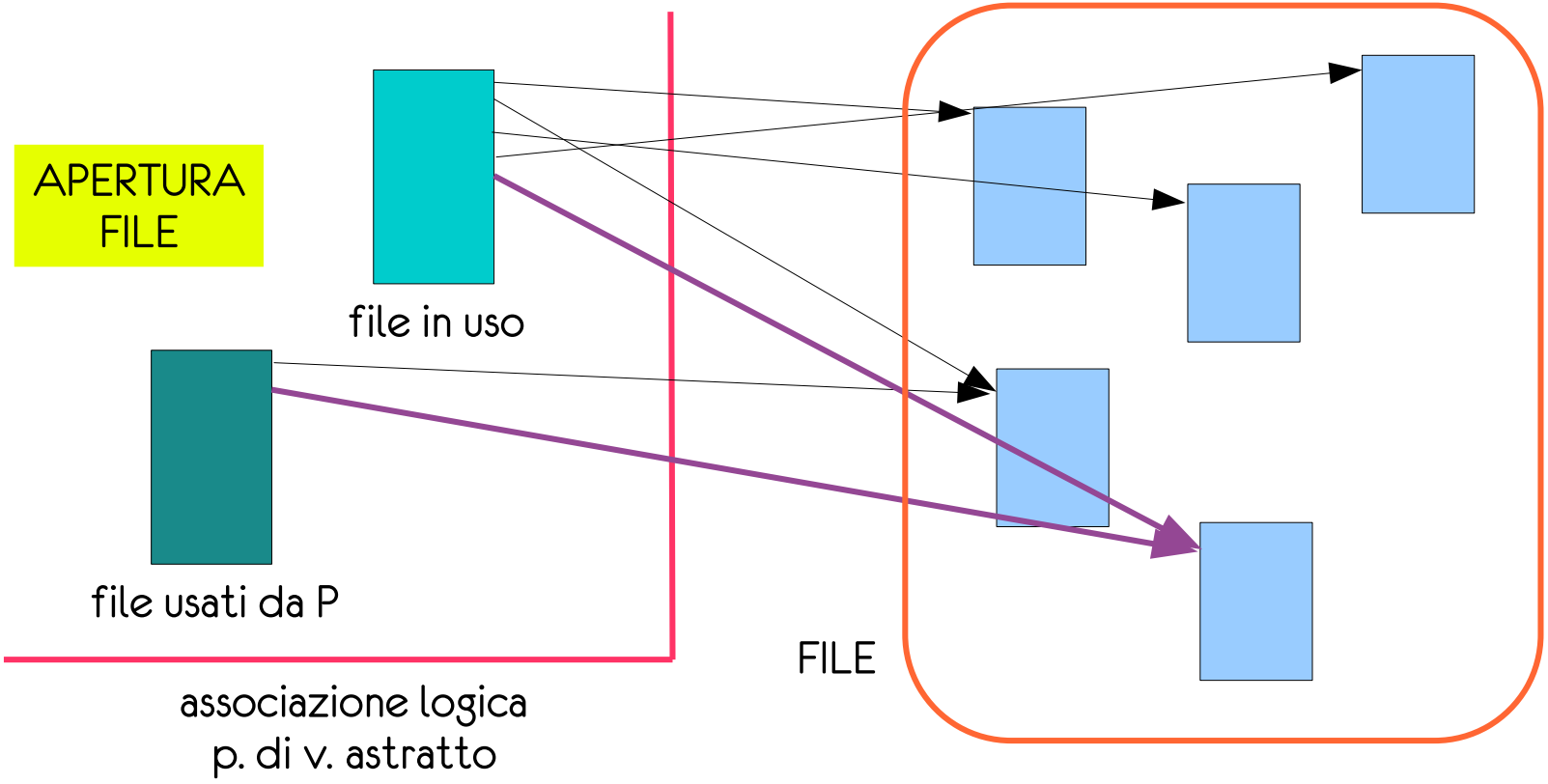
Lettura2:
esegui lettura

Lettura3:
esegui lettura

Chiudi il file

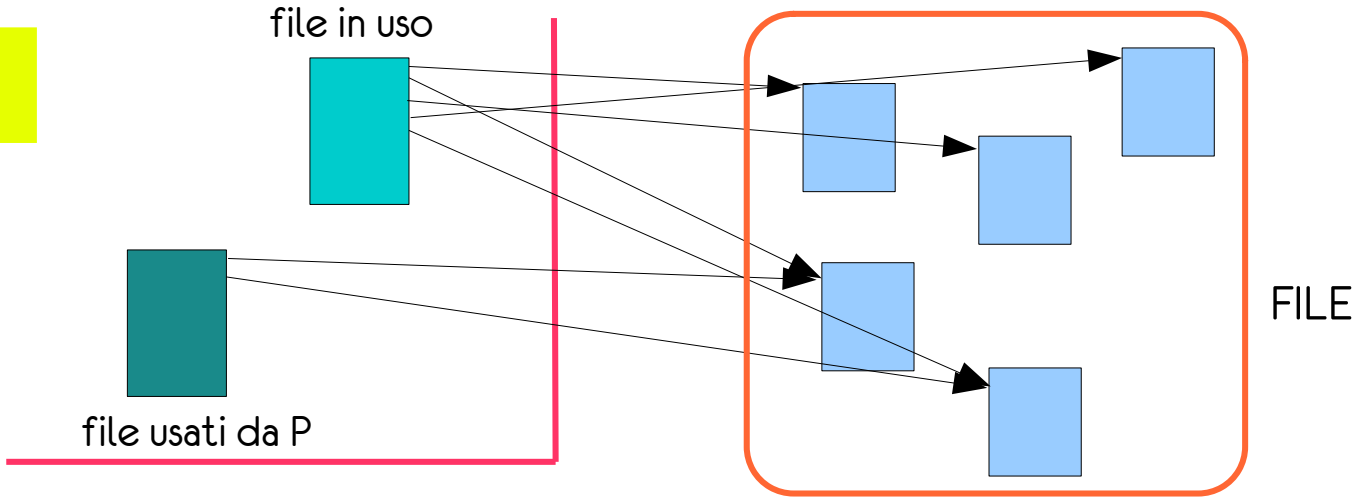
Apertura e chiusura di un file

- L'operazione di apertura di un file consente di ottenere un handle del file, cioè un riferimento che consente di operare effettivamente sul file stesso
- Questa operazione modifica alcune strutture gestite dal SO, che tracciano quali file sono in uso, in generale, e quali file sono in uso da ogni specifico processo

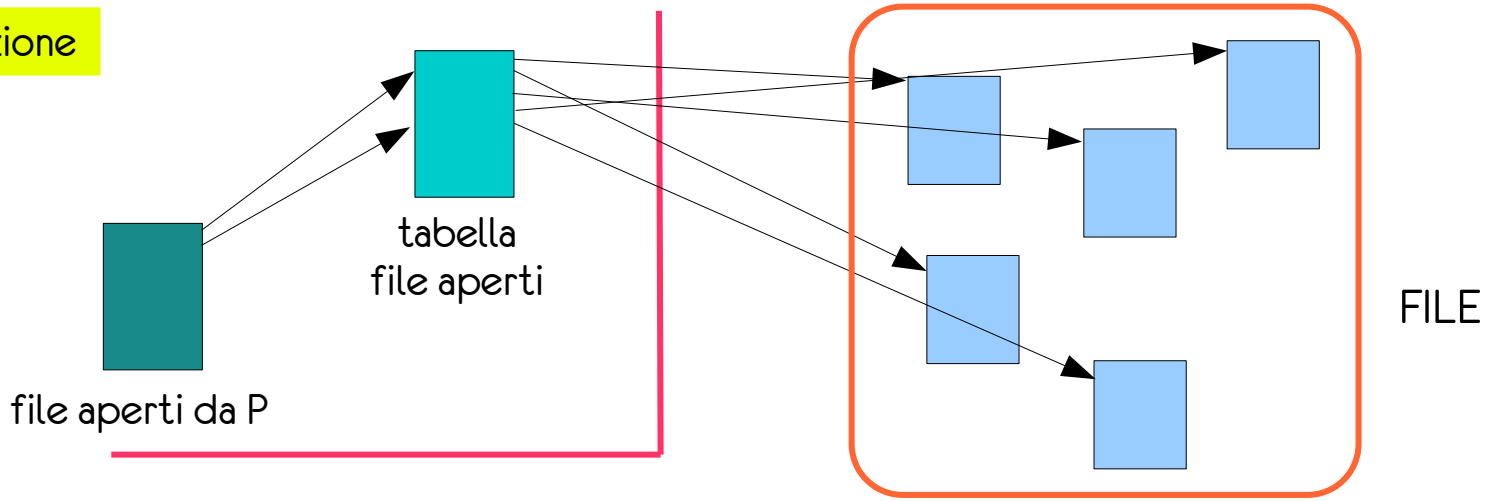


Apertura e chiusura di un file

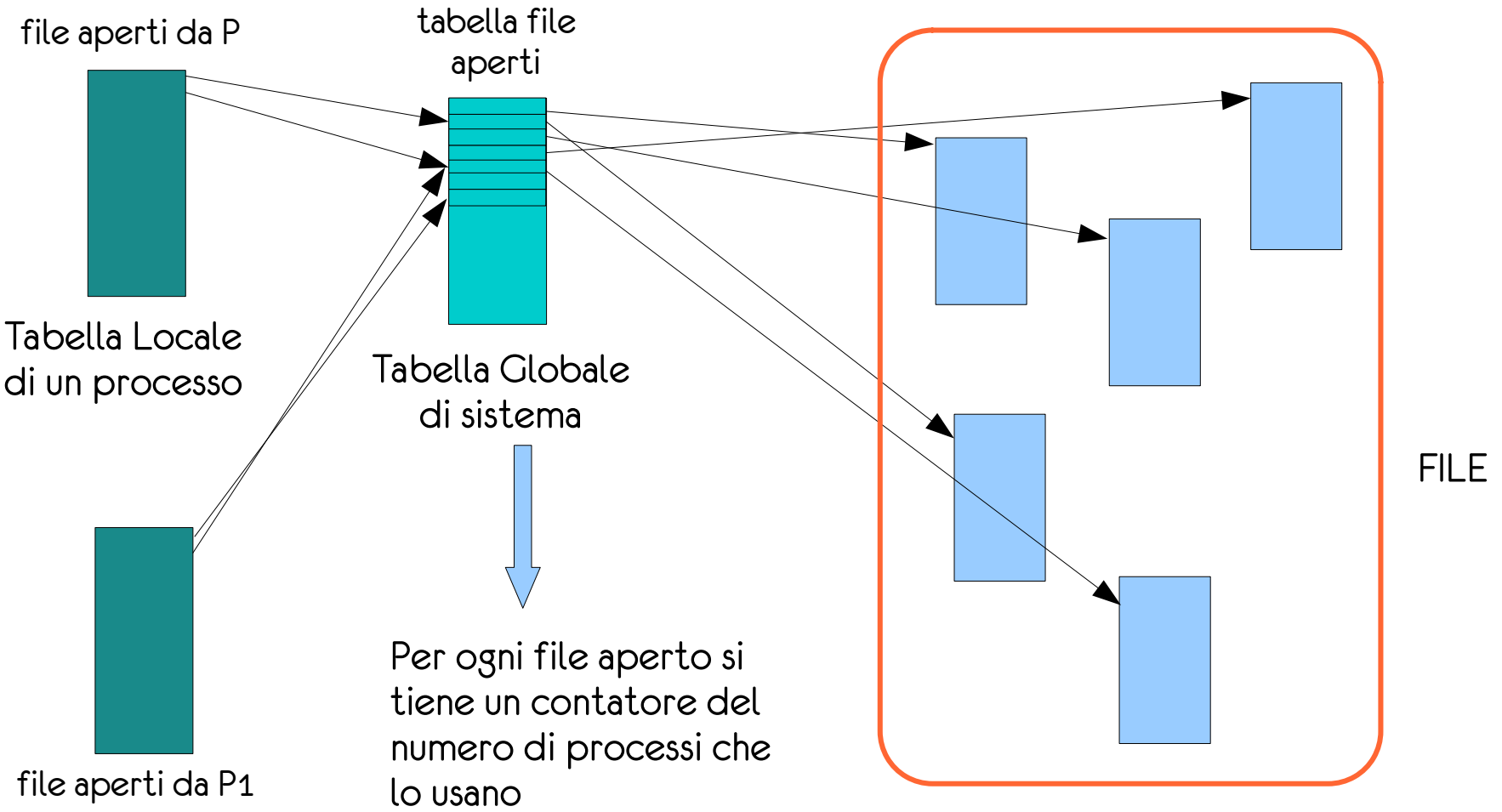
associazione logica
p. di v. astratto



implementazione



Apertura e chiusura di un file

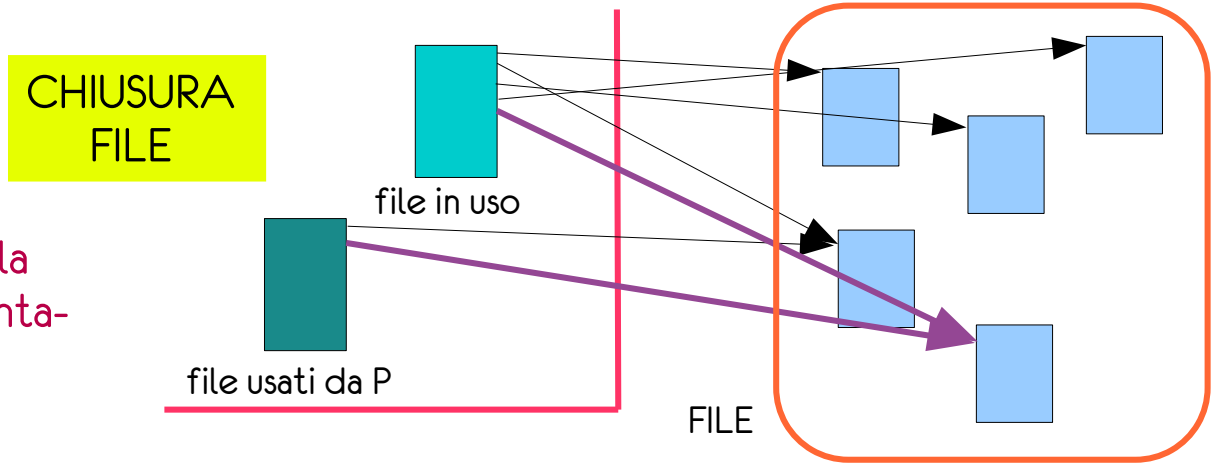


implementazione

Apertura e chiusura di un file

- Più in generale l'operazione di apertura di un file consente di ottenere un handle del file, cioè un riferimento che consente di operare effettivamente sul file stesso
- Questa operazione modifica alcune strutture gestite dal SO, che tracciano quali file sono in uso, in generale, e quali file sono in uso da ogni specifico processo
- Un file aperto da un processo è inteso come una risorsa in uso da parte del processo
- L'apertura di un file comporta (come vedremo in dttaglio) l'allocazione di un insieme di risorse di file system che consentono l'accesso al file stesso
- Chiudere un file significa rilasciare le risorse di file system allocate

Si rimuove una entry dalla tabella globale SSE il contatore va a 0



Apertura e chiusura di un file

- Più in generale l'operazione di apertura di un file consente di ottenere un handle del file, cioè un riferimento che consente di operare effettivamente sul file stesso
- Questa operazione modifica alcune strutture gestite dal SO, che tracciano quali file sono in uso, in generale, e quali file sono in uso da ogni specifico processo
- Un file aperto da un processo è inteso come una risorsa in uso da parte del processo
- L'apertura di un file comporta (come vedremo) l'allocazione di un insieme di risorse di file system che consentono l'accesso al file stesso
- Chiudere un file significa rilasciare le risorse di file system allocate
- La chiusura comporta un insieme di operazioni sulle strutture gestite dal SO per indicare che un certo processo ha terminato di usare un file.
- Se il processo era l'unico utilizzatore del file, le pagine di RAM usate per consentire un'elaborazione efficiente dei dati possono essere liberate
- A differenza da altri tipi di risorse, in genere i file sono risorse condivisibili

Apertura di un file

- **Esempio**, in C posso usare:
  - fopen: funzione di libreria, restituisce come handle un FILE *
  - open: system call, restituisce come handle un numero intero detto file descriptor

```
struct _IO_FILE {
  int _flags;         /* High-order word is _IO_MAGIC; rest is flags. */

  /* The following pointers correspond to the C++ streambuf protocol. */
  /* Note:  Tk uses the _IO_read_ptr and _IO_read_end fields directly. */
  char* _IO_read_ptr;   /* Current read pointer */
  char* _IO_read_end;   /* End of get area. */
  char* _IO_read_base;  /* Start of putback+get area. */
  char* _IO_write_base; /* Start of put area. */
  char* _IO_write_ptr;  /* Current put pointer. */
  char* _IO_write_end;  /* End of put area. */
  char* _IO_buf_base;   /* Start of reserve area. */
  char* _IO_buf_end;    /* End of reserve area. */

  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */
  ... ecc ...
}
```

Chiusura di un file

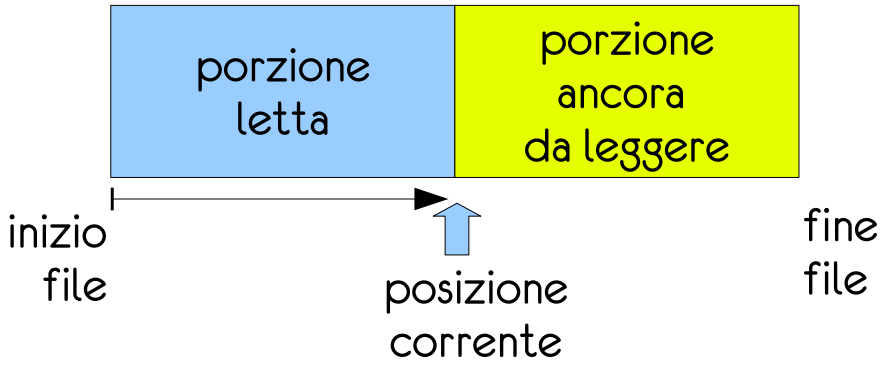
- Per esempio in C si utilizzano
  - fclose: funzione di libreria, ha come parametro un FILE *, l'handle ottenuta tramite fopen
  - close: system call, ha come parametro un file descriptor (un numero intero), l'handle ottenuta tramite open

Lettura e scrittura su file

- Anche lettura e scrittura richiedono di solito una handle come argomento
- Inoltre occorre definire il punto del file da cui leggere o in cui scrivere
  - accesso sequenziale: il programmatore non deve specificare la posizione in modo esplicito, il SO manterrà un puntatore alla posizione corrente di lettura/scrittura per il processo. Questa informazione è mantenuta nella tabella di file aperti locale (del processo).
  - accesso diretto: si può leggere/scrivere da/in specifiche posizioni del file, che vanno indicate espressamente

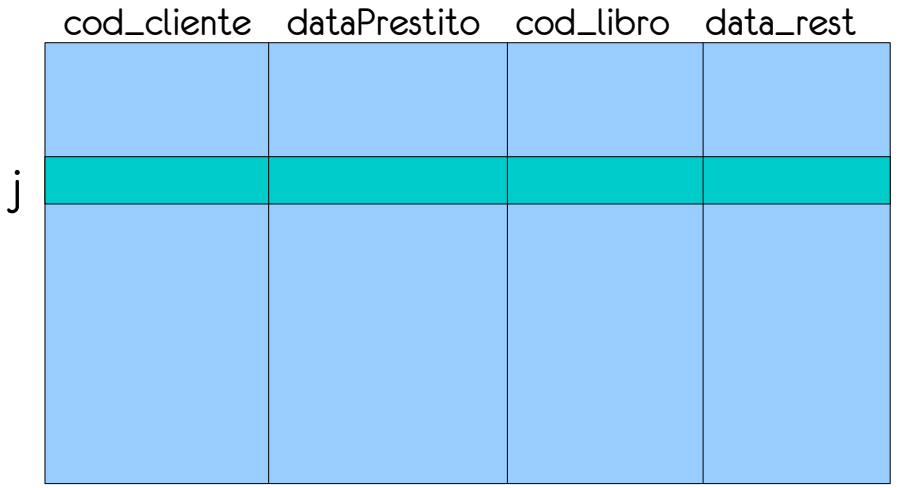
Accesso sequenziale

- è il metodo di accesso a file più comune
- i contenuti di un file vengono percorsi secondo l'ordine logico sequenziale dei dati stessi
- ogni operazione di lettura fa avanzare un puntatore che indica la posizione raggiunta correntemente all'interno del file
- ogni operazione di scrittura aggiunge del contenuto in fondo al file
- alcuni SO consentono una terza operazione: riportare il puntatore alla posizione corrente all'inizio del file



Accesso diretto

- presupposto: i dati contenuti in un file hanno un formato prestabilito
- il file è visto come una sequenza di record di pari dimensione
- conoscendo tale dimensione e la posizione del record di interesse è possibile accedervi senza scorrere l'intero file
- **es**. tabella di una base di dati



file ≡ tabella prestiti

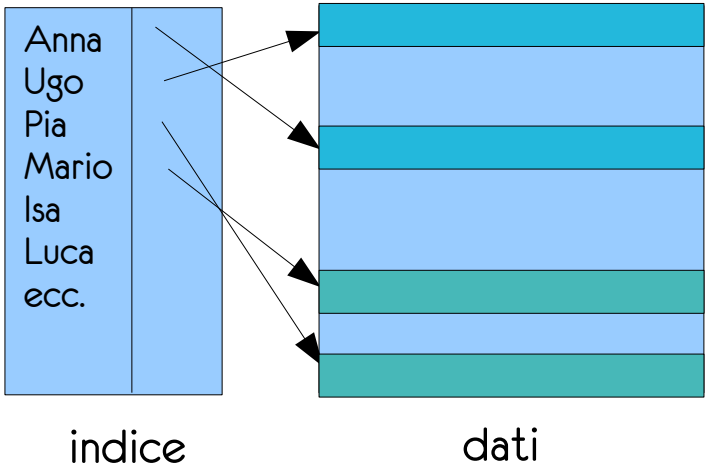
posizione: dim_record x j

read j = leggi il record j-mo
write j = scrivi il record j-mo

Accesso a indice

- definito sulla base del precedente
- un file indicizzato è in realtà costituito da due file:
  - il file dei contenuti veri e propri, memorizzati secondo un preciso formato
  - un file indice, contenente riferimenti ai record
- l'indice non è necessariamente numerico

mantenendo l'indice ordinato si possono effettuare ricerche veloci



Es. molti cosiddetti data base contenenti dati relativi a esperimenti medici/biologici sono in realtà file indicizzati