
CVMLab
Computer Vision
& Multimedia Lab

I file


Università
degli Studi
di Pavia

File binari e file di testo

distinzione tra

- file binari
- file di testo

si possono usare funzioni diverse per la gestione di tipi di file diversi

File binari e file di testo

File di testo

- contengono soltanto caratteri ASCII stampabili e alcuni caratteri di controllo come new-line e la tabulazione
- sono tipicamente letti e scritti per righe, ovvero ad ogni accesso si elabora una linea di testo delimitata da new-line
- sono visualizzabili e manipolabili usando i classici editor di testo
- esempi di file di testo sono i file sorgente dei programmi

File binari e file di testo

file binari

- non contengono solo i caratteri ASCII stampabili
- vengono letti e scritti in blocchi di byte
- esempi di file binari:
  - file eseguibili
  - i file compressi (zip, ecc.)
  - alcuni tipi di file di immagini (jpeg, gif, tiff, ...)

File binari e file di testo

- un file di testo può essere manipolato con le funzioni dedicate ai file binari
  - è possibile leggerli a blocchi di byte come per i file binari
- un file di testo è un "caso particolare" di file binario che utilizza un sottoinsieme dei caratteri ASCII (i caratteri stampabili)
- operare su file binari con le funzioni usate per i file di testo non è invece agevole

File binari e file di testo

- uno stesso dato può sempre essere memorizzato sia in file di testo che in file binari
  - è necessario usare una opportuna rappresentazione del dato stesso
- la scelta del tipo di rappresentazione dipende da considerazioni legate al tipo di utilizzo che viene fatto dei dati

Memorizzazione di un intero

- in un file di testo si presenta come una stringa di caratteri numerici
- in un file binario si usa la sua codifica binaria
- esempio: memorizzazione del numero decimale 214439
- file di testo:

```
'2' '1' '4' '4' '3' '9'
```

- file di binario (dipende dalla endianness):

```
0x00 0x03 0x45 0xa7
```

File binari e file di testo

- in un file di testo si presenta come una stringa di caratteri numerici
- in un file binario si usa la sua codifica binaria
- esempio: memorizzazione del numero decimale 214439
- file di testo:

```
'2' '1' '4' '4' '3' '9'
```

- file di binario (dipende dalla endianness):

```
0x00 0x03 0x45 0xa7
```

File binari e file di testo

sia data la dimensione di un numero intero

per esempio 32 bit

la memorizzazione in un file binario richiede sempre 4 byte indipendentemente dal valore memorizzato

la memorizzazione in un file di testo richiede un numero di byte che dipende dal valore da memorizzare

esempio:

- la memorizzazione del numero 10 richiede 2 byte, uno per ciascuna cifra decimale
- il valore 6745 ne richiede 4

L'accesso a file: il tipo FILE

- le funzioni per l'accesso a file sono dichiarate nel file di intestazione stdio.h
- il riferimento al file desiderato viene mantenuto per mezzo di un puntatore di tipo FILE
  - è una struttura che contiene tutte le informazioni utili a permettere l'interazione di un programma con i file gestiti dal sistema operativo
- esempio di dichiarazione di variabile di tipo FILE:

```
FILE *fp;
```

L'accesso a file: il tipo FILE

- tipicamente il programmatore non ha alcuna necessità di conoscere il contenuto dei campi delle variabili di tipo FILE
- il programma opera su un file chiamando le funzioni messe a disposizione dalla libreria standard
- esse usano le variabili di tipo FILE per identificare il file sul quale operare
- un programma può aprire e usare più file contemporaneamente

Apertura e chiusura di file

- per poter utilizzare un file, questo deve venire aperto
- aprire un file significa comunicare al sistema operativo che si intende accedere al contenuto del file
- si utilizza la funzione fopen (File OPEN)
- Quando non è più utilizzato il file deve essere "chiuso" utilizzando la funzione fclose (File CLOSE)
- le funzioni di apertura e chiusura di file operano su variabili di tipo puntatore a FILE
- la fopen non fa altro che inizializzare le variabili di tipo FILE

Apertura e chiusura di file

- prototipo di fopen:

```
FILE *fopen(char *path, char *mode);
```

- accetta due parametri di tipo stringa (puntatore a carattere):
- path contiene il percorso e il nome del file da aprire
- mode specifica il modo con il quale aprire il file

Apertura e chiusura di file

un file può essere aperto per diversi scopi: lettura, scrittura e append (scrittura alla fine del file)

la stringa mode può quindi valere:

"r" (lettura) : l'accesso al file avviene dal suo inizio

"r+" (lettura e scrittura) : l'accesso al file avviene dal suo inizio

Apertura e chiusura di file

- "w" (scrittura) : se il file esiste, esso viene sovrascritto, altrimenti un nuovo file viene creato; l'accesso al file avviene dal suo inizio
- "w+" (lettura e scrittura) : se il file esiste, esso viene sovrascritto, altrimenti un nuovo file viene creato; l'accesso al file avviene dal suo inizio;
- "a" (append) : la scrittura avviene a partire dalla fine del file; se il file non esiste, esso viene creato;
- "a+" (lettura e append) : se il file non esiste, un nuovo file viene creato; l'accesso in lettura avviene dall'inizio del file, mentre la scrittura è sempre effettuata alla fine del file

Apertura e chiusura di file

- è anche possibile aggiungere il carattere "b" alla fine o in mezzo a qualsiasi precedente combinazione di caratteri (es., ottenendo wb o ab+)
- questo carattere indica esplicitamente che il file da aprire è binario
- molti sistemi trattano i file binari e i file di testo allo stesso modo, quindi il fatto di specificare la b risulta superfluo, ma altri sistemi operativi potrebbero operare delle distinzioni, e quindi potrebbe essere necessario specificare esplicitamente il tipo di file trattato

Apertura e chiusura di file

- la funzione fopen restituisce un puntatore di tipo FILE correttamente inizializzato se l'operazione ha avuto successo
- altrimenti ritorna NULL
- tipiche fonti di errore:
  - il tentativo di aprire in lettura un file che non esiste
  - aprire in scrittura un file read-only (es. i file su un CD-ROM)
  - aprire un file per il quale non si dispongono dei necessari permessi di accesso

Apertura e chiusura di file

- il file deve essere chiuso utilizzando la funzione fclose
- la chiusura del file serve per liberare delle aree di memoria allocate dalle funzioni della libreria per memorizzare le informazioni lette e scritte sul file
- dichiarazione:

```
int fclose(FILE *fp);
```

- accetta come parametro il puntatore a FILE che identifica il file da chiudere
- ritorna 0 se la chiusura avviene con successo
- ritorna 1 in caso di errore

Apertura e chiusura di file

```
FILE *fin, *fout;

if (!(fin = fopen("matrice.dat", "r"))) {
  perror("matrice.dat");
  exit(1);
}
if (!(fout = fopen("documenti/info.txt", "w"))) {
  perror("documenti/info.txt");
  exit(1);
}

/* ... istruzioni di accesso al file ... */
fclose(fin);
fclose(fout);
```

Apertura e chiusura di file

- il file matrice.dat viene aperto in lettura
- il file documenti/info.txt viene aperto in scrittura
- i test verificano che i puntatori restituiti siano non nulli, cioè che il file sia stato aperto correttamente
- la funzione perror (Print ERROR) viene usata per visualizzare un eventuale messaggio
  - il messaggio descrive l'ultimo errore che si è verificato durante la chiamata ad una funzione di libreria
  - se il parametro della perror è diverso da NULL, la funzione prima visualizza la stringa passatale come parametro, seguita dai due punti e dal messaggio di errore
- i file vengono chiusi con la chiamata a fclose

I file di I/O standard

- in stdio.h sono definite tre variabili di tipo FILE:
  - stdin -> standard input (normalmente la tastiera)
  - stdout -> standard output (normalmente il video)
  - stderr -> standard error (normalmente il video)
- vengono aperti automaticamente dal programma quando questo viene eseguito ed ereditati dalla shell

I file di I/O standard

- in C, le azioni di scrivere su video o leggere un dato da tastiera sono assimilate rispettivamente alla scrittura del dato su file e della sua lettura da file
- i file standard sono utilizzati a questo scopo
  - "scrivere su standard output" equivale tipicamente alla visualizzazione a video
  - "leggere da standard input" equivale ad una lettura da tastiera
- invocando l'esecuzione da linea di comando, è possibile effettuare la redirezione di standard input e standard output, ovvero fare in modo che ciò che viene normalmente letto da tastiera sia letto da un file, mentre invece di scrivere a video si effettua la scrittura su un file

Lettura mediante fscanf

- con file di testo da leggere di struttura nota è possibile usare la funzione fscanf
  - si comporta come la scanf per la lettura da standard input
  - è possibile specificare da quale file leggere
  - si può indicare anche il file standard stdin; in questo caso il comportamento è identico a quello di scanf
- definizione di fscanf:

```
int fscanf(FILE *stream, const char *format, ...);
```

Scrittura mediante fprintf

- definizione di fprintf:

```
int fprintf(FILE *stream, const char *format, ...);
```

- le due funzioni seguenti sono del tutto equivalenti:

```
printf("%s %d\n", nome, anni);
fprintf(stdout, "%s %d\n", nome, anni);
```

- effettuano la scrittura su standard output di una stringa, seguita da un valore numerico decimale