File I/O in C

Le chiamate di input/output più utilizzate in C sono implementate nella libreria standard `stdio.h`. Si è già visto come usare lo standard input (`stdin` con `scanf()`) e lo standard output (`stdout` con `printf()`) per interagire col programma. Stavolta ci occupiamo della lettura e scrittura su file e per fare ciò useremo sempre la `stdio.h` con le funzioni:

- `fopen()` e `fclose()` per aprire e chiudere file
- `fgets()` e `fputs()` per leggere e scrivere linee di testo
- `fread()` e `fwrite()` per file binari

Bisogna anche fare una distinzione fra tipi di file binari e di testo: i primi non hanno necessariamente una struttra a righe, devono invece essere "interpretati", cioè bisogna conoscere come sono strutturati. I file di testo sono un sottoinsieme dei file binari, ma la loro struttura non cambia, sia nel senso di rappresentazione dei dati (e.g. la tabella ASCII che associa valori numerici a carateri) che nella struttura a righe.

L'esempio alla pagina successiva illustra l'utilizzo più frequente delle suddette funzioni (escludendo `fwrite()`).

Nella prima parte creiamo un file (con `fopen("nome_file","w")`) nel quale scriveremo due righe utilizzando la funzione `fputs(sorgente,destinazione)`; da notare che stiamo operando su un puntatore (`fout`) di tipo `FILE`.

La seconda parte dell'esempio riguarda la lettura di righe con `fgets(,"r")`, in queso caso è necessario allocare preventivamente dello spazio dove "mettere" ciò che abbiamo letto nel file: trattasi dei vettori `riga1-2`. Infatti la sintassi è di `fgets(destinazione,dimensione,sorgente)` è "prendo tutto ciò che è contenuto in `fin` fino a quando non trovo un carattere di new-line (`\n`), la fine del file (`EOF`) oppure supero il numero di caratteri indicato nel secondo argomento (`sizeof(destinazione)`, e lo metto nella destinazione (primo argomento: `riga1 o 2`)". In questo caso il primo `fgets()` si ferma a `prima riga` perché trova il (`\n`).

Alla fine, non dimentichiamoci di "liberare" i puntatori `fin` e `fout` prima di uscire dal programma, questo per evitare che rimangano segnati come aperti dal sistema operativo prevenendo quindi la loro riapertura in un secondo momento o da altri processi, anche se nella maggior parte dei casi il sistema operativo si occupa di pulire i file aperti non più usati al posto nostro.

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

int main() {

  //puntatori al file di tipo FILE
  FILE *fin, *fout;

   //##### SCRITTURA FILE CON fgets() ######

   //apro un file (se non esiste lo crea) fopen(nome, "in scrittura")
  fout = fopen("miofile.txt","w");
   //metto, "to put", due stringhe con l'accapo in fout
  fputs("prima riga\n seconda riga\n",fout);
  //chiudo fout perche' lo riutilizzeremo dopo
  fclose(fout);

   //##### LETTURA FILE CON fgets() ######

  //lo spazio concreto dove mettera' i dati caricati dal file
  char riga1[50], riga2[50];

  //apro il file di prima stavolta in lettura (r)
  fin = fopen("miofile.txt","r");

  //leggo due righe (so che il file e' di testo)
  fgets(riga1, sizeof(riga1), fin);
  fgets(riga2, sizeof(riga2), fin);

  printf("riga1: %s\nriga2: %s\n", riga1, riga2);
  //chiusura file
  fclose(fin);
  fclose(fout);

  return 0;
  }
```