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 `ciaone1` perché trova il (`\n`).

Nella terza parte ri-referenziamo il puntatore `fout` con `fopen(,"a+")`, che è un modo complicato per dire quello che abbiamo fatto prima, ovvero assegnamo a `fout` la posizione in memoria del file giacché l'avevamo chiuso alla riga 17. Notiamo inoltre che abbiamo aperto il file in modalità "append+" ovvero in scrittura alla fine del file, ma anche in lettura, per via del "+"; siamo in grado, quindi, di "appendere" nuove righe alla fine del file, ma non di sovrascrivere le precedenti.

Una volta scritta la terza riga, arriviamo all'ultima parte dove leggiamo il file per controllare se effettivamente siamo riusciti ad "appendere" la terza riga. Stavolta leggiamo in modalità binaria, con `fread()` la cui sintassi è un po' più complicata: il primo argomento (`temp`) è la destinazione (un po' come per `fgets()`), il secondo ed il terzo (`byte_da_leggere` e `1`) sono la dimensione in byte ed il numero di blocchi di tale dimensione da leggere rispettivamente. Nella fattispecie leggiamo da `fout` un blocco da 28 byte; 28 byte perché `strlen()` ci aiuta a scoprire quanto è lunga un stringa escludendo il byte di terminaizone (`\0`), quindi 8+8+13-1=28 byte (-1 perché non voglio un accapo alla fine).

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("ciaone1\nciaone2\n",fout);
  //chiudo fout perche' lo riutilizzeremo dopo
  fclose(fout);

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

  //lo spazio concreto dove metterò 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: %sriga2: %s\n", riga1, riga2);

  char riga3[] = "freebossetti\n";
  //apro il file alla fine del file stesso (con scrittura +)
  fout = fopen("miofile.txt","a+");

  //"appendo" una riga alla fine del file
  fputs(riga3,fout);

  //riporto il puntatore fout ad inizio file, come una cassetta lo "riavvolgo
   " all'inizio
  rewind(fout);

      //##### LETTURA BINARIA FILE CON FREAD() ######

  char temp[50];
  //strlen() mi dà la lungezza in byte di una stringa
  int byte_da_leggere= strlen(riga1)+strlen(riga2)+strlen(riga3)-1;

  //provo a leggere le stavolta tre righe col metodo binario fread(), devo
   anche specificargli quanti byte deve leggere
  fread(temp, byte_da_leggere, 1, fout);
  printf("rileggo il file (i primi %d byte):\n%s\n", byte_da_leggere, temp);

  //chiusura file
  fclose(fin);
  fclose(fout);

  return 0;}
```