UNIVERSITÀ
degli STUDI
di CATANIA

Operazioni su files

Corso di programmazione I (A-E / O-Z) AA 2022/23

Corso di Laurea Triennale in Informatica

Fabrizio Messina
fabrizio.messina@unict.it

Dipartimento di Matematica e Informatica

Files e stream

Un file è una sequenza di bytes.

Con tale astrazione si possono memorizzare dati su vari supporti in modo permanente (hard disk, nastri magnetici).

I files in C vanno manipolati con le funzioni della libreria standard che operano su **stream** (flussi).

1) "Apertura" di un file. Si usa la funzione di libreria `fopen()` con opportuni argomenti.

```
FILE *fopen(const char *filename, const char *mode);
```

Files e stream

- nome/percorso (nel file system) del file. ES: "/home/mrossi/input.txt" oppure "C:\Users\mrossi\documents\input.txt";
- modalità di apertura:
  - lettura ("r");
  - scrittura ("w");
  - append ("a"); scrittura, se il file esiste aggiunge bytes in "coda"; se il file non esiste ne viene creato uno nuovo
  - "r+" oppure "w+": lettura e scrittura, ma nel secondo caso se il file non esiste ne verrà creato uno nuovo;

La funzione `fopen()` restituisce un *puntatore* ad un tipo `FILE` (<stdio.h>).

2) Chiusura di un file (dopo opportune operazioni di lettura/scrittura).

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

NB: permette di liberare risorse nel sistema operativo.

Infatti un sistema operativo mette a disposizione un numero di `file descriptor` limitato, per tutte le applicazioni.

È quindi necessario chiudere il file quando questo non serve più.

Invocazione di `fclose()` implica `fflush()` automaticamente (per fflush vedi slide successive).

Esempi svolti

22_01.c – apertura di files.

IO a caratteri su file

```
int fprintf(FILE *restrict stream, const char *restrict format, ...);
int fscanf(FILE *restrict stream, const char *restrict format, ...);
char *fgets(char *restrict s, int size, FILE *restrict stream);
int fputs(const char *restrict s, FILE *restrict stream);
```

Funzioni presentate nelle precedenti lezioni.

Il parametro stream deve essere il file descriptor precedentemente aperto mediante funzione fopen().

IO a caratteri su file

Leggere linee di testo con la funzione `fscanf()`.

```
1 FILE *fp = fopen("a.txt", "r");
2 int ret;
3 //..
4 if((ret=fscanf(fp, "%s %s %u", s1, s2, x))==EOF)
5     fprintf(stderr, "\n EOF or read error!");
6 else if(ret<3)
7     printf("\n Partial read!");
```

IO a caratteri su file

Scrivere linee di testo con la funzione `fprintf()`.

```
1 FILE *fp = fopen("a.txt", "w");
2 //..
3 if(fprintf(fp, "%s %s %u", s1, s2, x)<0)
4     fprintf(stderr, "\n Write error! ");
```

IO a caratteri su file

Esempi svolti

22_02.c – scrittura di linee di testo su di un un file (fprintf()).

22_03.c – lettura di linee di testo da un file (fscanf()).

IO a caratteri su file

Scrivere testo con la funzione `fputs()`

```
1 FILE *fp = fopen("a.txt", "w");
2 const char s[] = "abracadabra mario rossi";
3 //..
4 if(fputs(s, fp)<0)
5     fprintf(stderr, "\n Write error!");
```

IO a caratteri su file

Leggere linee di testo con la funzione `fgets()`

```
1 FILE *fp = fopen("a.txt", "w");
2 const char s[50];
3 //..
4 if(fgets(s, 50, fp)==NULL)
5     fprintf(stderr, "\n Error!");
```

IO a caratteri su file

Esempi svolti

22_02_fputs.c – scrittura di linee di testo con la funzione fputs().
22_02_fgets_c – lettura di linee di testo da un file con la funzione fgets().

File ad accesso casuale. IO a “blocchi” su files

```
size_t fread(void *restrict ptr, size_t size, size_t nmemb, FILE *restrict
stream);
size_t fwrite(const void *restrict ptr, size_t size, size_t nmemb, FILE
*restrict stream);
```

La funzione `fread()` legge `nmemb` “blocchi“ di `size` bytes dal file puntato dal parametro stream.

La funzione `fwrite()` scrive `nmemb` “blocchi“ di `size` bytes sul file puntato dal parametro `stream`.

Entrambe le funzioni restituiscono il numero di blocchi letti dal file o scritti sul file.

File ad accesso casuale. IO a “blocchi” su files

```
 1 struct record {
 2     char s[30];
 3     float data;
 4 };
 5
 6 File *fp = fopen("output.txt", "w");
 7
 8 struct record my_record;
 9 strcpy(my_record.s, "pippo");
10 my_record.data = 6.5;
11
12 int ret = fwrite(&my_record, sizeof(struct record),  1, fp);
13 if(ret<1)
14     fprintf(stderr, "\n Write error..");
```

File ad accesso casuale. IO a “blocchi” su files

```
1 File *fp = fopen("output.txt", "r");
2
3 struct record my_record;
4
5 if(fread(&my_record, sizeof(struct record),  1, fp)<1)
6     if(feof(fp))
7         fprintf(stderr, "\n FEOF!");
8     else
9         fprintf(stderr, "\n Read error..");
```

File ad accesso casuale. IO a “blocchi” su files

Esempi svolti

22_04.c – scrittura di blocchi (record) su un file (fwrite()).

22_05.c – lettura di blocchi (record) da file (fread()).

File ad accesso casuale. IO a “blocchi” su files

A seguito di ogni lettura o scrittura, l’indicatore di posizione del file (testina) si sposta in avanti di un numero di bytes equivalente.

É possibile riposizionare la testina mediante la funzione `fseek()`

```
int fseek(FILE *stream, long offset, int whence);
```

`offset` rappresenta il numero di bytes per incrementare o decrementare il contatore, quindi può essere positivo o negativo

`whence` rappresenta il punto dal quale si inizia a contare per posizionare la testina.

SEEK_SET, SEEK_CUR, or SEEK_END (inizio del file, posizione corrente della testina, fine del file).

File ad accesso casuale. IO a “blocchi” su files

Esempio:

- scrivere N record di un certo tipo sul file;
- riposizionare la testina all’inizio del record numero due.
- quindi riscrivere il record numero due.

File ad accesso casuale. IO a “blocchi” su files

```
 1 File *fp = fopen("output.txt", "w+");
 2 struct record data_items[5];
 3 // ... riempimento dei record..
 4 //scrive 5 record sul file
 5 for(int i=0; i<5; i++)
 6     if(fwrite(&data_items[i]), sizeof(struct record), 1, fp)<1)
 7         fprintf(stderr, "\n Error writing on file ..");
 8
 9 //posiziona la testina
10 if(fseek(fp, sizeof(struct record), SEEK_SET)<0)
11     perror("fseek: ");
12
13 //aggiorna il dato sul file
14 data_items[1].data = 16.1234;
15 if(fwrite(&data_items[1], sizeof(struct record), 1, fp)<1)
16     fprintf(stderr, "\n Write error...");
```

File ad accesso casuale. IO a “blocchi” su files

Esempi svolti

22_06.c – lettura di record, aggiornamento di un record.

Funzioni `fflush()` e `feof()`

```
int feof(FILE *stream); int fflush(FILE *stream);
```

`feof()` restituisce "true" (non-zero value) se l'indicatore EOF per lo stream vale uno, altrimenti zero.

`fflush()` fa in modo che eventuali byte rimasti in memoria (buffer) a seguito di operazioni di scrittura, vengano scaricati su file.

Una invocazione di `fclose()` innesca una invocazione di `fflush()`.

Homework H22.1

1) Si generi una matrice di dimensione $N \times M$ – con N ed M a piacere – di numeri double.

2) Si codifichi una funzione che prenda in input la matrice e crei un file contenente tale matrice in formato testo. In particolare, la prima riga di tale file dovrà contenere i due numeri N ed M.

3) Si codifichi una funzione in grado di leggere un file come quello creato al punto precedente e quindi di caricare una tale matrice in memoria.

4) Si codifichi una ulteriore funzione che, data una matrice $N \times M$, ne calcoli la sua trasposta.

5) Si usi infine la funzione codificata al punto due per salvare tale matrice trasposta su file.