Lettura e scrittura a stringhe

- Funzioni nella Libreria Standard per lettura e scrittura di un'intera linea di un file testo
  - **`char *fgets(char *S, int n, FILE *FP)`**
  - **`int fputs(char *S, FILE *FP)`**
  - **`char *gets(char *S)`**
  - **`int puts(char *S)`**

fgets

- **`char *fgets(char *S, int n, FILE *FP)`**
  - Memorizza nella stringa puntata da **`S`** caratteri letti dal file puntato da **`FP`** fino a **`\n`** o al fine file o fino a quando non ha letto **`n`**-1 caratteri.
  - Un'eventuale **`\n`** letto viene memorizzato.
  - La stringa viene terminata con **`\0`**
  - Restituisce un puntatore al primo elemento della stringa o **`NULL`** in caso di errore

fputs

- **`int fputs(char *S, FILE *FP)`**
  - Scrive sul file puntato da **`FP`** la stringa puntata da **`S`**, seguita da **`\n`**
  - L'eventuale **`\0`** della stringa non viene scritto
  - Restituisce un valore diverso da zero in caso di errore, zero altrimenti

gets

- **`char *gets(char *S)`**
  - Memorizza nella stringa puntata da **`S`** i caratteri letti da stdin fino a **`\n`** o al fine file
  - Un'eventuale **`\n`** letto viene memorizzato.
  - La stringa viene terminata con **`\0`**
  - Restituisce un puntatore al primo elemento della stringa o **`NULL`** in caso di errore

puts

- **`int puts(char *S)`**
  - Scrive su stdout la stringa puntata da **`S`**, seguita da **`\n`**
  - L'eventuale **`\0`** della stringa non viene scritto
  - Restituisce un valore diverso da zero in caso di errore, zero altrimenti

Esempio

- Problema: dato un file “origine”, creare un file “destinazione” con lo stesso contenuto del file origine ma con spaziatura doppia
- Soluzione: Dopo ogni riga letta dal file origine, si scrive un **`\n`** nel file destinazione
- La lettura dal file origine si termina quando **`fgets`** restituisce **`NULL`**. Ciò segnala la fine del file.

Esempio (cont.)

- Il problema posto si risolve facilmente anche utilizzando **`getc`**, **`putc`**
- In altri casi, l'uso di **`fgets`**, **`fputs`** semplifica la soluzione

Esempio

- Confronto di due file di testo⇔ stampare la prima linea di entrambi in cui differiscono
- Il confronto tra stringhe si effettua tramite la funzione **`strcmp`** della Libreria Standard

strcmp

- **`#include <string.h>`**

  `int strcmp(char *s1, char *s2);`
- Confronta **`s1`** e **`s2`**
- Restituisce un intero minore di, uguale a, o maggiore di zero se **`s1`** è, rispettivamente, minore di, uguale a, o maggiore di **`s2`**

Input/Output formattato

- Avviene secondo le modalità simili a quelle introdotte per I/O standard con **`scanf`**, **`printf`**
- **`fprintf`** converte, formatta e scrive su file
- **`fscanf`** legge da file, formatta e memorizza

fscanf

- **`int fscanf(FILE *FP, char * format,...)`**
  - Legge il file puntato da **`FP`** e converte secondo il formato specificato dalla stringa puntata da **`format`**
  - I risultati delle conversioni sono memorizzati agli indirizzi che seguono la stringa **`format`**, nell'ordine in cui sono scritti
  - Restituisce il numero di conversioni effettuate; zero indica presenza di input, ma impossibilità di effettuare conversioni; **`EOF`** indica impossibilità di leggere prima che qualunque conversione sia tentata (es. a fine file)

fscanf

- Specificatori di formato nella stringa
  - Ogni specificatore è introdotto dal carattere **%**
  - **d** rappresenta un intero con segno
  - **e**,**f** un reale a virgola mobile
  - **s** una sequenza di caratteri diversi da spazio, **\t**, **\n**

fprintf

- **`int fprintf(FILE *FP, char * format,...)`**
- Analoga a **`printf`**