21. I/O e file

Corso di Algoritmi e Linguaggi di Programmazione Python/C

Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- Il concetto di stream
- Le funzioni **`scanf`** e **`printf`**
- File in C
- Le funzioni **`fopen`** ed **`fclose`**
- Le funzioni **`fscanf`** ed **`fprintf`**
- La funzione **`feof`**

Il concetto di stream

- Il concetto di **stream** viene utilizzato dal linguaggio C per aprire un ‘canale di comunicazione’ verso un dispositivo di I/O (e, quindi, verso l’utente)
- Lo stream è quindi associato a qualsiasi **sorgente in input** o **destinazione in output**

Le funzioni `scanf` e `printf`

- Due delle funzioni base dell'header **`stdio.h`**
  - *In generale, tutte le funzioni che tratteremo sono incluse in questo header!*
- Abbiamo già visto la **`printf`**: sappiamo che serve a stampare un messaggio a schermo
- Ha una sintassi di questo tipo:

```
printf(char* format, args)
```

- dove **`format`** è una stringa formattata, mentre **`args`** sono gli (eventuali) argomenti passati ed organizzati secondo diversi format specifier

Le funzioni `scanf` e `printf`

- La funzione **`scanf`** è duale alla **`printf`**
- Serve ad associare un valore dato in input dall'utente ad una variabile
- Ha una forma del tipo:

```
scanf(char* format, vars)
```

- dove **`format`** è il formato associato alla variabile, mentre **`vars`** sono le variabili che saranno popolate.
- La funzione **`scanf`** restituisce un valore intero, che è minore o uguale a zero nel caso qualcosa sia andato 'storto'.
  - *È quindi opportuno fare un controllo sul valore restituito!*

Le funzioni `scanf` e `printf`

- Supponiamo di voler associare una variabile in input da riga di comando al numero intero a. Per farlo, utilizzeremo la **`scanf`**.

```
if (scanf('%d', &a) == 1)
    return -1;
```

  - *Dato che il format specifier* **`%d`** *si aspetta un puntatore ad intero, in questo caso passiamo proprio questo, e non il valore dell'intero.*
  - *Lo stesso vale per gli altri formati di tipo numerico.*

- Se invece volessimo popolare una stringa:

```
if (scanf('%s', stringa) == 1)
    return -1;
```

- È importante sottolineare come stringa debba essere inizializzata per evitare dei comportamenti inaspettati.

Le funzioni `scanf` e `printf`

- ***Esercizio 1***: creiamo un programma che sia in grado di stampare a schermo il nostro nome e cognome, oltre che la nostra età. Per farlo, utilizziamo opportunamente le funzioni **`scanf`** e **`printf`**.

I file in C

- In C possiamo avere due tipi di file
- I **file binari** sono file composti esclusivamente da bit
- I **file di testo** sono classici file testuali, composti da una serie di righe
- Per interagirvi, dobbiamo usare il concetto di stream
- Usiamo quindi un apposito tipo di variabile, che rappresenta un puntatore al file, ed è possibile reperire da **`stdio.h`**:

```
FILE* fp
```

Le funzioni `fopen` ed `fclose`

- Come suggeriscono i nomi, le funzioni **`fopen`** ed **`fclose`** ci permettono di aprire e chiudere uno stream verso uno specifico file
- La **`fopen`** ha una sintassi di questo tipo:

```
fopen(FILE* fp, char* file_name, char* mode)
```

- dove **`mode`** indica la modalità di accesso al file
- È importante controllare che lo stream sia stato effettivamente aperto!
  - *Per farlo, controlliamo che non sia NULL; in caso contrario, restituiremo un errore*

```
if (fopen(fp, ‘esempio.txt.’, ‘w’) == NULL)
    return -1;
```

Le funzioni `fopen` ed `fclose`

Modo	Descrizione	Crea file?
`r`	Apertura file in lettura	No, il file deve esistere
`r+`	Apertura file in lettura/scrittura	No, il file deve esistere
`w`	Creazione file in scrittura	Sì. Se il file esiste, viene cancellato il contenuto
`w+`	Creazione file in lettura/scrittura	Sì. Se il file esiste, viene cancellato il contenuto
`a`	Aggiunge contenuto al termine del file	Sì. Se il file esiste, viene cancellato il contenuto
`a+`	Aggiunge contenuto e legge al termine del file	Sì. Se il file esiste, viene cancellato il contenuto

Le funzioni `fopen` ed `fclose`

- Oltre ai modificatori precedenti, si può specificare se il file è binario (b) o di testo (t)
  - *Di default, i file sono considerati di testo*
- La funzione **`fclose`** è la duale di **`fopen`**
- Viene utilizzata per chiudere lo stream, ed ha la seguente sintassi:

```
fclose(fp)
```

- È importante sottolineare come la **`fclose`** sia, nei fatti, chiamata anche quando viene terminato il programma.

Le funzioni `fprintf` ed `fscanf`

- Queste funzioni sono l'analogo su file delle (quasi omonime) **`printf`** e **`scanf`**
  - *La sintassi è molto simile, ma richiedono come primo argomento il puntatore a file*
- Avranno quindi una sintassi del tipo:

```
fprintf(FILE* fp, char* format, args)
fscanf(FILE* fp, char* format, vars)
```

- **Nota**: la **`fscanf`** (come la **`scanf`**) riconosce gli argomenti in base alla presenza degli spazi.
- **Nota**: entrambe le funzioni restituiscono dei valori interi, che vanno controllati come nelle controparti (specialmente la **`fscanf`**)

La funzione `feof`

- Questa funzione ci permette di sapere se siamo arrivati alla fine del file
- Questa è contraddistinta da una costante EOF
- È bene utilizzarla!

Domande?

42