



Input/Output File

Ing. Nadia Ranaldo
 Dipartimento di Ingegneria
 Università degli Studi del Sannio

Modello di input e output



- La libreria `<stdio.h>` fornisce un modello molto semplice di input e output riguardante i testi.
- Un flusso di testo consiste in **una sequenza di linee**
 - Ogni linea termina con un carattere di new line
- Finora abbiamo visto le funzione `scanf` e `printf` che permettono l'input e l'output "formattato"
- La funzione `scanf` opera sullo standard input (`stdin`), ovvero il dispositivo di ingresso di default, che normalmente coincide con la tastiera
- La funzione `printf` opera sullo standard output (`stdout`), ovvero il dispositivo di uscita di default, che normalmente coincide con il video

2

Funzione printf



```
int printf(char *fmt, ...);
```

- Scrive sul canale di output una serie di valori, effettuando le conversioni richieste ove necessario, utilizzando la stringa di formato `fmt`

- Restituisce il numero di caratteri emessi

```
#include <stdio.h>
main() {
int a;
printf("Immettere un carattere: ");
a = getchar();
printf("%c rappresenta %d come
intero decimale, %o in ottale
e %x in hex", a, a, a, a);
}
```

Caratteri di formattazione	Comportamento	Esempio
c	carattere	A
d	intero	120
u	intero decimale senza segno	12
o	ottale senza segno	56
x,X	esadecimale senza segno	0xA06
e,E	floating point	7.12e+00
f	floating point	7.12
g	più breve formato tra e ed f	
G	più breve formato tra E ed f	
s	stringa	Ciao!
p	puntatore (hex)	0xA06
%	carattere %	

Funzione scanf



```
int scanf(char *fmt, ...);
```

- Legge dal canale di input una serie di valori e li memorizza in variabili, effettuando le conversioni richieste ove necessario

- Restituisce il numero di campi letti e memorizzati correttamente

- In questo caso la **stringa di formato** `fmt` è una sequenza di specifiche **%carattere di formattazione**

- Un campo è separato dall'altro da un qualsiasi carattere di separazione (new line, spazio, tabulazione)

- Ad esempio:

```
scanf("%d %d %f", ...);
```

Funziona correttamente con i seguenti input:

```
"15 38 13.2" e
"32
3 3.4"
```

- Specificando altri caratteri tra i vari campi da leggere nella stringa di formato, questi devono essere necessariamente letti in input, esattamente come indicato altrimenti si verifica un errore

- Ad es. `scanf("%d/%d/%d", &g, &m, &a);`

```
"12/12/1993" ok "12 / 1/2000" NO! (solo 12 viene letto correttamente)
```

4



I/O a caratteri



- Poiché sui canali di I/O fluiscono sequenze di caratteri, il modello di I/O prevede anche due operazioni base:

- Scrivere un carattere sullo standard output

```
int putchar(int ch);
```

- Restituisce il carattere scritto, o EOF in caso di errore nell'eseguire l'operazione di scrittura

- Leggere un carattere dallo standard input

```
int getchar(void);
```

- Restituisce il codice ASCII del carattere letto, o EOF (end-of-file) in caso di errore o nel caso in cui la sequenza di input sia finita
- La funzione restituisce caratteri dopo aver inserito un new line

- Ogni altro tipo di operazione di I/O può essere costruita a partire da queste operazioni primitive

- Entrambe le funzioni leggono/scrivono un carattere convertito in int ⁵



Esempio



- Esempio: lettura dalla tastiera e scrittura a video (poiché la funzione getchar restituisce i caratteri dopo aver inserito lo \n, la funzione putchar visualizza a video un'intera linea letta precedentemente)

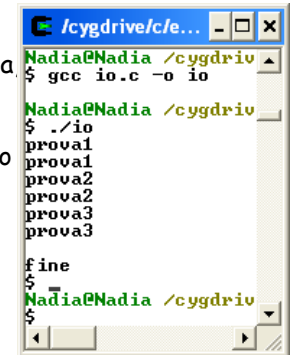
```
#include <stdio.h>
main(){
    int c;
    while(( c=getchar() ) != EOF)
        putchar(c);
}
```

- Per chiudere l'input producendo un EOF da tastiera, Windows, CTRL+D in Unix

- CTRL+D in Cygwin

- Esempio: lettura di una sola linea e scrittura a video

```
#include <stdio.h>
main(){
    int c;
    while(( c=getchar() ) != '\n')
        putchar(c);
}
```



Gestione dei file



- Un file permette di memorizzare in maniera permanente dei dati
 - La memorizzazione in variabili, array, etc. è solo temporanea
 - Quando il programma termina, il contenuto della memoria viene perso
- Ad esempio un file potrebbe essere utilizzato per memorizzare le informazioni di un insieme di strutture
 - Strutture corsoT, puntoT, matriceT, etc.

- Un insieme di file con informazioni correlate tra loro può anche essere chiamata **database**

Sally	Black	
Tom	Blue	
Judy	Green	
Iris	Orange	
Randy	Red	

File

- Nella libreria standard <stdio.h> viene definita una struttura **FILE** che contiene le informazioni per accedere ad un file (**descrittore di file**)
- Ogni file viene acceduto attraverso un puntatore ad una struttura **FILE**
- Nella libreria <stdio.h> sono definiti i descrittori di file (variabili di tipo puntatore a **FILE**):

- stdin (default: la tastiera)
- stdout (default: il video)
- stderr (default: il video)

```
FILE *f;
...
```

- Il C vede un file come una sequenza di byte

7



Apertura e chiusura di un file



- Per poter leggere o scrivere su un file occorre prima di tutto:
 - Dichiarare un puntatore a file


```
FILE *f;
```
 - Eseguire l'operazione di apertura mediante la funzione fopen


```
f = fopen("myFile.txt", openmode);
```

 - La funzione fopen restituisce un puntatore a FILE per il file specificato
 - Riceve due argomenti: il nome del file da aprire e la modalità di apertura del file
 - Se l'apertura fallisce, la funzione restituisce NULL
- Al termine di tutte le operazioni, il file deve essere chiuso mediante la funzione
 - fclose(FILE* f)
 - Al termine del programma i file vengono chiusi automaticamente
 - E' buona pratica di programmazione chiudere i file esplicitamente
- Altra funzione utile
 - feof(FILE *f)
 - Restituisce TRUE se si è raggiunti la fine del file (EOF)

8



Mode	Descrizione
r	Apertura di un file in lettura
w	Crea un file per la scrittura. Se il file esiste, il precedente contenuto viene cancellato
a	Crea un file per la scrittura. Se il file esiste, la scrittura inizia alla fine del contenuto esistente (append)
r+	Apri un file per la modifica (lettura e scrittura)



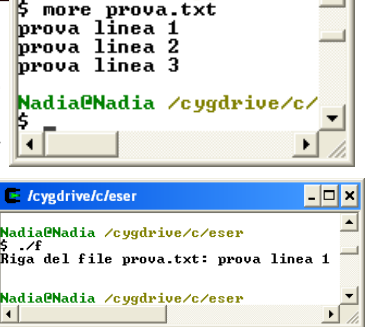
- Funzioni di lettura e scrittura nella libreria standard <stdio.h>
 - `int fgetc (FILE *f);`
 - Legge un carattere da un file
 - Riceve come argomento un puntatore a FILE f
 - Restituisce il codice ASCII del successivo carattere contenuto in f oppure EOF se incontra la fine del file
 - Generalizza la funzione getchar() che funziona solo sullo standard input
 - fgetc(stdin) equivale a getchar()
 - `int fputc(int c, FILE *f);`
 - Scrive un carattere su un file
 - Riceve come argomenti un puntatore a FILE f e il carattere c da scrivere
 - Restituisce il carattere scritto, oppure EOF in caso di errore
 - Generalizza la funzione putchar() che funziona solo sullo standard output
 - fputc('a', stdout) equivale a putchar('a')



- `char *fgets(char *s int n, FILE *f);`
 - Legge una linea da un file di al più n-1 caratteri
 - Riceve come parametri un puntatore a FILE f, la stringa s in cui memorizzare la linea letta e il numero n di caratteri da leggere
 - La funzione blocca prima la lettura se trova un new line
 - A differenza della funzione gets, inserisce anche il carattere new line nella stringa
 - Aggiunge il carattere '\0' alla fine della stringa
 - Restituisce s, oppure NULL se incontra la fine del file o se rileva un errore
- `int fputs(char *s, FILE *f);`
 - Scrive una linea su un file
 - Riceve come parametri un puntatore a FILE f la stringa s da scrivere nel file
 - A differenza della funzione puts, non aggiunge un new line
 - Restituisce un valore non negativo, oppure EOF se si verifica un errore
- `fscanf / fprintf`
 - Funzioni equivalenti a scanf e printf ma che operano su un qualsiasi file
 - Riceve un arg. in più, ovvero il puntatore a FILE (come primo argomento)
 - fscanf restituisce un intero che vale EOF quando viene raggiunta la fine del file

- File aperto in modalità lettura
 - Sono consentite solo operazioni di lettura
 - Operazioni di scrittura non producono nessun effetto
 - Esempio: lettura della prima linea dal file "prova.txt"
- ```

#include <stdio.h>
#define N 30
main(){
 FILE *f;
 char linea[N];
 f = fopen("prova.txt", "r");
 if (f==NULL){ /* controllo se il file esiste */
 printf("Errore durante l'apertura del file, verificare il nome
 del file");
 return;
 }
 /* Lettura di una linea del file (in questo caso la prima) */
 fgets(linea, N, f);
 printf("Riga del file prova.txt: %s \n", linea);
 fclose(f);
}

```
- 



- **File aperto in modalità scrittura**
- Sono consentite solo operazioni di scrittura
- Operazioni di lettura leggono valori indefiniti
- **Esempio:** scrittura di una stringa nel file "provascrittura.txt" non ancora esistente

```
#include <stdio.h>
main(){
 FILE *f;
 f = fopen("provascrittura.txt", "w");
 if (f==NULL) { /* Controllo se ci sono prob. nell'apertura */
 printf("Errore durante l'apertura del file, verificare i
 permessi di scrittura nella directory corrente");
 return;
 }
 /*Scrittura di una linea nel file */
 fputs("linea 1 di prova", f);
 fclose(f);
}
```



- Eseguendo più volte il programma precedente, il contenuto precedente viene sempre cancellato, quindi il file conterrà sempre una sola linea!
- **File aperto in modalità append**
- La scrittura inizia il coda al contenuto già esistente
- **Esempio:** scrittura di una linea nel file "provaappend.txt"
- Controllo se ci sono problemi durante l'apertura del file
- Scrittura di una linea nel file

```
#include <stdio.h>
main(){
 FILE *f;
 f = fopen("provaappend.txt", "a");
 if (f==NULL){
 printf("Errore durante l'apertura del file,
 verificare i permessi di scrittura nella directory corrente");
 return;
 }
 fputs("linea 1 di prova\n", f);
 fclose(f);
}
```

Dopo aver eseguito il programma tre volte →



- **File aperto in modalità lettura r+**
- Sono possibili operazioni di lettura e di scrittura
- La scrittura inizia il coda al contenuto già esistente
- **Esempio:** lettura della prima linea del "provaappend.txt", conversione in maiuscolo di tale stringa e scrittura nel file

```
#include <stdio.h>
void converti_maiuscolo(char *s);
main(){
 FILE *f;
 char linea[N];
 f = fopen("provaappend.txt", "r+");
 if (f==NULL){
 printf("Errore durante l'apertura del file, verificare i
 permessi di scrittura nella directory corrente");
 return;
 }
 fgets(linea, N, f);
 converti_maiuscolo(linea);
 fputs(linea, f);
 fclose(f);
}
```



- Scrivere un programma che stampa a video il contenuto di un file
- La lettura deve essere ripetuta fino a che non viene letto **tutto** il file
- La funzione fgets restituisce NULL quando incontra la fine del file

```
#include <stdio.h>
#define N
main(){
 FILE *f;
 char linea[N];
 f = fopen("prova.txt", "r");
 if (f==NULL) {
 printf("Errore durante l'apertura del file, verificare il nome
 del file");
 return;
 }
 while(fgets(linea,100, f)!=NULL) {
 printf("%s", linea);
 }
}
```



- Lettura da un file di un insieme di studenti
- Scrittura su un file di un insieme di studenti
- Ogni riga contiene i dati di uno studente: nome, cognome e voto
- Utilizziamo un array di strutture `studenteT`
- Una funzione per la lettura e una per la visualizzazione

```
#include<stdio.h>
#define N 30
#define MAX 20
typedef struct {
 char nome[N];
 char cognome[N];
 int voto;
} studenteT;
int leggiStudenti(studenteT studenti[], int cap);
void visualizza(studenteT studenti[], int riemp);
void memorizza(studenteT studenti[], char *nomeFile, int riemp);
main(){
 int riemp;
 studenteT stud_ing[MAX];
 riemp = leggiStudenti(stud_ing, MAX);
 visualizza(stud_ing, riemp);}
```

17

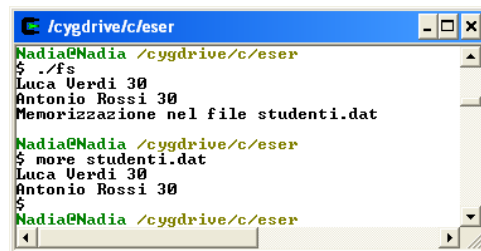


```
int leggiStudenti(studenteT studenti[], int cap){
 FILE *f;
 int fine, i=0;
 f = fopen("studenti.txt", "r");
 if (f==NULL) {
 printf("Err. apertura file, verificare il nome del file");
 return;
 }
 while (i<cap) {
 fine = fscanf(f, "%s %s %d", studenti[i].nome, studenti[i].cognome,
 &(studenti[i].voto));
 if (fine ==EOF) break;
 i++;
 }
 return i;
}
void visualizza(studenteT studenti[], int riemp) {
 int i;
 for (i = 0; i < riemp; i++) {
 printf("%s ", studenti[i].nome);
 printf("%s ", studenti[i].cognome);
 printf("%d\n", studenti[i].voto);
 }
}
```

18



```
void memorizza(studenteT studenti[], char *nomeFile, int riemp){
 FILE *f;
 int i;
 f = fopen(nomeFile, "w");
 if (f==NULL) {
 printf("Errore durante l'apertura del file %s, verificare i
 permessi\n", nomeFile);
 return;
 }
 printf("Memorizzazione nel file %s\n",nomeFile);
 for(i=0; i<riemp; i++) {
 fprintf(f, "%s %s %d\n", studenti[i].nome, studenti[i].cognome,
 studenti[i].voto);
 }
}
```



```
~/cygdrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$./fs
Luca Verdi 30
Antonio Rossi 30
Memorizzazione nel file studenti.dat

Nadia@Nadia /cygdrive/c/eser
$ more studenti.dat
Luca Verdi 30
Antonio Rossi 30
$
```