Lettura e scrittura formattata su File

Prof. Salvatore Venticinque

Lettura e scrittura formattata

Abbiamo studiato l'utilizzo di:

- *scanf*
- *printf*

Esistono le equivalenti funzioni per la lettura/stampa su file:

- ***fscanf***
- ***fprintf***

Esempio scrittura formattata

```
FILE *fp;

int i;

...

i = 2;

fprintf(fp, "Il valore di i è %d\n", i);
```

Esempio lettura formatta

```
FILE *fp;

int i;

...

fscanf(fp,"%d", &i);
```

Pro vs Cons

- Meno efficienti per il salvataggio dei dati di un programma
- Meno comode per ripristinare il contenuto di variabili
- Più comode per la stampa di documenti che devono essere letti da un utente

Leggere un file riga per riga

```
char buf[100];
FILE *fp;
char *s;
int n;
...
n = 100;
s = fgets(buf, n, fp);
```

- legge fino a 100 caratteri o al carattere '\n', includendolo nella stringa
- aggiunge automaticamente a fine riga il carattere di fine stringa “\0”.
- ritorna **NULL** in caso di errore oppure il valore di **buf**

Scrivere una riga in un file

```
char buf[100];

FILE *fp;

...

fputs(buf,fp);
```

buf è il vettore che contiene la riga da scrivere sul file fp terminata con il carattere di fine stringa “\0”

Lettura righe in un file

```
#include <stdio.h>
int main(int argc, char **argv)
{     char buf[100];  int linee;
        FILE *fp;
        if( argc < 2 ) printf("Errato numero di parametri\n");
        else {
                fp = fopen(argv[1], "r");
        if(fp!= NULL) {
                linee = 0;
                while(fgets(buf,100,fp) != NULL)
                        Linee++;
                fclose(fp);
                printf("Il file contiene %d linee\n", linee);
        }
        else
                printf("Il file %s non esiste\n",argv[1]);
        }
}
```

Lettura e scrittura carattere

int fgetc(FILE *)

c = fgetc(fp);

- Il valore di ritorno della funzione fgetc è di tipo int
- se coincide con la costante simbolica **EOF**, definita nel file di include stdio.h, significa che si è giunti a fine file
- la costante simbolica EOF è definita in modo tale che non possa mai eguagliare alcun carattere; solitamente vale −1

Lettura e scrittura carattere

void fput(int, FILE*)

fputc('a',fp)

- La funzione scrive un carattere nel file.

Conteggio caratteri numerici in file

Per sapere se il file pointer si trova posizionato a fine file:

```
int ret;

FILE *fp;

...
ret = feof(fp);
```

- Il valore di ritorno ret conterrà il numero 1 se il file pointer è posizionato a fine file e il numero 0 in caso contrario.

Stream di default

Quando un programma va in esecuzione il sistema apre automaticamente tre file pointer:

- Standard Input (stdin),
- Standard Output (stdout)
- e Standard Error (stderr).

- printf(…) → fprintf(stdout,…)
- scanf(…) → fscanf(stdin,…)
- getchar() → fgetc(stdin)
- putchar(c) → fputc(c, stdout)
- eof( ) → feof(stdin)

gets, puts

- ***gets*** legge una riga da tastiera ma elimina il carattere di newline
- ***puts*** scrive una riga a video aggiungendo automaticamente un carattere di newline.

Ne consegue che :

- gets(buf,n) non equivale a fgets(buf, n, stdin)
- puts(buf) non equivale a fputs(buf, stdout)

Inoltre … non abbiamo il controllo sulla dimensione del buffer