Lezione 13 e 14

- Introduzione ai file
- Nozioni sulla gestione dei file
- Esempio applicativo: gestione di una agenda

Fabio Scotti (2004-2009)

Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)

Laboratorio di programmazione

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 13 e 14

Introduzione ai file

Obiettivi :

- Comprendere il significato e l'uso dei file nella programmazione
- Comprendere che tutte le operazioni di Input/Output e la gestione dei file vengono trattati in C attraverso lo stesso formalismo: lo stream
- Comprendere e distinguere il concetto di accesso sequenziale e casuale ad un file

Perché i file

- **La memorizzazione dei dati nelle variabili e nei vettori è temporanea**
- **Non appena un programma terminerà la sua esecuzione tutti i nostri dati andranno persi**
- **La funzione dei file è quella di permettere la memorizzazione di piccole e grandi quantità di dati in un elaboratore**
- **I file sono memorizzati sui dispositivi di memorizzazione secondaria quali dischi, CDROM, i DVD, le pendrive USB, e più raramente nastri**

File visti come stream

- **Le operazioni di input/output viste fino ad ora:**
  - la tastiera come dispositivo di input
  - lo schermo come dispositivo di output
  - flussi di dati gestiti con il concetto di stream
- **Il linguaggio C gestisce dei file esattamente nello stesso modo**
- **E' necessario solo aggiungere qualche nuovo concetto**

Rappresentazione delle informazioni nei file

- **Tutti i file memorizzati sul disco sono semplicemente composti da 1 e 0**
- **Perché allora si parla di file di caratteri e non solo di file binari?**
  - Un numero scritto in binario sul disco può essere letto come
    - un numero (ad esempio un intero short senza segno)
    - oppure un carattere (un elemento della tabella ASCII)
- **Anche per i file dobbiamo sapere in che modo i valori binari memorizzati sui dischi dovranno essere interpretati**

Apertura, accesso, chiusura

- **In C le operazioni di I/O vengono semplificate attraverso l'uso degli stream**
- **Gli stream sono delle astrazioni rappresentative di un file o di un dispositivo fisico, che vengono manipolate attraverso l'uso di puntatori**
- **Vantaggio: potersi riferire ad un identificatore senza sapere come questo venga implementato.**
- **Operazioni su uno stream**
  - lo si apre
  - vi si accede (nel senso di operazioni di lettura e scrittura)
  - e lo si chiude

Bufferizzazione

- **Lo stream è bufferizzato:**
  - ovvero viene riservato un buffer per evitare ritardi o interruzioni nella fase di lettura e scrittura
- **Questo metodo rende efficiente l'I/O ma**
  - i dati scritti in un buffer non compaiono nel file finché il buffer non e' riempito o scaricato ("\n" serve a questo).
  - Qualsiasi uscita anormale del programma può causare problemi.

Stream predefiniti in C

- **Gli stream predefiniti nel linguaggio C (in <stdio.h>) sono :**
  - stdin
  - stdout
  - stderr
- **Utilizzano principalmente il testo come metodo di comunicazione I/O**
  - I primi due possono essere usati con i file, con i programmi, con la tastiera, con la consolle e lo schermo.
  - Lo stderr può scrivere soltanto su console o sullo schermo.
  - La console è il dispositivo di default per stdout e stderr, mentre per stdin il dispositivo di default è la tastiera.

Le funzioni fflush e fclose

- **Gli stream, qualunque uso ne sia stato fatto, devono essere**
  - prima "puliti"
  - e poi chiusi,
- **con le funzioni `fflush` e `fclose`:**
  - `fflush(FILE *stream);`
  - `fclose(FILE *stream);`
- **Per poter operare correttamente è necessario includere l'header file**
  - **<stdio.h>**
  - che contiene tutte funzioni per l'input/output, comprese quelle che operano sui file

Funzione fopen

- **La prima cosa da fare è aprire un file; per fare ciò si usa la funzione `fopen`:**
  - `FILE *fopen(char *nome, char *modo);`
  - Es.: `fstream = fopen("libro.txt", "a+");`
- **`nome` e' il nome del file al quale si intende accedere**
- **Restituisce un puntatore all'oggetto FILE. Se non si può accedere al file, viene restituito un puntatore a NULL.**

Funzione fopen

- `FILE *fopen(char *nome, char *modo);`

- **`modo` in cui si vuole aprirlo:**
  - "r" – apre un file in lettura
  - "w" – crea un file per la scrittura, se il file inizia già, ne elimina il contenuto
  - "a" – apre o crea un file per scrivere in fondo dello stesso (append)
  - "r+" – apre un file in aggiornamento (lettura e scrittura) le operazioni saranno eseguite all'inizio del file.
  - "a+" – apre o crea un file per lettura e scrittura, le operazioni saranno eseguite alla fine del file.

Esempio

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   /* dichiarazione dello stream */
   FILE *stream;

   /* apertura dello stream */
   stream = fopen("miofile.txt", "w");
   if (stream == NULL) // controlla se il file viene aperto
   {
      printf("Non posso aprire il file %s\n", "miofile.txt");
      exit(-1);
   }

   /*******************************/
   /* Codice che lavora sul file */
   /*******************************/

   /* chiusura dello stream*/
   fflush(stream);
   fclose(stream);
   return 0;
}
```

Funzioni fscanf e fprintf (1)

- **Le funzioni che useremo per leggere e scrivere i file sono `fscanf` e `fprintf`.**
- **Il loro funzionamento è del tutto simile a quello delle funzioni `scanf` e `printf`**
- **Al posto di leggere dalla tastiera e stampare a schermo:**
  - **`fscanf` e `fprintf`** leggono da file e stampano da file
- **Il prototipo delle due funzioni è il seguente:**

```
int fprintf(FILE *stream1,char *formato,argomenti);
  Es. fprintf(file1, "%s", cognome);
int fscanf(FILE *stream2,char *formato,argomenti);
  Es. fscanf(file1, "%s", cognome);
```


Funzioni fscanf e fprintf (2)

```
int fprintf(FILE *stream1,char *formato,argomenti);
int fscanf(FILE *stream2,char *formato,argomenti);
```

- **La `fprintf` scrive sullo stream `stream1`, (un file, il terminale, un socket, ecc.)**
- **La `fscanf` legge dallo stream `stream2` (un file, la tastiera, un socket, ecc.);**
- **La stringa `formato` ha due tipi di argomenti:**
  - i caratteri ordinari che vengono copiati nello stream di output
  - le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare le variabili presenti nella lista di argomenti.

Funzioni fscanf e fprintf (3)

Stringa di controllo	*Cosa viene stampato*
%d, %i	Intero decimale
%f	Valore in virgola mobile
%c	Un carattere
%s	Una stringa di caratteri
%o	Numero ottale
%x, %X	Numero esadecimale
%u	Intero senza segno
%f	Numero reale (float o double)
%e, %E	Formato scientifico
%%	Stampa il carattere %

Funzioni fscanf e fprintf (4)

- **La `fprintf` e la `fscanf` possono scrivere negli stream predefiniti,**
  - `stdout, stderr e stdin`
- **Esempio**

```
-fprintf(stderr, "Impossibile continuare!\n");

-fprintf(stdout, "Operazione completata!\n");
```

equivalente a

```
printf("Operazione completata!\n");

-fscanf(stdin, "%s", miastringa);
```

equivalente a

```
scanf("%s", miastringa);
```

Leggere e scrivere un carattere su file

- **Esistono funzioni che operano su file scrivendo un carattere per volta, come fanno la `getchar` e la `putchar` per gli stream predefiniti:**
  - `int getc(FILE *stream);`
  - `int fgetc(FILE *stream);`
  - `int putc(char ch, FILE *stream);`
  - `int fputc(char ch, FILE *stream);`
- **`getc` e `putc` sono macro del preprocessore**
- **`fgetc` e `fputc` sono funzioni di libreria**
- **Si comportano nello stesso modo**
- **Nel corso useremo sempre fscanf e fprintf**

Cancellare e rinominare un file

- **Per cancellare un file si usa**
  - `int remove(char * nomefile);`
  - Non si può eliminare un file aperto
- **Per rinominare un file si usa**
  - `int rename(char *vecchionome, char *nuovonome);`
  - Non si può rinominare un file aperto
- **Esempi:**

```
// se abbiamo una stringa con indirizzo nomefile
remove(nomefile);
// se abbiamo nella dir corrente il file f1.txt
rename("f1.txt" , "f1modif.txt");
```

Funzioni fseek() e rewind() (1)

- **Le due funzioni fseek e rewind che permettono di riposizionare la testina (cursore) all'interno del file**
- **Permettono un ACCESSO CASUALE (diretto) all'interno del file leggendo i record (struttura elementare di un file dati)**
- **La funzione rewind permette di posizionare la testina all'inizio del file**
  - `void rewind(FILE *f);`

Funzioni fseek()

- **La funzione `fseek` consente di spostare la testina di lettura su un qualunque byte del file:**
  - `int fseek (FILE *f, long offset, int origin);`
  - Es: `fseek( file1, 28, 1);`
- **Sposta la testina di `offset` byte a partire dalla posizione `origin` (che vale 0, 1 o 2)**
- **Origine dello spostamento (costanti definite in `stdio.h`)**
  - 0 → inizio file, SEEK_SET
  - 1 → posizione attuale nel file, SEEK_CUR
  - 2 → fine file, SEEK_END
- **Se lo spostamento ha successo ritorna 0**

Funzione feof()

- **La funzione `feof` consente di sapere se il puntatore al file è posizionato alla fine del file**
  - `int feof (FILE *f);`
- **Restituisce vero se f punta alla fine del file falso altrimenti.**

Etichetta EOF

- **L'etichetta `EOF` definita in stdio.h**
  - viene utilizzata per rappresentare un marcatore di fine file (dipendente dal sistema)
- **Fisicamente il carattere corrispondente alla etichetta EOF si trova alla fine del file come ultimo carattere**
- **Esempio** Per stampare su monitor il contenuto di un file è utile usare una delle seguenti istruzioni:

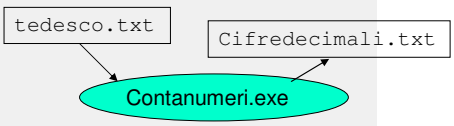
```
while(!feof(fp))   {fscanf(fp,"%c",&c); printf("%c",c);}
while(fscanf(fp,"%c", &c) == 1)  { printf("%c",c); }
while((c= fgetc(fp)) != EOF)     { printf("%c",c); }
```

Contanumeri.c (1)

- **Problema: dato il file `tedesco.txt` contare il numero di cifre decimali presenti nel file, e copiarle nel file `CifreDecimali.txt`.**

```
Schlechte Aussichten für deutsche Konjunktur

Der Geschäftsklimaindex des ifo-Instituts ist im März erneut überraschend
deutlich gefallen. Ifo-Chef Hans-Werner Sinn sagte, er sei zunehmend
besorgt, dass der Aufschwung ins Stocken gerate - sein Institut werde wohl
die Wachstumsprognose für Deutschland absenken müssen.


DDP
Banken in Frankfurt am Main: Für den Aufschwung sieht es düster aus
München - Der Indikator fiel von 96,4 Punkten im Vormonat auf 95,4 Punkte.
Volkswirte hatten hingegen mit einem Rückgang auf nur 95,7 Punkte
gerechnet. Bis Januar war der Indikator neun Mal in Folge gestiegen bevor
er sich im Februar überraschend wieder eintrübte.

Die Geschäftserwartungen verschlechterten sich von 100,3 Punkten im Februar
auf 98,9 Punkte im März. Die Lagebeurteilung habe sich ebenfalls
eingetrübt. Sie sei von 92,6 auf 92,1 Punkte gefallen.

....
```

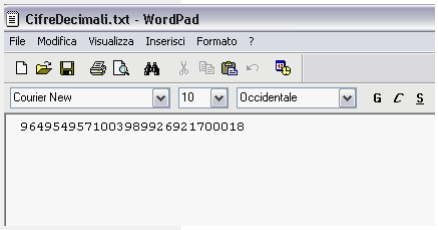

Contanumeri.c (2)




```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   /* dichiarazione */
   FILE *pfile_in, *pfile_out;

   int nc;
   char c;

   /* apertura */
   pfile_in = fopen("tedesco.txt", "r"); //in lettura
   if (pfile_in == NULL) // controlla se il file viene aperto
   {
      printf("Non posso aprire il file %s\n", "tedesco.txt");
      exit(-1);
   }

   pfile_out = fopen("CifreDecimali.txt", "w"); //in scrittura
   if (pfile_out == NULL) // controlla se il file viene aperto
   {
      printf("Non posso aprire il file %s\n", "CifreDecimali.txt");
      exit(-1);
   }
```

Contanumeri.c (3)

```
   /* elaborazione */
   nc = 0;
   while(!feof(pfile_in))
   {
      fscanf(pfile_in,"%c",&c);
      if (c >= '0' && c <= '9')
      {
         nc++;
         fprintf(pfile_out,"%c",c);
      }
   }

   printf("Numero di caretteri numerici: %d \n", nc);

   /* chiusura */
   fflush(pfile_in);
   fclose(pfile_in);

   fflush(pfile_out);
   fclose(pfile_out);

   getchar();
   return 0;
}
```



Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 13 e 14

Programmi per la gestione dei file

Obiettivi :

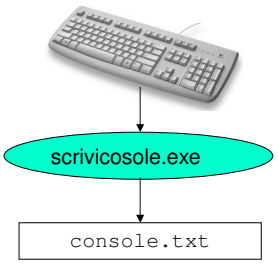
- Essere in grado di scrivere semplici programmi impieganti file
- Approfondire gli aspetti delle operazioni riguardanti i file mediante esempi di codice

Obiettivi

- **Scrivere un programma che salva in un file tutto quello che si scrive sulla tastiera.**
- **Capire le differenze fra lettura e scrittura di un file in modalità binaria o testuale.**

Scriviconsole.c (1)

```
// scriviconsole.c
// scrive in un file  tutto quello
// che viene digitato da console

#include <stdio.h>
#define NOME_FILE "console.txt"
main()
{
    FILE *file_out;
    char c;

    file_out = fopen(NOME_FILE, "w");
    if(file_out == NULL)
    {
      printf( "Non trovo il file.\n" );
      exit(-1);
    }
    printf("(Per terminare premi @)\n\n");
    scanf("%c",&c);
    while(c != '@')
    {
        fprintf(file_out,"%c",c);
        scanf("%c",&c);
    }
   fflush(file_out);
   fclose(file_out);
}
```



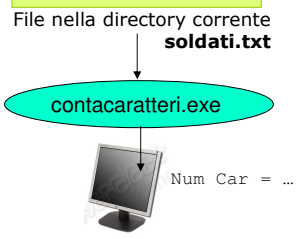
Modalità testuale e binaria

- **Un file si può aprire in modalità testuale:**
  `fopen(NOME_FILE, "rt");`
  oppure binaria:
  `fopen(NOME_FILE, "rb");`
- **In modalità testo i caratteri prima di essere posti nel buffer vengono interpretati.**
  Esempio:
  il fineriga, composto dai due caratteri LF (Line Feed: 10 o '\n') e CR (Carriage Return: 13 o '\r'), viene trasformato in LF.
- **In modalità binaria non viene operata nessuna trasformazione e tutti i caratteri vengono letti.**

contacaratteri.c (1)

```
// contacaratteri.c

#include <stdio.h>
#include <stdlib.h>
#define NOME_FILE "soldati.txt"

int main()
{
    FILE *file_in;
    char c;
    unsigned long numero_caratteri;

    file_in = fopen(NOME_FILE, "rt");
    if(file_in == NULL)
    {
      printf( "Non trovo il file.\n" );
      exit(-1);
    }
    numero_caratteri=0;
    while(fscanf(file_in, "%c", &c) == 1)
        numero_caratteri++;
    printf("[testo]: Il file e' composto da %u caratteri\n", numero_caratteri);
    fflush(file_in); fclose(file_in);
}
```

Si sta come
d'autunno
sugli alberi
le foglie.



contacaratteri.c (2)

```
  // riapriamo il file in modalità binaria
  // e rifacciamo le stesse operazioni

   file_in = fopen(NOME_FILE, "rb");
   if(file_in == NULL)
   {
     printf( "Non trovo il file.\n" );
     exit(-1);
   }
   numero_caratteri=0;
   while(fscanf(file_in, "%c", &c) == 1)
       numero_caratteri++;
   printf("[binaria]: Il file e' composto da %u caratteri\n", numero_caratteri);
   fflush(file_in);
   fclose(file_in);
}
```

```
C:\>contacaratteri.exe
[testo]: Il file è composto da 46 caratteri
[binaria]: Il file è composto da 50 caratteri
```


Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 13 e 14

Programma per la gestione di una agenda

Obiettivi :

- Essere in grado di impiegare i file all'interno di programmi complessi
- Essere in grado di passare alle funzioni le informazioni riguardanti i file

Obiettivi

- **Scrivere un programma che gestisce una agenda di indirizzi.**
- **Il programma deve poter caricare da file gli indirizzi, modificare i dati degli utenti registrati e salvare su un file le modifiche.**
- **Prime nozioni dell'input/output formattato.**

FILE rubríca.txt

Mario	Rossi	2323423423
Anna	Bellini	02324234
Massimo	Monti	0372323434
…		

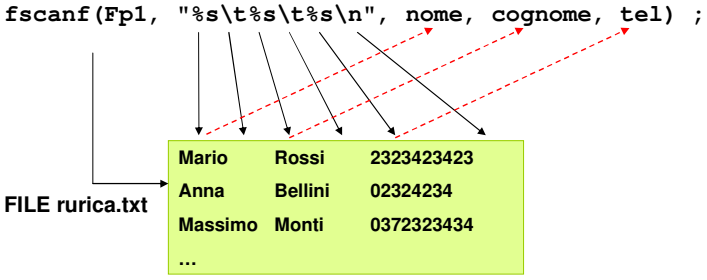
Lettura di record da file (1)

Come leggere o scrivere dei record da un file di testo?

Con `fprintf` e `fscanf` e una opportuna stringa di controllo.

Iniziamo con un'operazione di lettura:

```
char nome[128], cognome[128], tel[18];
…
fscanf(Fp1, "%s\t%s\t%s\n", nome, cognome, tel) ;
```



FILE rurica.txt

Mario	Rossi	2323423423
Anna	Bellini	02324234
Massimo	Monti	0372323434
…		

Lettura di record da file (2)

- **Come scandire tutto il file?**
- **E come ci si accorge di essere giunti in fondo al file?**

Usando un ciclo ed esaminando cosa ritorna la `fscanf`:

```
char nome[128], cognome[128], tel[18];
…
while(fscanf(Fp1, "%s\t%s\t%s\n", nome, cognome, tel) == 3 )
    {
        printf("%s %s, %s\n", nome, cognome, tel);
    }
```

FILE rurica.txt

Mario	Rossi	2323423423
Anna	Bellini	02324234
Massimo	Monti	0372323434
…		

agenda.c (apertura file) 1/4

```
#include <stdio.h>

int main()
{
    char nome[256], cognome[256], tel[256];
    char finito;
    char nomefile[]="rubrica.txt"   ;
    FILE *Fp1;


   // Apro il file in modalita' append testo
    Fp1 = fopen(nomefile, "a");
    if (Fp1==NULL){
        printf("File %s not found\n", nomefile);
        exit(-1);
    }

   printf("Il puntatore (*Fp) al file %s e' %d\n\n", nomefile, Fp1);
```

agenda.c (inserimento dati) 2/4

```
// Salvo nel file i dati delle persone


finito='n';
while((finito=='n') || (finito=='N'))
 {
     printf("Inserire Nome: "); scanf("%s", nome);
     printf("Inserire Cognome: "); scanf("%s", cognome);
     printf("Inserire TEL: "); scanf("%s", tel);

     fprintf(Fp1, "%s\t%s\t%s\n", nome, cognome, tel);
     fflush(stdin);

     printf("Finito? S(i) oppure N(o) --> ");
     scanf("%c", &finito);
 }
```


agenda.c (stampa file) 3/4

```
   // Chiudo il file in scrittura

close(Fp1);

Fp1 = fopen(nomefile, "r");
if (Fp1==NULL)
{
        printf("File %s not found\n", nomefile);
        exit(-1);
}

printf("\n\n\nRubrica:\n");
while(fscanf(Fp1, "%s\t%s\t%s\n", nome, cognome, tel) == 3 )

{

       printf("%s %s, %s\n", nome, cognome, tel);

}
```

agenda.c (output) 4/4

Uscita del programma `agenda.exe`:

```
C:\>agenda.exe
Il puntatore (*Fp) al file rubrica.txt e' 2013505264

Inserire Nome: Mario
Inserire Cognome: Rossi
Inserire TEL: 037235467
Finito? S(i) oppure N(o) --> n
Inserire Nome: Valeria
Inserire Cognome: Bianchi
Inserire TEL: 0373123456
Finito? S(i) oppure N(o) --> s


Rubrica:
Mario Rossi, 037235467
Valeria Bianchi, 0373123456
```


Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 13 e 14

Ripasso delle funzioni

Obiettivi :

- Ripassare il concetto di funzione
- Comprendere l'utilizzo del passaggio dei parametri per indirizzo

Restituire valori con una funzione

- **Una funzione può restituire uno o più risultati:**
  - con un return
  - scrivendo nelle variabili passate per indirizzo
- **E' possibile eseguire un solo return all'interno di una funzione**
- **Per restituire più di un valore**
  - la funzione scrive sulle variabili passate per indirizzo

square.c (uso del return)

```c
#include <stdio.h>
#include <stdlib.h>

int square( int y ) ; // prototipo

int main()
{
   int b,a=3;

   b=square(a); // chiamata

   printf( "Il quadrato di %d e' %d", a, b);
   fflush(stdin);
   getchar();
   return(0);
}

int square( int y ) // definizione
{
   return ( y * y ); // restituzione del risultato
}
```

square.c (passagio per indirizzo)

```c
#include <stdio.h>
#include <stdlib.h>

void square( int y, int *x  ) ;// prototipo

int main()
{
   int b,a=3;

   square(a,&b);// chiamata

   printf( "Il quadrato di %d e' %d", a, b);
   fflush(stdin);
   getchar();
   return(0);
}

void square( int y, int *x ) // definizione
{
   *x= ( y * y ); // il risultato viene inserito nella
                  // cella di memoria puntata da x
}
```