FILE

Fondamenti di Informatica, AA 2022/23

Luca Cassano

luca.cassano@polimi.it

I file

- I file sono “contenitori” di informazioni: sequenze di byte associate ad un nome
- Sono memorizzati su memoria di massa (**non volatile**)
- Possono continuare ad esistere indipendentemente dalla vita del programma che li ha creati
- Possono essere acceduti da più programmi
- Organizzano l’informazione in maniera **sequenziale**

I file

Il **sistema operativo** si occupa della loro gestione e offre ai programmi una serie di funzioni di libreria per

- creazione/cancellazione di file
- scrittura/lettura
- controllo dei casi di errore

Ci sono due tipi di file:

- **File binari**: le informazioni contenute sono memorizzate con la stessa codifica binaria con cui sono rappresentate in memoria
- **File testuali**: le informazioni sono convertite in stringhe (come quando si stampa a video) e salvate in termini di una sequenza di caratteri

I file nel linguaggio C

Per utilizzare un file all'interno di un programma è necessario:

- Aprire un "flusso di comunicazione", cioè aprire il file
- Accedere a file in lettura e/o scrittura
- Chiudere il "flusso di comunicazione", cioè chiudere il file

I file nel linguaggio C

Per ogni file aperto il sistema operativo gestisce un **descrittore di file**

Il descrittore contiene le seguenti informazioni:

- Modalità di utilizzo di un file (lettura, scrittura,...)
- Posizione corrente nel file (punta al prossimo byte da leggere o scrivere);
- Indicatore di errore
- Indicatore di end-of-file (eof)

Tutti i descrittori di file sono memorizzati nel sistema operativo nella **tabella dei file aperti**

I file nel linguaggio C – FILE*

Per utilizzare un file in C è necessario dichiarare un puntatore al suo descrittore:

```
FILE *fid;
```

- `FILE` è il tipo di dato che rappresenta il descrittore al file
- `fid` è una variabile puntatore al descrittore del file

I file nel linguaggio C – fopen()

```
fid = fopen(nomefile, modo);
```

Apre un flusso di comunicazione con il file il cui nome viene specificato come parametro

Riceve in ingresso

- Una stringa che contiene il nome del file da aprire (può includere il percorso del file se non si trova nella cartella corrente) e
- Una stringa che specifica il modo in cui lo si vuole aprire

Restituisce l'indirizzo del descrittore di tipo `FILE`

I file nel linguaggio C – fopen()

Possibili modi di apertura di un file sono:

- `"w"` accesso in scrittura in formato testuale; il file viene creato (se esiste già un file con lo stesso nome questo viene **distrutto**)
- `"a"` accesso in scrittura in append in formato testuale; il file viene creato (se esiste già un file con lo stesso nome la scrittura inizia dalla fine del file)
- `"r"` accesso in lettura in formato testuale di un file già esistente
- `"wb"` accesso in scrittura in formato binario
- `"ab"` accesso in scrittura append in formato binario
- `"rb"` accesso in lettura in formato binario

I file nel linguaggio C – fopen()

Alla chiamata del sottoprogramma il sistema operativo crea un nuovo descrittore di file nella tabella dei file aperti all'interno del sistema operativo

Restituisce il riferimento al descrittore

Una volta aperto il file, il puntatore `fid` sarà usato nel programma per accedere al file

I file nel linguaggio C – fopen()

Restituisce `NULL` se il file non può essere aperto:

- Il file aperto in lettura non esiste
- Il file aperto in scrittura è protetto oppure è protetta l'unità di memoria su cui si trova oppure lo stesso file è aperto da un altro programma
- Se si verifica un errore nell'interazione con il supporto di memorizzazione su cui il file risiede

Controllare sempre il valore restituito dalla `fopen()`

I file nel linguaggio C – fclose()

```
fclose(fid)
```

Una volta completate le operazioni di lettura/scrittura è necessario chiudere il file

Il sottoprogramma chiude il flusso di comunicazione con il file identificato da `fid`

I file nel linguaggio C – fclose()

E’ necessario chiudere i file perché:

- Altri programmi potrebbero aver bisogno del file che il mio programma non sta usando ma non ha chiuso
- Il programma potrebbe terminare i descrittori di file a sua disposizione

I file nel linguaggio C – fprintf()

```
fprintf(fid, stringa_di_controllo, lista_var)
```

Il sottoprogramma `fprintf` permette di scrivere una data stringa all'interno del file testuale puntato da `fid`

Il sottoprogramma `fprintf` funziona esattamente come la `printf` con l'unica differenza che scrive in un file e non su terminale

- Le informazioni vengono convertite dalla codifica interna utilizzata per i vari tipi di dato (`int`, `float`, …) ad una sequenza di caratteri

La scrittura è sequenziale

I file nel linguaggio C – esempio di scrittura

Scrivere un programma che apre il file `ciao.txt` in scrittura e vi scrive i numeri da 1 a 10

```
#include<stdio.h>

int main(){

  FILE* fp;

  int n;

  fp = fopen("ciao.txt", "w");

  if(fp){

    for(n = 1; n <= 10; n++)

      fprintf(fp, "%d ", n);

    fclose(fp);

  }else{

    printf("Errore di apertura del file\n");

  }

  return 0;  }
```

Dichiarazione del puntatore al file

Apertura del file in scrittura

Test per verificare la corretta apertura del file

Scrittura nel file

Chiusura del file

I file nel linguaggio C – fscanf()

```
fscanf(fid, stringa_di_controllo, lista_var)
```

Il sottoprogramma `fscanf` permette di leggere una serie di valori dal file testuale puntato da `fid` e salvarli nelle variabili specificate nella chiamata

Il sottoprogramma `fscanf` funziona esattamente come la `scanf` con la differenza che legge da file e non da tastiera

- La sequenza di caratteri letta viene interpretata in base alla stringa di controllo e convertita nella rappresentazione interna (`int`, `float`, …)

I file nel linguaggio C – fscanf()

```
fscanf(fid, stringa_di_controllo, lista_var)
```

La lettura è sequenziale

Se non ci sono più valori validi da leggere nel file e viene eseguita la `fscanf`, Il sottoprogramma non modifica il contenuto delle variabili (non legge niente!)

Il sottoprogramma restituisce il numero di elementi effettivamente letti o `EOF` (cioè -1) se non è stato letto niente poiché il file è terminato

I file nel linguaggio C – feof()

```
status = feof(fid)
```

Il sottoprogramma restituisce 1 se abbiamo raggiunto la fine del file (cioè se abbiamo fatto una lettura oltre l'ultimo dato valido) altrimenti 0

È sempre necessario controllare che i dati letti siano validi (cioè `feof` deve restituire 0) prima di utilizzare tali dati

I file nel linguaggio C – esempio di lettura

Scrivere un programma che apre in lettura il file `ciao.txt` che contiene una lista di lunghezza indefinita di interi e ne visualizza il contenuto

```
#include<stdio.h>
int main(){
  FILE* fp;
  int n;
  fp=fopen("ciao.txt", "r");
  if(fp){
    fscanf (fp, "%d", &n);
    while(!feof(fp)){
      printf("%d ", n);
      fscanf (fp, "%d", &n);
    }
    fclose(fp);
  }else
    printf("Errore di apertura del file\n");
  return 0;
}
```

Dichiarazione del puntatore al file

Apertura del file in lettura

Test per verificare la corretta apertura del file

Lettura da file

Test sulla validità dei dati letti

Elaborazione dei dati letti (stampo!)

Lettura da file

Chiusura del file

I file nel linguaggio C – esempio di lettura

Versione alternativa che utilizza il valore restituito da `fscanf`

```
#include<stdio.h>
int main(){
  FILE* fp;
  int n;
  fp=fopen("ciao.txt", "r");
  if(fp){
    while(fscanf (fp, "%d", &n)>0){
      printf("%d ", n);
    }
    fclose(fp);
  }else{
    printf("Errore di apertura del file\n");
  }
  return 0;
}
```

Lettura da file e controllo che sia stato letto qualcosa

I file nel linguaggio C – file binary - `fwrite()`

```
fwrite(ptr, dim, num, fid)
```

Il sottoprogramma `fwrite` permette di scrivere nel file binario puntato da `fid` una serie di «blocchi» di dati consecutivi in memoria

I blocchi scritti iniziano dall'indirizzo specificato dal puntatore `ptr`, ciascuno è grande `dim` byte ed il loro numero è `num`

- In altre parole il sottoprogramma scrive `dim*num` byte a partire da `ptr`
- Il numero di byte necessari per memorizzare una variabile di un dato tipo può esser ottenuta tramite l'operatore `sizeof()`

La scrittura è sequenziale e non avviene alcuna conversione dei dati scritti (differentemente dalla `fprintf`)

I file nel linguaggio C – reminder su sizeof()

`sizeof(type)` è un operatore (non un sottoprograma!) che restituisce il numero di byte necessari per memorizzare una data variabile di un tipo

`sizeof` può ricevere come operando sia il nome di un tipo che di una variabile e restituisce la dimensione del tipo (della variabile)

`sizeof` può ricevere come operando sia il nome di un array e restituisce la dimensione complessiva dell'array

Se `sizeof` riceve un puntatore a memoria allocata dinamicamente **NON** restituisce la dimensione della memoria allocata ma della variabile puntatore!

I file nel linguaggio C – file binary - `fwrite()`

Scrivere un programma che apre il file `ciao.bin` in scrittura e vi scrive il contenuto di un array di 10 interi

```
#include<stdio.h>
#define FILENAME "ciao.bin"
#define N 10

int main(){
  FILE* fp;
  int a[N];
  int i;

  for(i=0; i<N; i++)
    scanf("%d",&a[i]);

  fp=fopen(FILENAME, "wb");
  if(fp){
    fwrite(a, sizeof(int), M, fp);
    fclose(fp);
  }else{
    printf("Errore\n");
  }
  return 0;
}
```

Apertura del file binario in scrittura

Scrittura nel file

I file nel linguaggio C – file binary - `fread()`

```
fread (ptr, dim, num, fid)
```

Il sottoprogramma `fread` permette di leggere dal file binario puntato da `fid` una serie di «blocchi» di dati consecutivi e salvarli in memoria a partire dall'indirizzo `ptr`

I blocchi da leggere sono `num` e ciascuno è grande `dim` byte

- In altre parole il sottoprogramma legge `dim*num` byte e li salva in memoria a partire da `ptr`

La lettura è sequenziale e non avviene alcuna conversione dei dati scritti (differentemente dalla `fscanf`)

Il sottoprogramma restituisce il numero di blocchi effettivamente letti

I file nel linguaggio C – file binary - `fread()`

Scrivere un programma che apre il file `ciao.bin` in lettura e ne legge i 10 valori interi contenuti. Se la lettura è andata a buon fine visualizza i dati altrimenti restituisce errore

```
#include<stdio.h>
#define FILENAME "ciao.bin"
#define N 10
int main(){
  FILE* fp;
  int a[N];
  int n,i;
  fp=fopen(FILENAME, "rb");
  if(fp){
    n=fread(a, sizeof(int), N, fp);
    if(n==N)
      for(i=0;i<N;i++)
        printf("%d\n",a[i]);
    else
      printf("Letti meno valori: %d\n",n);
    fclose(fp);
  }else{
    printf("Errore\n");
  }
  return 0;
}
```

Apertura del file binario in lettura

Lettura da file file

Controllo della corretta lettura

I file nel linguaggio C – flussi standard

Tre flussi standard vengono automaticamente aperti quando inizia l'esecuzione di un programma ed assegnati a tre puntatori: `stdin`, `stdout`, e `stderr`

Normalmente questi tre flussi rappresentano

- L'output su terminale (`stdout` e `stderr`)
- L'input da tastiera (`stdin`)

I file nel linguaggio C – flussi standard

Tali puntatori sono dichiarati e inizializzati in `stdio.h`

`printf` e `scanf` utilizzano questi flussi standard

```
printf("%d ", n);
```

equivale a

```
fprintf(stdout, "%d ", n);
```

```
scanf("%d", &n);
```

equivale a

```
fscanf(stdin, "%d", &n);
```

I file