Programmazione I





I file

I file

- I file sono “contenitori” di informazione: 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 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

- In questo corso vedremo solo i file di testo!

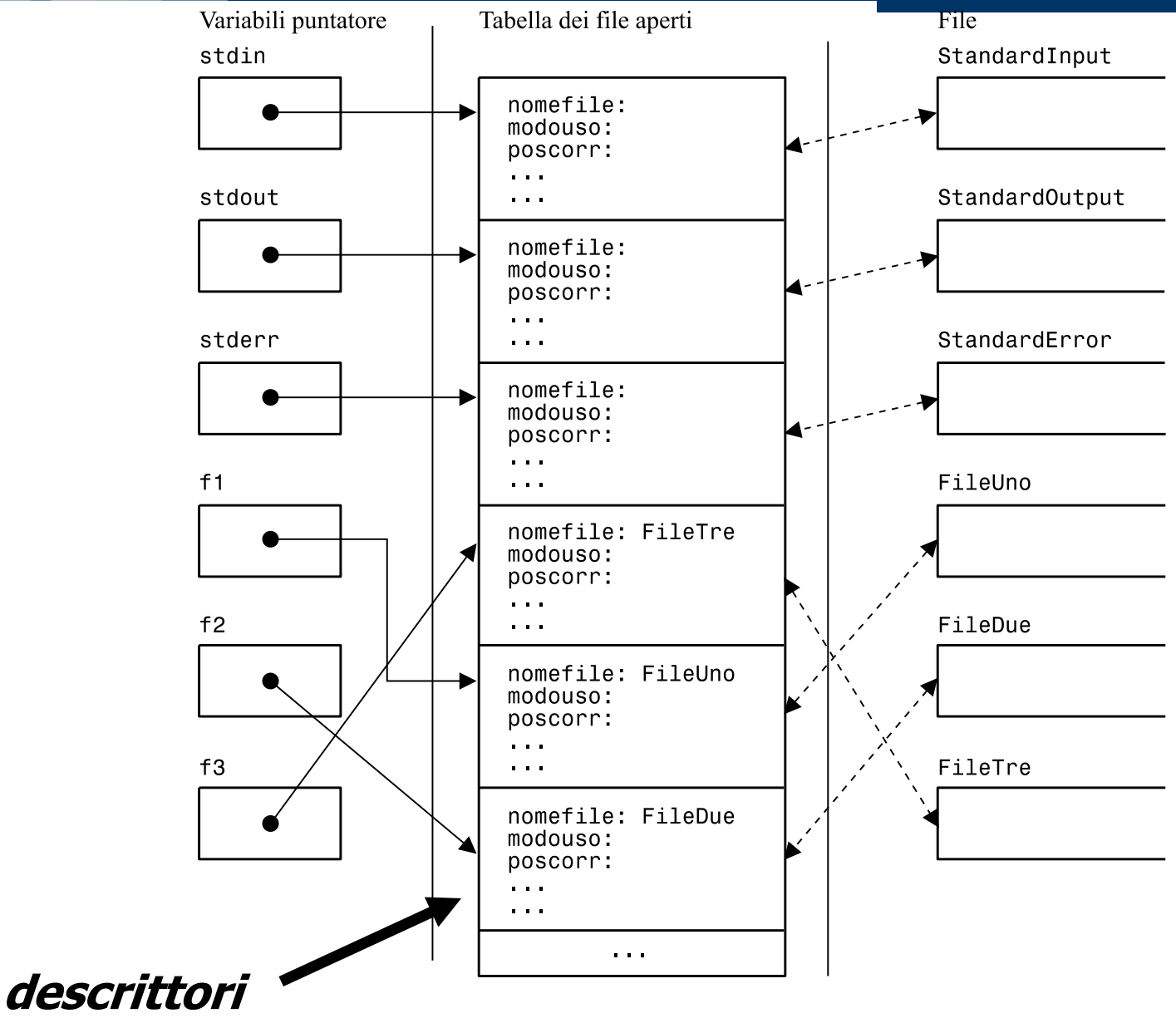
Descrittore del file

- 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
  - Stato dell'accesso (errore)
- Tutti i descrittori di file sono memorizzati nel sistema operativo nella tabella dei file aperti

Gestione dei file all'interno di un programma

- 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

Rappresentazione interna



FILE*

- Per utilizzare un file in C è necessario dichiarare un riferimento 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

fopen

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

- È la funzione per l'apertura del file
- 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
    - `"w"` accesso in scrittura; il file viene creato (se esiste già un file con lo stesso nome questo viene **distrutto**)
    - `"r"` accesso in lettura di un file già esistente
- Restituisce l'indirizzo del descrittore di tipo `FILE`

fopen

- Alla chiamata della funzione 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

fopen

- Restituisce `NULL` se il file non può essere aperto:
  - Il file aperto in lettura non esiste
  - Il file aperto in scrittura è protetto da scrittura oppure è protetta da scrittura 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()`

fclose

`fclose(fid)`

- Una volta completate le operazioni di lettura/scrittura è necessario chiudere il file
- La funzione chiude il flusso di comunicazione con il file identificato da `fid`

fprintf

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

- La funzione `fprintf` permette di scrivere una data stringa all'interno del file puntato da `fid`
- La funzione `fprintf` funziona esattamente come la `printf` con l'unica differenza che scrive in un file e non stdout (cioè il terminale)
- La scrittura è sequenziale

Esempio di scrittura in un file

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

```
#include<stdio.h>

void 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");
  }
}
```

Dichiarazione del puntatore al file

Apertura del file in scrittura

Test per verificare la corretta apertura del file

Scrittura nel file

Chiusura del file

fscanf

`fscanf (fid, stringa_di_controllo, lista_var)`

- La funzione `fscanf` permette di leggere una serie di valori dal file puntato da `fid` e salvarli nelle variabili specificate nella chiamata
- La funzione `fscanf` funziona esattamente come la `scanf` con la differenza che legge da file e non da stdin (cioè da tastiera)
- La lettura è sequenziale
- Se non ci sono più valori validi da leggere nel file e viene eseguita la `fscanf`, la funzione non modifica il contenuto delle variabili (non legge niente!)

feof

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

- La funzione 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

Esempio di lettura da un file

- 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>

void 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");
  }
}
```

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