Gestione file

La gestione dei file in C

- Il **file** è un'astrazione molto ampia nella descrizione di un sistema informatico:
  - Nastro di I/O
  - Supporto di comunicazione macchina/ambiente di ogni tipo (sensori, attuatori, ….)
  - Zona di memoria di massa
- È un supporto di memoria, ma profondamente diverso dalla memoria centrale
- In C i file sono utilizzati tramite funzionalità della standard library
  - I file sono strutture sostanzialmente sequenziali, anche se, quando è possibile, permettono un accesso diretto ai vari record

File e programmi C

- Un programma C che desidera utilizzare un file deve aprire un flusso di comunicazione
- Il flusso di comunicazione viene chiuso chiudendo il file utilizzato
  - Le operazioni di input/output sono sia le operazioni che coinvolgono un dispositivo di ingresso/uscita sia le operazioni di memorizzazione permanente
- Una entità di tipo **file** contiene diversi campi:
  - Modalità di utilizzo del file (lettura, scrittura, o lettura e scrittura)
  - Posizione corrente sul file (punta al prossimo byte da leggere o scrivere)
  - Indicatore di errore
  - Indicatore di end-of-file (eof)

Apertura di un file

- Bisogna dichiarare una variabile di tipo puntatore a FILE e chiedere l'apertura del flusso tramite una funzione di libreria (**fopen**)
  - Il puntatore serve al programma per far riferimento al file corrispondente
  - La chiusura del flusso (tramite **fclose**) impedisce ulteriori riferimenti al file
- Un flusso di comunicazione può essere
  - Binario (sequenza di byte) o di tipo testo (sequenza di caratteri)

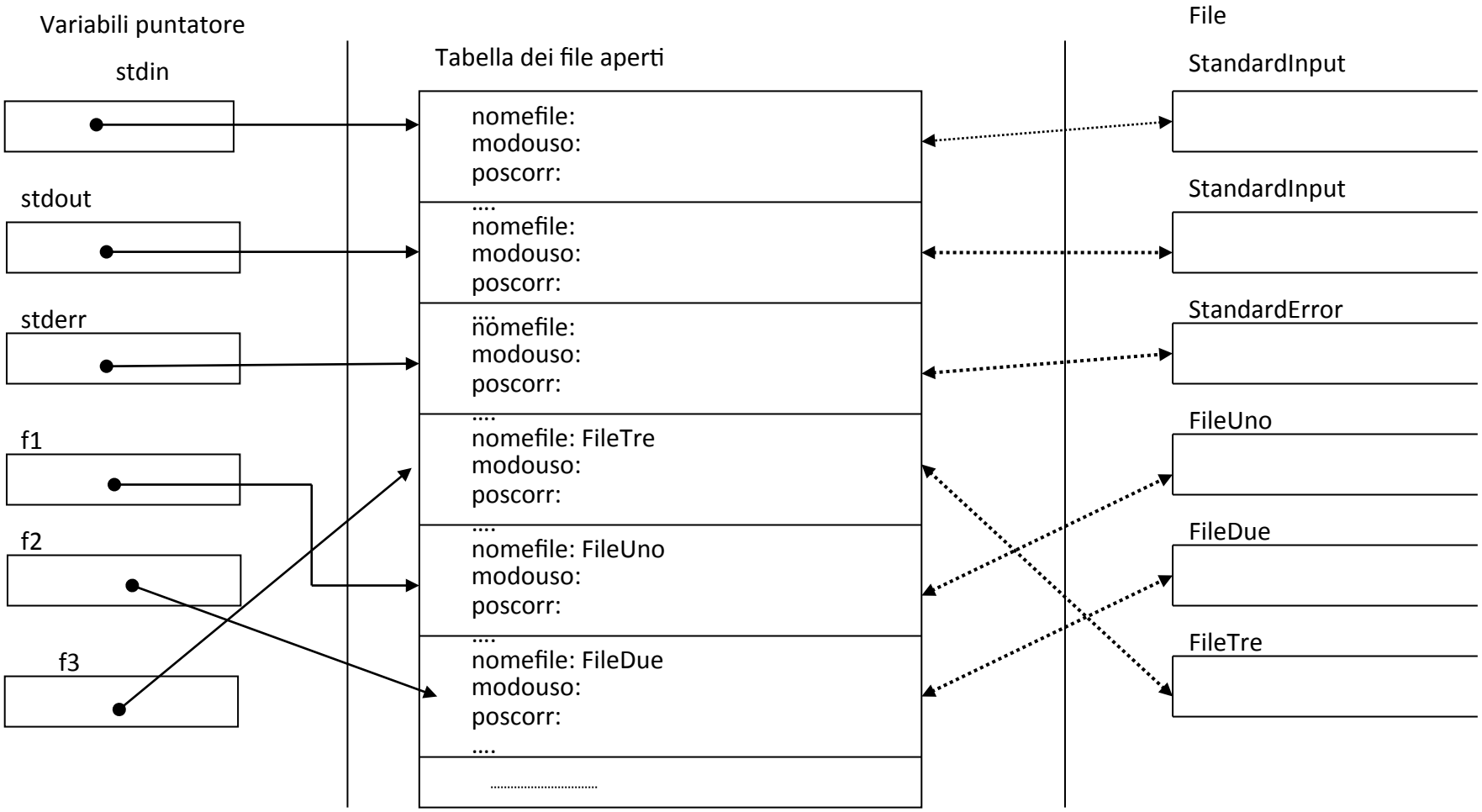
File e sistema operativo

- Ogni variabile che punta a un file deve essere definita come segue
  - FILE *fp;
- Una “tabella file aperti”

  FILE TabellaFileAperti[MaxNumFileGestibili];

  è gestita dal SO (file system) e costituisce il “ponte” tra il programma e la macchina astratta gestita dal SO

File e sistema operativo



Flussi standard

- Tre flussi standard vengono automaticamente aperti quando inizia l'esecuzione di un programma: **stdin**, **stdout**, e **stderr**
- Normalmente questi tre flussi rappresentano
  - Il video del terminale (stdout e stderr)
  - La tastiera del terminale (stdin)
- **printf** e **scanf** utilizzano questi flussi standard

Operazioni di gestione dei file

- FILE *fopen(nomefile, modalità)
  - Richiede in ingresso il nome del file da aprire e la modalità di apertura
- int fclose(FILE *fp)
  - Chiude il file cui fa riferimento il puntatore fp
- int remove(nomefile)
  - Cancella il file identificato da nomefile
- int rename(vecchionome, nuovonome)
  - Modifica il nome di un file da vecchionome a nuovonome

Mode	Meaning
"r"	Open text file for reading
"w"	Create a text file for writing
"a"	Append to text file
"rb"	Open binary file for reading
"wb"	Create binary file for writing
"ab"	Append to a binary file
"r+"	Open text file for read/write
"w+"	Create text file for read/write
"a+"	Open text file for read/write
"rb+" or "r+b"	Open binary file for read/write
"wb+" or "w+b"	Create binary file for read/write
"ab+" or "a+b"	Open binary file for read/write

Operazioni di gestione degli errori

- int ferror(FILE *fp)
  - Controlla se è stato commesso un errore nella precedente operazione di lettura o scrittura
  - Restituisce 0 se nessun errore è stato commesso, un valore diverso da 0 in caso contrario
- int feof(FILE *fp)
  - Controlla se è stata raggiunta la fine del file nella precedente operazione di lettura o scrittura
  - Restituisce 0 se la condizione di fine file non è stata raggiunta, un valore diverso da 0 in caso contrario

Operazioni di lettura e scrittura

- Le operazioni di lettura e scrittura su file possono essere effettuate in quattro modi diversi:
  - precisando il formato dei dati in ingresso e in uscita
  - accedendo ai dati carattere per carattere
  - linea per linea
  - blocco per blocco
- Generalmente si adotta l'accesso linea per linea nel caso di flussi di testo e l'accesso carattere per carattere o blocco per blocco in presenza di flussi binari

Lettura e scrittura formattata

- Le funzioni **fprintf** e **fscanf** consentono operazioni formattate analoghe a quelle di scanf e printf
- Restituiscono il numero degli elementi effettivamente letti o stampati o restituiscono un numero negativo in caso di errore
  - int fprintf(FILE *fp, stringa di controllo, elementi)
  - int fscanf(FILE *fp, stringa di controllo, indirizzo elementi)

Lettura e scrittura di caratteri

- **getchar** legge da Standard Input il prossimo carattere restituendolo come intero
- **putchar** scrive come prossimo carattere sul file di Standard Output il carattere che riceve come parametro restituendo il carattere scritto
- **getc** e **fgetc** leggono il prossimo carattere del file specificato tra i parametri di ingresso restituendolo come intero
- **putc** e **fputc** scrivono come prossimo carattere del file il carattere specificato tra i parametri di ingresso restituendolo come intero

Lettura e scrittura di caratteri: esempio

```
#include <stdio.h>
#include <stddef.h>

main()
{
        FILE    *fp;
        char    c;

        if ((fp = fopen("filechar", "r")) != NULL) {

                while ((c = fgetc(fp)) != EOF)
                        putchar(c);

                fclose(fp);
        }
        else
                printf("Il file non può essere aperto\n");
}
```

Contiene la definizione del tipo FILE e i prototipi delle funzioni che operano su file.

Contiene la definizione di NULL.

File aperto in lettura con modalità testo.

Dal file viene letto, e successivamente stampato a vide, un carattere alla volta.

Lettura e scrittura di stringhe

- **gets** e **puts** leggono da Standard Input e scrivono su Standard Output
- **fgets** e **fputs** leggono o scrivono linee (stringhe di caratteri terminate da un newline) dal o sul file specificato come parametro di ingresso

Lettura e scrittura di stringhe: esempio

```
#define ERROR           0
#define MAXLINE         100

void copiaselettiva(char refstr[]) {
        char    line[MAXLINE];
        FILE    *fin, *fout;

        if ((fin = fopen("filein", "r")) == NULL)
                return ERROR;
        if ((fout = fopen("fileout", "w")) == NULL) {
                fclose(fin);
                return ERROR;
        }
        while (fgets(line,MAXLINE,fin) != NULL)
                if (strstr (line,refstr) != NULL)
                        fputs(line,fout);
        fclose(fin);
        fclose(fout);
        return OK;
}
```

File aperto in lettura con modalità testo.

File aperto in scrittura con modalità testo.

Legge da filein al più MAXLINE-1 caratteri e li assegna a line, con il terminatore \0 in fondo.

Ritorna la prima occorrenza di refstr in line, oppure NULL.

Lettura e scrittura di strutture (per blocchi)

- int **fread**(void *ptr, dimelemento, numelementi, FILE *fp);
  - Legge un blocco di dati binari o testuali dal file cui fa riferimento fp e li memorizza nel vettore identificato da ptr
- int **fwrite**(void *ptr, dimelemento, numelementi, FILE *fp);
  - Scrive un blocco …

Lettura e scrittura di strutture (per blocchi): esempio (1/3)

- Un file Persone è costituito da record di tipo Persona
- Ogni Persona contiene i campi nome, cognome, indirizzo
- Si vuole modificare il file aggiungendo a ogni persona il campo CodiceFiscale
- Un file CodiciFiscali contiene i codici fiscali delle persone contenute in persone, nello stesso ordine
- Si vuole costruire un file NuovePersone
- I tre file sono binari
- Questa operazione è svolta, in maniera parametrica rispetto ai file utilizzati, dalla seguente funzione, cui sono premesse le necessarie dichiarazioni di tipo

Lettura e scrittura di strutture (per blocchi): esempio (2/3)

```
typedef  struct  {  char    nome[20];
                    char    cognome[20];
                    char    indirizzo[50];
                 } Persona;
typedef  char CodFisc[16];
typedef  struct  {  char    nome[20];
                    char    cognome[20];
                    char    indirizzo[50];
                    CodFisc      CodiceFiscale;
                 } NuovaPersona;
```

Lettura e scrittura di strutture (per blocchi): esempio (3/3)

```
void AggiornaPersone (FILE *pp, FILE *cf, FILE *np) {
        Persona PersonaCorrente;
        CodFisc CodFiscCorrente;
        NuovaPersona NuovaPersonaCorrente;

        rewind(pp);
        rewind(cf);
        rewind(np);

        while (fread(&PersonaCorrente,sizeof(Persona),1,pp) != EOF) {
                fread(CodFiscCorrente,sizeof(CodFisc),1,cf);
                strcpy(NuovaPersonaCorrente.nome, PersonaCorrente.nome);
                strcpy(NuovaPersonaCorrente.cognome,PersonaCorrente.cognome);
                strcpy(NuovaPersonaCorrente.indirizzo, PersonaCorrente.indirizzo);
                strcpy(NuovaPersonaCorrente.CodiceFiscale, CodFiscCorrente);
                fwrite(&NuovaPersonaCorrente,sizeof(NuovaPersona),1,np);
        }
}
```

Supponiamo che i file siano già stati aperti dal programma chiamante..

Ripartiamo dall'inizio di tutti e tre i file… (necessario se altre operazioni sono già state effettuare sugli stessi file)

Iteriamo fino alla fine del primo file.

Uniamo le informazioni e scriviamo un nuovo record nel terzo file.

Accesso diretto

- int **fseek**(FILE *fp, long offset, int refpoint)
  sposta l'indicatore di posizione per effettuare accessi diretti al file a cui fa riferimento fp
  - Lo scostamento offset (può assumere valori positivi o negativi ed è espresso in byte) si riferisce alla posizione fissa indicata da refpoint
  - Refpoint può assumere tre diversi valori
    - SEEK_SET rispetto all'inizio del file
    - SEEK_CUR rispetto alla posizione corrente
    - SEEK_END rispetto alla fine del file
- La funzione fseek restituisce zero se la richiesta è corretta, un valore diverso da zero altrimenti
  - Ad esempio, quando si cerca di superare la fine del file…

Accesso diretto (continua)

- long **ftell**(FILE *fp) restituisce il valore corrente dell'indicatore di posizione del file specificato
  - Per file binari la posizione è il numero di byte rispetto all'inizio del file, mentre per file testuali è un valore dipendente dall'implementazione
- rewind(f) equivale a fseek (f, 0, SEEK_SET);
  - A differenza di fseek, rewind non restituisce alcun valore

Accesso diretto: esempio

```
main() {
        FILE            *f;
        long int        inizio, fine;
        int             tempi, tempf;
        if ((f = fopen("numint", "rb+")) == NULL) {
                exit(1);
        }
        inizio = 0;
        fseek(f, –sizeof(int), SEEK_END);  fine = ftell(f);
        while (inizio < fine) {
                fseek(f, inizio, SEEK_SET);
                fread(&tempi, sizeof(int), 1, f);
                fseek (f, fine,  SEEK_SET);
                fread (&tempf, sizeof(int), 1, f);
                fseek (f, fine, SEEK_SET);
                fwrite (&tempi, dim, 1, f);
                fseek (f, inizio, SEEK_SET);
                fwrite (&tempf, dim, 1, f);
                inizio = inizio + sizeof(int);
                fine = fine – sizeof(int);
        }
}
```

Causa l'interruzione del programma e l'uscita con il valore di ritorno 1.

Legge il primo intero da inizio.

Legge il primo intero da fine.

E poi… ?

Si riposiziona e scrive alla fine l'intero trovato ad inizio.

Si riposiziona e scrive ad inizio fine l'intero trovato a inizio.