



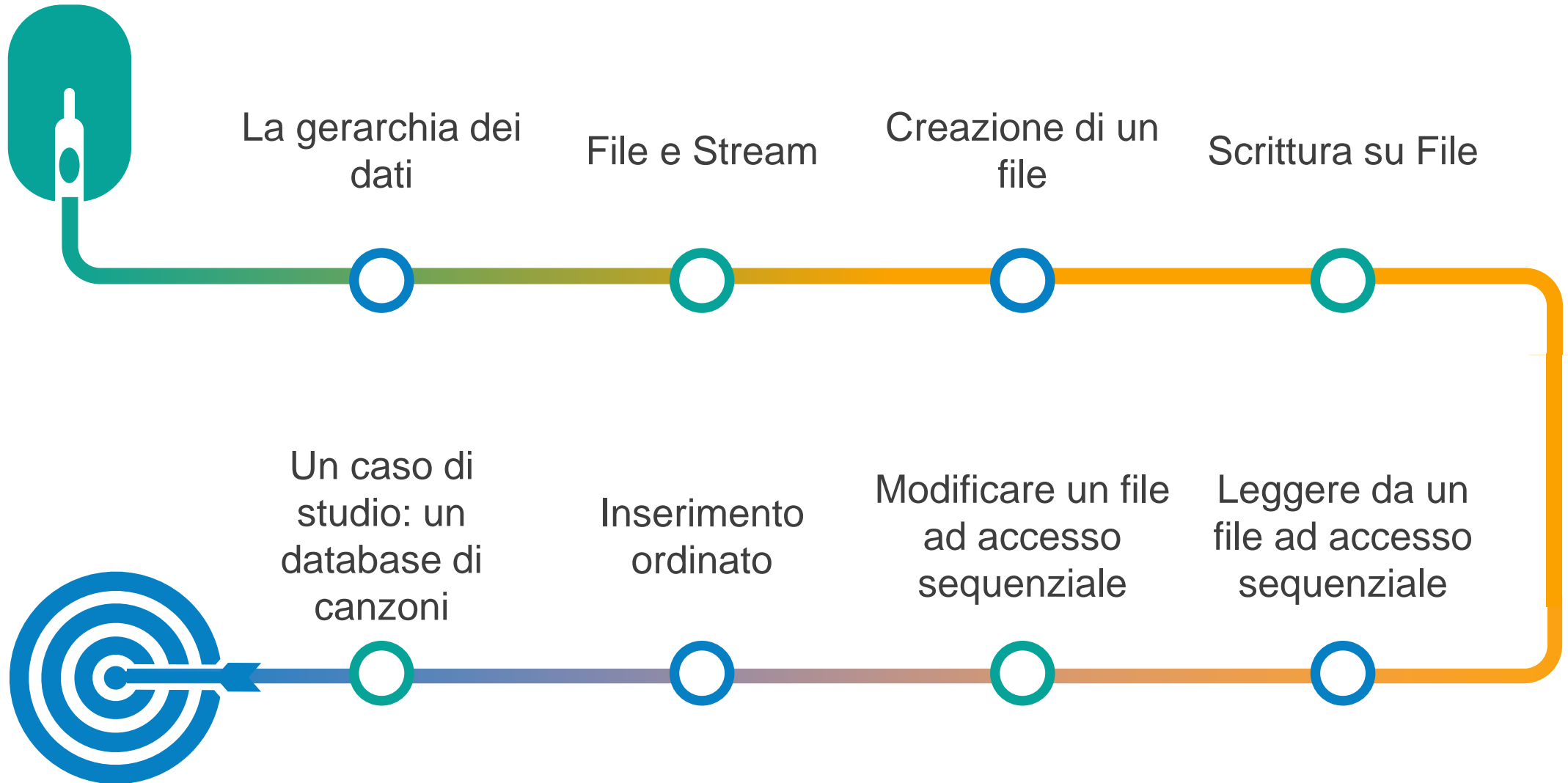
BASI DI DATI

FILE PROCESSING

Polese G. Caruccio L. Breve B.

a.a. 2023/2024

Outline

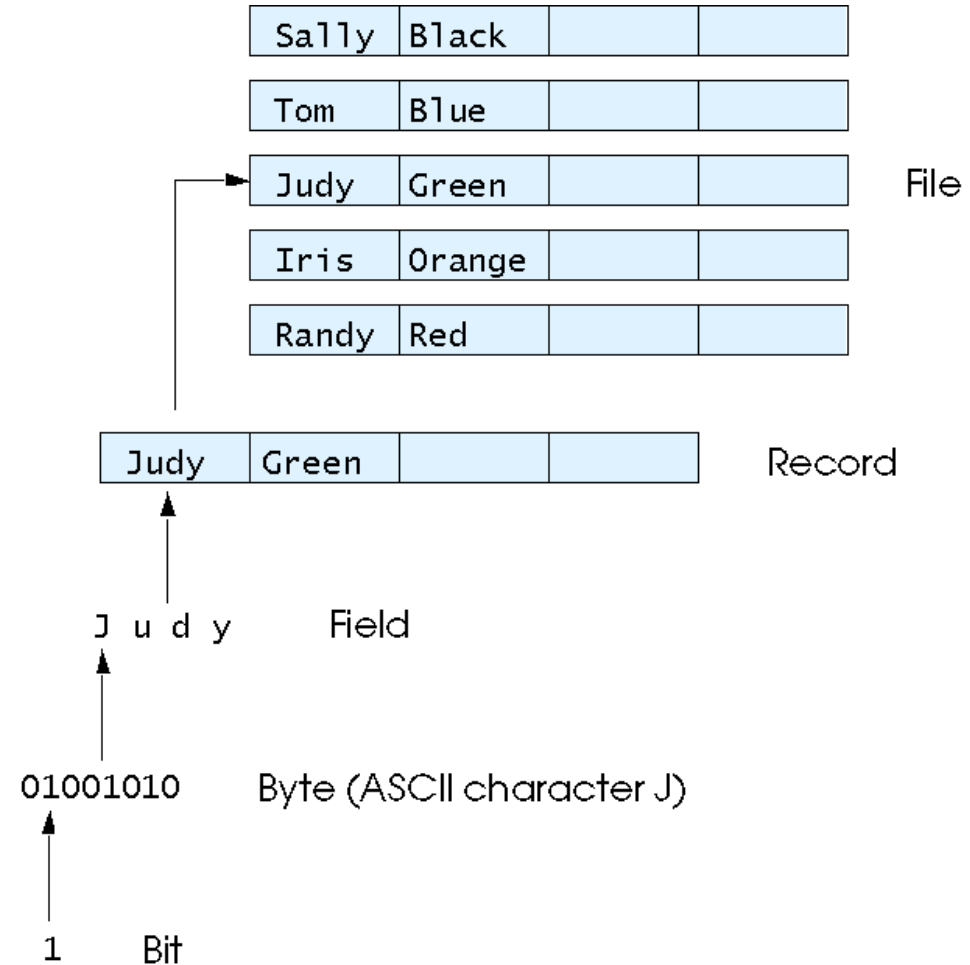


LA GERARCHIA DEI DATI (1)

- Bit – il dato più piccolo
 - Due possibili valori: 0/1
- Byte – 8 bit
 - Usato ad esempio per memorizzare i caratteri
 - Digit decimali, lettere o caratteri speciali
- Campo/Attributo – gruppo di caratteri con un significato
 - Esempio: nome
- Record – gruppo di campi/attributi correlati
 - Rappresentato attraverso una `struct` o una `class`
 - Esempio: un record per uno specifico studente che contiene la sua matricola, nome, indirizzo, ecc.

LA GERARCHIA DEI DATI (2)

- File – gruppo di record correlati
 - Esempio: un file di studenti
- Database –
 - gruppo di file correlati



FILE (1)

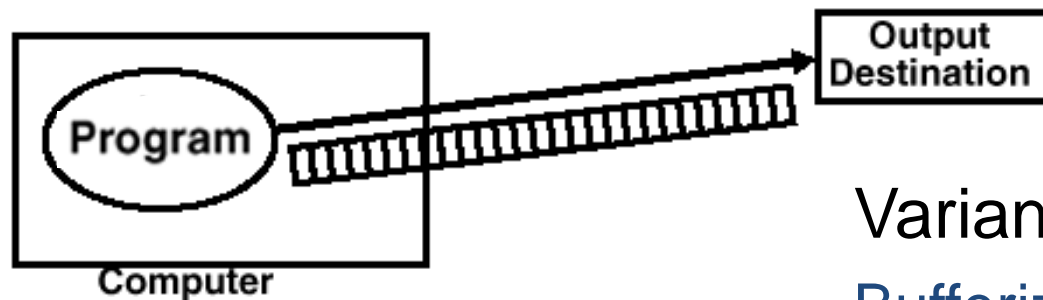
- File su disco: un insieme di dati sul disco cui è associato un nome
 - **Persistenza**, i dati vivono oltre il tempo dell'elaborazione;
 - **Grande capacità**, dipende solo dalla capienza del disco;
 - **Condivisibilità**, più applicazioni/utenti possono accedere agli stessi file.

FILE (2)

- I file su disco hanno due attributi
 - Un contenuto (i dati)
 - Un nome (del file)
- Contenuto
 - Qualsiasi: compito d'esame, foto, video, foglio di calcolo, programma C
- Operazioni
 - Creazione, cancellazione, cambiamento del nome, cambiamento del contenuto, lettura del contenuto

FILE E STREAM (1)

- Uno stream viene creato quando un file viene aperto
- Output Stream
 - Una sequenza di dati da un programma ad un dispositivo d'uscita
 - Monitor, file, case acustiche, connessioni di rete



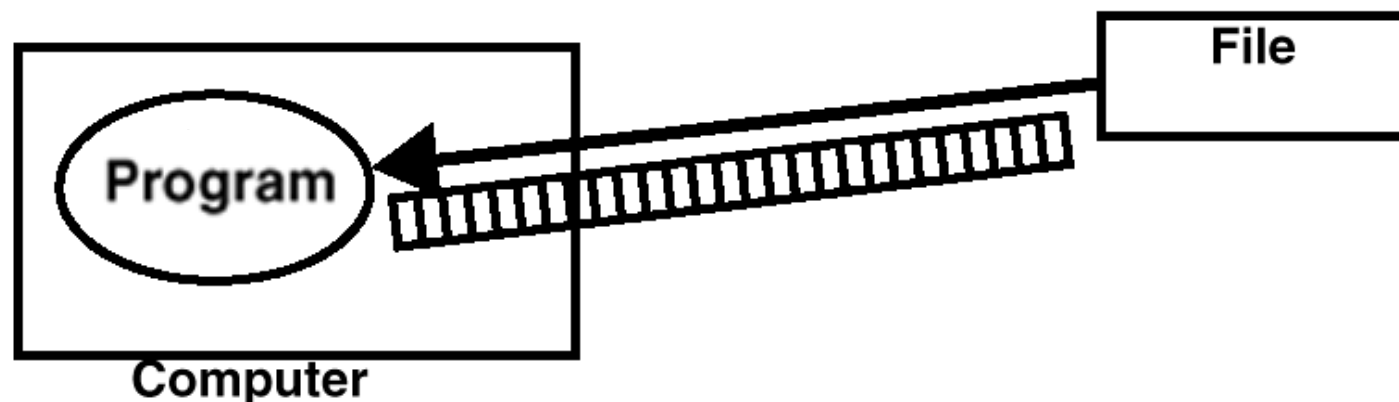
Varianti

Bufferizzati / non bufferizzati

Unità dei dati: byte, string, linee ...

FILE E STREAM (2)

- Uno stream viene creato quando un file viene aperto
- Input Stream
 - Una sequenza di dati da un dispositivo d'ingresso ad un programma
 - file, tastiera, connessioni di rete



FILE E STREAM (3)

- C considera ogni file come una sequenza di byte
 - Un file termina con l'*end-of-file marker*
- Uno stream viene creato quando un file viene aperto
 - Fornisce un canale di comunicazione tra i file e i programmi
 - Quando si apre un file viene restituito un puntatore alla struttura **FILE**
 - Esempi di puntatori a file:
 - › stdin – standard input (tastiera)
 - › stdout – standard output (schermo)
 - › stderr – standard error (schermo)

FILE E STREAM (4)

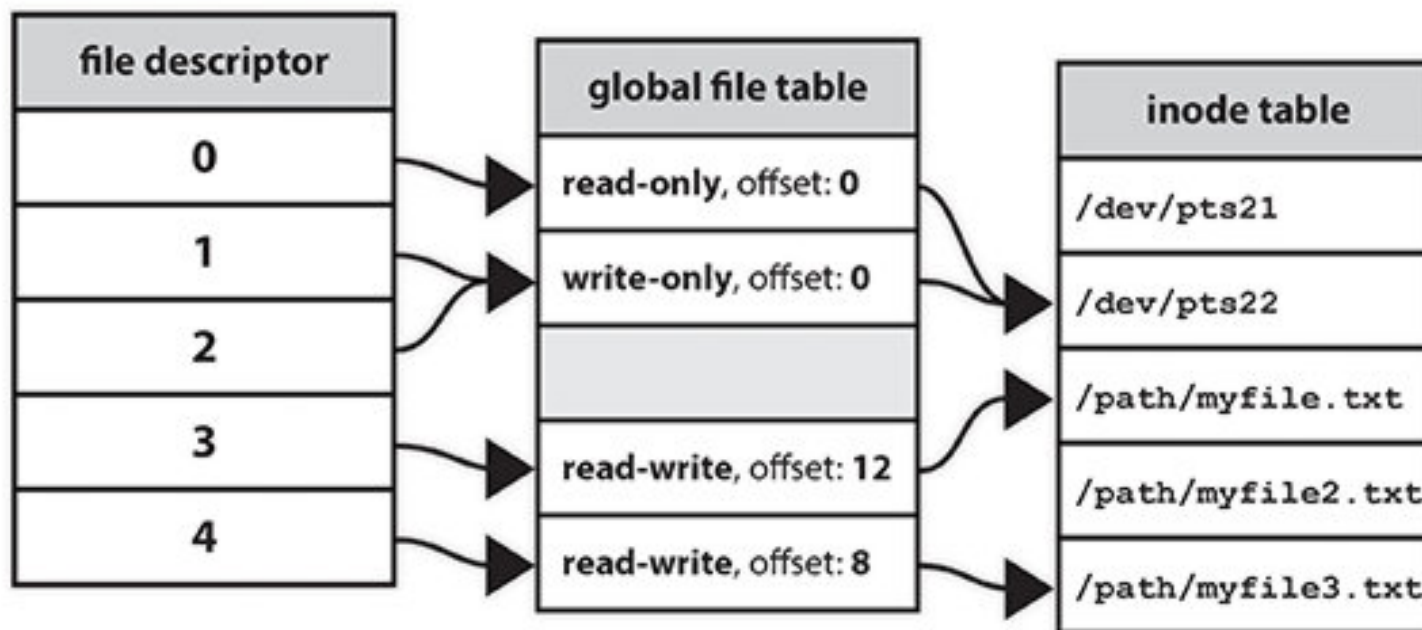
- FILE structure

- File descriptor

- Indice nell'array del sistema operativo chiamato tabella dei file aperti

- File Control Block (FCB)

- Mantiene tutte le informazioni sul file
 - Viene usato dal sistema per gestire il file



FILE E STREAM (5)

- Funzioni di lettura e scrittura nella libreria standard
 - `fgetc`
 - Legge un carattere dal file
 - Prende un puntatore a `FILE` come argomento
 - `fgetc(stdin)` equivale a `getchar()`
 - `fputc`
 - Scrive un carattere in un file
 - Prende un carattere e un puntatore a `FILE` come argomenti
 - `fputc('a', stdout)` equivale a `putchar('a')`
 - `fgets/fputs`
 - Legge/scrive una linea dal/nel file
 - `fscanf/fprintf`
 - Processi sui file equivalenti a `scanf` e `printf`

CREAZIONE DI UN FILE (1)

- C non impone/prevede nessuna struttura nei file
 - Non è considerata una struttura a record
 - Il programmatore deve fornire una struttura al file
- Creare un File
 - `FILE *f;`
 - Crea un puntatore a FILE denominato f
 - `f = fopen("file.txt", "w");`
 - La funzione fopen restituisce un puntatore al file FILE specificato
 - Prende come argomenti in nome del file da aprire e la modalità d'apertura
 - Se fallisce l'apertura, viene restituito NULL

CREAZIONE DI UN FILE (2): ESEMPIO

```
/*Creazione di un file sequenziale */

#include <stdio.h>

int main() {

    FILE *f;                /*f=puntatore del file "file.txt" */

    /*si esce dal programma se non è possibile creare il file */
    if((f=fopen("file.txt", "w")) == NULL) {
        printf("Il file non può essere aperto");
    } /* end if */
    else {
        printf("Ho creato/aperto il file");
    }

    fclose(f); /*chiude il file */

    return 0;
}
```

CREAZIONE DI UN FILE (3)

Modalità	Descrizione
r	Apri un file in lettura.
w	Crea un file per la scrittura. Se il file già esiste non considera il contenuto attuale.
a	Append; apre e crea un file per scrivere in coda.
r+	Apri un file per la modifica (lettura e scrittura).
w+	Crea un file per la modifica. Se il file già esiste non considera il contenuto attuale.
a+	Append; crea un file per la modifica; la scrittura viene effettuata in coda.
rb	Apri un file per leggerlo in modalità binaria.
wb	Crea un file per scrivere in modalità binaria. Se il file già esiste, non considera il contenuto attuale.
ab	Append; apre o crea un file per scrivere in coda in modalità binaria.
rb+	Apri un file per la modifica (lettura e scrittura) in modalità binaria.
wb+	Crea un file per la modifica in in modalità binaria. Se il file già esiste, non considera il contenuto attuale.
ab+	Append; apre o crea un file per la modifica in modalità binaria; la scrittura viene effettuata in coda.

SCRITTURA SU FILE (1)

- `fpr int f`
 - Usato per scrivere in un file
 - Simile a `pr int f`, eccetto per il primo argomento che è un puntatore a `FILE` (il file in cui si vuole scrivere)
- `f eof (FILE poi nter)`
 - Restituisce vero se si è raggiunto l'end-of-file (non c'è nessun altro dato da processare)
- `fcl ose (FILE poi nter)`
 - Chiude il file specificato
 - Viene effettuato automaticamente quando il programma termina
 - È comunque buona pratica chiudere i file esplicitamente

SCRITTURA SU FILE (2): ESEMPIO

```
/*Scrittura in un file sequenziale */
#include <stdio.h>

int main(void) {

    char matricola[30]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    FILE *f;            /*f=puntatore del file "studenti.txt" */

    /*si esce dal programma se non è possibile creare il file */
    if((f=fopen("studenti.txt", "w"))== NULL) {
        printf("Il file non può essere aperto\n");
    } /* end if */
    else {
        printf("Inserisci la matricola, il nome e la media dei voti di uno studente\n");
        printf("Inserisci EOF per concludere l'inserimento dei dati\n");
        printf("? ");
        scanf("%s%s%lf", matricola, nome, &media_voti);
    }
}
```


SCRITTURA SU FILE (3): ESEMPIO

```
/*Scrive i dati inseriti nel file*/
while(!feof(stdin)) {
    fprintf(f, "%s %s %lf \n", matricola, nome, media_voti);
    printf("? ");
    scanf("%s%s%lf", matricola, nome, &media_voti);
} /* end while */

fclose(f); /*chiude il file */
}

return 0;
}
```

```
Inserisci la matricola, il nome e la media dei voti di uno studente
Inserisci EOF per concludere l'inserimento dei dati
? 001 Mario_Rossi 25.4
? 002 Giuseppe_Verdi 26
? 003 Anna_Bianchi 28.3
? ^Z
```

SCRITTURA SU FILE (4)

Sistema	Combinazione di tasti
UNIX systems	<i><return> <ctrl> d</i>
IBM PC and compatibles	<i><ctrl> z</i>
Macintosh	<i><ctrl> d</i>
Combinazione di tasti per indicare l'end-of-file di stdin	

LEGGERE DA UN FILE AD ACCESSO SEQUENZIALE

- I dati vengono letti dall'inizio alla fine
- Si utilizza un **file position pointer** per determinare il numero di byte che devono essere ancora letti/scritti
 - **fopen**
 - Crea il puntatore al file che si vuole leggere
 - **fscanf**
 - Simile a scanf, eccetto per il primo argomento che è un puntatore a FILE (il file in cui si vuole leggere)
 - **rewind(*FILE* pointer)**
 - Riposiziona il file position pointer all'inizio del file da leggere o scrivere (byte 0)

LETTURA DA FILE (1): ESEMPIO

```
/*Lettura di un file sequenziale */

#include <stdio.h>

int main() {

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    FILE *f;            /*f=puntatore del file "studenti.txt" */

    /*si esce dal programma se non è possibile creare il file */
    if((f=fopen("studenti.txt", "r"))== NULL) {
        printf("Il file non puo essere aperto\n");
    } /* end if */
}
```

...

LETTURA DA FILE (2): ESEMPIO

...

```
else {  
    printf("%s\t %s\t %s \n", "Matricola", "Nome", "Media_Voti");  
    fscanf(f, "%s%s%lf", matricola, nome, &media_voti);  
    /*legge i dati dal file*/  
    while(!feof(f)) {  
        printf("%s\t %s\t %lf\n", matricola, nome, media_voti);  
        fscanf(f, "%s%s%lf", matricola, nome, &media_voti);  
    } /* end while */  
  
    fclose(f); /*chiude il file */  
}  
  
return 0;  
}
```

Matricola	Nome	Media_Voti
001	Mario_Rossi	25.400000
002	Giuseppe_Verdi	26.000000
003	Anna_Bianchi	28.300000

MODIFICARE UN FILE AD ACCESSO SEQUENZIALE

- Modificare un file di testo
 - Cambiare la media dei voti di Giuseppe Verdi

Matricola	Nome	Media_Voti
001	Mario_Rossi	25.400000
002	Giuseppe_Verdi	26.000000
003	Anna_Bianchi	28.300000

26.0 → 27.0

- Con l'accesso sequenziale l'unico modo di fare questa operazione è
 1. usare un file temporaneo
 2. ricopiare il file modificato
 3. cancellare il file originale
 4. dare al file temporaneo il nome del file originale

MODIFICA DI FILE (1): ESEMPIO

```
/*Modifica di in un file sequenziale */

#include <stdio.h>

int main() {

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    FILE *f_in;         /*f_in=puntatore del file "studenti.txt" */
    FILE *f_temp;       /*f_temp=puntatore del file temporaneo */

    /*si esce dal programma se non e' possibile aprire il file */
    if((f_in=fopen("studenti.txt", "r"))== NULL) {
        printf("Il file non puo essere aperto\n");
    } /* end if */
    else {
        /*si esce dal programma se non e' possibile creare il file */
        if((f_temp=fopen("temp.txt", "w"))== NULL) {
            printf("Il file non puo essere creato\n");
        } /* end if */
    }
}
```

...

MODIFICA DI FILE (2): ESEMPIO

```
...  
    else {  
        fscanf(f_in, "%s%s%lf", matricola,nome,&media_voti);  
        /*legge i dati dal file*/  
        while(!feof(f_in)) {  
            if(strcmp(matricola,"002") == 0) {  
                media_voti=27.0;  
                printf("Sto cambiando la media dei voti di %s\n", nome);  
            }  
            fprintf(f_temp,"%s\t%s\t%lf\n", matricola, nome, media_voti);  
            fscanf(f_in,"%s%s%lf", matricola,nome,&media_voti);  
        } /* end while */  
  
        fclose(f_temp); /*chiude il file temporaneo*/  
    }  
    fclose(f_in); /*chiude il file "studenti.txt"*/
```

- Il programma può ritenersi concluso?

CANCELLAZIONE E RIDENOMINAZIONE

- Il programma può ritenersi concluso?
 1. usare un file temporaneo
 2. ricopiare il file modificato
 3. cancellare il file originale
 4. dare al file temporaneo il nome del file originale
- Funzioni di cancellazione e ridenominazione nella libreria standard
 - `remove`
 - cancella il file il cui nome è specificato come argomento
 - `rename`
 - rinomina il file indicato come primo argomento con il nome specificato come secondo argomento

MODIFICA DI FILE (3): ESEMPIO

...

```
remove("studenti.txt");  
rename("temp.txt", "studenti.txt");  
  
return 0;  
}
```

```
Sto cambiando la media dei voti di Giuseppe_Verdi
```

```
-----
```

```
Process exited after 0.2009 seconds with return value 0
```

```
Premere un tasto per continuare . . .
```

INSERIMENTO ORDINATO

- Utilizziamo un campo per inserire i dati nel file in maniera ordinata
 - Per il nostro esempio usiamo la matricola
- Ogni volta che si inserisce una riga
 - Deve essere posizionata in modo corretto
 - Tutti gli altri dati devono continuare a persistere nel file
 - Se usiamo un campo di ordinamento possiamo controllare che i valori di quel campo siano univoci

INSERIMENTO ORDINATO: MAIN

- È utile costruire una funzione che si occupi dell'inserimento della riga nel file

```
/* Dichiarazione della funzione di inserimento */  
int insert_into(char *matricola, char *nome, double media_voti);
```

- La procedura generale può essere strutturata nel modo seguente:
 1. finché esistono dati da inserire
 - utilizza la funzione “insert_into”
 2. alla fine termina l'esecuzione

INSERIMENTO ORDINATO (1): ESEMPIO

```
/*Inserimento ordinato in un file */

#include <stdio.h>

/* Dichiarazione della funzione di inserimento*/
int insert_into(char *matricola, char *nome, double media_voti);

int main() {

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    printf("Inserisci la matricola, il nome e la media dei voti di uno studente\n");
    printf("Inserisci EOF per concludere l'inserimento dei dati\n");
    printf("? ");
    scanf("%s%s%lf", matricola, nome, &media_voti);

    /*Richiama la funzione per scrivere nel file*/
    while(!feof(stdin)) {
        insert_into(matricola, nome, media_voti);
        printf("? ");
        scanf("%s%s%lf", matricola, nome, &media_voti);
    } /* end while */

    return 0;
}
```

INSERIMENTO ORDINATO (2): ESEMPIO

- Possiamo arricchire la procedura generale per mostrare il successo/insuccesso di ogni inserimento:
 - Usiamo una variabile intera che memorizzerà il valore restituito dalla funzione
 - `res`
 - Associamo i seguenti valori per effettuare il controllo
 - 0: successo
 - - 1: insuccesso
 - Memorizziamo il valore restituito dalla funzione in `res`
 - Produciamo delle stampe sulla base del valore della variabile `res`

INSERIMENTO ORDINATO (3): ESEMPIO

```
int main(){

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti; /* media dei voti */

    int res = -1; /* Variabile di controllo sull'output della funzione */

    printf("Inserisci la matricola, il nome e la media dei voti di uno studente\n");
    printf("Inserisci EOF per concludere l'inserimento dei dati\n");
    printf("? ");
    scanf("%s%s%lf", matricola, nome, &media_voti);

    /*Richiama la funzione per scrivere nel file*/
    while(!feof(stdin)) {
        res = insert_into(matricola, nome, media_voti);
        if(res==0){
            printf("? Studente %s inserito con successo\n", nome);
        }
        else {
            printf("? Studente %s NON inserito\n", nome);
        }
        printf("? ");
        scanf("%s%s%lf", matricola, nome, &media_voti);
    } /* end while */

    return 0;
}
```

INSERIMENTO ORDINATO (4): ESEMPIO

- In generale dobbiamo utilizzare il metodo del file temporaneo per aggiornare il file originale in modo consistente

```
FILE *f_in;           /*f_in=puntatore del file "studenti.txt" */  
FILE *f_temp;         /*f_temp=puntatore del file temporaneo */
```

- La funzione insert_into può effettuare il seguente controllo iniziale:
 1. Se il file non esiste
 - Si apre in scrittura
 - Si scrive direttamente sul file «studenti.txt»
 2. Un errore verrà generato se non si può aprire neanche in scrittura

INSERIMENTO ORDINATO (5): ESEMPIO

```
int insert_into(char *matricola_new, char *nome_new, double media_voti_new) {

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    FILE *f_in;          /*f_in=puntatore del file "studenti.txt" */
    FILE *f_temp;        /*f_temp=puntatore del file temporaneo */

    /*si esce dal programma se non e' possibile aprire il file */
    if((f_in=fopen("studentiOrd.txt", "r"))== NULL) {
        if((f_in=fopen("studentiOrd.txt", "w"))== NULL) {
            printf("Il file non puo essere ne aperto ne creato\n");
            return -1;
        } /* end if */
        else {
            printf("? Inserisco la prima riga con lo studente %s\n", nome_new);
            fprintf(f_in,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);
            fclose(f_in);
            return 0;
        }
    } /* end if */

    ...
}
```

INSERIMENTO ORDINATO (6): INSERT_INT0

- Si parte con il processo di lettura dal file originale e inserimento nel file temporaneo
- È importante controllare il punto in cui inserire la nuova riga:
 1. Si legge una riga
 2. Finché si può continuare a leggere
 - Controllo se la riga letta ha una matricola maggiore di quella inserita
 - › In caso affermativo si inserisce la riga nuova
 - Si inserisce la riga letta dal file

INSERIMENTO ORDINATO (7): ESEMPIO

```
else {  
    /*si esce dal programma se non e' possibile creare il file */  
    if((f_temp=fopen("temp.txt", "w"))== NULL) {  
        printf("Il file non puo essere creato\n");  
    } /* end if */  
    else {  
        fscanf(f_in, "%s%s%lf", matricola,nome,&media_voti);  
  
        /*legge i dati dal file*/  
        while(!feof(f_in)) {  
            if(strcmp(matricola,matricola_new) > 0){  
                fprintf(f_temp,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);  
            }  
            fprintf(f_temp,"%s\t %s\t %lf\n", matricola, nome, media_voti);  
            fscanf(f_in,"%s%s%lf", matricola,nome,&media_voti);  
        } /* end while */  
  
        fclose(f_temp); /*chiude il file temporaneo*/  
    }  
    fclose(f_in); /*chiude il file "studenti.txt"*/  
}
```

...

INSERIMENTO ORDINATO (8): ESEMPIO

- Possiamo arricchire la funzione di inserimento controllando che nessun studente abbia la stessa matricola:
 - Controllo sull'uguaglianza di due stringhe
 - `strcmp(matricola, matricola_new) == 0`
 - Quando la condizione è verificata è necessario
 - Stampare un messaggio di errore
 - Chiudere tutti i file aperti
 - Rimuovere il file temporaneo
 - Restituire - 1

INSERIMENTO ORDINATO (9): ESEMPIO

```
else {
    /*si esce dal programma se non e' possibile creare il file */
    if((f_temp=fopen("temp.txt", "w"))== NULL) {
        printf("Il file non puo essere creato\n");
    } /* end if */
    else {
        fscanf(f_in, "%s%s%lf", matricola,nome,&media_voti);

        /*legge i dati dal file*/
        while(!feof(f_in)) {
            if(strcmp(matricola,matricola_new) == 0) {
                printf("? Non puoi inserire uno studente con la stessa matricola\n");
                fclose(f_temp);
                fclose(f_in);
                remove("temp.txt");
                return -1;
            }
            else if(strcmp(matricola,matricola_new) > 0){
                fprintf(f_temp,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);
            }
            fprintf(f_temp,"%s\t %s\t %lf\n", matricola, nome, media_voti);
            fscanf(f_in,"%s%s%lf", matricola,nome,&media_voti);
        } /* end while */

        fclose(f_temp); /*chiude il file temporaneo*/
    }
    fclose(f_in); /*chiude il file "studenti.txt"*/
}
```

...

INSERIMENTO ORDINATO (10): ESEMPIO

- Se provassimo ad eseguire il codice prodotto
 - Esiste un caso in cui il programma non funziona
 - Quando si inserisce come ultimo studente uno studente che deve essere posizionato in coda
 - È necessario utilizzare una variabile che controlli se la riga scritta da console sia stata inserita

```
int inserted = 0; /*Variabile di controllo per l'inserimento */
```

- Associamo i seguenti valori per effettuare il controllo
 - 1: riga inserita
 - 0: riga non inserita
- Controlliamo alla fine del while se la riga non è stata ancora inserita

...

INSERIMENTO ORDINATO (11): ESEMPIO

```
int insert_into(char *matricola_new, char *nome_new, double media_voti_new) {

    char matricola[3]; /*numero di matricola*/
    char nome[50];      /*nominativo*/
    double media_voti;  /* media dei voti */

    FILE *f_in;          /*f_in=puntatore del file "studenti.txt" */
    FILE *f_temp;        /*f_temp=puntatore del file temporaneo */

    int inserted = 0; /*Variabile di controllo per l'inserimento */

    /*si esce dal programma se non e' possibile aprire il file */
    if((f_in=fopen("studentiOrd.txt", "r"))== NULL) {
        if((f_in=fopen("studentiOrd.txt", "w"))== NULL) {
            printf("Il file non puo essere ne aperto ne creato\n");
            return -1;
        } /* end if */
    } else {
        printf("? Inserisco la prima riga con lo studente %s\n", nome_new);
        fprintf(f_in,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);
        fclose(f_in);
        return 0;
    }
} /* end if */
else {
    ...
}
```

INSERIMENTO ORDINATO (12): ESEMPIO

```
else {
    /*si esce dal programma se non e' possibile creare il file */
    if((f_temp=fopen("temp.txt", "w"))== NULL) {
        printf("Il file non puo essere creato\n");
    } /* end if */
    else {
        fscanf(f_in, "%s%s%lf", matricola,nome,&media_voti);
        /*legge i dati dal file*/
        while(!feof(f_in)) {
            if(strcmp(matricola,matricola_new) == 0) {
                printf("? Non puoi inserire uno studente con la stessa matricola\n");
                fclose(f_temp);
                fclose(f_in);
                remove("temp.txt");
                return -1;
            }
            else if(strcmp(matricola,matricola_new) > 0){
                fprintf(f_temp,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);
                inserted = 1;
            }
            fprintf(f_temp,"%s\t %s\t %lf\n", matricola, nome, media_voti);
            fscanf(f_in,"%s%s%lf", matricola,nome,&media_voti);
        } /* end while */
        if(inserted == 0){
            fprintf(f_temp,"%s\t %s\t %lf\n", matricola_new, nome_new, media_voti_new);
        }
        fclose(f_temp); /*chiude il file temporaneo*/
    }
    fclose(f_in); /*chiude il file "studenti.txt"*/
}
remove("studentiOrd.txt");
rename("temp.txt", "studentiOrd.txt");
return 0;}
```


CASO DI STUDIO: DATABASE DI CANZONI

- Problema

- WOLD, una stazione radio locale, vuole costruire un database di canzoni, per automatizzare le ricerche
- Si è creato un file in cui sono stati inseriti degli elementi composti dai titoli e dai compositori delle canzoni
- Si intende dare al disk-jockey la possibilità di
 - cercare nel database tutte le canzoni di un particolare artista
 - inserire una nuova canzone
 - modificare il nome di un artista
 - cancellare un artista con tutte le sue canzoni

CASO DI STUDIO: DATABASE DI CANZONI

- Scenario di esempio

Inserisci il nome del file contenente il database di canzoni:

Classi cRock. txt

File Classi cRock. txt loaded.

Inserisci l'artista da cercare:

Beatles

Canzoni dei Beatles trovate:

Back_in_the_USSR

Paperback_writer

She_Loves_You

Inserisci l'artista da cercare:

Mozart

Nessuna canzone di Mozart trovata

CASO DI STUDIO: RICERCA DI CANZONI

- Scenario di esempio

Inserisci il nome della canzone da inserire:

She_Loves_You

Inserisci l'artista da associare alla canzone

Beatles

Canzone già presente nel database

Inserisci il nome della canzone da inserire:

Hey_Jude

Inserisci l'artista da associare alla canzone

Beatles

Canzone inserita con successo

CASO DI STUDIO: MODIFICA DI CANZONI

- Scenario di esempio

Inserisci il nome dell'artista da modificare:

Beatle

Artista non trovato

Inserisci il nome dell'artista da modificare:

Beatles

Inserisci il NUOVO nome dell'artista:

The_Beatles

Canzoni a cui è stato sostituito l'artista:

Back_in_the_USSR

Paperback_writer

She_Loves_You

Hey_Jude

CASO DI STUDIO: CANCELLAZIONE DI CANZONI

- Scenario di esempio

Inserisci il nome dell'artista da cancellare:

Beatles

Artista non trovato

Inserisci il nome dell'artista da cancellare:

The_Beatles

Artista cancellato:

The_Beatles

Canzoni cancellate:

Back_in_the_USSR

Paperback_writer

She_Loves_You

Hey_Jude