Gestione dei files

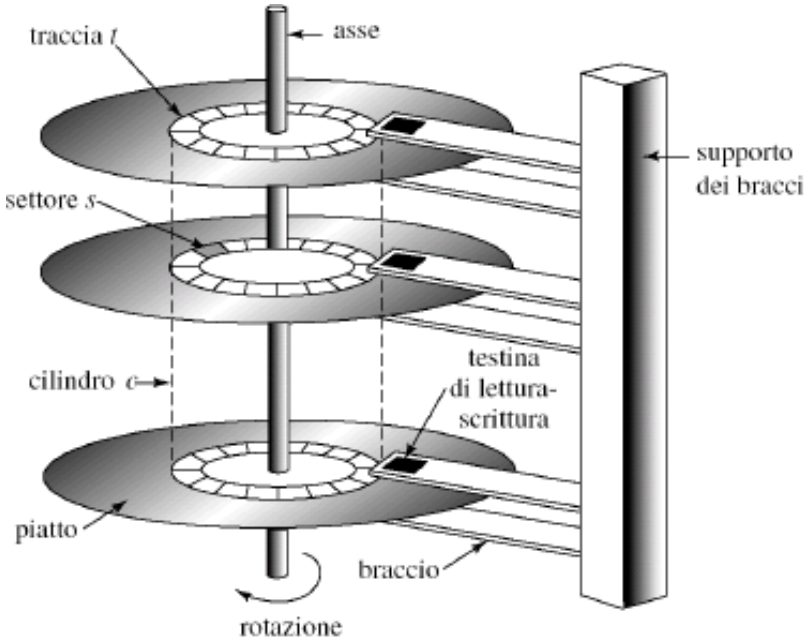
Prof. Francesco Accarino

IIS Altiero Spinelli Sesto San Giovanni

Via Leopardi 132

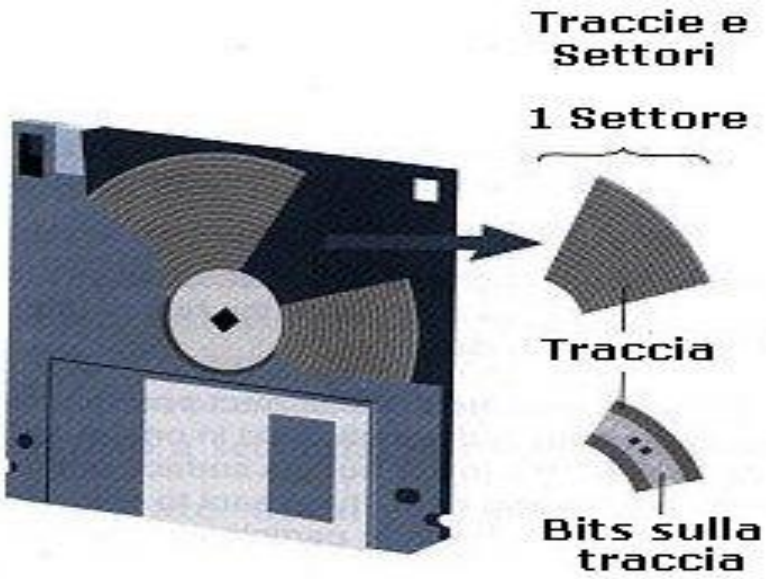
Struttura del disco fisso

- Un disco fisso è composto da una serie
  di **piatti** sovrapposti
- Ogni piatto è suddiviso in **tracce** circolari concentriche
- Ogni traccia è suddivisa in una serie di
  settori
- L'insieme delle tracce nella stessa posizione sui diversi piatti prende il nome di **cilindro**
- Un braccio mobile supporta una testina
  di lettura e scrittura per ogni piatto



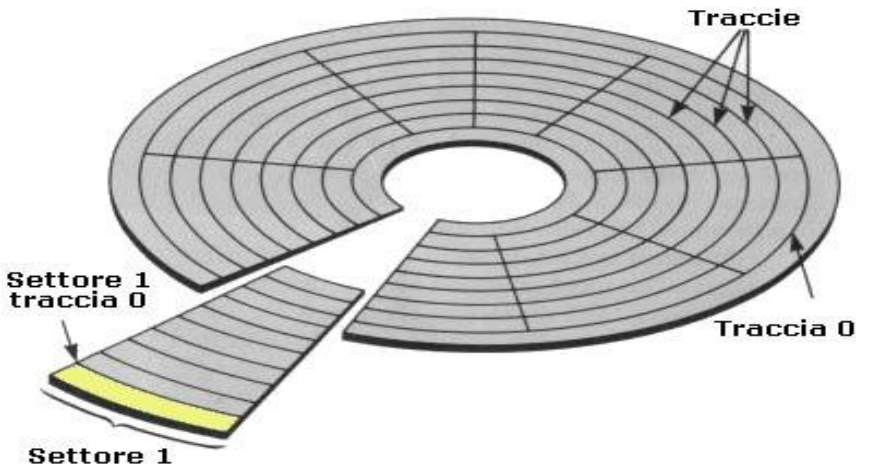
Un indirizzo fisico
è una terna (c,t,s)

Struttura di un Floppy Disk



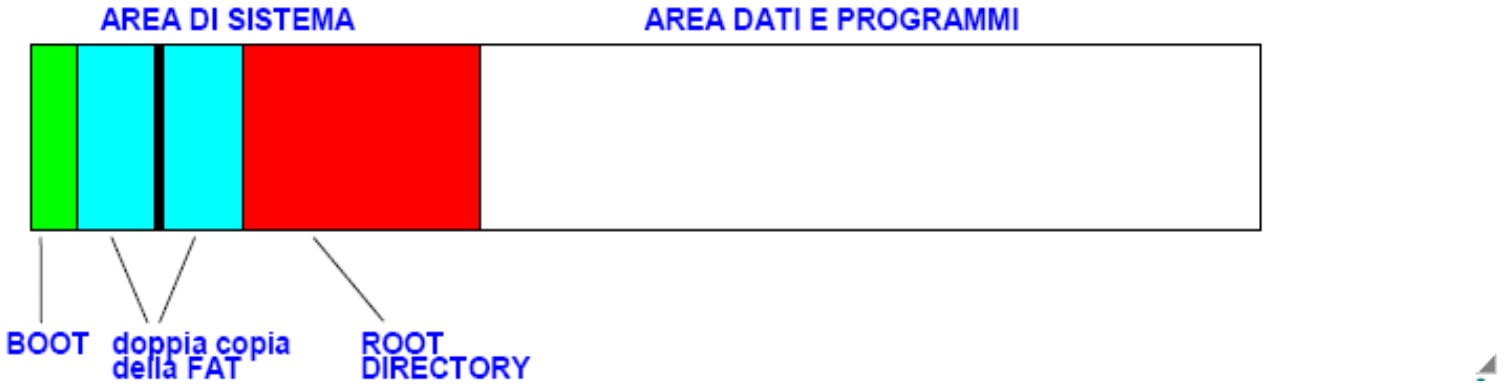
FLOPPY DISK

80 tracce per faccia, 18 settori per traccia, 512 byte per settore
1,44MB = 512 byte * 18 settori * 80 tracce * 2 lati del disco.

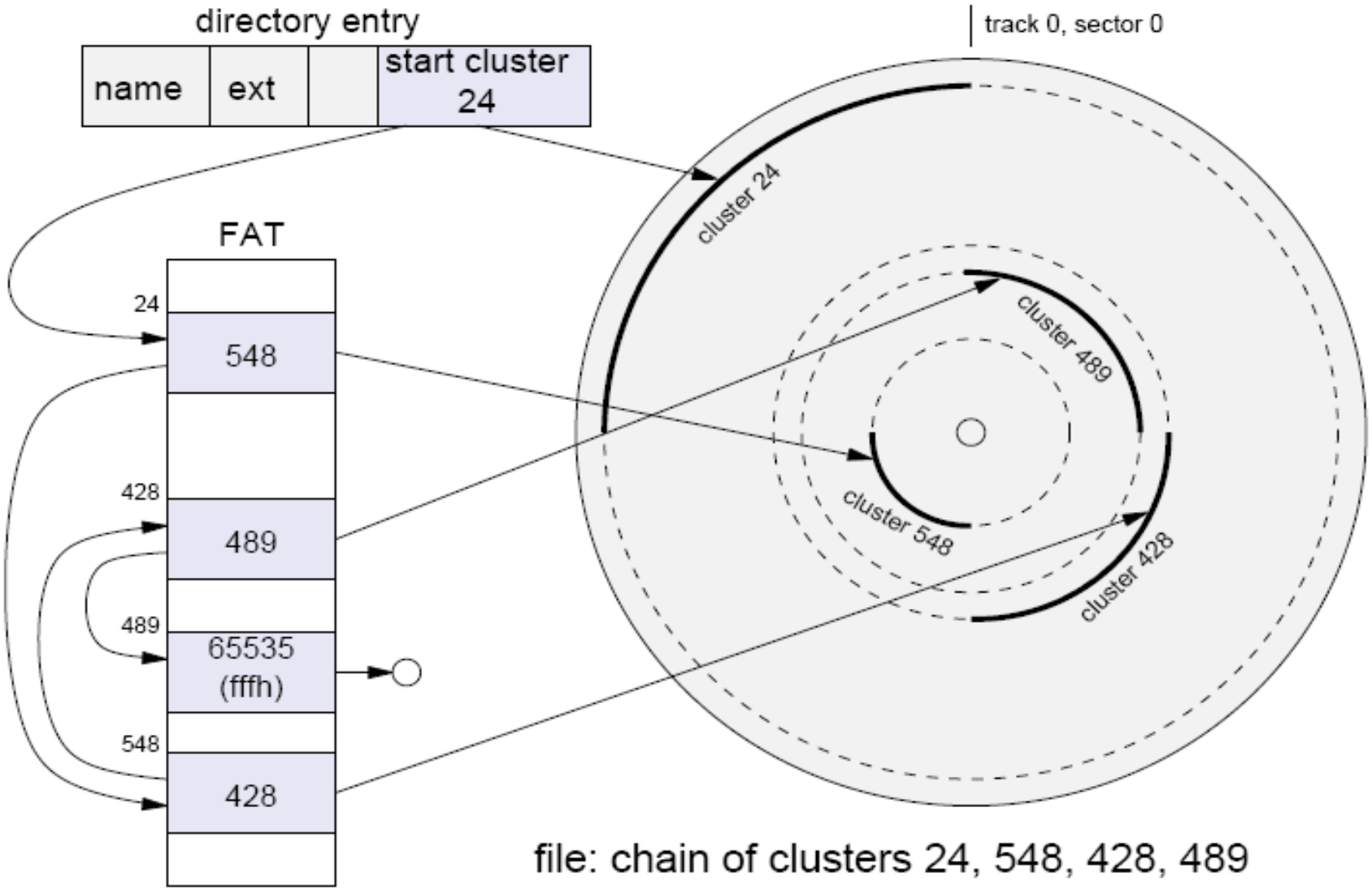


Organizzazione di un disco DOS

- **BOOT** : Contiene un piccolo programma che avvia il processo di caricamento del DOS nella memoria principale, nel caso di partizione "bootable".
- **FAT**: è la mappa di allocazione della partizione; le prime due voci della FAT sono riservate per informazioni sul tipo di disco; le altre numerate a partire da 2 contengono una voce per ogni cluster e ne indicano lo stato.
- **ROOT DIRECTORY**: il direttorio radice serve a memorizzare informazioni fondamentali sull'etichetta del volume, sui file di disco e sui sottodirettori (subdirectory), visti come file speciali. Per ogni file o subdirectory esiste nel direttorio radice una voce che contiene: nome, tipo (es. EXE), attributi, riservato per usi futuri, ora, data, entry iniziale nella FAT, dimensione.
- **SUBDIRECTORY**: file di tipo speciale che appartengono al direttorio radice o ad altri sottodirettori e sono organizzati in modo analogo al root directory.



File Allocation Table (FAT)



File e Sistema Operativo

- Un file è una astrazione fornita dal sistema operativo, per consentire la memorizzazione di informazioni su memoria di massa, cioè:
  - Il sistema operativo opera una astrazione

    rispetto alle modalità con cui vengono effettivamente memorizzati i dati su un dispositivo di memoria esterna

I File in C

I file in C vengono distinti in due categorie

- ***file di testo***, trattati come sequenze di caratteri. Organizzati in linee (ciascuna terminata da EOL)
- ***file binari***, visti come sequenze di byte

Operazioni con i file

- Prima di poter effettuare qualsiasi operazione sui file è necessario **“aprire il file”**
- Al termine del suo utilizzo è necessario **“chiudere il file”**

Sintassi per apertura file:

<Identificatore>=**fopen(<**NomeFile>, <modalità>**)**

<Identificatore>	Variabile puntatore di tipo “FILE” detto “filepointer”, all’interno del programma il programmatore deve usare il “file pointer” per accedere al file.
<NomeFile>	Stringa contenente il nome del file compreso il path (prova.txt; ../prova.txt; c:/..../prova.txt; Dati/prova.txt)
<Modalità>	Uno dei modi elecati nella prossima slide

Nota: nel programma deve essere incluso “**stdio.h**”

Modalità di apertura file

- **“r”** Solo lettura - se il file al momento dell’apertura non esiste allora la funzione di apertura restituisce NULL
- **“w”** Solo scrittura - se il file al momento dell’apertura non esiste allora sarà **utomaticamente creato** - se il file al momento dell’apertura esiste allora il **contenuto** del file preesistente **andrà perso**
- **“a”** Append (sul file sarà possibile scrivere in fondo al file) - se il file al momento dell’apertura non esiste allora sarà **automaticamente creato** - se il file al momento dell’apertura esiste allora il **contenuto** del file preesistente **rimarrà invariato**
- **“r+”** Lettura e Scrittura - se il file al momento dell’apertura non esiste allora la funzione di apertura restituisce NULL
- **“w+”** Lettura e Scrittura - se il file al momento dell’apertura non esiste allora sarà **automaticamente creato** - se il file al momento dell’apertura esiste allora il **contenuto** del file preesistente **andrà perso**
- **“a+”** Lettura ed Append (sul file sarà possibile eseguire sia operazioni di lettura che di scrittura) - se il file al momento dell’apertura non esiste allora sarà **automaticamente creato**- se il file al momento dell’apertura esiste allora il **contenuto** del file preesistente **rimarrà invariato**

Aggiungendo la ettera b si ottengolno le stesse funzioni ma per file binari

Esempio di apertura in C

- Esempio di apertura:

```
//Esempio di apertura di un file in sola lettura
#include <stdio.h>
void main ()
{
FILE *fp;
fp=fopen ("carattere.txt","r");
……..
//Esempio di apertura di un file in append
#include <stdio.h>
void main ()
{
FILE *fp;
fp=fopen ("../prova.txt",“a");
……..
```

Apertura e chiusura di un file

- Sintassi per chiusura file: **fclose (**<Identificatore>**)**
  <Identificatore> nome della variabile “file pointer” usato in apertura

Esempio di chiusura:

```
#include <stdio.h>
void main ()
{
FILE *fp;
fp=fopen ("carattere.txt","r");
……..
fclose (fp)
……..
```

Lettura e scrittura

Per file di testo sono disponibili funzioni di:

- Lettura/scrittura **con formato**
- Lettura/scrittura di **caratteri**
- Lettura/scrittura di **stringhe di caratteri**

Per file binari si utizzano funzioni di:

- Lettura/scrittura di **blocchi**

Ogni operazione di lettura (o scrittura) provoca l'avanzamento del puntatore al file al primo record logico (carattere , linea o blocco) non letto (o libero, nel caso discrittura).

Lettura con formato

Si usa la funzione fscanf:

int fscanf **(FILE *fp, *stringa-controllo*, *ind-elem*);**

dove:

- fp e` il puntatore al file
- *stringa-controllo* indica il formato dei dati da leggere
- *ind-elem* e` la lista degli indirizzi delle variabili a cui assegnare i valori letti.

Esempio:

```
FILE *fp;
int A; char B; float C;
fp=fopen("dati.txt", "r");
fscanf(fp, "%d%c%f", &A, &B, &C);
...
fclose(fp);
```

Restituisce il numero di elementi letti, oppure un valore negativo in caso di errore.

Esempio1

Visualizzazione del contenuto di un file di testo:

```
#include <stdio.h>
#include <conio.h>
void main(){
char buf[50];
FILE *fp;
fp=fopen("testo.txt", "r");
fscanf(fp,"%s",buf);
while(!feof(fp))
{ printf("%s", buf);
fscanf(fp,"%s",buf);
}
fclose(fp);
getche();
}
```

Esempio1

Visualizzazione del contenuto di un file di testo:

```
#include <stdio.h>
#include <conio.h>
void main(){
char buf[10];
FILE *fp;
fp=fopen("testo.txt", "r");
while (fscanf(fp,"%s",buf)>0)
printf("%s", buf);
fclose(fp);
getche();
}
```

Scrittura con Formato

Si usa la funzione fprintf:

int fprintf **(FILE *fp, *stringa-controllo*, *elem*);**

dove:

- fp e` il puntatore al file
- *stringa-controllo* indica il formato dei dati da scrivere
- *elem* e` la lista delle variabili che contengono i valori da scrivere.

Restituisce il numero di elementi scritti, oppure un valore negativo in caso di errore.

Scrittura con formato

Esempio:

```
#include <stdio.h>
#include <conio.h>
void main(){
FILE *fp;
float r;
printf("inserisci il raggio del cerchio");
scanf("%f",&r);
fp=fopen("area.txt", "w");
fprintf(fp,"L'area del cerchio e': %f", r*r*3.14);
fclose(fp);
getche();
}
```

Lettura/scrittura di **caratteri**

Scrittura carattere su file

- E' possibile operare in scrittura singoli caratteri su di un file precedentemente aperto utilizzando la seguente procedura:

fputc(<carattere>, <Identificatore>**)**

Nota: Tutti i caratteri che aggiungiamo successivamente in scrittura si accodano ai caratteri eventualmente già scritti.

Lettura carattere da file

- E' possibile operare in lettura singoli caratteri su di un file

precedentemente aperto utilizzando la seguente funzione:

<carattere> = **fgetc(**<Identificatore>**)**

Nota: Tutti i caratteri che leggiamo successivamente saranno quelli successivi al carattere appena letto.

Lettura carattere da file

Visualizzazione del contenuto di un file di testo:

```
#include <stdio.h>
#include <conio.h>
void main(){
char car;
FILE *fp;
fp=fopen("testo.txt", "r");
while (!feof(fp)){
  car=fgetc(fp);
  printf("%c",car);
}
fclose(fp);
getche();
}
```

Scrittura carattere su file

Esempio:

```
#include <stdio.h>
#include <conio.h>
void main(){
FILE *fp;
char c;
fp=fopen("testo.txt", "w");
while((c=getche())!='\r')
 fputc(c,fp);
fclose(fp);
getche();
}
```

Lettura e scrittura per stringhe

- **int fgets (char *s, int n, FILE *fp)**
  - La funzione fgets legge dal file puntato da fp al massimo n caratteri compreso il termintatore di linea EOL e li memorizza nella stringa s aggiungendo il terminatore di stringa
- **int fputs (char *s, FILE *fp)**
  - La funzione fputs memorizza nel file puntato da fp alla posizione corrente i caratteri contenuti nella stringa s

Lettura e scrittura di stringhe

```
#include <stdio.h>
#include <string.h>
#define  OK                 1
#define  ERROR              0
#define  MAXLINE            100
int copiafile()
{
        char     line[MAXLINE];
        FILE     *fin, *fout;

        If ((fin = fopen(“testo.txt", "r")) == NULL)return ERROR;
        if ((fout = fopen(“copiatesto.txt", "w")) == NULL)
        {
                fclose(fin);
                return ERROR;
        }
      while (fgets(line,MAXLINE,fin) != NULL) fputs(line,fout);

 fclose(fin);
fclose(fout);
return OK;
}
```

File binari letura scrittura di blocchi

Si puo` leggere o scrivere da un file binario un intero blocco di dati (binari). Un file binario memorizza dati di qualunque tipo, in particolare dati che non sono caratteri (interi, reali, vettori o strutture).

Per la lettura/scrittura a blocchi e` necessario che il file sia stato aperto in modo ***binario*** (modo "b")**.**

- **int fread (void **vet*, int *size*, int *n*, FILE **fp*);**

Legge (al piu`) *n* oggetti dal file puntato da *fp*, collocandoli nel vettore *vet*, ciascuno di dimensione *size*. Restituisce un intero che rappresenta il numero di oggetti effettivamente letti.

- **int fwrite (void **vet*, int *size*, int *n*, FILE **fp*);**

Scrive sul file puntato da *fp*, prelevandoli dal vettore *vet*, *n* oggetti, ciascuno di dimensione *size*. Restituisce un intero che rappresenta il numero di oggetti effettivamente scritti (inferiore ad *n* solo in caso di errore, o fine del file).

Esempio:

Programma che scrive una sequenza di record (dati da input) in un file binario:

```
#include<stdio.h>
typedef struct  { char nome[20];
                  char cognome[20];
                  int reddito;
                 }persona;
void main() {
     FILE *fp;
     persona p;
    int fine=0;
        fp=fopen("archivio.dat","wb");
       do{
           printf("Nome:?\n");scanf("%s",p.nome);
           printf("Cognome:?\n");scanf("%s",p.cognome);
           printf("Reddito:?\n");scanf("%s",&p.reddito);
           fwrite(&p,sizeof(persona),1,fp);
           printf("Fine (si=1,no=0)?");
           scanf("%d", &fine);
          }while(!fine);
     fclose(fp);
 }
```

Esempio:

Programma che legge una sequenza di record da un file binario e li visualizza

```
#include<stdio.h>
typedef struct  { char nome[20];
                  char cognome[20];
                  int reddito;
                 }persona;
void main() {
     FILE *fp;
     persona p;
   fp=fopen("archivio.dat",“rb");
       while(!feof(fp) ){
            fread(&p,sizeof(persona),1,fp);
            printf("Nome: %s\n“,p.nome);
            printf("Cognome: %s\n",p.cognome);
            printf("Reddito: %d\n",p.reddito);
       }
     fclose(fp);
 }
```