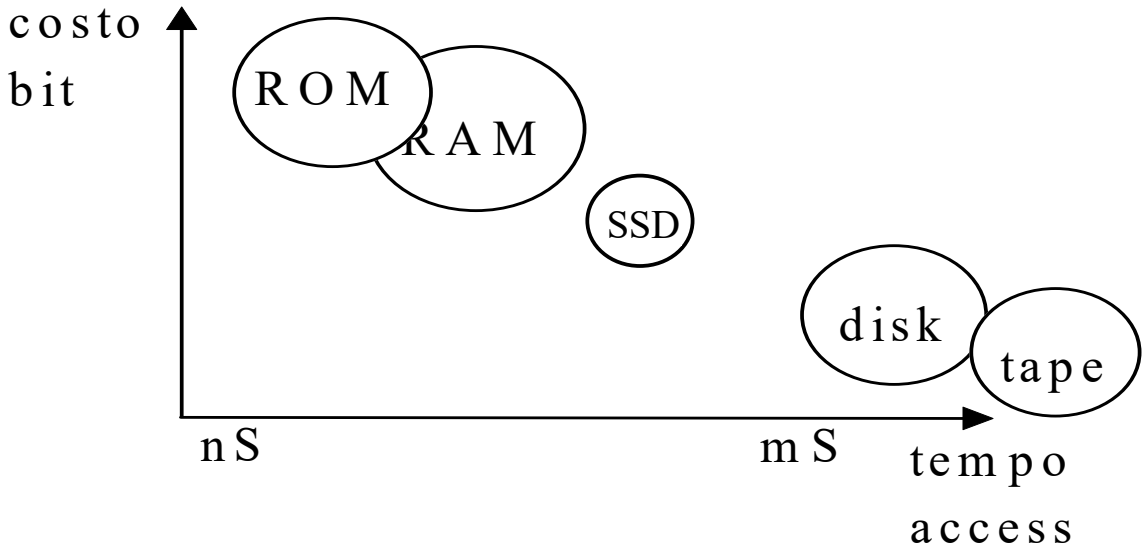
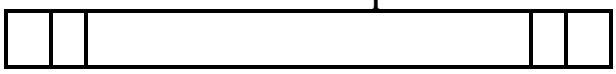
La gerarchia di memoria

CPU (registri, ROM)
Cache (CPU KB)
⇓
Memoria centrale (RAM GB)
⇓
Memoria persistente (GB/TB)

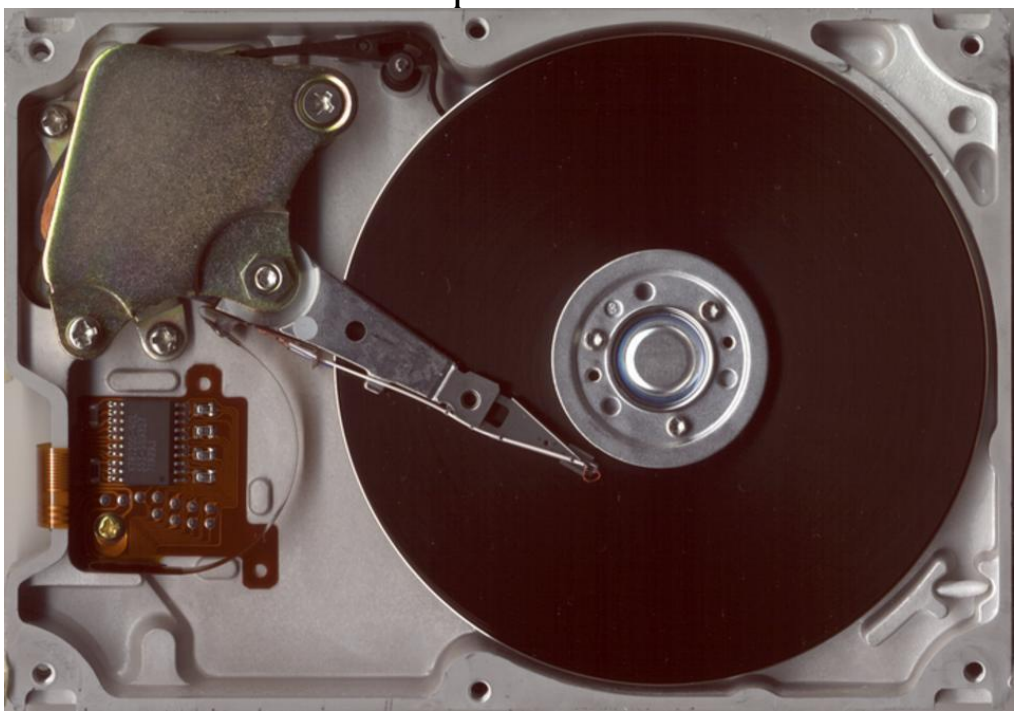


La tecnologia magnetica

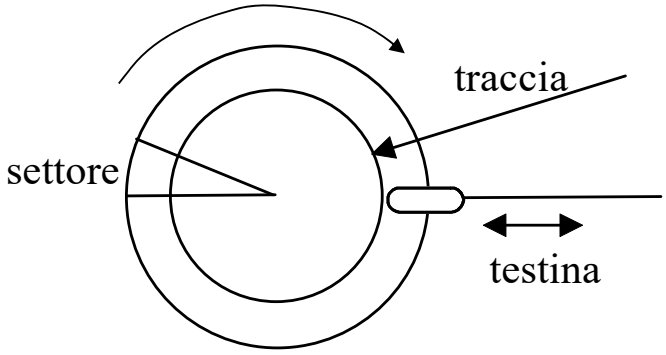
Nastro: struttura sequenziale



Il disco: struttura non sequenziale



Formattazione del S.O. a circonferenze concentriche



tempo di latenza e seek(msec)

La tecnologia ottica

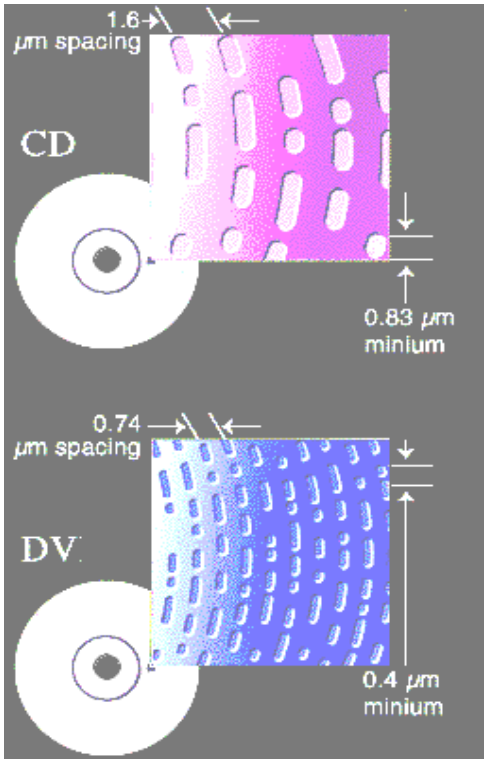
Formattazione del S.O. a spirale dall'interno verso l'esterno

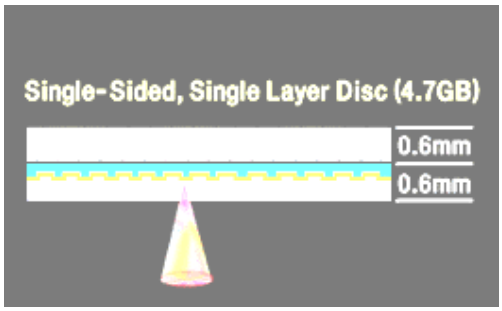
Compact disk (CD)
Formattazione ISO 9660 file system
Capacità ≈ 700MB
Velocità: nX = n*150KB

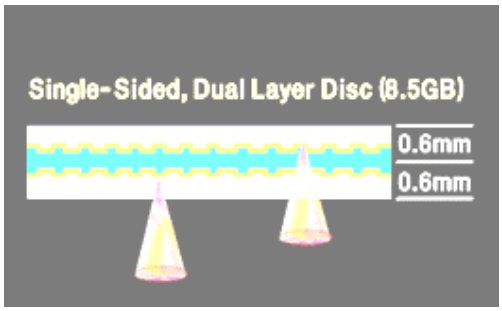
Digital versatile Disk (DVD)

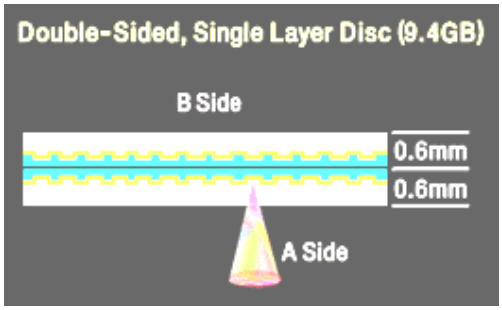
Formattazione UDF bridge
SingleSideSingleLayer (4,7GB)
....
DoubleSideDualLayer (17GB)

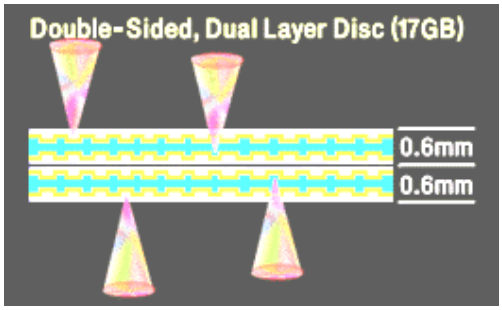
Velocità: nX = n*1,3MB/s











Una tecnologia sostitutiva?

Solid state drive (SSD)
Per confronto con i dischi:
- più veloci (≈ 50 volte)
- più robuste,
- più silenziose
- minor consumo
- capacità inferiori ai dischi e costi superiori per bit
- durata memoria (10^4 – 10^6 cicli di scrittura)

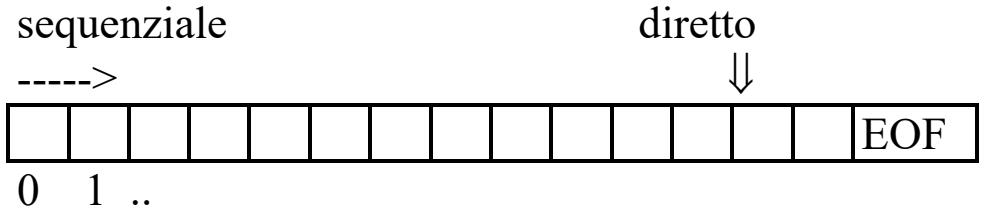
Esecuzione di una richiesta al disco

I/O request(user)
⇓
File System (SO)
⇓
R/W(unità di R/W)
⇓
Driver disco(SO) (algoritmo)
BUS⇓
Controller disco
⇓
disco
⇓
Restituzione risultato/dato

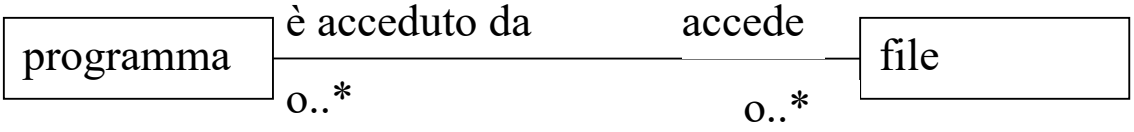
IL CONCETTO DI FILE

Struttura dati persistente di arbitraria dimensione

- file speciali (dispositivi)
- file directory
- file normali:
  - formato testo (dati, sorgenti) - portabilità, XML
  - formato binario (dati, eseguibili)
  - accesso sequenziale
  - accesso diretto (solo per formato binario)
- identificazione singolo byte



Un file ha vita indipendente dai programmi che vi accedono



Uso concorrente di un file

S.O.

Program P2		Program P1 R	Program P1 W
	R	SI	NO
	W	NO	NO

Come si connette un file ad un programma?

#include <stdio.h>

- FILE *f1; descrittore del file
- f1 = fopen(nome, mode);

- nome: stringa tra ” ” o nome vettore (stringa con \0)
- mode

“r”, “w” ,“a” lettura/scrittura/append file testo

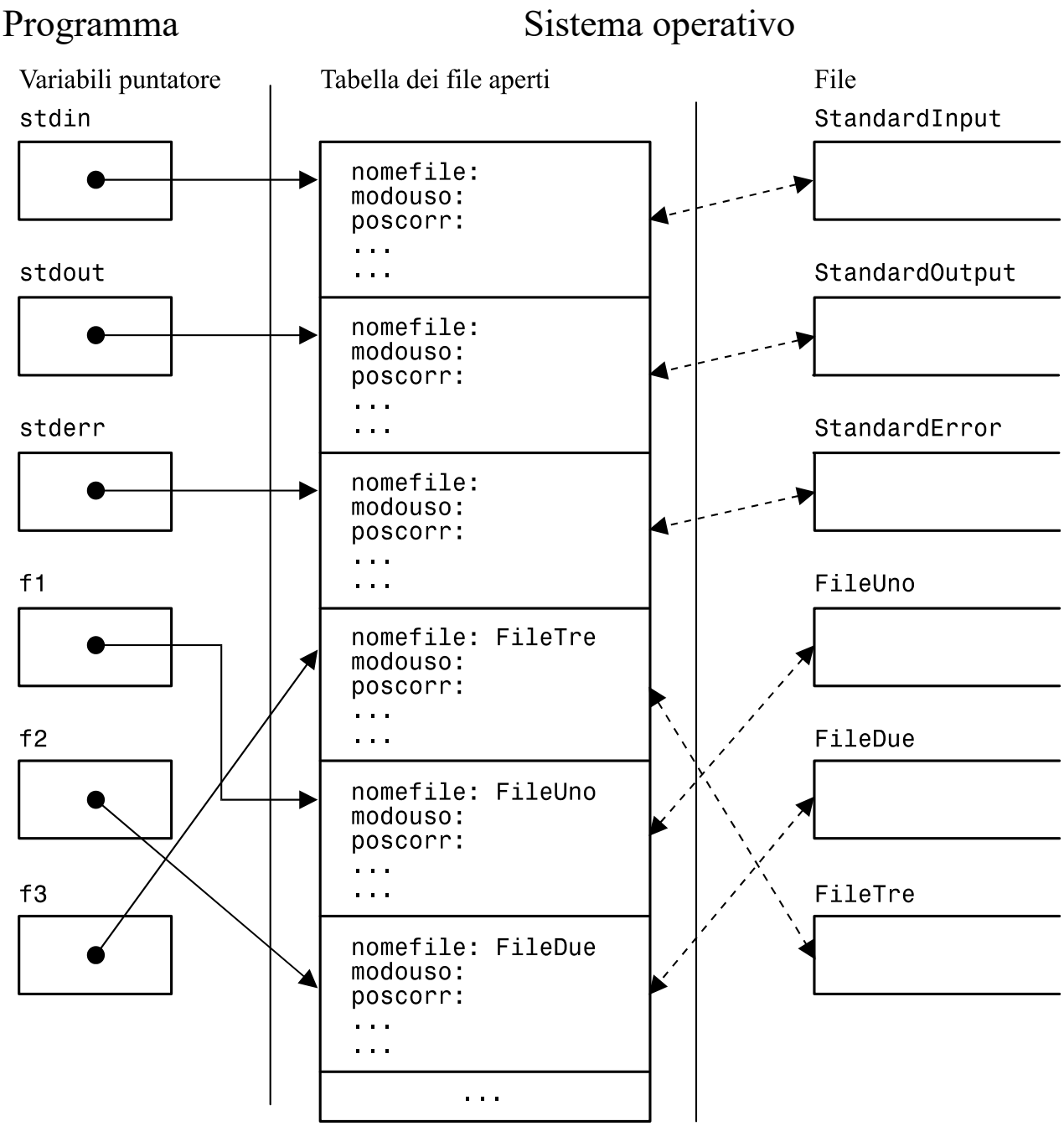
“rb” ,“wb”, “ab” lettura/scrittura/append file binario

varianti(lettura e scrittura): “r+”, “rb+” “w+” “wb+”, “a+” “ab+”

- viene valutata presenza file
- viene determinata la posizione corrente

- error ⇒ return(NULL) DA CONTROLLARE
- ok ⇒ return (indirizzo descrittore file)

Il descrittore del file



- int fclose(f1)
- OK ⇒ return (0)
- error ⇒ return(EOF)
- chiudere sempre il file per garantire l'applicazione delle write
- close automatica a fine esecuzione programma

- int fflush(fp) forza scrittura del buffer sul disco

Funzioni C di manipolazione dei dati

Tipo di accesso	Video	Tastiera	File testo	File binario
Formattato	printf	scanf	fscanf/fprintf	NO
a char	putchar	getchar	getc/putc fgetc/fputc	NO
a stringhe	puts	gets	fgets/fputs	NO
a blocchi	NO	NO	NO	fread/fwrite

Un'operazione di lettura/scrittura su file:

- viene eseguita a partire dalla posizione corrente
- sposta la posizione corrente del numero di byte letti

Accesso formattato file testo

- int fscanf(fp, stringa controllo, adr(variabili));
- legge a partire dalla posizione corrente;
- sposta la posizione corrente del numero di caratteri letti;
- NO ok ⇒ return(numero caratteri letti >0, escluso eof)
- NO error or end of file ⇒ return(EOF)

- int fprintf(fp, stringacontrollo, variabili);
- scrive a partire dalla posizione corrente;
- sposta la posizione corrente del numero di caratteri scritti
- NO ok ⇒ return(numero elementi scritti ≥ 0)
- NO error ⇒ return(<0)

Accesso a stringhe nel file testo (sequenza char terminata da newline)

- char *fgets(char *str, int n, fp);
- Lettura di n-1 chars (se non incontra eof, <ret>) dalla posizione corrente del file e li carica in str con \0.
- sposta la posizione corrente dei caratteri letti;
- NO ok ⇒ return (str) error ⇒ return(NULL)

- int *fputs(char *str, fp);
- scrive str sino a \0 (escluso) sul file a partire dalla posizione corrente
- sposta la posizione corrente dei caratteri scritti;
- NO ok ⇒ return(0) error ⇒ return(EOF)

Accesso a carattere nel file testo

- int fgetc(fp); funzione (int getc(fp); macro)
- legge a partire dalla posizione corrente;
- sposta la posizione corrente di un carattere;
- NO ok ⇒ return(codifica numerica ASCII del carattere letto)
- NO error ⇒ return(EOF)
-
- int fputc(c, fp); funzione (int putc(c, fp); macro)
- scrive a partire dalla posizione corrente;
- sposta la posizione corrente di un carattere;
- scrive il carattere c (ASCII)
- NO ok ⇒ return(carattere scritto)
- NO error ⇒ return(EOF)

Esempio 1: lettura file a caratteri e visualizzazione

```
#include <stdio.h>
#include <stddef.h>  /* NULL */
int main()
{ FILE *fp;   char c;
  fp = fopen("filechar", "r");
  if ( fp != NULL )
  { c = fgetc(fp);
    while (c  != EOF)  {putchar(c); c = fgetc(fp);}
    fclose(fp);
  }
  else
    printf("Il file non può essere aperto\n");
}
```

Esempio 2: copia di un file in un altro file carattere x carattere

```
#include <stdio.h>
#include <stddef.h>
int main()
{ FILE *fr, *fw; int  c;
   fr = fopen("filer", "r"); fw = fopen("filew", "w");
   if ((fr != NULL) &&(fw != NULL))
   { c = fgetc(fr);
      while (c  != EOF)  {fputc(c,fw); c = fgetc(fr);}
      fclose(fr); fclose(fw);
   }
   else
      printf("problemi di apertura”);
}
```

Accesso file binario

int fread(adr(var), sizeof(elemento), numelementi, fp)
- legge dalla posizione corrente (numelementi*sizeof:bytes);
- sposta la posizione corrente di numero bytes letti;
- NO return (numelementi letti)
- NO numelementi letti < numelementi ⇒ errore ⇒ feof, ferror)

int fwrite(adr(var), sizeof(elemento), numelementi, fp)
- scrive dalla posizione corrente (numelementi*sizeof:bytes);
- sposta la posizione corrente di numero bytes scritti;
- NO return (numelementi scritti)
- NO numelementi scritti < numelementi ⇒ errore)

int feof(fp);
- ultimo byte letto è EOF ⇒ return (!0) altrimenti return(0)
- deve essere preceduta da almeno una lettura su file (feof() dopo fopen ritorna 0)

Accesso diretto
- int fseek(fp, long offset, int origine)
- origine = SEEK_SET ⇒ pos.corrente = BOF + offset
  SEEK_CUR ⇒ pos.corrente = pos.corrente + offset
- NO ok ⇒ return(0)
- NO error ⇒return(<0)

- long int ftell(fp)
- return (posizione corrente) se numero >0 altrimenti return <0

- rewind (fp);
- posizione corrente= BOF

Interscambio file senza interoperabilità

Esempio:

```
#include <stdio.h>
#define N 10
FILE       *fw; struct{int a;int b;} V[N];
void main()
{int i;
//esempio di caricamento struttura dati del mittente
for (i=0;i<N;i++){V[i].a=100+i;V[i].b=200+i;}

fw = fopen("filewb", "wb");
       if (fw != NULL)
              {for (i=0;i<N;i++) fwrite(&V[i],sizeof(V[i]),1,fw);
              fclose(fw);
              }
       else
              printf("problemi di apertura");
  }
```

Osservazione:

l'istruzione fwrite(&V[i],sizeof(V[i]),1,fw); trasferisce un elemento del vettore

E' possibile trasferire l'intero vettore con una write

- fwrite(&V[0],sizeof(V[0]),N,fw);
- fwrite(&V[0],sizeof(V[i])*N,1,fw);

Interoperabilità tra applicazioni

- tipo di file (caratteri)
- convenzione di transcodifica: (ad es., XML,....)

Esempio: scrittura di un vettore di struct in un file testo. Convenzione: ogni riga contiene i due campi della struct separati da un '-'.

```
#include <stdio.h>
#include <stdlib.h>/*itoa*/
#define N 10
FILE      *fw; struct{int a;int b;} V[N];
void main()
{int i,j;char temp1[20], temp2[20];

for (i=0;i<N;i++){V[i].a=100+i;V[i].b=200+i;}

fw = fopen("filew", "w");
      if (fw != NULL)
            {for (i=0;i<N;i++)
                  {     itoa( V[i].a, temp1, 10);itoa( V[i].b, temp2, 10);
                        j=0;
                        while (temp1[j]!='\0')
                              {fputc(temp1[j],fw);j++;}
                        fputc('-',fw);
                        j=0;
                        while (temp2[j]!='\0')
                              {fputc(temp2[j],fw);j++;}
                        fputc('\n',fw);
                  }
            fclose(fw);
            }
      else
            printf("problemi di apertura");
  }
```

Contenuto del file letto con un editor

100-200
101-201
102-202
103-203
104-204
105-205
106-206
107-207
108-208
109-209

Formato xml alternative

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
.....

<elemento><campo a> 100 </campo a><del> - </del><campo b> 200 </campo b></elemento>
<elemento><campo a> 101 </campo a><del> - </del><campo b> 201 </campo b></elemento>
....
<elemento><campo a> 109 </campo a><del> - </del><campo b> 209 </campo b></elemento> …