La memoria secondaria

- Caratteristiche della *memoria secondaria*

  Persistenza I dati sono conservati per lungo tempo anche in assenza di energia

  Tempo di accesso dell'ordine dei millisecondi

  Costo per byte molto inferiore alla memoria interna o primaria: oggi ≈ 0.5 Euro/GB contro ≈ 100 Euro/GB

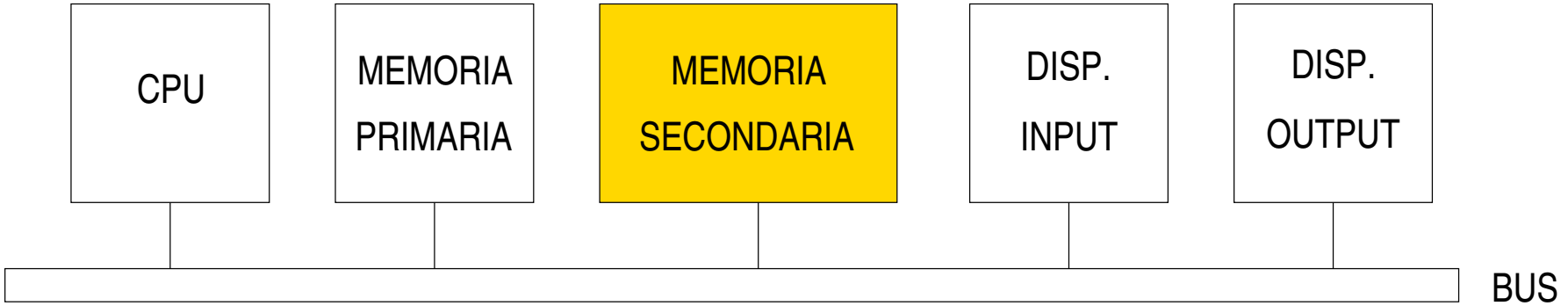
- Realizzazioni della memoria secondaria

  Hard disk Supporto magnetico di grande capacità: $\sim$ centinaia di GB, scrivibile, fragile

  CD/DVD Supporto ottico di bassa capacità: CD $\approx 700 \div 800$ MB, DVD ≈ 8 GB robusto, scritture lente

La memoria secondaria

- Utilità della memoria secondaria
  - La quasi totalità dei dati oggi deve essere memorizzata in modo persistente
  - La memoria primaria non ha dimensioni sufficienti per molte applicazioni
    - Collezioni di programmi di uso comune (fino a varie decine di GB)
    - Basi di dati di grandi dimensioni (fino a vari Tera Byte) (1 Tera Byte $= 10^3$ GB)
- Architettura con memoria secondaria



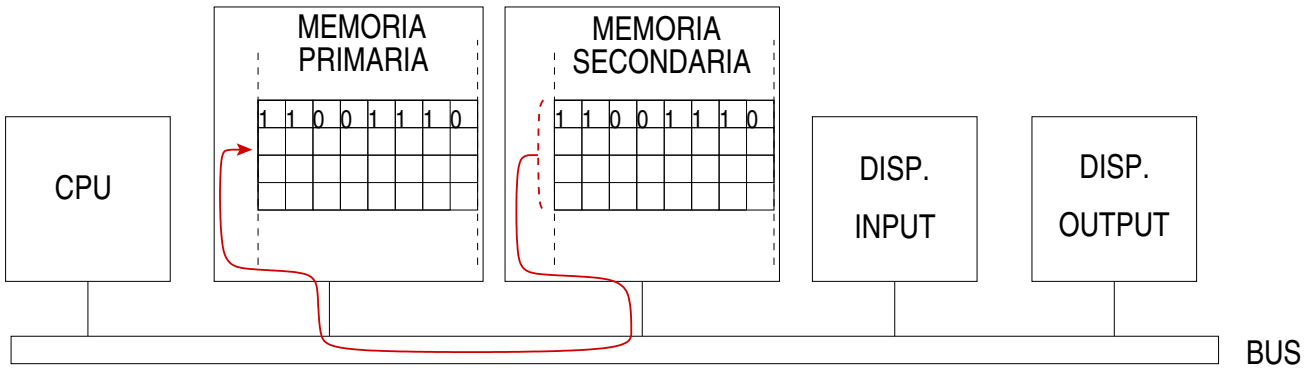
Il file

File strutturato sequenza, di lunghezza non prefissata, di valori dello stesso tipo

- Accesso a un componente di un file
  - Si individua la posizione nella memoria secondaria del componente cercato

    Sequenziale Per accedere a un componente, tutti i componenti precedenti devono essere letti

    Diretto Specificando l'indirizzo del componente nel file
  - Si effettua una operazione di ingresso
    - Il componente viene copiato in memoria primaria



Il file

- Il file in C è definito come una sequenza di lunghezza non prefissata di byte
- Si parla di file testo quando i byte del file sono considerati come codici di caratteri; altrimenti si parla genericamente di file binario
- In `stdio.h` è definita la struttura `FILE`; contiene
  - nome del file
  - modalità di accesso al file
  - posizione corrente
- Per utilizzare i file è necessaria la direttiva

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

Utilizzare un file in C

1. Dichiarare un puntatore a `FILE`

   ```
   FILE *FP;
   ```

2. Aprire il file con la funzione `fopen`, la cui intestazione è

   ```
   FILE *fopen(const char *NomeFile, const char *Modo);
   ```

   dove

   - `NomeFile` è un puntatore a una stringa che contiene un nome di file valido,
   - `Modo` può essere solo `"r"`, `"w"`, `"a"`, `"r+"`, `"w+"`, `"a+"`
   - `fopen` restituisce un puntatore a una struttura di tipo `FILE` allocata dal sistema operativo in caso di successo, `NULL` altrimenti.

3. Eseguire letture e/o scritture

4. Chiudere il file con la funzione

   ```
   int fclose(FILE *FP);
   ```

Utilizzare un file in C

`"r"`	lettura; la posizione corrente è l'inizio del file
`"w"`	scrittura; tronca il file a lunghezza zero, se esiste; altrimenti crea il file; la posizione corrente è l'inizio del file
`"a"`	scrittura alla fine (*append*); il file è creato se non esiste; la posizione corrente è la fine del file
`"r+"`	lettura e scrittura; la posizione corrente è l'inizio del file
`"w+"`	lettura e scrittura; tronca il file a lunghezza zero, se esiste; altrimenti crea il file; la posizione corrente è l'inizio del file
`"a+"`	scrittura alla fine (*append*) e lettura; il file è creato se non esiste; la posizione corrente è la fine del file

Utilizzare un file in C

Posizione corrente numero (di byte) che misura la posizione attuale sul file

- una operazione di lettura o scrittura viene effettuata alla posizione successiva a quella corrente

- Nota:
  - Immediatamente dopo l'apertura in modo `"r"`, `"r"`, `"w+"`, `"w+"` la posizione vale 0
  - Immediatamente dopo l'apertura in modo `"a"`, `"a+"`, la posizione vale la lunghezza del file in byte

Utilizzare un file in C

Esempio.

```
#include <stdio.h>
#define NOME_FILE "prova.txt"
main() {
  FILE *FP;
  if ( (FP = fopen(NOME_FILE, "r")) == NULL )
    printf ("Impossibile aprire %s\n", NOME_FILE);
  else {
    /* elabora il file */
    fclose(FP);
  }
}
```

- Funzione utile:

`int feof(FILE *FP);`	restituisce non zero se non è stata raggiunta la fine del file, 0 se è stata raggiunta

File testo

- Una sequenza di caratteri, organizzata in linee
- i dispositivi di ingresso e uscita sono disponibil come file testo

  `stdin` variabile che punta al file che rappresenta la tastiera

  `stdout` variabile che punta al file che rappresenta il video
- Tre famiglie di funzioni, per la lettura/scrittura
  - a caratteri
  - a stringhe
  - formattata

File testo

- Lettura/scrittura a caratteri

`int getc(FILE *FP);`	Restituisce il prossimo carattere in `FP`, o `EOF`, o un errore
`int putc(int Ch, FILE *FP);`	Scrive `Ch` restituendo come intero il carattere scritto
`int getchar(void);`	Legge da `stdin` il prossimo carattere e lo restituisce
`int putchar(int Ch);`	Scrive `Ch` su `stdout` e lo restituisce come intero

File testo

Esempio. Lettura e visualizzazione di un file testo carattere per carattere

```
                              #include <stdio.h>
                              #define NOME_FILE "prova.txt"
                              main() {
                                FILE *FP;
                                char Ch;
- apri il file in lettura     FP=fopen(NOME_FILE, "r");
```

File testo

- *se il file esiste e si può aprire*
  - *leggi carattere dal file, assegnalo a Ch*
  - *finché non è stata raggiunta la fine file*
    - *visualizza il carattere Ch*
    - *leggi carattere, assegnalo a Ch*
  - *chiudi il file*

altrimenti

- *visualizza messaggio di errore*

```
if (FP != NULL) {
  Ch=getc(FP);

  while (!feof(FP)) {

    putchar(Ch);
    Ch=getc(FP);
  }
  fclose(FP);
}
else
  printf("Impossibile aprire %s\n
         NOME_FILE);
}
```

File testo

- Versione piú sintetica

```
#include <stdio.h>
#define NOME_FILE "prova.txt"
main() {
  FILE *FP;
  char Ch;
  if ((FP=fopen(NOME_FILE, "r")) != NULL) {
    while ((Ch=getc(FP)) != EOF)
      putchar(Ch);
    fclose(FP);
  }
  else
    printf("Impossibile aprire %s\n",
            NOME_FILE);
}
```

File testo

- I/O formattato su file avviene con modalità simili a quelle dell'I/O da tastiera e video
- Le funzioni `fprintf`, `fscanf` si comportano in modo simile a `printf`, `scanf`
  - `int fprintf(FILE *FP, const char *formato, ...);`
  - `int fscanf(FILE *FP, const char *formato, ...);`
- È possibile scrivere programmi utilizzabili sia con con file testo persistenti che con `stdin` e `stdout`
  - La logica di controllo della ripetizione può sfruttare in entrambi i casi `feof(FP)`, o la costante `EOF`
  - Nel caso di tastiera e video, l'uso di `feof(FP)` o `EOF` è un'alternativa alla richiesta all'utente di un carattere di scelta

File testo

Esempio. Il file testo "reali.txt" contiene solo numeri reali in notazione decimale, separati da spazio bianco. Visualizzare la media di tutti i reali del file.

```
#include <stdio.h>
#define NOME_FILE "reali.txt"
main() {
  FILE *FP;
  float X, Somma;
  int N;
  if ( (FP = fopen(NOME_FILE, "r")) == NULL )
    printf("Impossibile aprire %s\n", NOME_FILE);
  else {
    Somma = 0;
    N = 0;
    while  ( fscanf(FP, "%f", &X) != EOF ) {
      Somma += X;
      N++;
    }
    printf("%f\n", Somma/N);
    fclose(FP);
  }
}
```

File testo

- Versione con lettura da tastiera

```
#include <stdio.h>
main() {
  FILE *FP;
  float X, Somma;
  int N;
  FP = stdin;
  Somma = 0;
  N = 0;
  while ( fscanf(FP, "%f", &X) != EOF ) {
    Somma += X;
    N++;
  }
  printf("%f\n", Somma/N);
}
```