Lezione 9: I File

Laboratorio di Elementi di Architettura e Sistemi Operativi

9 Aprile 2013

File sequenziali

- Accesso tramite variabile di tipo *stream* (flusso di caratteri)
- Definiti in `stdio.h`
- **Definizione:** (fare attenzione al maiuscolo!)

  ```
  FILE *<identificatore>;
  ```

- Al momento dell'attivazione di un programma vengono automaticamente attivati (aperti) tre stream:

  `stdin` standard input (tastiera)
  `stdout` standard output (schermo)
  `stderr` standard error (schermo)

  sono direttamente utilizzabili nelle istruzioni per l'accesso ai file

Apertura di un file

Per accedere ad un file è necessario aprirlo:

- `FILE* fopen(char* nomefile, char* modo);`

  collega un *file fisico* (su disco) ad un *file logico* (stream)

- `nomefile`: nome del file fisico (eventualmente con percorso assoluto o relativo rispetto alla cartella di lancio del programma)
- `modo`: il tipo di accesso al file

`"r"` sola lettura
`"w"` sola scrittura (cancella il file se esiste)
`"a"` append (aggiunge in coda ad un file o crea file se non esiste)
`"r+"` lettura/scrittura su file esistente
`"w+"` lettura/scrittura (cancella il file se esiste o crea file se non esiste)
`"a+"` lettura/scrittura-in-coda (crea file se non esiste)

- ritorna il puntatore allo stream in caso di successo, altrimenti ritorna `NULL`

Chiusura di un file

Quando l'utilizzo di un file è terminato, è consigliabile chiuderlo:

- `int fclose(FILE* stream);`

  cancellazione della connessione di un file fisico con uno stream

- `stream`: uno stream aperto in precedenza con `fopen()`
- Valore di ritorno

  `0` se la chiusura avviene correttamente
  `EOF` in caso di errore

Riapertura di un file

Chiusura e riapertura di uno stream:

- `FILE *freopen(char* nomefile, char* modo, FILE* stream);`

  chiude uno stream e lo riapre assegnandolo ad un file diverso

- `stream`: uno stream aperto in precedenza con `fopen()`
- `nomefile`, `modo` e valore di ritorno come in `fopen()`
- equivalente a `fclose(stream); stream = fopen(nomefile, modo);`
- obbligatorio per riassegnare `stdin`, `stdout` e `stderr`, per i quali non si può usare `fopen`

Esempio di apertura e chiusura di un file

```
FILE *fp; /* variabile di tipo stream */
...

fp = fopen("testo.dat","r");
/* apro 'testo.dat' in lettura*/
if (fp == NULL)
printf("Errore nell'apertura del file.\n");
else {
  /* qui posso accedere a 'testo.dat' usando fp */
  ...

  if ( fclose(fp) != 0 ) {
    printf("Errore nella chiusura del file.\n");
  }
}
...
```

Lettura/scrittura a caratteri

```
int getc (FILE* stream);
int fgetc (FILE* stream);
```

- Legge un carattere alla volta dallo `stream`

- Restituisce il carattere letto o `EOF` in caso di fine file o errore
- *NOTA:* `getchar()` equivale a `getc(stdin)` e `fgetc(stdin)`

```
int putc (int c, FILE* stream);
int fputc (int c, FILE* stream);
```

- Scrive un carattere alla volta sullo `stream`
- Restituisce il carattere scritto o `EOF` in caso di errore
- *NOTA:* `putchar(c)` equivale a `putc(c, stdout)` e `fputc(c, stdout)`

Lettura/scrittura a righe

```
char* fgets(char* s, int n, FILE* stream);
```

- `n` è la lunghezza del vettore di caratteri `s`
- Legge una riga dallo `stream` fermandosi al più dopo `n-1` caratteri
- L'eventuale `'\n'` NON viene eliminato (diverso da `gets`!)
- Restituisce il puntatore alla stringa o `NULL` in caso di fine file o errore

```
int fputs(char* s, FILE* stream);
```

- Scrive la stringa `s` sullo `stream` senza aggiungere `'\n'` (diverso da `puts`!)
- Restituisce l'ultimo carattere scritto, oppure `EOF` in caso di errore

Lettura/scrittura formattata

```
int fscanf(FILE* stream, char *formato, ... );
```

- Come `scanf()`, con un parametro addizionale che rappresenta uno `stream`
- Restituisce il numero di campi convertiti, oppure `EOF` in caso di fine file

```
int fprintf(FILE* stream, char *formato, ... );
```

- Come `printf()`, con un parametro addizionale che rappresenta uno `stream`
- Restituisce il numero di byte scritti, oppure `EOF` in caso di errore