**Fgetc, getc, getchar()**

Si utilizza la libreria #include <stdio.h>

int fgetc(FILE \*stream);

int getc(FILE \*stream);

int getchar(void)

La funzione fgetc() legge un carattere da stream.

Ritorna il carattere letto oppure EOF se e' stato riscontrato un errore o la fine del file.

Il tipo ritornato e' un intero, poiche' la funzione, oltre a ritornare ogni carattere del tipo char, deve ritornare anche EOF (il cui valore tipico e' -1).

La funzione getc() equivale a fgetc() eccetto che potrebbe essere implementata come macro e valutare stream piu' di una volta.

La funzione getchar() legge un carattere da stdin.

Equivale a getc(stdin)

Le funzioni fgetc(), getc() e getchar() ritornano il carattere c letto come unsigned char con cast ad int oppure EOF in caso di fine file o di errore.

**fopen(), fdopen(), freopen() - apre uno stream**

FILE \*fopen(const char \*path, const char \*mode);

FILE \*fdopen(int fildes, const char \*mode);

FILE \*freopen(const char \*path, const char \*mode, FILE \*stream);

La funzione **fopen()** apre il file **path** associandolo ad uno stream.

Il file **path** viene predisposto per le operazioni di I/O come e' indicato da **mode**.

L'argomento **mode** puo' valere:

|  |  |
| --- | --- |
| **"r"** | Apre un file di testo per la lettura. Lo stream e' posizionato all'inizio del file. |
| **"r+"** | Apre un file di testo per la lettura e scrittura. Lo stream e' posizionato all'inizio del file. |
| **"w"** | Apre un file di testo per la scrittura. Il file viene creato se non esiste, altrimenti viene troncato (forza la lunghezza a 0). Lo stream e' posizionato all'inizio del file. |
| **"w+"** | Apre un file di testo per la lettura e scrittura. Il file viene creato se non esiste, altrimenti viene troncato (forza la lunghezza a 0). Lo stream e' posizionato all'inizio del file. |
| **"a"** | Apre un file di testo per la scrittura. Il file viene creato se non esiste. Lo stream e' posizionato alla fine file. |
| **"a+"** | Apre un file di testo per la lettura e scrittura. Il file viene creato se non esiste. Lo stream e' posizionato alla fine file. |

Inoltre **mode** puo' comprendere anche la lettera **b** come secondo o terzo carattere.  
In tal caso il file e' considerato binario.

Tra una operazione di lettura ed una di scrittura e viceversa, occorre chiamare la funzione [fflush()](http://digilander.libero.it/uzappi/C/librerie/funzioni/fflush.html) o [fseek()](http://digilander.libero.it/uzappi/C/librerie/funzioni/fseek.html).

La funzione **fdopen()** associa uno stream al file descriptor **filedes**. In tal caso **mode** dello stream deve essere compatibile con mode del file descriptor.  
In questa situazione, **mode** valorizzato con "w" o con "w+" non causa il troncamento del file.

Le funzioni **fopen()**, **fdopen()** e **freopen()**, ritornano in caso di successo, lo stream da impiegare nelle successive chiamate alle funzioni di I/O.  
In caso di fallimento viene ritornato NULL e la variabile **errno** contiene il codice di errore.

**fflush() - scarica uno stream**

int fflush(FILE \*stream);

La funzione **fflush()** forza la scrittura dei dati bufferizzati sullo **stream** tramite la funzione di basso livello **write()**.  
Se **stream** vale **NULL** la scrittura avviene per tutti i files aperti in output.

La funzione **fflush()** ritorna 0 in caso di successo. In caso di fallimento viene tornato EOF e la variabile globale **errno** e' settata propriamente per indicare l'errore che e' avvenuto.

**fputc(), putc(), putchar() - output di caratteri**

int fputc(int c, FILE \*stream);

int putc(int c, FILE \*stream);

int putchar(int c);

La funzione **fputc()** scrive il carattere **c** su **stream** eseguendo un cast a **unsigned char**.  
Ritorna il carattere scritto se la funzione ha successo, altrimenti **EOF** in caso di errore.

La funzione **putc()** equivale a **fputc()** eccetto che potrebbe essere implementata come macro e valutare **stream** piu' di una volta.

La funzione **putchar()** scrive il carattere **c** su **stdout**.  
E' equivalente a putc(c,stdout)

Le funzioni **fputc()**, **putc()**, e **putchar()** ritornano il carattere scritto o EOF in caso di errore.

**read() - legge da un file descriptor**

#include <unistd.h>

ssize\_t read(int fd, void \*buf, size\_t count);

La funzione **read()** legge da **fd** numero **count** bytes immagazzinandoli nel buffer puntato da **buf**.  
**buf** deve essere di dimensioni adeguate, cioe' almeno di **count** bytes.

La funzione **read()** ritorna il numero di elementi letti. In caso di raggiungimento della fine file viene restituito un numero minore di **count**. In caso di errore viene restituito -1.

**write() - scrive su un file descriptor**

#include <unistd.h>

ssize\_t write(int fd, const void \*buf, size\_t count);

La funzione **write()** scrive su **fd** numero **count** bytes di dati presenti nel buffer puntato da **buf**.

La funzione **write()** ritorna il numero di elementi scritti. In caso di errore viene restituito -1.

**close() - chiude un file descriptor**

#include <unistd.h>

int close(int fd);

La funzione **close()** chiude il file descriptor **fd**.

La funzione **close()** ritorna 0 in caso di successo e -1 in caso di errore.

**exit() - termina normalmente il programma**

#include <stdlib.h>

void exit(int status);

La funzione **exit()** effettua il termine normale del programma, ritornando il valore di **status** al parent.  
Quindi la funzione **exit()** esegue in sequenza le seguenti operazioni:

* Tutte le funzioni registrate con le funzioni [atexit()](http://digilander.libero.it/uzappi/C/librerie/funzioni/atexit.html) e [on\_exit()](http://digilander.libero.it/uzappi/C/librerie/funzioni/on_exit.html) sono chiamate nell'ordine inverso alla loro registrazione.
* Tutti gli stream aperti vengono scaricati tramite la funzione della libreria standard [fflush()](http://digilander.libero.it/uzappi/C/librerie/funzioni/fflush.html).
* Tutti gli stream aperti (compresi stdin, stdout e stderr) vengono chiusi impiegando le funzioni della libreria standard.
* Viene chiamata la funzione [\_exit()](http://digilander.libero.it/uzappi/C/librerie/funzioni/_exit.html).

La funzione **exit()** non ritorna. Termina l'esecuzione del processo.

**atoi()**

1. int atoi **(** const char \* str **)**;

questa funzione traduce un char in un int.

1) se gli passi un qualcosa che non e' un numero ti restituisce zero  
2) Se invece gli passi un numero troppo grosso , restiuisce il valore costante INT\_MAX   
3) se gli passi un numero troppo piccolo, restituisce INT\_MIN