



## Referencia Rápida - Librería Estándar C (2)

### E/S binaria- stdio.h

#### FILE\* fopen(char\* nombre, char\* modo)

Abre un archivo. Devuelve NULL si la apertura fué errónea.  
modos : r = lectura texto, w = escritura texto, a= adición texto  
rb = lectura binaria, wb = escritura binaria, ab = adición binaria

#### int fclose(FILE\* ptr)

Cierra un archivo. Devuelve 0 si todo fué correcto, EOF en caso de error

#### size\_t fread(void\* ptr, size\_t tam, size\_t num, FILE\* ptr)

Intenta leer *num* elementos de un archivo, donde cada elemento tiene un tamaño de *tam* bytes. Los datos quedan guardados en *ptr*. Devuelve el número de elementos leídos

#### size\_t fwrite(void\* ptr, size\_t tam, size\_t num, FILE\* ptr)

Intenta escribir *num* elementos de un archivo, donde cada elemento tiene un tamaño de *tam* bytes. Los datos quedan guardados en *ptr*. Devuelve el número de elementos escritos.

#### long int ftell(FILE\* ptr)

Devuelve la posición actual desde el inicio. Devuelve -1 en caso de error

#### int fseek(FILE\* ptr, long int posicion, int modo)

Establece el archivo en la posición indicada. Posibles valores de modo son:  
SEEK\_SET : desde el inicio del archivo, SEEK\_END : desde el final  
SEEK\_CUR : desde la posición actual

#### int feof(FILE\* ptr)

Devuelve un valor !=0 si el archivo está en EOF, 0 en caso contrario

#### int ferror(FILE\* ptr)

Devuelve el código de error actualmente asociado al archivo, 0 si no hay ninguno.

#### int fflush(FILE\* ptr)

Fuerza la escritura de los datos pendientes

#### void clearerr(FILE\* ptr)

Borra el indicador de error asociado al archivo.

### E/S texto - stdio.h

#### int fgetc(FILE\* ptr)

Lee un carácter. Devuelve EOF en caso de error o fin de archivo

#### int fputc(int ch, FILE\* ptr)

Escribe un carácter. Devuelve EOF en caso de error.

#### char\* fgets(char\* s, int max, FILE\* ptr)

Lee una línea (hasta \n inclusive) del archivo y la guarda en s. Lee un máximo de max-1 caracteres. Devuelve un puntero al resultado o NULL si error

#### int fputs(char\* s, FILE\* ptr)

Escribe una cadena en el archivo. Devuelve EOF si se produjo algún error

#### int fprintf(FILE\* ptr, char\* formato, ...)

#### int sprintf(char\* str, char\* formato ...)

Escriben una cadena con formato, bien en un archivo (fprintf), bien en una cadena resultante (sprintf). Devuelven el total de caracteres impresos, o -1 en caso de error

Caracteres de formato :

%d - signed int	%u - unsigned int	%x - unsigned int, hex
%X - unsigned int, HEX	%f - double	%c - char
%s - cadena	%p - puntero	%% - %
%ld - long int	%lld - long long int	%Ld - long double

#### int fscanf(FILE\* ptr, char\* formato, ...)

#### int sscanf(char\* str, char\* formato ...)

Idem que fprintf/ fscanf, pero leen datos con formato.

**stdin** Entrada estándar

**stdout** Salida estándar

**stderr** Flujo de error estándar

### Fechas y horas - time.h

#### struct tm {

int tm_sec;	Segundos (0..61)
int tm_min;	Minutos (0..59)
int tm_hour;	Horas (0..23)
int tm_mday;	Día del mes (1..31)
int tm_mon;	Mes (0..11)
int tm_year;	Años desde 1900 (01 = 1901)
int tm_wday;	Día de la semana (0 = Domingo, 6 = Sábado)
int tm_yday;	Día del año (0 .. 365)
int tm_isdst;	¿Horario de verano? >0:si, =0:no, <0:desconocido

#### char\* asctime(tm\* fecha)

Representación legible de la fecha

#### time\_t time(time\_t\* ptr)

Devuelve la fecha y hora actual. Además, la guarda en ptr (si ptr != NULL)

#### char\* ctime(time\_t\* ptr)

Igual que asctime, pero toma un time\_t\* como parámetro en lugar de tm\*

#### double difftime(time\_t t1, time\_t t2)

Devuelve el tiempo transcurrido entre t1 y t2 (t2-t1), expresado en seg.

#### tm\* gmtime(time\_t\* ptr)

Expande el tiempo almacenado en ptr, referenciando el resultado a GMT

#### tm\* localtime(time\_t\* ptr)

Expande el tiempo almacenado en ptr, referenciando el resultado a la zona local

#### time\_t mktime(tm\* ptr)

Empaqueta una fecha referenciada a la zona horaria local

#### size\_t strftime(char\* s, size\_t max, char\* fmt, tm\* ptr)

Almacena en s una representación de la fecha guardada en ptr, siguiendo el formato contenido en fmt. Hasta un máximo de max caracteres se guardan.

Caracteres de formato:

%a = nombre corto día.semana	%Y = año con cuatro dígitos
%A = nombre largo día.semana	%y = año con dos dígitos
%b = nombre corto del mes	%H = hora
%B = nombre largo del mes	%m = mes
%d = día del mes	%M = Minutos
%S = Segundos	%Z = Zona horaria local

#### clock\_t clock()

Impulsos de reloj desde el inicio del proceso. CLOCKS\_PER\_SEC indica el número de impulsos que hay por segundo

### Señales - signal.h

#### void (\*signal(int sig, void (\*func)(int)))(int);

Establece una función gestora para la señal sig. Devuelve SIG\_ERR en caso de error, o un puntero a la función gestora anterior si todo ha ido bien

#### int raise(int sig)

Hace que la señal sig sea generada. Devuelve 0 si todo fue bien.

**SIGABRT** Abortar proceso

**SIGCHLD** Proceso hijo finalizado

**SIGFPE** Error Punto Flotante

**SIGABRT** Abortar proceso

**SIGILL** Instrucción ilegal

**SIGKILL** Finalizar inmediatamente

**SIGSTOP** Suspender ejecución

**SIGALRM** Alarma de tiempo

**SIGCONT** Continuar

**SIGHUP** Cerrar Terminal

**SIGALRM** Alarma de tiempo

**SIGINT** Interrupción

**SIGQUIT** Finalizar + core dump

**SIGTERM** Solicitar finalización