Referencia Rápida - Librerías Estándar de C

Memoria -

void* malloc(size_t num_bytes)
Asigna una zona de memoria. Si no es posible, devuelve NULL

void* realloc(void* ptr, size_t num_bytes)
Modifica el tamaño de la zona de memoria a la que apunta ptr, haciendo que sea num_bytes. Devuelve un puntero a la nueva zona de memoria, o NULL si no se pudo redimensionar.

void free(void* ptr)
Libera una zona de memoria previamente asignada por calloc o malloc

void* calloc(size_t num, size_t tamaño)
Asigna una zona de memoria apropiada para *num* elementos, de *tamaño* bytes cada uno. Devuelve un puntero a la zona, o NULL si no hay memoria.

Cadenas - string.h

char* strcat(char* s1, char* s2)
char* strncat(char* s1, char *s2, size_t max)
Concatena s2 a s1. Devuelve un puntero a s1. No se permiten solapamientos

char* strcpy(char* s1, char* s2)
char* strncpy(char* s1, char *s2, size_t max)
Copia s2 sobre s1. Devuelve un puntero a s1. No se permiten solapamientos

int strcmp(char* s1, char* s2)
int strncmp(char* s1, char *s2, size_t max)
Devuelve >0 si s1 > s2, 0 si son iguales, <0 si s1 < s2.

int strlen(char* str)
Devuelve la longitud de una cadena. (En la longitud no se incluye el \0 final)

char* strchr(char* s, int c)
Devuelve un puntero a la primera posición de *c* en *s*, o NULL si no existe

char* strrchr(char* s, int c)
Devuelve un puntero a la última posición de *c* en *s*, o NULL si no existe

char* strstr(char* s1, char* s2)
Devuelve un puntero a la primera posición de *s2* en *s1*, o NULL si no existe

Varias - stdlib.h

int rand()
Devuelve un valor aleatorio entre 0 y RAND_MAX

void srand(unsigned int semilla)
Inicializa el generador de números aleatorios con la semilla especificada

void qsort(void* datos, size_t numElem, size_t longElem, int (*comp)(void*,void*))
Ordenación QuickSort. El puntero *datos* apunta al inicio de los mismos, *numElem* es el número de datos, *longElem* el tamaño en bytes de cada uno y *comp* - un puntero a una función de comparación

void* bsearch(void* k, void* datos, size_t numElem, size_t longElem, int (*comp)(void*,void*))
Realiza una búsqueda binaria de *k* dentro de una matriz de datos que comienza en *datos* La matriz debe estar ordenada de forma ascendente. Devuelve un puntero al dato encontrado o NULL si no se encuentra. En caso de duplicados cuál de ellos se devuelve es indeterminado.

char* getenv(char* nombre)
Devuelve el valor de una variable de entorno, o NULL si no existe.

int system(char* comando)
Ejecuta el comando especificado y devuelve su código de salida

Conversiones - stdlib.h

double atof(char* ptr)
int atoi(char* ptr)
long atol(char *ptr)
Convierten la cadena a la que apunta ptr a un número del tipo apropiado. La conversión se detiene en el primer carácter no válido. En caso de fallo, se devuelve 0.

double strtod(char* str, char fin)**
long strtol(char* str, char fin, int base)**
unsigned long strtoul(char* str, char* fin, int base)
Convierten la cadena a la que apunta ptr a un número del tipo apropiado. La conversión se detiene en el primer carácter no válido, y la posición de este carácter queda almacenada en el puntero *fin*. El parámetro *base* indica la base numérica (10, 16, 8, etc.) a utilizar para la conversión

Zonas de memoria - string.h

void* memchr(void* ptr, int c, size_t max)
Devuelve un puntero a la primera posición de *c* en *ptr*, o NULL si no existe

void* memcpy(void*s1,void* s2, size_t max)
Copia s2 encima de s1. Devuelve s1. No se permiten solapamientos

void* memmove(void*s1,void* s2, size_t max)
Copia s2 encima de s1. Devuelve s1. Se permiten solapamientos

void* memset(void*s1,int c, size_t max)
Rellena una zona de memroia con el byte c

Caracteres - ctype.h

la función...	...devuelve un valor != 0 (verdadero) para:
int isanum(int c)	alfanuméricos (A..Z,a..z,0..9)
int isalpha(int c)	letras (A..Z,a..z)
int iscntrl(int c)	caracteres de control (00 a 31 y 126)
int isdigit(int c)	dígitos (0..9)
int isgraph(int c)	car. visibles (todos menos 0 a 31 y espacio)
int islower(int c)	minúsculas (a..z)
int isprint(int c)	car. imprimibles (todos menos 0 a 31)
int isspace(int c)	blancos (espacio, tabulador, etc.)
int isupper(int c)	mayúsculas (A..Z)
int isxdigit(int c)	dígitos en hexadecimal (0..9 y A..F o a..f)
int toupper(int c)	Convierte c a mayúsculas
int tolower(int c)	Convierte c a minúsculas

Funciones Matemáticas - math.h

Salvo indicación contraria, todos los parámetros son double

double exp(x)	e^x	**double log(x)**	ln(x)
double pow(x,y)	x^y	**double sqrt(x)**	$\sqrt{x}$
double ceil(x)	redondeo dcha.	**double floor(x)**	redondeo izda.

sin(x) cos(x) tan(x) Funciones trigonométricas
asin(x) acos(x) atan(x) Funciones trigonométricas inversas
sinh(x) cosh(x) tanh(x) Funciones hiperbólicas

Programación Estructurada y Dinámica

www.profmatiasgarcia.com.ar

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 in	%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* c time(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 dia.semana	%Y = año con cuatro dígitos
%A = nombre largo dia.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	**SIGALRM**	Alarma de tiempo
SIGCHLD	Proceso hijo finalizado	**SIGCONT**	Continuar
SIGFPE	Error Punto Flotante	**SIGHUP**	Cerrar Terminal
SIGABRT	Abortar proceso	**SIGALRM**	Alarma de tiempo
SIGILL	Instrucción ilegal	**SIGINT**	Interrupción
SIGKILL	Finalizar inmediatamente	**SIGQUIT**	Finalizar + core dump
SIGSTOP	Suspender ejecución	**SIGTERM**	Solicitar finalización