**LIST OF MOST USED HEADER FILES IN C PROGRAMMING LANGUAGE:**

* Check the below table to know all the C library functions and header files in which they are declared.
* Click on the each header file name below to know the list of inbuilt functions declared inside them.

|  |  |
| --- | --- |
| **Header file** | **Description** |
| [stdio.h](http://fresh2refresh.com/c/c-function/stdio-h-library-functions/) | This is standard input/output header file in which Input/Output functions are declared |
| [conio.h](http://fresh2refresh.com/c/c-function/conio-h-library-functions/) | This is console input/output header file |
| [string.h](http://fresh2refresh.com/c/c-function/string-h-library-functions/) | All string related functions are defined in this header file |
| [stdlib.h](http://fresh2refresh.com/c/c-function/c-stdlib-h-library-functions/) | This header file contains general functions used in C programs |
| [math.h](http://fresh2refresh.com/c/c-function/c-math-h-library-functions/) | All maths related functions are defined in this header file |
| [time.h](http://fresh2refresh.com/c/c-function/c-time-h-library-functions/) | This header file contains time and clock related functions |
| [ctype.h](http://fresh2refresh.com/c/c-function/c-ctype-h-library-functions/) | All character handling functions are defined in this header file |
| [stdarg.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | Variable argument functions are declared in this header file |
| [signal.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | Signal handling functions are declared in this file |
| [setjmp.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | This file contains all jump functions |
| [locale.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | This file contains locale functions |
| [errno.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | Error handling functions are given in this file |
| [assert.h](http://fresh2refresh.com/c/c-function/c-all-other-library-functions/) | This contains diagnostics functions |

### <ctype.h>

int isalnum(int c);

isalpha(c) or isdigit(c)

int isalpha(int c);

isupper(c) or islower(c)

int iscntrl(int c);

is control character

int isdigit(int c);

is decimal digit

int isgraph(int c);

is printing character other than space

int islower(int c);

is lower-case letter

int isprint(int c);

is printing character (including space)

int ispunct(int c);

is printing character other than space, letter, digit

int isspace(int c);

is space, formfeed, newline, carriage return, tab, vertical tab

int isupper(int c);

is upper-case letter

int isxdigit(int c);

is hexadecimal digit

int tolower(int c);

return lower-case equivalent

int toupper(int c);

return upper-case equivalent

Notes:

* In ASCII (7-bit), printing characters are 0x20 (' ') to 0x7E ('~'); control characters are 0x00 (NUL) to 0x1F (US) and 0x7F (DEL)

### <errno.h>

extern int errno;

An error code value set by some functions. It is generally the responsibility of the programmer to clear errno before calling such a function.

### <math.h>

double sin(double x);

double cos(double x);

double tan(double x);

double asin(double x);

double acos(double x);

double atan(double x);

double atan2(double y, double x);

double sinh(double x);

double cosh(double x);

double tanh(double x);

double exp(double x);

double log(double x);

double log10(double x);

double pow(double x, double y);

x raised to power y

double sqrt(double x);

double ceil(double x);

smallest integer not less than x

double floor(double x);

largest integer not greater than x

double fabs(double x);

double ldexp(double x, int n);

double frexp(double x, int\* exp);

double modf(double x, double\* ip);

double fmod(double x, double y);

### <

### <string.h>

char\* strcpy(char\* s, const char\* ct);

Copy ct to s including terminating NUL. Return s.

char\* strncpy(char\* s, const char\* ct, int n);

Copy at most n characters of ct to s Pad with NULs if ct is of length less than n. Return s.

char\* strcat(char\* s, const char\* ct);

Concatenate ct to s. Return s.

char\* strncat(char\* s, const char\* ct, int n);

Concatenate at most n characters of ct to s. Terminate s with NUL and return it.

int strcmp(const char\* cs, const char\* ct);

Compare cs and ct. Return negative if cs < ct, zero if cs == ct, positive if cs > ct.

int strncmp(const char\* cs, const char\* ct, int n);

Compare at most n characters of cs and ct. Return negative if cs < ct, zero if cs == ct, positive if cs > ct.

char\* strchr(const char\* cs, int c);

Return pointer to first occurrence of c in cs, or NULL if not found.

char\* strrchr(const char\* cs, int c);

Return pointer to last occurrence of c in cs, or NULL if not found.

size\_t strspn(const char\* cs, const char\* ct);

Return length of prefix of cs consisting entirely of characters in ct.

size\_t strcspn(const char\* cs, const char\* ct);

Return length of prefix of cs consisting entirely of characters not in ct.

char\* strpbrk(const char\* cs, const char\* ct);

Return pointer to first occurrence within cs of any character of ct, or NULL if not found.

char\* strstr(const char\* cs, const char\* ct);

Return pointer to first occurrence of ct in cs, or NULL if not found.

size\_t strlen(const char\* cs);

Return length of cs.

char\* strerror(int n);

Return pointer to implementation-defined string corresponding with error n.

char\* strtok(char\* s, const char\* t);

A sequence of calls to strtok returns tokens from s delimted by a character in ct. Non-NULL s indicates the first call in a sequence. ct may differ on each call. Returns NULL when no such token found.

void\* memcpy(void\* s, const void\* ct, int n);

Copy n characters from ct to s. Return s. **Does not work correctly if objects overlap.**

void\* memmove(void\* s, const void\* ct, int n);

Copy n characters from ct to s. Return s. Works correctly even if objects overlap.

int memcmp(const void\* cs, const void\* ct, int n);

Compare first n characters of cs with ct. Return negative if cs < ct, zero if cs == ct, positive if cs > ct.

void\* strchr(const char\* cs, int c, int n);

Return pointer to first occurrence of c in first n characters of cs, or NULL if not found.

void\* strchr(char\* s, int c, int n);

Replace each of the first n characters of s by c. Return s.

### <time.h>

clock\_t

An arithmetic type representing time.

CLOCKS\_PER\_SEC

The number of clock\_t units per second.

time\_t

An arithmetic type representing time.

struct tm

Represents the components of calendar time:

int tm\_sec;

seconds after the minute

int tm\_min;

minutes after the hour

int tm\_hour;

hours since midnight

int tm\_mday;

day of the month

int tm\_ymon;

months **since** January

int tm\_year;

years since 1900

int tm\_day;

days since Sunday

int tm\_yday;

days since January 1

int tm\_isdst;

Daylight Saving Time flag : is positive if DST is in effect, zero if not in effect, negative if information unavailable.

clock\_t clock();

Returns processor time used by program or -1 if not available.

time\_t time(time\_t\* tp);

Returns current calendar time or -1 if not available. If tp is non-NULL, return value is also assigned to \*tp.

double difftime(time\_t time2, time\_t time1);

Returns the difference is seconds between time2 and time1.

time\_t mktime(struct tm\* tp);

Returns the local time corresponding to \*tp, or -1 if it cannot be represented.

char\* asctime(const struct tm\* tp);

Returns the given time as a string of the form:  
Sun Jan 3 14:14:13 1988\n\0

char\* ctime(const time\_t tp);

Converts the given calendar time to a local time and returns the equivalent string. Equivalent to:  
[asctime](https://www.csse.uwa.edu.au/programming/ansic-library.html#asctime)([localtime](https://www.csse.uwa.edu.au/programming/ansic-library.html" \l "localtime)(tp))

struct tm\* gmtime(const time\_t tp);

Returns the given calendar time converted into Coordinated Universal Time, or NULL if not available.

struct tm\* localtime(const time\_t tp);

Returns calendar time \*tp converted into local time.

size\_t strftime(char\* s, size\_t smax, const char\* fmt, const struct tm\* tp);

Formats \*tp into s according to fmt.

Notes:

* Local time may differ from calendar time, for example because of time zone.