H

header — Overview

The Standard mandates that every function be declared in a *header*, whose contents are available to the program through the **#include** preprocessor directive. A header usually is a file, but it may also be built into the translator.

The Standard describes 15 headers, as follows:

assert.h	Run-time assertion checking
ctype.h	Character-handling functions
errno.h	**errno** and related macros
float.h	Limits to floating-point numbers
limits.h	General implementation limits
locale.h	Establish or modify a locale
math.h	Mathematics function
setjmp.h	Non-local jumps
signal.h	Signal-handling functions
stdarg.h	Handle variable numbers of arguments
stddef.h	Common definitions
stdio.h	Standard input and output
stdlib.h	General utilities
string.h	String-handling functions
time.h	Date and time functions

Each header contains only those functions described within the Standard, plus attending data types and macros. Every external identifier in every header is reserved for the implementation. Also reserved is every external identifier that begins with an underscore character '_', whether it is described in the Standard or not. If a reserved external name is redefined, behavior is undefined, even if the function that replaces it has the same specification as the original. This is done to assure the user that moving code from one implementation to another will not generate unforeseen collisions with implementation-defined identifiers. It is also done to assure the implementor that functions called by other library functions will not be derailed by user-defined external names.

Every header can be included any number of times, and any number of headers can be included in any order without triggering problems.

Let's C also includes the following, implementation-specific headers:

access.h	Define manifest constants used by **access()**
bios.h	Outline ROM BIOS data area
canon.h	Canonical conversion for the 68000
dos.h	Define MS-DOS functions and devices
larges.h	Support model-independent assembly language
mtype.h	List processor code numbers
path.h	Declare **path()**
stat.h	Definitions and declarations to obtain file status
xctype.h	Declare/define extended character handling routines
xmath.h	Declare extended mathematics functions
xstdio.h	Declare/define extended STDIO routines
xtime.h	Declare/define extended date and time routines

Cross-references

Standard, §4.1.3
The C Programming Language, ed. 2, p. 241

See Also

header names, Library

header names — Definition

A *header name* is a token that gives the name of a header. There are two varieties of header name: **<***filename***.h>** and "*filename***.h"**.

The two varieties of header names are both searched in an implementation-defined manner. The name of the file can be enclosed within angle brackets (**<***file***.h>**) or quotation marks ("*file***.h"**). Angle brackets tell **Let's C** to look for *file***.h** in the directories named with the **-I** options to the **cc** command, and then in the directory named by the environmental variable **INCDIR**. Quotation marks tell **Let's C** to look for *file***.h** in the source file's directory, then in directories named with the **-I** options, and then in the directory named by the environmental variable **INCDIR**.

If any of the characters **'**, ****, **,** or **/*** appear between the '<' and '>' of a bracketed header name, behavior is undefined. Likewise, if any of the characters **'**, ****, or **/*** appear between the "" and the '"' of a quoted header name, behavior is undefined.

Cross-references

Standard, §3.1.7

See Also

#include, header, lexical elements

hypot() — Extended function (libm)

Compute hypotenuse of right triangle
#include <xmath.h>
double hypot(double *x***, double** *y***);**

hypot computes the hypotenuse, or distance from the origin, of its arguments *x* and *y*. The result is the square root of the sum of the squares of *x* and *y*.

See Also

cabs, extended mathematics

Notes

hypot is not described in the ANSI Standard. Any program that uses it does not conform strictly to the Standard, and may not be portable to other compilers or environments.