1 接口说明

1.1 assert.h

1.1.1 宏

assert()是一个诊断宏，用于动态辨识程序的逻辑错误条件。其原型是： void assert(int expression);

如果宏的参数求值结果为非零值，则不做任何操作（no action）；如果是零值，用[宽字符](https://zh.wikipedia.org/wiki/%E5%AE%BD%E5%AD%97%E7%AC%A6)打印诊断消息，然后调用abort()。诊断消息包括：

* 源文件名字（在stdlib.h中声明的宏\_\_FILE\_\_的值）
* 所在的源文件的行号（在stdlib.h中声明的宏\_\_LINE\_\_的值）
* 所在的函数名（在stdlib.h中声明的宏\_\_func\_\_的值）
* 求值结果为0的表达式

诊断信息的显示目标依赖于被调用程序的类型。如果是控制台程序，诊断信息显示在[stderr](https://zh.wikipedia.org/wiki/Stderr)设备；如果是基于窗口的程序，assert()产生一个Windows MessageBox来显示诊断信息。

程序可以屏蔽掉所有的assert()而无需修改源代码。这只需要在命令行调用C语言的编译器时添加宏定义的命令行选项，定义NDEBUG宏;也可以在源程序程序引入<assert.h>之前就使用#define NDEBUG来定义宏。被屏蔽的assert()甚至不对传递给它的参数表达式求值，因此使用assert()时其参数表达式不能有副作用（side-effects）。

1.2 complex.h

complex.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，提供了[复数](https://zh.wikipedia.org/wiki/%E5%A4%8D%E6%95%B0_(%E6%95%B0%E5%AD%A6))算术所需要的宏定义与函数声明。

1.2.1 类型

[C99](https://zh.wikipedia.org/wiki/C99)及[C++11](https://zh.wikipedia.org/wiki/C++11)都规定了[关键字](https://zh.wikipedia.org/wiki/%E9%97%9C%E9%8D%B5%E5%AD%97)\_Complex。因而有3种复数类型：

* double \_Complex
* float \_Complex
* long double \_Complex

次序不是必须遵守的，比如float \_Complex也可以写成\_Complex float。\_Complex\_I扩展为类型为const float \_Complex的常量值，其值为虚数单位。 [C99](https://zh.wikipedia.org/wiki/C99)规定complex作为宏扩展为\_Complex。但C++未定义complex宏。 [gcc](https://zh.wikipedia.org/wiki/Gcc)仅支持complex type，不支持imaginary type。因此宏I扩展为\_Complex\_I。 定义复数变量时，可以如此：

double \_Complex v1=3.1+5\*I;

double \_Complex v2=3.2+5\*\_Complex\_I;

float \_Complex v3=4.6f+12.0IF; //必须先写i，后写f

\_Complex float v4=8.2f+2.3if;

long double \_Complex v5=1.23456789L+3.456789999iL; //必须先写i，后写L

也可以用CMPLX/CMPLXF/CMPLXL宏来表示相应复数表达式。

1.3 函数

[复三角函数](https://zh.wikipedia.org/wiki/%E4%B8%89%E8%A7%92%E5%87%BD%E6%95%B8)

[反余弦](https://zh.wikipedia.org/wiki/%E5%8F%8D%E4%BD%99%E5%BC%A6)

* cacos 双精度版本
* cacosf 单精度版本
* cacosl 长双精度版本

[反正弦](https://zh.wikipedia.org/wiki/%E5%8F%8D%E6%AD%A3%E5%BC%A6)

* casin 双精度版本
* casinf 单精度版本
* casinl 长双精度版本

[反正切](https://zh.wikipedia.org/wiki/%E5%8F%8D%E6%AD%A3%E5%88%87)

* catan 双精度版本
* catanf 单精度版本
* catanl 长双精度版本

[余弦](https://zh.wikipedia.org/wiki/%E4%BD%99%E5%BC%A6)

* ccos 双精度版本
* ccosf 单精度版本
* ccosl 长双精度版本

[正弦](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%BC%A6)

* csin 双精度版本
* csinf 单精度版本
* csinl 长双精度版本

[正切](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%87)

* ctan 双精度版本
* ctanf 单精度版本
* ctanl 长双精度版本

[复双曲函数](https://zh.wikipedia.org/wiki/%E5%8F%8C%E6%9B%B2%E5%87%BD%E6%95%B0)

反双曲余弦

* cacosh 双精度版本
* cacoshf 单精度版本
* cacoshl 长双精度版本

反双曲正弦

* casinh 双精度版本
* casinhf 单精度版本
* casinhl 长双精度版本

反双曲正切

* catanh 双精度版本
* catanhf 单精度版本
* catanhl 长双精度版本

双曲余弦

* ccosh 双精度版本
* ccoshf 单精度版本
* ccoshl 长双精度版本

双曲正弦

* csinh 双精度版本
* csinhf 单精度版本
* csinhl 长双精度版本

双曲正切

* ctanh 双精度版本
* ctanhf 单精度版本
* ctanhl 长双精度版本

[指数](https://zh.wikipedia.org/wiki/%E6%8C%87%E6%95%B8%E5%87%BD%E6%95%B8)与[对数](https://zh.wikipedia.org/wiki/%E5%AF%B9%E6%95%B0)函数

指数

* cexp 双精度版本
* cexpf 单精度版本
* cexpl 长双精度版本

[自然对数](https://zh.wikipedia.org/wiki/%E8%87%AA%E7%84%B6%E5%AF%B9%E6%95%B0)

* clog 双精度版本
* clogf 单精度版本
* clogl 长双精度版本

[幂](https://zh.wikipedia.org/wiki/%E5%B9%82)运算和[绝对值](https://zh.wikipedia.org/wiki/%E7%BB%9D%E5%AF%B9%E5%80%BC)

绝对值

* cabs 双精度版本
* cabsf 单精度版本
* cabsl 长双精度版本

幂运算

* cpow 双精度版本
* cpowf 单精度版本
* cpowl 长双精度版本

[平方根](https://zh.wikipedia.org/wiki/%E5%B9%B3%E6%96%B9%E6%A0%B9)

* csqrt 双精度版本
* csqrtf 单精度版本
* csqrtl 长双精度版本

操作

[相角](https://zh.wikipedia.org/wiki/%E7%9B%B8%E4%BD%8D)

* carg 双精度版本
* cargf 单精度版本
* cargl 长双精度版本

[虚部](https://zh.wikipedia.org/wiki/%E8%A4%87%E6%95%B8_(%E6%95%B8%E5%AD%B8))

* cimag 双精度版本
* cimagf 单精度版本
* cimagl 长双精度版本

[复共轭](https://zh.wikipedia.org/wiki/%E5%A4%8D%E5%85%B1%E8%BD%AD)

* cong 双精度版本
* congf 单精度版本
* congl 长双精度版本

[黎曼球面](https://zh.wikipedia.org/wiki/%E9%BB%8E%E6%9B%BC%E7%90%83%E9%9D%A2)[投影](https://zh.wikipedia.org/wiki/%E6%8A%95%E5%BD%B1)

* cproj 双精度版本
* cprojf 单精度版本
* cprojl 长双精度版本

[实部](https://zh.wikipedia.org/wiki/%E8%A4%87%E6%95%B8_(%E6%95%B8%E5%AD%B8))

* creal 双精度版本
* crealf 单精度版本
* creall 长双精度版本

1.4 ctype.h

ctype.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，定义了一批[C语言](https://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)字符分类函数（C character classification functions），用于测试字符是否属于特定的字符类别，如字母字符、控制字符等等。既支持单字节字符，也支持[宽字符](https://zh.wikipedia.org/wiki/%E5%AE%BD%E5%AD%97%E7%AC%A6)。

1.4.1 函数

字节字符处理函数在ctype.h(C++的cctype )中声明。宽字节字符处理函数在wctype.h(C++的cwctype)中声明.

|  |  |  |
| --- | --- | --- |
| 单字节 | 宽字节 | 描述 |
| [isalnum](http://en.cppreference.com/w/c/string/byte/isalnum) | iswalnum | 是否为字母数字 |
| [isalpha](http://en.cppreference.com/w/c/string/byte/isalpha) | [iswalpha](http://en.cppreference.com/w/c/string/wide/iswalpha) | 是否为字母 |
| [islower](http://en.cppreference.com/w/c/string/byte/islower) | [iswlower](http://en.cppreference.com/w/c/string/wide/iswlower) | 是否为小写字母 |
| [isupper](http://en.cppreference.com/w/c/string/byte/isupper) | [iswupper](http://en.cppreference.com/w/c/string/wide/iswupper) | 是否为大写字母 |
| [isdigit](http://en.cppreference.com/w/c/string/byte/isdigit) | [iswdigit](http://en.cppreference.com/w/c/string/wide/iswdigit) | 是否为数字 |
| [isxdigit](http://en.cppreference.com/w/c/string/byte/isxdigit) | [iswxdigit](http://en.cppreference.com/w/c/string/wide/iswxdigit) | 是否为16进制数字 |
| [iscntrl](http://en.cppreference.com/w/c/string/byte/iscntrl) | [iswcntrl](http://en.cppreference.com/w/c/string/wide/iswcntrl) | 是否为控制字符 |
| [isgraph](http://en.cppreference.com/w/c/string/byte/isgraph) | [iswgraph](http://en.cppreference.com/w/c/string/wide/iswgraph) | 是否为图形字符（例如，空格、控制字符都不是） |
| [isspace](http://en.cppreference.com/w/c/string/byte/isspace) | [iswspace](http://en.cppreference.com/w/c/string/wide/iswspace) | [是否为空格字符（包括制表符、回车符、换行符等）](https://zh.wikipedia.org/w/index.php?title=%E7%A9%BA%E6%A0%BC%E5%AD%97%E7%AC%A6&action=edit&redlink=1) |
| [isblank](http://en.cppreference.com/w/c/string/byte/isblank) | [iswblank](http://en.cppreference.com/w/c/string/wide/iswblank) | 是否为空白字符 (C99/C++11新增)（包括水平制表符） |
| [isprint](http://en.cppreference.com/w/c/string/byte/isprint) | [iswprint](http://en.cppreference.com/w/c/string/wide/iswprint) | 是否为可打印字符 |
| [ispunct](http://en.cppreference.com/w/c/string/byte/ispunct) | [iswpunct](http://en.cppreference.com/w/c/string/wide/iswpunct) | 是否为标点 |
| [tolower](http://en.cppreference.com/w/c/string/byte/tolower) | [towlower](http://en.cppreference.com/w/c/string/wide/towlower) | 转换为小写 |
| [toupper](http://en.cppreference.com/w/c/string/byte/toupper) | [towupper](http://en.cppreference.com/w/c/string/wide/towupper) | 转换为大写 |
| 不适用 | [iswctype](http://en.cppreference.com/w/c/string/wide/iswctype) | 检查一个wchar\_t是否是属于指定的分类 |
| 不适用 | [towctrans](http://en.cppreference.com/w/c/string/wide/towctrans) | 使用指定的变换映射来转换一个wchar\_t（实际上是大小写的转换） |
| 不适用 | [wctype](http://en.cppreference.com/w/c/string/wide/wctype) | 返回一个宽字符的类别，用于iswctype函数 |
| 不适用 | [wctrans](http://en.cppreference.com/w/c/string/wide/wctrans) | 返回一个变换映射，用于 towctrans |

1.5 errno.h

errno.h是[C语言](https://zh.wikipedia.org/wiki/C%E8%AA%9E%E8%A8%80)[C标准库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E5%BC%8F%E5%BA%AB)里的头文件，定义了透过[错误码](https://zh.wikipedia.org/w/index.php?title=%E9%8C%AF%E8%AA%A4%E7%A2%BC&action=edit&redlink=1)来回报错误信息的宏：

errno宏定义为一个int类型的[左值](https://zh.wikipedia.org/wiki/%E5%B7%A6%E5%80%BC), 包含任何函数使用errno功能所产生的上一个错误码。

一些表示错误码，定义为整数值的宏：

* EDOM源自函数的参数超出范围,例如sqrt(-1)
* ERANGE源自函数的结果超出范围,例如strtol("0xfffffffff",NULL,0)
* EILSEQ源自不合法的字符顺序,例如wcstombs(str, L"\xffff", 2)

1.6 fenv.h

fenv.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，提供了对浮点环境的支持。浮点环境（floating-point environment）是指任何实现支持的浮点状态标志与控制模式。浮点状态标志（floating-point status flag）是一个系统变量，当浮点异常被抛出时该系统变量被置位。浮点控制模式（floating-point control mode）是一个系统变量，用户可以对其置位，以影响随后的浮点运算行为。

1.6.1 类型

fenv\_t 表示整体浮点环境

fexcept\_t 表示浮点状态的集合

1.6.2 宏

表示各种浮点运算异常：

* FE\_DIVBYZERO
* FE\_INEXACT
* FE\_INVALID
* FE\_OVERFLOW
* FE\_UNDERFLOW
* FE\_ALL\_EXCEPT

近似舍入方法

* FE\_DOWNWARD
* FE\_TONEAREST
* FE\_TOWARDZERO
* FE\_UPWARD

缺省浮点环境

* FE\_DFL\_ENV

1.6.3 函数

feclearexcept 清除给定的浮点异常

fegetexceptflag 保存一个浮点状态标志到对象中

feraiseexcept 抛出一个浮点异常

fesetexceptflag 设置一个浮点状态标志到对象中

fetestexcept 测试那些浮点异常标志被置位

fegetround 获取当前近似舍入方法

fesetround 设置近似舍入方法

fegetenv 存储当前浮点环境到对象中

feholdexcept 存储当前浮点环境到对象中，清楚浮点状态标志，安装non-stop模式

fesetenv 从对象存储的浮点环境恢复现场

feupdateenv 保存当前浮点异常，从对象存储的浮点环境恢复现场，再重新抛出当前浮点异常

1.7 float.h

1.7.1 宏

双精度浮点类型

* DBL\_DIG 十进制的精度位数：15
* DBL\_EPSILON 保持运算的最小值：2.2204460492503131e-016
* DBL\_MANT\_DIG 尾数的位数：53
* DBL\_MAX 最大值：1.7976931348623158e+308
* DBL\_MAX\_10\_EXP 10进制最大指数值：308
* DBL\_MAX\_EXP 2进制最大指数值：1024 （即可以表示到21024这个数量级的值）
* DBL\_MIN 最小的正值：2.2250738585072014e-308
* DBL\_MIN\_10\_EXP10进制最小指数值： (-307)
* DBL\_MIN\_EXP 2进制最小指数值： (-1021)
* \_DBL\_RADIX 指数的进制基数： 2
* \_DBL\_ROUNDS 额外的舍入方法： 1

单精度浮点类型

* FLT\_DIG 10进制的精度位数 6
* FLT\_EPSILON 保持加法运算的最小值 1.192092896e-07F
* FLT\_GUARD 0
* FLT\_MANT\_DIG 尾数的位数： 24
* FLT\_MAX 最大值：3.402823466e+38F
* FLT\_MAX\_10\_EXP 十进制的最大指数值： 38
* FLT\_MAX\_EXP 二进制的最大指数值： 128
* FLT\_MIN 最小正值：1.175494351e-38F
* FLT\_MIN\_10\_EXP 最小10进制指数值： (-37)
* FLT\_MIN\_EXP 最小二进制指数值： (-125)
* FLT\_NORMALIZE 0
* FLT\_RADIX 指数的进制基数： 2
* FLT\_ROUNDS 额外的舍入方法： 1

长双精度浮点类型

均规定为双精度浮点类型的极限值：

* LDBL\_DIG 即DBL\_DIG
* LDBL\_EPSILON 即DBL\_EPSILON
* LDBL\_MANT\_DIG 即DBL\_MANT\_DIG
* LDBL\_MAX 即DBL\_MAX
* LDBL\_MAX\_10\_EXP 即DBL\_MAX\_10\_EXP
* LDBL\_MAX\_EXP 即DBL\_MAX\_EXP
* LDBL\_MIN 即DBL\_MIN
* LDBL\_MIN\_10\_EXP 即DBL\_MIN\_10\_EXP
* LDBL\_MIN\_EXP 即DBL\_MIN\_EXP
* \_LDBL\_RADIX 即DBL\_RADIX
* \_LDBL\_ROUNDS 即DBL\_ROUNDS

1.8 inttypes.h

inttypes.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，提供了各种位宽的整数类型输入输出时的转换标志宏。

1.8.1 宏

下述定义的宏用于stdint.h中定义的各种位宽的整形在格式化输入/输出时的格式标志。

前3个字符：

* PRI 用于printf format
* SCN 用于scanf format

第4个字符

* x 用于hexadecimal formatting
* u 用于unsigned formatting
* 用于octal formatting
* i 用于integer formatting
* d 用于decimal formatting

其他字符

* 8 用于eight bit
* 16 用于sixteen bit
* 32 用于thirty-two bit
* 64 用于sixty-four bit
* FAST8 用于"fast" eight bit
* FAST16 用于"fast" sixteen bit
* FAST32 用于"fast" thirty-two bit
* FAST64 用于"fast" sixty-four bit
* LEAST8 用于"least" eight bit
* LEAST16 用于"least" sixteen bit
* LEAST32 用于"least" thirty-two bit
* LEAST64 用于"least" sixty-four bit
* PTR 用于指针
* MAX 用于maximum supported bit size

1.8.2 类型

imaxdiv\_t 结构化类型，用于保存函数imaxdiv返回的除商与余数。

1.8.3 函数

imaxabs 计算绝对值

imaxdiv 计算商与余数

strtoimax 字符串转换为整数

strtoumax 字符串转换为无符号整数

wstrtoimax 宽字符串转换为整数

wstrtoumax 宽字符串转换为无符号整数

1.9 limits.h

limits.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，定义了整数类型的一些极限值。

1.9.1 宏

以下常量以32位[电脑](https://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6)的常见数值为例，根据不同[硬件](https://zh.wikipedia.org/wiki/%E7%A1%AC%E9%AB%94)、[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)、[编译器](https://zh.wikipedia.org/wiki/%E7%B7%A8%E8%AD%AF%E5%99%A8)可能会有不同的常量值定义。

CHAR\_BIT 字节的最小位数：8

SCHAR\_MIN 有符号字符类型的最小值：-128

SCHAR\_MAX 有符号字符类型的最大值：+127

UCHAR\_MAX 无符号字符类型的最大值：255

CHAR\_MIN 字符类型的最小值

CHAR\_MAX 字符类型的最大值

MB\_LEN\_MAX 多字节字符在任何locale中可能的最长字节数：4/5/8/16

SHRT\_MIN 短整型最小值：-32768，即- 215

SHRT\_MAX 短整型最大值：+32767，即 215 - 1

USHRT\_MAX 无符号短整型最大值：65535 ，即 216 - 1

INT\_MIN 整型最小值：-2147483648，即 -(231)

INT\_MAX 整型最大值：+2147483647 ，即231 - 1

UINT\_MAX 无符号整型最大值：4294967295，即232 - 1

LONG\_MIN 长整型最小值：-2147483648 ，即-(231 )

LONG\_MAX 长整型最大值：+2147483647 ，即231 - 1

ULONG\_MAX 无符号长整型最大值：4294967295 ，即232 - 1

LLONG\_MIN 长长整型最小值：-9223372036854775808 ，即-(263 )

LLONG\_MAX 长长整型最大值：+9223372036854775807 ，即263 - 1

ULLONG\_MAX 无符号长长整型最大值：18446744073709551615 ，即264- 1

1.10 Locale.h

locale.h是[C程序设计语言](https://zh.wikipedia.org/wiki/C%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E8%AF%AD%E8%A8%80)[标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)的一个[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，声明了C语言[本地化](https://zh.wikipedia.org/wiki/%E6%9C%AC%E5%9C%B0%E5%8C%96)函数。这些函数用于在处理多种自然语言的软件编程设计时，把程序调整到特定的[区域设置](https://zh.wikipedia.org/wiki/%E5%8C%BA%E5%9F%9F%E8%AE%BE%E7%BD%AE). 这些区域设置影响到C语言标准库的[输入/输出函数](https://zh.wikipedia.org/wiki/Stdio.h).

1.10.1 函数

|  |  |
| --- | --- |
| Function | Description |
| [setlocale](http://en.cppreference.com/w/c/locale/setlocale) | 设置与读取当前C locale |
| [localeconv](http://en.cppreference.com/w/c/locale/localeconv) | 返回当前locale的货币与数值的格式细节 |

1.11 math.h

math.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件(标头档)](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)。提供用于常用高级数学运算的运算函数。

1.11.1 函数

|  |  |
| --- | --- |
| [函数原型](https://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B0%E5%8E%9F%E5%9E%8B) | 描述 |
| double sin(double); | [正弦](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%BC%A6) |
| double cos(double); | [余弦](https://zh.wikipedia.org/wiki/%E4%BD%99%E5%BC%A6) |
| double tan(double); | [正切](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%87) |
| double asin(double); | [反正弦，结果介于 [ − π 2 , π 2 ]](https://zh.wikipedia.org/wiki/%E5%8F%8D%E6%AD%A3%E5%BC%A6) |
| double acos(double) | [反余弦，结果介于 [ 0 , π ]](https://zh.wikipedia.org/wiki/%E5%8F%8D%E9%A4%98%E5%BC%A6) |
| double atan(double); | [反正切(主值)，结果介于 [ − π 2 , π 2 ]](https://zh.wikipedia.org/wiki/%E5%8F%8D%E6%AD%A3%E5%88%87) |
| double atan2(double, double); | [反正切(整圆值)，结果介于 [ − π , π ]](https://zh.wikipedia.org/wiki/Atan2) |
| double sinh(double); | [双曲正弦](https://zh.wikipedia.org/wiki/%E5%8F%8C%E6%9B%B2%E6%AD%A3%E5%BC%A6) |
| double cosh(double); | [双曲余弦](https://zh.wikipedia.org/wiki/%E5%8F%8C%E6%9B%B2%E4%BD%99%E5%BC%A6) |
| double tanh(double); | [双曲正切](https://zh.wikipedia.org/wiki/%E5%8F%8C%E6%9B%B2%E6%AD%A3%E5%88%87) |
| double exp(double); | [指数函数](https://zh.wikipedia.org/wiki/%E6%8C%87%E6%95%B8%E5%87%BD%E6%95%B8) |
| double sqrt(double); | 开平方根 |
| double log(double); | [自然对数](https://zh.wikipedia.org/wiki/%E8%87%AA%E7%84%B6%E5%B0%8D%E6%95%B8) |
| double log10(double); | [常用对数](https://zh.wikipedia.org/wiki/%E5%B8%B8%E7%94%A8%E5%B0%8D%E6%95%B8) |
| double pow(double, double); | 计算 x y |
| float powf(float, float); | 功能等同于 pow()，只是输入参数与输出参数皆为单浮点数 |
| double ceil(double); | [上取整](https://zh.wikipedia.org/wiki/%E5%8F%96%E6%95%B4%E5%87%BD%E6%95%B0) |
| double floor(double); | [下取整](https://zh.wikipedia.org/wiki/%E5%8F%96%E6%95%B4%E5%87%BD%E6%95%B0) |
| int abs(int); | 求整型的绝对值 |
| double fabs(double); | [求绝对值](https://zh.wikipedia.org/wiki/%E7%BB%9D%E5%AF%B9%E5%80%BC) |
| double cabs(struct complex); | 求复数的绝对值 |
| double frexp(double f, int \*p); | 标准化浮点数 |
| double ldexp(double x, int p); | 与 frexp() 相反 |
| double modf(double, double \*); | 将参数的整数部分通过指针回传，返回小数部分 |
| double fmod(double, double); | 返回两参数相除的余数 |
| double hypot(double, double); | 已知直角三角形两个直角边长度，求斜边长度 |
| double ldexp(double x, int exponent); | 计算 x ∗ 2 exponent |
| double poly(double x, int degree, double coeffs [] ); | 计算多项式 |
| int matherr(struct exception \*); | 数学错误计算处理程序 |

1.12 setjmp.h

setjmp.h是C标准函数库中提供“非本地跳转”的头文件：控制流偏离了通常的子程序调用与返回序列。互补的两个函数setjmp与longjmp提供了这种功能。

setjmp/longjmp的典型用途是异常处理机制的实现：利用longjmp恢复程序或线程的状态，甚至可以跳过栈中多层的函数调用。

1.12.1 函数

|  |  |
| --- | --- |
| int setjmp(jmp\_buf env) | [创建本地的jmp\_buf缓冲区并且初始化，用于将来跳转回此处。这个子程序保存程序的调用环境于env参数所指的缓冲区，env将被longjmp使用。如果是从setjmp直接调用返回，setjmp返回值为0。如果是从longjmp恢复的程序调用环境返回，setjmp返回非零值。](https://zh.wikipedia.org/wiki/Setjmp.h#cite_note-macro-1) |
| void longjmp(jmp\_buf env, int value) | [恢复env所指的缓冲区中的程序调用环境上下文，env所指缓冲区的内容是由setjmp子程序调用所保存。value的值从longjmp传递给setjmp。longjmp完成后，程序从对应的setjmp调用处继续执行，如同setjmp调用刚刚完成。如果value传递给longjmp零值，setjmp的返回值为1；否则，setjmp的返回值为value。](https://zh.wikipedia.org/wiki/Setjmp.h#cite_note-macro-1) |

1.13 signal.h

signal.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的信号处理部分， 定义了程序执行时如何处理不同的[信号](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6))。信号用作进程间通信， 报告异常行为（如除零）、用户的一些按键组合（如同时按下Ctrl与C键，产生信号[SIGINT](https://zh.wikipedia.org/wiki/SIGINT)）。

1.13.1 信号

C语言标准定义了6个信号。都定义在signal.h头文件中[[1]](https://zh.wikipedia.org/wiki/Signal.h" \l "cite_note-c99-1)。

* SIGABRT - 异常中止。
* SIGFPE - 浮点异常。
* SIGILL - 无效指令。
* SIGINT - 交互的用户按键请求，默认情况下，这会导致进程终止。
* SIGSEGV - 无效内存访问。
* SIGTERM - 程序的中止请求。

signal.h可能还定义了其它信号，这依赖于具体实现。例如，类Unix系统还定义了15个以上的信号[[2]](https://zh.wikipedia.org/wiki/Signal.h" \l "cite_note-sus-2)。[Visual C++](https://zh.wikipedia.org/wiki/Visual_C++)的C标准库只支持C语言标准规定的6个信号，即对信号处理只提供最小的支持。

1.13.2 函数

|  |  |
| --- | --- |
| **函数** | **描述** |
| [raise](http://en.cppreference.com/w/c/program/raise) | 人工抛出一个信号 |
| [signal](http://en.cppreference.com/w/c/program/signal) | 当程序收到特定信号时应该执行的处理函数 |

1.14 stdalign.h

stdalign.h是C语言标准2011版新增加的[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，为向后兼容目的定义了4个宏：

* alignas 被扩展为\_Alignas，\_Alignas为C11的关键字。
* alignof 被扩展为\_Alignof，\_Alignof为C11的关键字。
* \_\_alignas\_is\_defined 被扩展为1
* \_\_alignof\_is\_defined 被扩展为1

在C++11中，alignas与alignof是关键字。alignas用于变量定义、数据成员声明、class\struct\union的声明与定义。alignas不可用于基本类型的别名、数组类型等。alignas只能用于更严格（更大对齐值）。alignas的参数可以为数、类型。

1.15 string.h

string.h是[C标准库](https://zh.wikipedia.org/wiki/C%E6%A0%87%E5%87%86%E5%BA%93)的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，其中包含了[宏](https://zh.wikipedia.org/wiki/%E5%AE%8F)(宏)定义、常量以及函数和类型的声明，涉及的内容除了[字符串](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2)处理之外，还包括大量的内存处理函数；

1.15.1 常量和类型

|  |  |
| --- | --- |
| **名称** | **说明** |
| NULL | [表示空指针常量的宏，即表示一个不指向任何有效内存单元地址的指针常量。](https://zh.wikipedia.org/wiki/%E7%A9%BA%E6%8C%87%E9%92%88) |
| size\_t | 无符号整型，被用于sizeof运算符的返回值类型。 |

1.15.2 函数

|  |  |
| --- | --- |
| **名称** | **说明** |
| [void \*memcpy(void \*dest, const void \*src, size\_t n);](https://zh.wikipedia.org/wiki/Memcpy) | 将n字节长的内容从一个内存地址复制到另一个地址；如果两个地址存在重叠，则最终行为未定义 |
| void \*memmove(void \*dest, const void \*src, size\_t n); | 将n字节长的内容从一个内存地址复制到另一个地址；与memcpy不同的是它可以正确作用于两个存在重叠的地址 |
| void \*memchr(const void \*s, char c, size\_t n); | 在从s开始的n个字节内查找c第一次出现的地址并返回，若未找到则返回NULL |
| int memcmp(const void \*s1, const void \*s2, size\_t n); | 对从两个内存地址开始的n个字符进行比较 |
| void \*memset(void \*, int, size\_t); | 用某种字节内容覆写一段内存空间 |
| [char \*strcat(char \*dest, const char \*src);](https://zh.wikipedia.org/w/index.php?title=Strcat&action=edit&redlink=1) | 在字符串dest之后连接上src |
| [char \*strncat(char \*dest, const char \*src, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncat&action=edit&redlink=1) | 从src截取n个字符连接在字符串dest之后，返回dest字符串 |
| [char \*strchr(const char\* str, int ch);](https://zh.wikipedia.org/w/index.php?title=Strchr&action=edit&redlink=1) | 从字符串str头开始查找字符ch首次出现的位置 |
| [char \*strrchr(const char\* str,int ch);](https://zh.wikipedia.org/w/index.php?title=Strrchr&action=edit&redlink=1) | 从字符串str尾开始查找字符ch首次出现的位置 |
| [int strcmp(const char \*, const char \*);](https://zh.wikipedia.org/wiki/Strcmp) | 基于字典顺序比较两个字符串 |
| [int strncmp(const char \*, const char \*, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncmp&action=edit&redlink=1) | 基于字典顺序比较两个字符串，最多比较n个字节 |
| [int strcoll(const char \*, const char \*);](https://zh.wikipedia.org/w/index.php?title=Strcoll&action=edit&redlink=1) | 基于当前区域设置的字符顺序比较两个字符串 |
| [char \*strcpy(char\* str1, const char\* str2);](https://zh.wikipedia.org/wiki/Strcpy) | 将str2拷贝给str1 |
| [char \*strncpy(char\* str1, const char\* str2, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncpy&action=edit&redlink=1) | 截取str2的n个字符拷贝给str1 |
| [char \*strerror(int);](https://zh.wikipedia.org/w/index.php?title=Strerror&action=edit&redlink=1) | [返回错误码对应的解释字符串，参见errno.h（非线程安全函数）](https://zh.wikipedia.org/wiki/Errno.h) |
| [size\_t strlen(const char \*);](https://zh.wikipedia.org/w/index.php?title=Strlen&action=edit&redlink=1) | 返回一个字符串的长度 |
| [size\_t strspn(const char \*s, const char \*strCharSet);](https://zh.wikipedia.org/w/index.php?title=Strspn&action=edit&redlink=1) | 从字符串s的起始处开始，寻找第一个不出现在strCharSet中的字符，返回其位置索引值。换句话说，返回从字符串s的起始位置的完全由strCharSet中的字符构成的子串的最大长度。strspn为string span的缩写。不支持多字节字符集。 |
| [size\_t strcspn(const char \*s, const char \*strCharSet);](https://zh.wikipedia.org/w/index.php?title=Strcspn&action=edit&redlink=1) | 从字符串s的起始处开始，寻找第一个出现在strCharSet中的字符，返回其位置索引值。换句话说，返回从字符串s的起始位置的完全由不属于strCharSet中的字符构成的子串的最大长度。strcspn为string complement span的缩写。不支持多字节字符集。 |
| [char \*strpbrk(const char \*s, const char \*breakset);](https://zh.wikipedia.org/w/index.php?title=Strpbrk&action=edit&redlink=1) | 在字符串s中查找breakset中任意字符第一次出现的位置的指针值。strpbrk为string pointer break缩写。通常，breakset是分隔符的集合。不支持多字节字符集。 |
| [char \*strstr(const char \*haystack, const char \*needle);](https://zh.wikipedia.org/w/index.php?title=Strstr&action=edit&redlink=1) | 在字符串haystack中查找字符串needle第一次出现的位置，heystack的长度必须长于needle |
| [char \*strtok(char \*strToken, const char \*strDelimit );](https://zh.wikipedia.org/w/index.php?title=Strtok&action=edit&redlink=1) | 将一个字符串strToken依据分界符（delimiter）分隔成一系列字符串。此函数非线程安全，且不可重入；但MSVC实现时使用了thread-local static variable因而是线程安全的但仍然是不可重入，即在单线程中不能对两个源字符串交替调用该函数来分析token，应当对一个字符串分析完成后再处理别的字符串。 |
| [size\_t strxfrm(char \*dest, const char \*src, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strxfrm&action=edit&redlink=1) | 根据当前locale转换一个字符串为strcmp使用的内部格式 |
| 名称 | 说明 |
| [void \*memcpy(void \*dest, const void \*src, size\_t n);](https://zh.wikipedia.org/wiki/Memcpy) | 将n字节长的内容从一个内存地址复制到另一个地址；如果两个地址存在重叠，则最终行为未定义 |
| void \*memmove(void \*dest, const void \*src, size\_t n); | 将n字节长的内容从一个内存地址复制到另一个地址；与memcpy不同的是它可以正确作用于两个存在重叠的地址 |
| void \*memchr(const void \*s, char c, size\_t n); | 在从s开始的n个字节内查找c第一次出现的地址并返回，若未找到则返回NULL |
| int memcmp(const void \*s1, const void \*s2, size\_t n); | 对从两个内存地址开始的n个字符进行比较 |
| void \*memset(void \*, int, size\_t); | 用某种字节内容覆写一段内存空间 |
| [char \*strcat(char \*dest, const char \*src);](https://zh.wikipedia.org/w/index.php?title=Strcat&action=edit&redlink=1) | 在字符串dest之后连接上src |
| [char \*strncat(char \*dest, const char \*src, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncat&action=edit&redlink=1) | 从src截取n个字符连接在字符串dest之后，返回dest字符串 |
| [char \*strchr(const char\* str, int ch);](https://zh.wikipedia.org/w/index.php?title=Strchr&action=edit&redlink=1) | 从字符串str头开始查找字符ch首次出现的位置 |
| [char \*strrchr(const char\* str,int ch);](https://zh.wikipedia.org/w/index.php?title=Strrchr&action=edit&redlink=1) | 从字符串str尾开始查找字符ch首次出现的位置 |
| [int strcmp(const char \*, const char \*);](https://zh.wikipedia.org/wiki/Strcmp) | 基于字典顺序比较两个字符串 |
| [int strncmp(const char \*, const char \*, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncmp&action=edit&redlink=1) | 基于字典顺序比较两个字符串，最多比较n个字节 |
| [int strcoll(const char \*, const char \*);](https://zh.wikipedia.org/w/index.php?title=Strcoll&action=edit&redlink=1) | 基于当前区域设置的字符顺序比较两个字符串 |
| [char \*strcpy(char\* str1, const char\* str2);](https://zh.wikipedia.org/wiki/Strcpy) | 将str2拷贝给str1 |
| [char \*strncpy(char\* str1, const char\* str2, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strncpy&action=edit&redlink=1) | 截取str2的n个字符拷贝给str1 |
| [char \*strerror(int);](https://zh.wikipedia.org/w/index.php?title=Strerror&action=edit&redlink=1) | [返回错误码对应的解释字符串，参见errno.h（非线程安全函数）](https://zh.wikipedia.org/wiki/Errno.h) |
| [size\_t strlen(const char \*);](https://zh.wikipedia.org/w/index.php?title=Strlen&action=edit&redlink=1) | 返回一个字符串的长度 |
| [size\_t strspn(const char \*s, const char \*strCharSet);](https://zh.wikipedia.org/w/index.php?title=Strspn&action=edit&redlink=1) | 从字符串s的起始处开始，寻找第一个不出现在strCharSet中的字符，返回其位置索引值。换句话说，返回从字符串s的起始位置的完全由strCharSet中的字符构成的子串的最大长度。strspn为string span的缩写。不支持多字节字符集。 |
| [size\_t strcspn(const char \*s, const char \*strCharSet);](https://zh.wikipedia.org/w/index.php?title=Strcspn&action=edit&redlink=1) | 从字符串s的起始处开始，寻找第一个出现在strCharSet中的字符，返回其位置索引值。换句话说，返回从字符串s的起始位置的完全由不属于strCharSet中的字符构成的子串的最大长度。strcspn为string complement span的缩写。不支持多字节字符集。 |
| [char \*strpbrk(const char \*s, const char \*breakset);](https://zh.wikipedia.org/w/index.php?title=Strpbrk&action=edit&redlink=1) | 在字符串s中查找breakset中任意字符第一次出现的位置的指针值。strpbrk为string pointer break缩写。通常，breakset是分隔符的集合。不支持多字节字符集。 |
| [char \*strstr(const char \*haystack, const char \*needle);](https://zh.wikipedia.org/w/index.php?title=Strstr&action=edit&redlink=1) | 在字符串haystack中查找字符串needle第一次出现的位置，heystack的长度必须长于needle |
| [char \*strtok(char \*strToken, const char \*strDelimit );](https://zh.wikipedia.org/w/index.php?title=Strtok&action=edit&redlink=1) | 将一个字符串strToken依据分界符（delimiter）分隔成一系列字符串。此函数非线程安全，且不可重入；但MSVC实现时使用了thread-local static variable因而是线程安全的但仍然是不可重入，即在单线程中不能对两个源字符串交替调用该函数来分析token，应当对一个字符串分析完成后再处理别的字符串。 |
| [size\_t strxfrm(char \*dest, const char \*src, size\_t n);](https://zh.wikipedia.org/w/index.php?title=Strxfrm&action=edit&redlink=1) | 根据当前locale转换一个字符串为strcmp使用的内部格式 |

1.16 tgmath.h

tgmath.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，提供了数学函数的类型通用的宏定义。使用这些宏调用数学函数时，会根据参数自动对应到类型适合的数学函数，其效果类似于[C++](https://zh.wikipedia.org/wiki/C++)的[函数重载](https://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B0%E9%87%8D%E8%BD%BD)，使得编程者不必繁琐地去调用数学库函数的单精度、双精度、长双精度、单精度复数、双精度复数、长双精度复数等各个版本。

1.16.1 宏

1.16.2 math.h与complex.h共有

一些常见数学函数既在math.h有实数版本，也在complex.h有复数版本。tgmath.h提供了下述类型通用的宏定义：

acos、asin、atan、acosh、asinh、atanh、cos、sin、tan、cosh、sinh、tanh、exp、log、pow、sqrt、fabs

math.h专用

对于包含在math.h中，但在complex.h中没有对应的函数，提供了下述通用类型的宏：

atan2、cbrt、ceil、copysign、erf、erfc、exp2、expm1、fdim、floor、fma、fmax、fmin、fmod、frexp、hypot、ilogb、ldexp、lgamma、llrint、llround、log10、log1p、log2、logb、lrint、lround、nearbyint、nextafter、nexttoward、remainder、remquo、rint、round、scalbn、scalbln、tgamma、trunc

complex.h专用

对于包含在complex.h中，但在math.h中没有对应的函数，提供了下述通用类型的宏：

carg、cimag、conj、cproj、creal

1.17 time.h

time.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中获取时间与日期、对时间与日期数据操作及格式化的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)。

1.17.1 数据类型

日历时间（Calendar Time），是从一个标准时间点（epoch）到现在的时间经过的秒数，不包括插入[闰秒](https://zh.wikipedia.org/wiki/%E9%97%B0%E7%A7%92)对时间的调整。开始计时的标准时间点，各种编译器一般使用[UTC](https://zh.wikipedia.org/wiki/%E5%8D%8F%E8%B0%83%E4%B8%96%E7%95%8C%E6%97%B6) 1970-01-01 00:00:00。日历时间用数据类型time\_t表示。[[1]](https://zh.wikipedia.org/wiki/Time.h" \l "cite_note-APUE-1):20time\_t类型实际上一般是32位整数类型，因此表示的时间不能晚于[UTC](https://zh.wikipedia.org/wiki/%E5%8D%8F%E8%B0%83%E4%B8%96%E7%95%8C%E6%97%B6) 2038-01-18 19:14:07。为此，某些编译器引入了64位甚至更长的整型来保存日历时间，如Visual C++支持\_\_time64\_t数据类型，通过\_time64()函数获取日历时间，可支持到[UTC](https://zh.wikipedia.org/wiki/%E5%8D%8F%E8%B0%83%E4%B8%96%E7%95%8C%E6%97%B6) 3001-01-01 00:00:00的时间。

处理器时间（Processor Time），也被称为[CPU](https://zh.wikipedia.org/wiki/CPU)时间（CPU Time），用以度量[进程](https://zh.wikipedia.org/wiki/%E8%BF%9B%E7%A8%8B)使用的CPU资源。处理器时间以时钟滴答数（Clock Tick）计算，通常从进程启动开始计时，因此这是相对时间。时钟滴答数用[系统基本数据类型](https://zh.wikipedia.org/w/index.php?title=%E7%B3%BB%E7%BB%9F%E5%9F%BA%E6%9C%AC%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B&action=edit&redlink=1)clock\_t来表示，每[秒钟](https://zh.wikipedia.org/wiki/%E7%A7%92%E9%92%9F)包含CLOCKS\_PER\_SEC（time.h中定义的常量，一般为1000）个时钟滴答，也可使用sysconf函数得到每秒的时钟滴答数。clock\_t类型一般是32位整数类型。[[1]](https://zh.wikipedia.org/wiki/Time.h" \l "cite_note-APUE-1):20

分解时间(broken-down time)，用结构数据类型tm表示，tm包含下列结构成员：

|  |  |
| --- | --- |
| **成员** | **描述** |
| [int tm\_hour](https://zh.wikipedia.org/wiki/%E6%95%B4%E6%95%B0_(%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6)) | 小时 (0 – 23) |
| int tm\_isdst | [夏令时启用 (> 0)、禁用 (= 0)、未知 (< 0)](https://zh.wikipedia.org/wiki/%E5%A4%8F%E4%BB%A4%E6%97%B6) |
| int tm\_mday | 一月中的哪一天 (1 – 31) |
| int tm\_min | 分 (0 – 59) |
| int tm\_mon | 月 (0 – 11, 0 = 一月) |
| int tm\_sec | 秒 (0 – 60, 60 = 闰秒) |
| int tm\_wday | 一周中的哪一天 (0 – 6, 0 = 周日) |
| int tm\_yday | 一年中的哪一天 (0 – 365) |
| int tm\_year | 1900 以来的年数 |

1.17.2 函数

time\_t time(time\_t\* timer)

得到从标准计时点（一般是UTC 1970年1月1日午夜）到当前时间的秒数。

clock\_t clock(void)

得到从进程启动到此次函数调用的累计的时钟滴答数。

struct tm\* gmtime(const time\_t\* timer)

从日历时间time\_t到分解时间tm（[世界协调时](https://zh.wikipedia.org/wiki/%E4%B8%96%E7%95%8C%E5%8D%8F%E8%B0%83%E6%97%B6)UTC）的转换。函数返回的是一个静态分配的tm结构存储空间，该存储空间被gmtime, localtime与ctime函数所共享. 这些函数的每一次调用会覆盖这块tm结构存储空间的内容。

struct tm\* gmtime\_r(const time\_t\* timer, struct tm\* result)

该函数是gmtime函数的线程安全版本.

struct tm\* localtime(const time\_t\* timer)

从日历时间time\_t到分解时间tm的转换，即结果数据已经调整到本地时区与夏令时。

time\_t mktime(struct tm\* ptm)

从基于本地时区(与夏令时)的分解时间tm到日历时间time\_t的转换。忽略tm\_wday与tm\_yday的输入值。如果tm\_isdst不确定则输入设为-1。其它各输入域的值可以任意设定，输出时被规范化到正确范围。例如，想要计算2012年的第200天的日期，设为1月200日，函数会输出正确的结果

time\_t timegm(struct tm\* brokentime)

从分解时间tm（被视作UTC时间，不考虑本地时区设置）到日历时间time\_t的转换。该函数较少被使用。

char \*asctime(const struct tm\* tmptr)

把分解时间tm输出到字符串，结果的格式为"Www Mmm dd hh:mm:ss yyyy"，即“周几 月份数 日数 小时数:分钟数:秒钟数 年份数”。函数返回的字符串为静态分配，长度不大于26，与ctime函数共享。函数的每次调用将覆盖该字符串内容。

char\* ctime(const time\_t\* timer)

把日历时间time\_t timer输出到字符串，输出格式与asctime函数一样.

size\_t strftime(char\* s, size\_t n, const char\* format, const struct tm\* tptr)

把分解时间tm转换为自定义格式的字符串，类似于常见的字符串格式输出函数sprintf。例如：strftime(buf, 64, "%Y-%m-%d %H:%M:%S", localtime);

char \* strptime(const char\* buf, const char\* format, struct tm\* tptr)

strftime的逆操作，把字符串按照自定义的格式转换为分解时间tm。

double difftime(time\_t timer2, time\_t timer1)

比较两个日历时间之差。

1.18 ctype.h

ctype.h是[C标准函数库](https://zh.wikipedia.org/wiki/C%E6%A8%99%E6%BA%96%E5%87%BD%E6%95%B8%E5%BA%AB)中的[头文件](https://zh.wikipedia.org/wiki/%E5%A4%B4%E6%96%87%E4%BB%B6)，定义了一批[C语言](https://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)字符分类函数（C character classification functions），用于测试字符是否属于特定的字符类别，如字母字符、控制字符等等。既支持单字节字符，也支持[宽字符](https://zh.wikipedia.org/wiki/%E5%AE%BD%E5%AD%97%E7%AC%A6)。

1.18.1 函数

单字节字符处理函数在ctype.h(C++的cctype )中声明。宽字节字符处理函数在wctype.h(C++的cwctype)中声明.

|  |  |  |
| --- | --- | --- |
| 单字节 | 宽字节 | 描述 |
| [isalnum](http://en.cppreference.com/w/c/string/byte/isalnum) | [iswalnum](http://en.cppreference.com/w/c/string/wide/iswalnum) | 是否为字母数字 |
| [isalpha](http://en.cppreference.com/w/c/string/byte/isalpha) | [iswalpha](http://en.cppreference.com/w/c/string/wide/iswalpha) | 是否为字母 |
| [islower](http://en.cppreference.com/w/c/string/byte/islower) | [iswlower](http://en.cppreference.com/w/c/string/wide/iswlower) | 是否为小写字母 |
| [isupper](http://en.cppreference.com/w/c/string/byte/isupper) | [iswupper](http://en.cppreference.com/w/c/string/wide/iswupper) | 是否为大写字母 |
| [isdigit](http://en.cppreference.com/w/c/string/byte/isdigit) | [iswdigit](http://en.cppreference.com/w/c/string/wide/iswdigit) | 是否为数字 |
| [isxdigit](http://en.cppreference.com/w/c/string/byte/isxdigit) | [iswxdigit](http://en.cppreference.com/w/c/string/wide/iswxdigit) | 是否为16进制数字 |
| [iscntrl](http://en.cppreference.com/w/c/string/byte/iscntrl) | [iswcntrl](http://en.cppreference.com/w/c/string/wide/iswcntrl) | 是否为控制字符 |
| [isgraph](http://en.cppreference.com/w/c/string/byte/isgraph) | [iswgraph](http://en.cppreference.com/w/c/string/wide/iswgraph) | 是否为图形字符（例如，空格、控制字符都不是） |
| [isspace](http://en.cppreference.com/w/c/string/byte/isspace) | [iswspace](http://en.cppreference.com/w/c/string/wide/iswspace) | [是否为空格字符（包括制表符、回车符、换行符等）](https://zh.wikipedia.org/w/index.php?title=%E7%A9%BA%E6%A0%BC%E5%AD%97%E7%AC%A6&action=edit&redlink=1) |
| [isblank](http://en.cppreference.com/w/c/string/byte/isblank) | [iswblank](http://en.cppreference.com/w/c/string/wide/iswblank) | 是否为空白字符 (C99/C++11新增)（包括水平制表符） |
| [isprint](http://en.cppreference.com/w/c/string/byte/isprint) | [iswprint](http://en.cppreference.com/w/c/string/wide/iswprint) | 是否为可打印字符 |
| [ispunct](http://en.cppreference.com/w/c/string/byte/ispunct) | [iswpunct](http://en.cppreference.com/w/c/string/wide/iswpunct) | 是否为标点 |
| [tolower](http://en.cppreference.com/w/c/string/byte/tolower) | [towlower](http://en.cppreference.com/w/c/string/wide/towlower) | 转换为小写 |
| [toupper](http://en.cppreference.com/w/c/string/byte/toupper) | [towupper](http://en.cppreference.com/w/c/string/wide/towupper) | 转换为大写 |
| 不适用 | [iswctype](http://en.cppreference.com/w/c/string/wide/iswctype) | 检查一个wchar\_t是否是属于指定的分类 |
| 不适用 | [towctrans](http://en.cppreference.com/w/c/string/wide/towctrans) | 使用指定的变换映射来转换一个wchar\_t（实际上是大小写的转换） |
| 不适用 | [wctype](http://en.cppreference.com/w/c/string/wide/wctype) | 返回一个宽字符的类别，用于iswctype函数 |
| 不适用 | [wctrans](http://en.cppreference.com/w/c/string/wide/wctrans) | 返回一个变换映射，用于 towctrans |

©统信软件技术有限公司。访问者可将本网站提供的内容或服务用于个人学习、研究或欣赏，以及其他非商业性或非盈利性用途，但同时应遵守著作权法及其他相关法律的规定，不得侵犯本网站及相关权利人的合法权利。除此以外，将本网站任何内容或服务进行转载，须备注：**该文档出自【faq.uniontech.com】统信软件知识分享平台**。否则统信软件将追究相关版权责任。