Standard C 语言标准函数库速查 (Cheat Sheet)

说明

本页面包含C语言标准版的15个头文件解释以及函数，137个函数和演示，可以点击函数名字进去查看。
纯粹无责任乱译，参照本页产生的任何错误概不负责。本页也不一定会保证能持续更新。(查看 更新记录)

C语言函数搜索框：

最后更新：2013年4月24日
如果您喜欢它，请考虑 捐赠 一点资金，感激不尽！

分享到...

C语言标准头

<assert.h> 断言

头文件<assert.h>唯一的目的是提供宏assert的定义。如果断言非真（expression==0），则程序会在标准错误流输出提示信息，并使程序异常中止调用abort() 。

定义：**void assert (int expression);**

```
//#define NDEBUG
#include <assert.h>
int main(int argc, char* argv[]){
        int a = 12;
        int b = 24;
        assert( a > b );
        printf("a is larger than b!");
        return 0;
}
```

上面的程序会发现程序中止，printf并未执行，且有这样的输出： **main: Assertion `a > b' failed.** 原因就是因为a其实小于b，导致断言失败，assert 输出错误信息，并调用abort()中止了程序执行。

<ctype.h> 字符测试

<ctype.h> 主要提供两类重要的函数：字符测试函数和字符大小转化函数。提供的函数中都以int类型为参数，并返回一个int类型的值。实参类型应该隐式转换或者显示转换为int类型。

int isalnum(int c); 判断是否是字母或数字。
int isalpha(int c); 判断是否是字母。
int iscntrl(int c); 判断是否是控制字符。
int isdigit(int c); 判断是否是数字。
int isgraph(int c); 判断是否是可显示字符。
int islower(int c); 判断是否是小写字母。
int isupper(int c); 判断是否是大写字母。
int isprint(int c); 判断是否是可显示字符。
int ispunct(int c); 判断是否是标点字符。
int isspace(int c); 判断是否是空白字符
int isxdigit(int c); 判断字符是否为16进

<math.h> 数学函数

<math.h> 是C语言中的数学函数库

三角函数

double sin(double x); 正弦
double cos(double x); 余弦
double tan(double x); 正切
*cot三角函数，可以使用tan(PI/2-x)来实现。

反三角函数

double asin(double x); 结果介于[-PI/2, PI/2]
double acos(double x); 结果介于[0, PI]
double atan(double x);
反正切(主值)，结果介于[-PI/2, PI/2]
double atan2(double y,double);
反正切(整圆值)，结果介于[-PI, PI]

双曲三角函数

double sinh(double x); 计算双曲正弦
double cosh(double x); 计算双曲余弦
double tanh(double x); 计算双曲正切

指数与对数

double exp(double x); 求取自然数e的幂
double sqrt(double x); 开平方
double log(double x); 以e为底的对数
double log10(double x); 以10为底的对数
double pow(double x, double y);
计算以x为底数的y次幂
float powf(float x, float y);
与pow一致，输入与输出皆为浮点数

取整

double ceil(double); 取上整
double floor(double); 取下整

标准化浮点数

double frexp(double f, int *p);
标准化浮点数，f = x * 2^p，已知f求x，p (x介于[0.5, 1])
double ldexp(double x, int p);
与frexp相反，已知x，p求f

取整与取余

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

<stdio.h> 输入和输出

<stdio.h>头文件定义了用于输入和输出的函数、类型和宏。最重要的类型是用于声明文件指针的FILE。另外两个常用的类型是 size_t和fpos_t，size_t是由运算符sizeof产生的无符号整类型；fpos_t类型定义能够唯一说明文件中的每个位置的对象。由头部 定义的最有用的宏是EOF，其值代表文件的结尾。

变量：
typedef size_t
typedef FILE
typedef fpos_t

常量 ：
NULL 空值
_IOFBF 表示完全缓冲
_IOLBF 表示线缓冲
_IONBF 表示无缓存
BUFSIZ setbuf函数所使用的缓冲区的大小
EOF EOF是负整数表示END OF FILE
FOPEN_MAX (20)同时打开的文件的最大数量
FILENAME_MAX 文件名的最大长度
L_tmpnam 整数，最大长度的临时文件名
SEEK_CUR 取得目前文件位置
SEEK_END 将读写位置移到文件尾时
SEEK_SET 将读写位置移到文件开头
TMP_MAX tmpnam最多次数
stderr 标准错误流，默认为屏幕，可输出到文件。
stdin 标准输入流，默认为键盘
stdout 标准输出流，默认为屏幕

所有函数(点击可查看介绍和DEMO) ：

clearerr(); 复位错误标志
fclose(); 关闭一个流。
feof(); 检测文件结束符
ferror(); 检查流是否有错误
fflush(); 更新缓冲区
fgetpos(); 移动文件流的读写位置
fopen(); 打开文件
fread(); 从文件流读取数据
freopen(); 打开文件
fseek(); 移动文件流的读写位置
fsetpos(); 定位流上的文件指针
ftell(); 取得文件流的读取位置
fwrite(); 将数据写至文件流
remove(); 删除文件
rename(); 更改文件名称或位置
rewind(); 重设读取目录的位置为开头位置
setbuf(); 把缓冲区与流相联
setvbuf(); 把缓冲区与流相关
tmpfile(); 以wb+形式创建一个临时二进制文件
tmpnam(); 产生一个唯一的文件名
fprintf(); 格式化输出数据至文件
fscanf(); 格式化字符串输入
printf(); 格式化输出数据
scanf(); 格式输入函数
sprintf(); 格式化字符串复制
sscanf(); 格式化字符串输入
vfprintf(); 格式化输出数据至文件
vprintf(); 格式化输出数据

制。

`int tolower(int c);` 转换为小写字母。

`int toupper(int c);` 转换为大写字母。

<errno.h> 错误代码

error.h 是 C语言 C标准函式库里的头文件，定义了通过错误码来返回错误信息的宏：

errno 宏定义为一个int型态的左值，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 源自于函数的参数超出范围,例如 sqrt(-1)
ERANGE 源自于函数的结果超出范围,例如strtol("0xfffffffff",NULL,0)
EILSEQ 源自于不合法的字符顺序,例如wcstombs(str, L"\xffff", 2)

<float.h> 浮点数运算

float头文件定义了浮点型数值的最大最小限 浮点型数值以下面的方式定义：符号-value E 指数 符号是正负，value是数字的值
下面的值是用#define定义的，这些值是详细的实现，但是可能没有比这里给出的更详细，
在所有实例里FLT指的是float，DBL是double，LDBL指的是long double

FLT_ROUNDS
定义浮点型数值四舍五入的方式，-1是不确定，0是向0，1是向最近，2是向正无穷大，3是负无穷大

FLT_RADIX 2
定义指数的基本表示（比如base-2是二进制，base-10是十进制表示法，16是十六进制）

FLT_MANT_DIG, DBL_MANT_DIG, LDBL_MANT_DIG
定义数值里数字的个数

FLT_DIG 6, DBL_DIG 10, LDBL_DIG 10
在四舍五入之后能不更改表示的最大小数位

FLT_MIN_EXP, DBL_MIN_EXP, LDBL_MIN_EXP
FLT_RADIX 的指数的最小负整数值

FLT_MIN_10_EXP -37, DBL_MIN_10_EXP -37, LDBL_MIN_10_EXP -37
10进制表示法的的指数的最小负整数值

FLT_MAX_EXP ,DBL_MAX_EXP ,LDBL_MAX_EXP
FLT_RADIX 的指数的最大整数值

FLT_MAX_10_EXP +37 ,DBL_MAX_10_EXP ,LDBL_MAX_10_EXP +37 +37
10进制表示法的的指数的最大整数值

FLT_MAX 1E+37, DBL_MAX 1E+37, LDBL_MAX 1E+37
浮点型的最大限

FLT_EPSILON 1E-5, DBL_EPSILON 1E-9, LDBL_EPSILON 1E-9
能表示的最小有符号数

<limits.h> 取值范围

CHAR_BIT 一个ASCII字符长度

SCHAR_MIN 字符型最小值

SCHAR_MAX 字符型最大值

UCHAR_MAX 无符号字符型最大值

CHAR_MIN
CHAR_MAX
char字符的最大最小值，如果char字符正被表示有符号整数。它们的值就跟有符号整数一样。 否则char字符的最小值就是0，最大值就是无符号char字符的最大值。

MB_LEN_MAX 一个字符所占最大字节数

SHRT_MIN 最小短整型

SHRT_MAX 最大短整形

USHRT_MAX 最大无符号短整型

INT_MIN 最小整型

INT_MAX 最大整形

UINT_MAX 最大无符号整型

LONG_MIN 最小长整型

<setjmp.h> “非局部跳转”

在该头文件中定义了一种特别的函数调用和函数返回顺序的方式。这种方式不同于以往的函数调用和返回顺序，
它允许程序流程立即从一个深层嵌套的函数中返回。
<setjmp.h> 中定义了两个宏：

```
int setjmp(jmp_buf env); /*设置调转点*/
longjmp(jmp_buf jmpb, int retval); /*跳转*/
```

宏setjmp的功能是将当前程序的状态保存在结构env，为调用宏longjmp设置一个跳转点。setjmp将当前信息保存在env中供longjmp使用。其中env是jmp_buf结构类型的。
Demo :

```
#include  <stdio.h>
#include  <setjmp.h>
static jmp_buf buf;
void second(void) {
    printf("second\n");
        // 打印
    longjmp(buf,1);
        // 跳回setjmp的调用处
        - 使得setjmp返回值为1
}
 void first(void) {
    second();
    printf("first\n");
        // 不可能执行到此行
}
 int main() {
    if ( ! setjmp(buf) ) {
        first();
        // 进入此行前，setjmp返回0
    } else {
        // 当longjmp跳转回，
        setjmp返回1，因此进入此行
        printf("main\n");
                // 打印
    }
     return 0;
}
```

直接调用setjmp时，返回值为0，这一般用于初始化（设置跳转点时）。以后再调用longjmp宏时用env变量进行跳转。程序会自动跳转到setjmp宏的返回语句处，此时setjmp的返回值为非0，由longjmp的第二个参数指定。
一般地，宏setjmp和longjmp是成对使用的，这样程序流程可以从一个深层嵌套的函数中返回。

<signal.h> 信号

在signal.h头文件中，提供了一些函数用以处理执行过程中所产生的信号。

宏：

SIG_DFL

SIG_ERR

SIGN_IGN

SIGABRT

SIGFPE

SIGILL

SIGINT

SIGSEGV

SIGTERM

函数：

signal();

raise();

变量：

typedef sig_atomic_t

sig_atomic_t 类型是int类型，用于接收signal函数的返回值。

以SIG_开头的宏用于定义信号处理函数

vsprintf(); 格式化字符串复制
fgetc(); 由文件中读取一个字符
fgets(); 文件中读取一字符串
fputc(); 将一指定字符写入文件流中
fputs(); 将一指定的字符串写入文件内
getc(); 由文件中读取一个字符
getchar(); 由标准输入设备内读进一字符
gets(); 由标准输入设备内读进一字符串
putc(); 将一指定字符写入文件中
putchar(); 将指定的字符写到标准输出设备
puts(); 送一字符串到流stdout中
ungetc(); 将指定字符写回文件流中
perror(); 打印出错误原因信息字符串

<stdlib.h> 实用功能

<stdlib.h> 头文件里包含了C语言的中最常用的系统函数

宏：
NULL 空
EXIT_FAILURE 失败状态码
EXIT_SUCCESS 成功状态码
RAND_MAX rand的最大返回值
MB_CUR_MAX 多字节字符中的最大字节数

变量：
typedef size_t 是unsigned integer类型
typedef wchar_t 一个宽字符的大小
struct div_t 是结构体类型 作为div函数的返回类型
struct ldiv_t 是结构体类型 作为ldiv函数的返回类型

函数：
字符串函数
atof(); 将字符串转换成浮点型数
atoi(); 将字符串转换成整型数
atol(); 将字符串转换成长整型数
strtod(); 将字符串转换成浮点数
strtol(); 将字符串转换成长整型数
strtoul(); 将字符串转换成无符号长整型数

内存控制函数
calloc(); 配置内存空间
free(); 释放原先配置的内存
malloc(); 配置内存空间
realloc(); 重新分配主存

环境函数
abort(); 异常终止一个进程
atexit(); 设置程序正常结束前调用的函数
exit(); 正常结束进程
getenv(); 取得环境变量内容
system(); 执行shell 命令

搜索和排序函数
bsearch(); 二元搜索
qsort(); 利用快速排序法排列数组

数学函数
abs(); 计算整型数的绝对值
div(); 将两个整数相除，返回商和余数
labs(); 取长整型绝对值
ldiv(); 两个长整型数相除，返回商和余数
rand(); 随机数发生器
srand(); 设置随机数种子

多字节函数
mblen(); 根据locale的设置确定字符的字节数
mbstowcs(); 把多字节字符串转换为宽字符串
mbtowc(); 把多字节字符转换为宽字符
wcstombs(); 把宽字符串转换为多字节字符串
wctomb(); 把宽字符转换为多字节字符

LONG_MAX 最大长整型

ULONG_MAX 无符号长整型

<locale.h> 本土化

国家、文化和语言规则集称为区域设置，**<locale.h>** 头文件中定义了区域设置相关的函数。setlocale函数用于设置或返回当前的区域特性，localeconv用于返回当前区域中的数字和货币信息（保存在struct lconv结构实例中）。setlocale的第一个实参指定要改变的区域行为类别，预定义的setlocale类别有：

LC_ALL
全部本地化信息

LC_COLLATE
影响strcoll和strxfrm

LC_CTYPE
影响字符处理函数和多行字符处理函数

LC_MONETARY
影响localeconv返回的货币格式化信息

LC_NUMERIC
影响格式化输入输出字符中的小数点符号

LC_TIME
影响strftime函数

<locale.h> 头文件中提供了2个函数
setlocale() 设置或恢复本地化信息
localeconv() 返回当前地域设置的信息

setlocale(constant,location) 用法
如果这个函数成功执行，将返回当前的场景属性；如果执行失败，将返回False。

constant 参数（必要参数。指定设置的场景信息）

LC_ALL - 所有下属的常量

LC_COLLATE - 排列顺序

LC_CTYPE - 字符分类和转换（例如：将所有的字符转换成小写或大写形式）

LC_MESSAGES - 系统信息格式

LC_MONETARY - 货币 / 通货格式

LC_NUMERIC - 数值格式

LC_TIME - 日期和时间格式

location （必要参数）

必要参数。指定需要进行场景信息设置的国家或区域。它可以由一个字符串或一个数组组成。如果本地区域是一个数组，那么setlocale()函数将尝试每个数组元素直到它从中获取有效的语言和区域代码信息为止。如果一个区域处于不同操作系统中的不同名称下，那么这个参数将非常有用。

struct lconv *localeconv(void); 用法

localeconv 返回lconv结构指针 lconv结构介绍：保存格式化的数值信息，保存数值包括货币和非货币的格式化信息，localeconv返回指向该对象的指针，以下为结构中的成员及信息：

char *decimal_point; 数字的小数点号

char *thousands_sep; 数字的千分组分隔符

每个元素为相应组中的数字位数，索引越高的元素越靠左边。一个值为CHAR_MAX的元素表示没有更多的分组了。一个值为0的元素表示前面的元素能用在靠左边的所有分组中

char *grouping; 数字分组分隔符

char *int_curr_symbol; 前面的三个字符ISO 4217中规定的货币符号，第四个字符是分隔符，第五个字符是'\0' */

char *currency_symbol; 本地货币符号

char *mon_decimal_point; 货币的小数点号

char *mon_thousands_sep; 千分组分隔符

char *mon_grouping; 类似于grouping元素

char *positive_sign; 正币值的符号

char *negative_sign; 负币值的符号

char int_frac_digits; 国际币值的小数部分

char frac_digits; 本地币值的小数部分

char p_cs_precedes; 如果currency_symbol

SIG_DFL 默认信号处理函数。

SIG_ERR 表示一个错误信号，当signal函数调用失败时的返回值。

SIG_IGN 信号处理函数，表示忽略该信号。

SIG开头的宏是用来在下列情况下，用来表示一个信号代码：

SIGABRT 异常终止（abort函数产生）。

SIGFPE 浮点错误（0作为除数产生的错误，非法的操作）。

SIGILL 非法操作（指令）。

SIGINT 交互式操作产生的信号（如CTRL - C）。

SIGSEGV 无效访问存储（片段的非法访问，内存非法访问）。

SIGTERM 终止请求。

signal 函数

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

上面的函数定义中，sig 表示一个信号代码（相当于暗号类别），即是上面所定义的SIG开头的宏。当有信号出现（即当收到暗号）的时候，参数func所定义的函数就会被调用。如果func等于SIG_DFL，则表示调用默认的处理函数。如果等于SIG_IGN，则表示这个信号被忽略（不做处理）。如果func是用户自定义的函数，则会先调用默认的处理函数，再调用用户自己定义的函数。 自定义函数，有一个参数，参数类型为int，用来表示信号代码（暗号类别）。同时，函数必须以return、abort、exit 或 longjump等语句结束。当自定义函数运行结束，程序会继续从被终止的地方继续运行。（除非信号是SIGFPE导致结果未定义，则可能无法继续运行）

如果调用signal函数成功，则会返回一个指针，该指针指向为所指定的信号类别的所预先定义的信号处理器。
如果调用失败，则会返回一个SIG_ERR，同时errno的值也会被相应的改变。

raise 函数

int raise(int sig);

发出一个信号sig。信号参数为SIG开头的宏。

如果调用成功，返回0。否则返回一个非零值。

<stdarg.h> 可变参数

<stdarg.h> 头文件定义了一些宏，当函数参数未知时去获取函数的参数

变量：**typedef va_list**

宏：
va_start()
va_arg()
va_end()

变量和定义
va_list类型通过stdarg宏定义来访问一个函数的参数表，参数列表的末尾会用省略号省略

声明：**void va_start(va_list ap, last_arg);**
用va_arg和va_end宏初始化参数ap，last_arg是传给函数的固定参数的最后一个，省略号之前的那个参数 注意va_start必须在使用va_arg和va_end之前调用

声明：**type va_arg(va_list ap, type);**
用type类型扩展到参数表的下个参数
注意ap必须用va_start初始化，如果没有下一个参数，结果会是undefined

声明：**void va_end(va_list ap) ;** 允许一个有参数表（使用va_start宏）的函数返回，如果返回之前没有调用va_end,结果会是undefined。参数变量列表可能不再使用（在没调用va_start的情况下调用va_end）

<string.h> 字符串函数

<stdlib.h> 头文件里包含了C语言的最常用的字符串操作函数

宏：
NULL 空

变量：
typedef size_t

函数：
memchr(); 在某一内存范围中查找一特定字符
memcmp(); 比较内存内容
memcpy(); 拷贝内存内容
memmove(); 拷贝内存内容
memset(); 将一段内存空间填入某值
strcat(); 连接两字符串
strncat(); 连接两字符串
strchr(); 查找字符串中第一个出现的指定字符
strcmp(); 比较字符串
strncmp(); 比较2个字符串的前N个字符
strcoll(); 采用目前区域的字符排列比较字符串
strcpy(); 拷贝字符串
strncpy(); 拷贝字符串
strcspn(); 返回字符连续不含指定字符的字符数
strerror(); 返回错误原因的描述字符串
strlen(); 计算字符串长度
strpbrk(); 查找字符串中第一个出现的指定字符
strrchr(); 查找字符串中最后出现的指定字符
strspn(); 返回字符串连续不含指定字符的字符数
strstr(); 在一字符串中查找指定的字符串
strtok(); 分割字符串
strxfrm(); 转换字符串

<time.h> 时间和日期函数

<time.h> 是C标准函数库中获取时间与日期、对时间与日期数据操作及格式化的头文件。

宏：
NULL null是一个null指针常量的值
CLOCKS_PER_SEC 每秒的时钟数

变量：
typedef size_t 类型定义
typedef clock_t 类型定义
struct tm 结构体

```
struct tm {
int tm_sec;  /* 秒 - 取值区间为[0,59] */

int tm_min;  /* 分 - 取值区间为[0,59] */

int tm_hour;  /* 时 - 取值区间为[0,23] */

int tm_mday;  /* 一个月中的日期 - 取值区间为[1,31] */

int tm_mon;  /* 月份（从一月开始，0代表一月） - 取值区间为[0,11] */

int tm_year;  /* 年份，其值等于实际年份减去1900 */

int tm_wday;  /* 星期 - 取值区间为[0,6]，其中0代表星期天，1代表星期一，以此类推 */

int tm_yday;  /* 从每年的1月1日开始的天数 - 取值区间为[0,365]，其中0代表1月1日，1代表1月2日，以此类推 */

int tm_isdst;  /* 夏令时标识符，实行夏令时的时候，tm_isdst为正。不实行夏令时的进
候，tm_isdst为0；不了解情况时，tm_isdst()为负。*/
};
```

函数：
asctime(); 将时间和日期以字符串格式表示
clock(); 确定处理器时间

放在正币值之前则为1，否则为0

char p_sep_by_space; 当且仅当currency_symbol与正币值之间用空格分开时为1

char n_cs_precedes; < 如果currency_symbol放在负币值之前则为1，否则为0/dt>

char n_sep_by_space; 当且仅当currency_symbol与负币值之间用空格分开时为1

char p_sign_posn; 格式化选项

0 – 在数量和货币符号周围的圆括号
1 – 数量和货币符号之前的 + 号
2 – 数量和货币符号之后的 + 号
3 – 货币符号之前的 + 号
4 – 货币符号之后的 + 号

char n_sign_posn 格式化选项

0 – 在数量和货币符号周围的圆括号
1 – 数量和货币符号之前的 – 号
2 – 数量和货币符号之后的 – 号
3 – 货币符号之前的 – 号
4 – 货币符号之后的 – 号

最后提示：可以使用setlocale(LC_ALL,NULL)函数将场景信息设置为系统默认值。

<stddef.h> 一些常数，类型和变量

<stddef.h> 头文件定义了一些标准定义，许多定义也会出现在其他的头文件里

宏命令：NULL 和 offsetof()

变量：
typedef ptrdiff_t
typedef size_t
typedef wchar_t

变量和定义：
ptrdiff_t 是两个指针相减的结果
size_t 是sizeof一个关键词得到的无符号整数值
wchar_t 是一个宽字符常量的大小，是整数类型
NULL 是空指针的常量值

offsetof(type, member-designator); 这个宏返回一个结构体成员相对于结构体起始地址的偏移量（字节为单位），type是结构体的名字，member-designator是结构体成员的名字。

ctime(); 把日期和时间转换为字符串
difftime(); 计算两个时刻之间的时间差
gmtime(); 把日期和时间转换为(GMT)时间
localtime(); 取得当地目前时间和日期
mktime(); 将时间结构数据转换成经过的秒数
strftime(); 将时间格式化
time(); 取得目前的时间