

Borland C++ 3.1 库函数手册

李文通 刘天放 毕卫明 等编

北京航空航天大学出版社

(京)新登字 166 号

内 容 简 介

Borland C++ 3.1 支持多重覆盖窗口和自动覆盖管理,既是面向对象的程序设计语言,又是一个编辑、编译、调试、运行和剖视优化一体化的优秀的语言开发环境,且与 Turbo C 相兼容。本书是上述两个软件的最新参考系列书,按字母顺序介绍了 Borland C++ 的每个库函数,它们的功能、用法、原型所在的头文件、返回值、相关函数等信息,并介绍了有关全局变量的参考和用法。本书是 Borland C++ 库函数的快速参考,适合于所有用 Borland C++ 的 C 编写应用程序的读者。

- 书 名: Borland C++ 3.1 库函数手册
Borland C++ 3.1 KUHANSHU SHOUCHE
- 编 著 者: 李文通 刘天放 毕卫明 等编
- 组稿编辑: 杨昌竹
- 责任编辑: 曾昭奇
- 出 版 者: 北京航空航天大学出版社
- 地 址: 北京市海淀区学院路 37 号(100083)
- 印 刷 者: 通县觅子店印刷厂印刷
- 发 行: 新华书店总店科技发行所
- 经 售: 全国各地新华书店
- 开 本: 787×1092 1/16
- 印 张: 16
- 字 数: 409.6 千字
- 印 数: 5000 册
- 版 次: 1994 年 7 月第 1 版
- 印 次: 1994 年 7 月第 1 次印刷
- 书 号: ISBN 7-81012-513-3/TP·128
- 定 价: 16.00 元

目 录

第一章 C 库函数参考

abort	1	biosprint	21
abs	1	_bios_printer	22
absread	2	_bios_serialcom	22
abswrite	2	biostime	24
access	3	_bios_timeofday	24
acos,acosl	3	brk	25
alloca	4	bsearch	25
allocmem,_dos_allocmem	4	cabs,cabsl	26
arc	5	calloc	26
arg	5	ceil,ceil	27
asctime	6	_c_exit	27
asin,asinl	6	_cexit	27
assert	7	cgets	28
atan,atanl	7	_chain_intr	28
atan2,atan2l	8	chdir	29
atexit	8	_chdrive	29
atof,_atold	8	_chmod	29
atoi	9	chmod	30
atol	10	chsize	30
bar	10	circle	31
bar3d	10	_clear87	31
bcd	11	cleardevice	31
bdos	11	clearerr	32
bdosptr	11	clearviewport	32
bioscom	12	clock	32
_bios_disk	13	_close,close	32
biosdisk	15	closedir	33
biosequip	17	closegraph	33
_bios_equiplist	18	clreol	33
bioskey	19	clrscr	34
_bios_keybrd	20	complex	34
biosmemory	21	conj	34
_bios_memsiz	21	_control87	35
		coreleft	35
		cos,cosl	36

cosh, coshl	37	execvp, execvpe	56
country	37	_exit	58
cprintf	38	exit	58
cputs	38	exp, expl	59
_creat, _dos_creat	39	fabs, fabsl	59
creat	40	farcalloc	60
creatnew	40	farcoreleft	60
creattemp	41	farfree	60
cscanf	41	farheapcheck	61
ctime	42	farheapcheckfree	61
ctrlbrk	42	farheapchecknode	61
delay	43	farheapfillfree	62
delline	43	farheapwalk	62
detectgraph	43	farmalloc	62
difftime	45	farrealloc	63
disable, _disable, enable, _enable	45	fclose	63
div	45	fcloseall	63
_dos_close	46	fcvt	64
_dos_creatnew	46	fdopen	64
dosexterr	47	feof	65
_dos_findfirst	47	ferror	65
_dos_findnext	48	fflush	65
_dos_getdiskfree	49	fgetc	66
_dos_getdrive, _dos_setdrive	49	fgetchar	66
_dos_getfileattr, _dos_setfileattr	50	fgetpos	66
_dos_getftime, _dos_setftime	50	fgets	66
_dos_gettime, _dos_settime	51	filelength	67
_dos_getvect	51	fileno	67
_dos_setvect	52	fillellipse	67
_dos_write	52	fillpoly	68
dostounix	53	findfirst	68
drawpoly	53	findnext	69
dup	53	floodfill	70
dup2	54	floor, floorl	70
ecvt	54	flushall	70
ellipse	55	fmod, fmodl	71
_emit	55	fnmerge	71
eof	56	fnsplit	72
execl, execl, execlp, execlpe, execv, execve,		fopen	73

FP_OFF,FP_SEG	74	getdisk,setdisk	89
_fpreset	74	_getdrive	89
fprintf	74	getdrivename	89
fputc	75	getdta	89
fputchar	75	getenv	90
fputs	75	getfat	90
fread	75	getfatd	91
free	76	getfillpattern	91
freemem,_dos_freemem	76	getfillsettings	92
freopen	76	getftime,setftime	93
frexp,frexpl	77	getgraphmode	93
fscanf	77	getimage	94
fseek	78	getlinesettings	94
fsetpos	78	getmaxcolor	95
_fsopen	79	getmaxmode	95
fstat,stat	80	getmaxxx	95
_fstr *	81	getmaxy	96
ftell	81	getmodename	96
ftime	81	getmoderange	96
_fullpath	82	getpalette	97
fwrite	82	getpalettesize	97
gcvt	83	getpass	97
geninterrupt	83	getpid	98
getarccoords	83	getpixel	98
getaspectratio	84	getpsp	98
getbkcolor	84	gets	99
getc	84	gettext	99
getcbrk	85	gettextinfo	99
getch	85	gettextsettings	100
getchar	85	gettime,settime	100
getche	85	getvect,setvect	101
getcolor	86	getverify	101
getcurdir	86	getviewsettings	102
getcwd	86	getw	102
getdate,_dos_getdate,_dos_setdate,setdate	87	getx	102
_getdcwd	87	gety	103
getdefaultpalette	88	gmtime	103
getdfree	88	gotoxy	104
		graphdefaults	104

grapherrormsg	104	isprint	122
_graphfreemem	105	ispunct	123
_graphgetmem	105	isspace	123
graphresult	105	isupper	123
harderr,hardresume,hardretn	106	isxdigit	124
_harderr	107	itoa	124
_hardresume	108	kbhit	124
_hardretn	108	keep,_dos_keep	125
heapcheck	108	labs	125
heapcheckfree	109	ldexp,ldexpl	125
heapchecknode	109	ldiv	126
heapfillfree	109	lfind	126
heapwalk	110	line	127
highvideo	110	linerel	127
hypot,hypotl	111	lineto	127
imag	111	localeconv	127
imagesize	111	localtime	128
initgraph	112	lock	128
inp	114	locking	129
inport	114	log,logl	129
inportb	115	log10,log10l	130
inpw	115	longjmp	131
inline	115	lowvideo	131
installuserdriver	116	_lrotl	132
installuserfont	116	_lrotr	132
int86	117	lsearch	132
int86x	117	lseek	133
intdos	118	ltoa	133
intdosx	118	_makepath	134
intr	119	malloc	134
ioctl	119	matherr,_matherrl	135
isalnum	120	max	136
isalpha	120	mblen	136
isascii	121	mbstowcs	137
isatty	121	mbtowc	137
isctrl	121	memcpy,_fmemcpy	138
isdigit	122	memchr,_fmemchr	138
isgraph	122	memcmp,_fmemcmp	138
islower	122	memcpy, fmemcpy	139

memicmp, _fmicmp	139	putc	158
memmove	140	putch	159
memset, _fmemset	140	putchar	159
min	140	putenv	159
mkdir	141	putimage	160
MK_FP	141	putpixel	160
mktemp	141	puts	160
mktime	141	puttext	161
modf, modfl	142	putw	161
movedata	142	qsort	161
movmem	143	raise	162
moverel	143	rand	163
movetext	143	randbrd	163
moveto	143	randbwr	163
norm	144	random	164
normvideo	144	randomize	164
nosound	144	_read, _dos_read	164
_open, _dos_open	144	read	165
open	145	readdir	165
opendir	147	real	166
outp	147	realloc	166
outport, outportb	147	rectangle	167
outpw	148	registerbgidriver	167
outtext	148	registerbgifont	168
outtextxy	149	remove	168
_OvrInitEms	149	rename	168
_OvrInitExt	149	restorecrtmode	169
parsfnm	150	rewind	169
peek	150	rewinddir	169
peekb	150	rmdir	170
perror	151	rmtmp	170
pieslice	151	_rotr	170
poke	151	_rotr	170
pokeb	152	sbrk	171
polar	152	scanf	171
poly, polyl	152	_searchenv	176
pow, powl	153	searchpath	177
pow10, pow10l	154	sector	177
printf	154	segrad	178

setactivepage	178	sprintf	202
setallpalette	178	sqrt, sqrtl	202
setaspectratio	179	srand	203
setbkcolor	180	sscanf	203
setblock, _dos_setblock	180	_status87	203
setbuf	181	stime	203
setcbreak	181	strcpy	204
setcolor	182	strcat, _fstrcat	204
_setcursortype	182	strchr, _fstrchr	204
setdata	183	strcmp	205
setfillpattern	183	strcmpi	205
setfillstyle	183	strcoll	206
setgraphbufsize	184	strcpy	206
setgraphmode	184	strcspn, _fstrcspn	206
setjmp	185	_strdate	206
setlinestyle	186	strdup, _fstrdup	207
setlocale	186	_strerror	207
setmem	187	strerror	208
setmode	187	strftime	208
set_new_handler	187	stricmp, _fstricmp	209
setpalette	188	strlen, _fstrlen	209
setrgbpalette	189	strlwr, _fstrlwr	210
settextjustify	189	strncat, _fstrncat	210
settextstyle	190	strncmp, _fstrncmp	210
setusercharsize	191	strncmpi	211
setvbuf	191	strncpy, _fstrncpy	211
setverify	192	strnicmp, _fstrnicmp	212
setviewport	192	strnset, _fstrnset	212
setvisualpage	193	strpbrk, _fstrpbrk	212
setwritemode	193	strrchr, _fstrchr	213
signal	193	strrev, _fstrrev	213
sin, sinl	196	strset, _fstrset	213
sinh, sinh1	197	strspn, _fstrspn	214
sleep	197	strstr, _fstrstr	214
sopen	197	_strtime	214
sound	199	strtod, _strtold	215
spawnl, spawnle, spawnlp, spawnlpe, spawnv,		strtok, _fstrtok	215
spawnve, spawnvp, spawnvpe	199	strtol	216
splitpath	201	strtoul	217

strupr, _fstrupr	217	vprintf	232
strxfrm	217	vscanf	232
swab	217	vsprintf	233
system	218	vsscanf	233
tan, tanl	218	wcstombs	233
tanh, tanhl	219	wctomb	234
tell	219	wherex	234
tempnam	220	wherey	234
textattr	220	window	235
textbackground	221	_write	235
textcolor	222	write	236
textheight	223	第二章 全局变量	
textmode	223	_8087	237
textwidth	224	_argc	237
time	224	_argv	237
tmpfile	224	_ctype	237
tmpnam	225	daylight	238
toascii	225	directvideo	238
_tolower	225	environ	238
tolower	226	errno, _doserrno, sys_errlist, sys_nerr ...	239
_toupper	226	_fmode	240
toupper	226	_heaplen	241
tzset	226	_new_handler	241
ultoa	227	_osmajor, _osminor	242
umask	227	_ovrbuffer	242
ungetc	228	_psp	243
ungetch	228	_stklen	243
unixtodos	228	timezone	243
unlink	229	tzname	244
unlock	229	_version	244
utime	229	_wscroll	244
va_arg, va_end, va_start	230		
vfprintf	231		
vscanf	231		

第一章 C 库函数参考

DOS ✓	UNIX ✓	Windows ✓
函 数 名 函数的功能		
ANSI C ✓		C++ ×

上面是函数名和函数功能概述,如果函数适用于 DOS、UNIX、Windows、ANSI C 和/或 C++,则在其后跟有✓;否则,在其后用×表示。上面的函数适用于 DOS、UNIX、Windows 和 ANSI C,但不适用 C++。

用 法 #include <header.h>

本部分列出包含函数原型的头文件或定义函数用到的常量、枚举类型等等的头文件。

function (modifier parameter [, ...]);

本部分给出函数的用法。[, ...]代表其它参数,而它们的修饰符可以跟随其后。

说 明 本部分说明函数的作用,采用的参数和使用函数的细节,并列出了相关子程序。

返 回 值 函数的返回值在这里给出。如果函数设置了全局变量,则也列出它们的值。

参 见 在这里列出了与本函数相关的子程序。如果所列的该子程序名都已经包含省略号(funcname...funcname,func...name),则是指函数族(例如,exec...指的是 exe 函数族项:execl, execl, execlp, execlpe, execv, execve, execvp 和 ex-
ecvpe)。

DOS ✓	UNIX ✓	Windows ✓
abort 异常终止一进程		
ANSI C ✓		C++ ×

用 法 #include <stdlib.h>

void abort (void);

说 明 本函数向 stderr 写终止信息"Abnormal program termination",接着调用带终止码 3 的_exit 来终止程序。

返 回 值 abort 将终止码 3 返回给父进程或 DOS。

参 见 assert, atexit, exit, _exit, raise, signal, spawn...

DOS ✓	UNIX ✓	Windows ✓
实数 abs 返回整型数的绝对值		
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
实数 abs 返回整型数的绝对值		
ANSI C ×		C++ ✓

- 用 法** 实数版本 复数版本
 #include <math.h> include <complex.h>
 int abs (int x); double abs (complex x);
- 说 明** abs 返回整型参数 x 的绝对值。如果包含 stdlib.h 头文件,则调用 abs 时,abs 将作为可扩展为插入代码的宏看待。如果想用 abs 函数代替宏,就要在 #include <stdlib.h> 后面包含 #undef abs。
- 返 回 值** abs 的实型版本返回了 0 到 32767 间的一个整数,但也有一个例外:当参数值为 -32768 时,返回 -32768。abs 的复数版本返回 double 型值。
- 参 见** cabs, complex, fabs, labs

DOS ✓	UNIX ×	Windows ×
absread 读取磁盘绝对扇区		
ANSI C ×		C++ ×

- 用 法** #include <dos.h>
 int absread (int drive, int nsects, long lsect, void * buffer);
- 说 明** absread 读取磁盘扇区内容。它忽略磁盘的逻辑结构,不管是文件,FAT 还是目录。
 absread 通过 DOS 中断 0x25 读取指定磁盘扇区内容。
 drive = 要读取的驱动器号(0=A,1=B,等等)
 nsects = 要读取的扇区数
 lsect = 起始逻辑扇区号
 buffer = 要读取的数据的内存地址
 读取的扇区数限制为 64K 或者小于等于缓冲区的大小。
- 返 回 值** 如果调用成功,absread 返回 0。
- 参 见** abswrite, biosdisk.

DOS ✓	UNIX ×	Windows ×
abswrite 写磁盘绝对扇区		
ANSI C ×		C++ ×

- 用 法** #include <dos.h>
 int abswrite (int drive, int nsects, long lsect, void * buffer);
- 说 明** abswrite 写指定磁盘扇区,它忽略磁盘的逻辑结构,不管是文件,FAT 还是目录。
 注意:如果使用不合理,abswrite 可能会覆盖文件,目录和 FAT。abswrite 通过调用 DOS 中断 0x26 写内容到指定的磁盘扇区。
 drive = 要写的驱动器号(0=A,1=B,等等)
 nsects = 要写的扇区数
 lsect = 起始逻辑扇区号
 buffer = 要写入数据的内存起始地址

要写的磁盘扇区限制为 64K 或者小于等于缓冲区的大小。

返回值 如果调用成功, `abswrite` 返回 0。如果出错, 返回 -1, 并置全局变量 `errno` 为系统调用后返回的 AX 寄存器的值。

参 见 `absread`, `biosdisk`

DOS ✓	UNIX ✓	Windows ✓
access 确定文件的存取权限		
ANSI C ×		C++ ×

用 法 `#include <io.h>`

```
int access (const char * filename, int amode);
```

说 明 `access` 检查由 `filename` 所命名的文件是否存在, 以及它是否可读, 可写或可执行。

`amode` 的值列表如下:

06 检查读/写允许

04 检查读允许

02 检查写允许

01 执行(被忽略)

00 检查文件是否存在

注意: 在 DOS 下, 所有存在的文件具有可读性(`amode` 等于 04), 所以 00 和 04 有同样的结果。同理, `amode` 值 06 和 02 是等价的, 因为在 DOS 下, 写访问隐含了可读性。

如果 `filename` 指向一目录, `access` 仅确定该目录是否存在。

返回值 如果允许所请求的存取, 则 `access` 返回 0; 否则返回 -1, 并且全局变量 `errnu` 被置为下列值之一:

ENOENT 没有找到路径或文件名

EACCES 无比权限

参 见 `chmod`, `fstat`, `stat`

DOS ✓	UNIX ×	Windows ✓
acosl 计算反余弦值		
ANSI C ×		C++ ×
DOS ✓	UNIX ✓	Windows ✓
实数 acos 计算反余弦值		
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
复数 acos 计算反余弦值		
ANSI C ×		C++ ✓

用 法 实数版本

复数版本

```
#include <math.h>
```

```
double acos(double x);
```

```
long double acosl(long double x);
```

```
#include <complex.h>
```

```
complex acos(complex x);
```

用法 `acos` 返回输入值的反余弦值。`acosl` 是长双精度版本, 它以一个长双精度值为参数并返回一个长双精度值。传给 `acos` 和 `acosl` 的实参数必须在 -1 到 1 之间, 否则 `acos` 和 `acosl` 返回 NAN, 并且置全局变量 `errno` 为:

EDOM 域错误

复数的反余弦定义为:

$$\operatorname{acos}(z) = -i \log(z + i \sqrt{1 - z^2})$$

返回值 传给 `acos` 和 `acosl` 一个 -1 到 +1 之间的实参数, 则返回值在 0 到 π 之间。函数的错误处理程序可以通过 `matherr` 和 `_matherrl` 函数修改。

参见 `asin`, `atan`, `atan2`, `complex`, `cos`, `matherr`, `sin`, `tan`

DOS ✓	UNIX ✓	Windows ✓
	<code>alloca</code> 分配临时堆栈空间	
ANSI C ×		C++ ×

用法

```
#include <malloc.h>
```

```
void *alloca (size_t size);
```

说明 `alloca` 分配堆栈空间, 以字节为单位; 当调用函数终止时, 分配的空间自动释放。因为 `alloca` 修改堆栈指针, 所以在函数的参数表达式中不放置对 `alloca` 的调用。如果调用函数不提供堆栈中局部变量的任何信息, 则当函数终止时, 堆栈不能被正确存储, 从而引起程序崩溃。为确保正确地存储堆栈, 在调用函数中使用下列代码:

```
char *p
char dummy[1]
dummy[0]=0;
...
p=alloca(nbytes);
```

返回值 如果堆栈空间足够大, 则 `alloca` 返回一个指向所分配栈空间的指针。否则, 返回 NULL。

参见 `malloc`

DOS ✓	UNIX ✓	Windows ✓
	<code>allocmem</code> , <code>_dos_allocmem</code> 分配 DOS 内存段	
ANSI C ✓		C++ ✓

用法

```
#include <dos.h>
```

```
int allocmem(unsigned size, unsigned *segp);
```

```
unsigned dos_allocmem(unsigned size, unsigned *segp);
```

说明 `allocmem` 和 `_dos_allocmem` 使用 DOS 系统调用 `0x48` 来分配自由内存块, 并返回所分配块的段地址。size 是请求的以段为单位的内存大小(一段是 16 字节)。`segp` 是一个指针, 它指向将被赋值为新分配块段地址的地址字。

对于 `allocmem`, 如果没有足够的空间, 则 `segp` 所指向的地址字没有赋值。

对于 `_dos_allocmem`, 如果没有足够的空间, 则最大的可分配块的大小将放在 `segp` 所指向的地址字。

注意: `allocmem` 和 `_dos_allocmem` 不能与 `malloc` 并存。

返回值 如果调用成功, `allocmem` 返回 -1。如果出现错误, `allocmem` 返回段中代表最大可分配块的大小。如果调用成功, `_dos_allocmem` 返回 0。如果出现错误, `_dos_allocmem` 返回 DOS 错误代码, 并设置 `segp` 所指向的地址字为段中最大可用块的大小。

`allocmem` 或 `_dos_allocmem` 的出错返回将置全局变量 `_doserrno` 和 `errno` 为:

ENOMEM 没有足够的内存

参见 `coreleft`, `freemem`, `malloc`, `setblock`

DOS ✓	UNIX ×	Windows ×
	arc 画一圆弧	
ANSI C ×		C++ ×

用法 `#include <graphics.h>`

`void far arc (int x, int y, int stangle, int endangle, intradius);`

说明 `arc` 以 (x,y) 为圆心, 以 radius 为半径, 按当前绘图颜色画一圆弧。arc 轨迹从起始角 stangle 到终止角 endangle。如果 stangle=D 且 endangle=360, 则调用 `arc` 将画出一个整圆。

`arc` 的角度是逆时针方向的, 0 度在 3 点钟, 90 度在 12 点钟等。

说明: `linestyle` (线型) 参数对弧、圆、椭圆或圆饼图无影响。仅用到了 `thickness` 参数。

注意: 如果用户使用的是 CGA 的高分辨率模式或是单色图形适配器, 本书中关于如何使用图形函数的样例程序可能得不到预期的结果。如果系统采用的是 CGA 或单色适配器, 要把值 1 传给这些函数以改变填充或绘制颜色 (例如, `setcolor`, `setfillstyl` 和 `setlinestyle`), 而不要使用符号颜色常数 (在 `graphics.h` 中定义)。

返回值 无

参见 `circle`, `ellipse`, `fillellipse`, `getarccoords`, `getaspectratio`, `graphresult`, `pieslice`, `sector`

DOS ✓	UNIX ×	Windows ✓
	arg 求复平面中一个复数的角度	
ANSI C ×		C++ ✓

用法 `#include <complex.h>`

double arg (complex x);

说 明 arg 给出复平面中某个复数以弧度为单位的角度值。

正实轴的弧度值为 0, 正虚轴的弧度值为 $\pi/2$ 。如果输入参数是复数 0, 则 arg 返回 0。

返 回 值 arg(x) 返回 atan2(imag(x), real(x))。

参 见 complex, norm, polar

DOS ✓	UNIX ✓	Windows ✓
asctime 将日期和时间转换为 ASCII 码		
ANSI C ✓		C++ ×

用 法 #include <time.h>

char * asctime (const struct tm * tblock);

说 明 asctime 把以结构形式存储在 * tblock 里的时间转换为同 ctime 字符串中形式相对应的 26 个字符的字符串:

Sun Sep 16 D1:03:52 1973\n\o

返 回 值 asctime 返回指向包含日期和时间字符串的指针。该字符串是静态的, 每次调用 asctime 都被重写。

参 见 ctime, difftime, ftime, gmtime, localtime, mktime, strftime, stime, time, tzset

DOS ✓	UNIX ×	Windows ✓
asinh 计算反正弦值		
ANSI C ×		C++ ×
DOS ✓	UNIX ✓	Windows ✓
实数 asin 计算反正弦值		
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
复数 asin 计算反正弦值		
ANSI C ×		C++ ✓

用 法 实数版本:

#include <math.h>

double asin (double x);

long double asinh (long double x);

复数版本

#include <complex.h>

complex asin (complex x);

说 明 asin 返回输入值的反正弦值。

asinh 是长双精度版本; 它以一个长双精度值为参数并返回一个长双精度值。传给 asin 和 asinh 的实参数必须在 -1 到 1 之间, 否则 asin 和 asinh 返回 NAN, 并置全局变量 errno 为:

EDOM 域错误

复数的反正弦定义为:

$$\operatorname{asin}(z) = -i * \log(i * Z + \sqrt{1 - z})$$

返回值 传给 `asin` 和 `asinl` 的实参数的返回值在 $-\pi/2$ 到 $\pi/2$ 之间。它们的错误处理程序可以通过 `matherr` 和 `-matherrl` 修改。

参 见 `acos, atan, atan2, complex, cos, matherr, sin, tan`

DOS ✓	UNIX ✓	Windows ✓
assert 测试条件并可能使程序终止		
ANSI C ✓		C++ ×

用 法 `#include <assert.h>`
`void assert (int test);`

说 明 `assert` 是一个可扩展为 `if` 语句的宏。如果 `test` 值为 0, 则 `assert` 将在 `stderr` 中输出一信息并终止程序(通过调用 `abort`)。

`assert` 输出的信息为:

Assertion failed; test, file filename, line linenum

其中 `filename` 和 `linenum` 是源文件名和 `assert` 宏出现的行号。

如果在源文件代码的 `#include <assert.h>` 指令之前放置 `#define NDEBUG` 指令, 其作用是对 `assert` 进行注释。

返回值 无

参 见 `abort`

DOS ✓	UNIX ×	Windows ✓
atanl 计算反正切		
ANSI C ×		C++ ×
DOS ✓	UNIX ✓	Windows ✓
实数 atan 计算反正切		
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
复数 atan 计算反正切		
ANSI C ×		C++ ✓

用 法 实数版本

`#include <math.h>`

`double atan (double x);`

`long double atanl (long double x);`

复数版本

`#include <complex.h>`

`complex atan (complex x)`

说 明 `atan` 计算输入值的反正切值。

`atanl` 是长双精度版本; 它以一个长双精度值为参数并返回一个长双精度值。复数的反正切定义为:

$$\operatorname{atan}(z) = -0.5 i \log((1 + iz)/(1 - iz))$$

返回值 对 `atan` 和 `atanl` 输入实参数时返回值在 $-\pi/2$ 和 $\pi/2$ 之间。它们的错误处理程序

可通过 `matherr` 和 `_matherrl` 函数修改。

参 见 `acos`, `asin`, `atan2`, `complex`, `cos`, `matherr`, `sin`, `tan`

DOS ✓	UNIX ✓	Windows ✓
	<code>atan2</code> 计算 y/x 的反正切值	
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
	<code>atan2l</code> 计算 y/x 的反正切值	
ANSI C ×		C++ ×

用 法 `#include <math.h>`

`double atan2 (double y, double x);`

`long double atan2l (long double y, long double x);`

说 明 `atan2` 返回 y/x 的反正切值。即使结果接近 $\pi/2$ 或 $-\pi/2$ (即 x 接近 0) 仍能得到正确结果。

如果 x 和 y 都设置为 0, 则本函数置全局变量 `errno` 为 `EDOM`。

`atan2l` 是长双精度版本; 它接受长双精度参数并返回一长双精度值。

返 回 值 `atan2` 和 `atan2l` 的返回值在 $-\pi$ 和 π 之间。

这两个函数的错误处理程序可以通过 `matherr` 和 `_matherrl` 修改。

参 见 `acos`, `asin`, `atan`, `cos`, `matherr`, `sin`, `tan`

DOS ✓	UNIX ×	Windows ✓
	<code>atexit</code> 注册终止函数	
ANSI C ✓		C++ ×

用 法 `#include <stdlib.h>`

`int atexit (void (* func) (void));`

说 明 `atexit` 注册由 `func` 指定的函数作为 `exit` 函数。在程序正常终止情况下, `exit` 返回操作系统前调用 `(* func)()`。

每次 `atexit` 调用注册另一个 `exit` 函数。最多可以注册 32 个函数。它们按后进先出的顺序执行 (即最后注册的函数最先执行)。

返 回 值 `atexit` 调用成功时返回 0。否则返回非零值 (没有足够的空间注册函数)。

参 见 `abort`, `_exit`, `exit`, `spawn...`

DOS ✓	UNIX ✓	Windows ✓
	<code>atof</code> 把一字符串转换成浮点数	
ANSI C ✓		C++ ×
DOS ✓	UNIX ×	Windows ✓
	<code>_atold</code> 把一字符串转换成浮点数	
ANSI C ×		C++ ×