第一章　运行库交叉参考

本章提供 Borland C++库函数例程和头文件的概览。库例程由类、函数和宏组成，用这些例程在 C 和 C++程序中能完成众多的标准服务，包括低级和高级的 I/O、字符串和文件处理、内存分配、进程控制、数据转换和数学计算等等。

本章提供下面信息：

- 静态和动态链接库名字、在 LIB 和 BIN 下的文件、子目录，并将讨论用法。
- 解释 Borland C++运行时库的源代码有何用处。
- 列出和描述头文件。
- 按服务类型为库例程分类。

1.1　直接使用运行时库源代码的理由

用户会由于下述原因需要使用运行时库例程源代码：

- 用户要写的函数可能与某些 Borland C++函数很相似，但并不完全一样。有了运行时库源代码，可以修改库函数以适合用户的需要，可以避免重写一个全新的代码。
- 当调试代码，有时需要了解库函数的内部细节。
- 如果想删除 C 符号的前导下划线，可以修改运行时库源代码。
- 有些程序员想研究一个库函数源代码。

1.2　运行时库

运行时库分成静态(OBJ 和 LIB)和动态链接库(DLL)。这些不同的库放在不同的子目录中。静态和动态库在独立的表的描述。

运行时库的版本有很多，比如依内存模式、诊断版本和 16 和 32 位可分成不同的库文件。还有一些可选的库，包括数学、容器、ObjectWindows 开发和各国通用应用程序。

下面是运行时库选择的指南：

- 段相关的内存模式库只支持 16 位程序。微和巨内存模式不支持。
- 16 位 DLL 只支持大内存模式。
- 对于 32 位程序，只支持 flat 内存模式。

- 32 位和 GUI 程序需要不同的启动代码。
- 多线程应用程序只支持 32 位程序。

1.2.1 静态库

Borland C++运行库的静态(OBJ 和 LIB)库包含在 LIB 子目录,对于每个库文件名,"?"字符代码四个不同内存模式(紧缩、小、中和大)中的一种。每种模式有它自己的库文件和数学库文件。

下面的表指出每个编译器使用的缺省运行时。参见《Borland C++ 4.0 百科大全》一书中关于编译和连接的讨论。

表 1.1 缺省运行时库

编译器	应用程序	缺省库
BC4\LIB 目录		
BCC.EXE	16 位 Windows	C0WS. OBJ, CWS. LIB, MATHWS.LIB,IMPORT.LIB
BCC32.EXE	Win32	C0X32. OBJ, CW32. LIB, IMPORT32.LIB
BCW.EXE	和 BCC.EXE 相同	和 BCC.EXE 相同
BCWS32.EXE	和 BCC32.EXE 相同	和 BCC32.EXE 相同

表 1.2 列出了 Borland C++静态库的名字和用法。它列出了某一种操作系统有何种库;参见《Borland C++ 4.0 百科大全》有关连接器、连接器选项、需求和库选择的信息。

表 1.2 静态运行时库小结

文件名	应用程序	用法
BC4\LIB 目录		
BIDSDL.LIB	Win 16	BIDS40D.DLL 的 16 位诊断、动态 BIDS 导入库。
BIDSDL.LIB	Win 16	BIDS40.DLL 的 16 位动态 BIDS 导入库。
BIDSF.LIB	Win32s,Win32	32 位 BIDS 库。
BIDSDF.LIB	Win32s,Win32	32 位诊断 BIDS 库。
BIDSFI.LIB	Win32s,Win32	BIDS40DF.DLL 的 32 位诊断、动态 BIDS 导入库。
BIDSDB?.lib	Win 16	16 位诊断 BIDS 库。
BIDS?.LIB	Win 16	16 位 BIDS 库。
BWCC.LIB	Win 16	BWCC32.DLL 的 16 位导入库。
BWCC32.LIB	Win32s,Win32	BWCC32.DLL 的 32 位导入库。
C0D32.OBJ	Win32s,Win32	32 位 DLL 启动模块。
C0W32.OBJ	Win32s,Win32	32 位 GUI.EXE 启动代码。
C0W?.OBJ	Win16	16 位 DLL 启动模块。

C0X32.OBJ	Win32	32 位控制台模式 EXE 启动模块。
CRTDLL.LIB	Win 16	BC40RTL.DLL 的 16 位动态导入库。
CW32.LIB	Win32s,Win32	32 位 GUI 单线程库。
CW?.LIB	Win 16	16 位库。
CW32I.LIB	Win32s,Win32	CW32.DLL 的 32 位单线程、GUI、动态 RTL 导入库。
CW32MT.LIB	Win32	32 位 GUI 多线程库。
CW32MTI.LIB	Win32	CW32MT.DLL 的 32 位多线程、GUI、动态 CTL 导入库。
IMPORT.LIB	Win16	Windows 3.1 的 16 位导入库。
IMPORT32.LIB	Win32s,Win32	和 IMPRTW32.LIB 一起使用的 32 位导入库。
MATHW?.LIB	Win 16	16 位数学库。
W32SUT16.LIB	Win 16	16 位大学用库。
S32SUT32.LIB	Win32s	32 位大学用库。
OBSOLETE.LIB	Win16,Win32,Win32s	提供取消的全局变量。
BC4\LIB\16-BIT		
目录		
FILES.C	Win 16	增加文件句柄数。
FILES2.C	Win 16	增加文件句柄数。
MATHERR.C	Win 16	用户定义的 float 和 double 浮点类型数学异常处理程序的例子。
MATHERRL.C	Win 16	用户定义的 long double 浮点类型数学异常处理程序的例子。
BC4\LIB\32-BIT		
目录		
FILES.C	Win32,Win32	增加文件句柄数。
FILES2.C	Win32,Win32	增加文件句柄数。
FILEINFO.OBJ	Win32s,Win32	传递打开文件句柄信息给子进程。
GP.OBJ	Win32s,Win32	当出现异常错误时打印寄存器转储信息。
MATHERR.C	Win 16	用户定义的 float 和 double 浮点类型数学异常处理程序的例子。
MATHERRL.C	Win 16	用户定义的 long double 浮点类型数学异常处理程序的例子。
WILDARGS.OBJ	Win32	转换通配参数到控制台模式应用程序的 main 参数数组中。

BC4 \ LIB \ STARTUP 目录		
BUILD-C0. BAT	Win 16	建立 C0D?. OBJ, C0F?. OBJ 和 C0W?. OBJ 的批处理文件。
C0D. ASM	Win 16	C0D?. OBJ 的源代码。
C0W. ASM	Win 16	C0W?. OBJ 的源代码。
RULES. ASI	Win 16	C0D. ASM 和的 C0W. ASM 的汇编规则文件。

1.2.2 动态链接库

运行时库的动态链接库(DLL)版本包含在 BIN 子目录中,有几种 DLL 版本。比如,有诊断类型、有 16 和 32 位相关的版本和支持多线程应用程序的版本。

在 16 位专用的版本中,只有大内存模式 DLL。其它的内存模式不支持。

表 1.3 列出了 Borland C++ DLL 名字和用处,在何种操作系统中适用。参见《Borland C++ 4.0 百科大全》关于连接器、连接器选项、需求和库选择的信息。

表 1.3 动态链接库小结

文件名	应用程序	用处
BC4\BIN 目录		
BC40RTL. DLL	Win 16	16 位,大内存模式
BIDS40. DLL	Win 16	16 位,BIDS
BID40D. DLL	Win 16	16 位,诊断 BIDS
BIDS40F. DLL	Win32s,Win32	32 位 BIDS
BIDS40DF. DLL	Win32s,Win32	32 位、诊断 BIDS
CW32. DLL	Win32s,Win32	32 位、单线程
CW32MT. DLL	Win32	32 位多线程
LOCALE. BIL	WIn32s,Win32	场所库

1.3 Borland C++头文件

C++头文件,以及 ANSI C 定义的头文件在 margin 中说明。

头文件也称为包含文件,提供了库函数的函数原型说明。其中还定义了数据类型、符号常量、Borland C++以及库函数定义的全局变量。Borland C++库在头文件命名及其内容上遵循 ANSI C 标准:

	alloc. h	声明内存管理函数(分配,释放等)。
ANSI C	assert. h	定义 assert 调试宏。
C++	bcd. h	声明 C++的 bcd 类以及 bcd 的重载操作符和数学函数

	bios. h	声明用于调用 IBM PC ROM BIOS 子程序的各种函数
C++	checks. h	定义 PRECONDITION，WARN 和 TRACE 诊断宏
C++	complex. h	声明 C++ 的复数数学函数。
	conio. h	声明用于调用 DOS 控制台 I/O 子程序的各种函数。
C++	constrea. h	声明支持控制台输出的 C++ 类型和媒介。
C++	cstring. h	声明 ANSI C++ 字符串类支持
ANSI C	ctype. h	包含有关字符分类和字符转换宏。(如 isalpha 和 toascii)
	dir. h	包含与目录和路径名有关的结构宏和函数。
	direct. h	为处理目录和路径名所定义的结构宏和函数。
	dirent. h	为 POSIX 目录操作声明的函数和结构。
	dos. h	定义为 DOS 和专用于 8086 调用所需的各种常量和说明。
ANSI C	errno. h	定义错误代码的常量助记符。
C++	excep. h	声明 ANSI C++ 异常支持。
	except. h	声明 C 异常支持。
	fcntl. h	定义用于与库子程序 open 连接的符号常量。
ANSI C	float. h	包含浮点子程序的参数。
C++	fstream. h	声明支持 C++ 输入和输出的 C++ 流的种类。
C++	generic. h	包含类说明的宏。
	graphics. h	包含图形函数包的各种子程序的原型。
	io. h	包含低级输入/输出子程序的结构和说明。
C++	iomanip. h	声明 C++ I/O 流的操作符，并包含创建参量化操作符的宏。
C++	iostream. h	声明基本 C++(2.0 版本)流(I/O)子程序。
ANSI C	limits. h	包含环境参量，编译时间限制信息以及整型量范围。
ANSI C	local. h	声明有关可以提供国家与语言专用信息的函数。
	sys\locking. h	定义 locking 函数的 mode 参数。
	malloc. h	存储管理的函数和变量。
ANSI C	math. h	声明数学函数原型，和数学错误处理子程序。
	mem. h	声明内存操作函数(其中许多函数在 string. h 中定义)。
	memory. h	内存操作函数。
C++	new. h	存取操作数 new 和 newhandler。
	process. h	包含 spwn…和 exec…函数的结构和声明。
C++	ref. h	提供引用计数的支持。和字符串类一起使用。
C++	regexp. h	实现正则表达式搜索。
	search. h	声明搜索和排序函数
ANSI C	setjmp. h	定义 longjmp 和 setjmp 函数所用的 jmp _buf 类型。并声明子程序 longjmp 和 setjmp。
	share. h	定义函数中用于文件共享的参数
ANSI C	signal. h	定义用于 signal 和 raise 函数所用的常量和说明。
ANSI C	stdarg. h	定义用于读取说明为参数可变的函数(如 vprint，vscanf 等)参数表的宏。
ANSI C	stddef. h	定义几个公用的数据类型和宏

ANSI C	stdio.h	定义 Kernighan 和 Ritchie 及在 UNIX 系统 V 下扩充的标准 I/O 函数包所需的类型和宏。定义标准 I/O 预定义流 stdin，stdout，stdprn 和 stderr，并说明 I/O 流子程序。
C++	stdiostr.h	定义和 stdio 的 FILE 结构一起使用的 C++(2.0 版本)流类。
ANSI C	stdlib.h	声明几个公用的子程序，转换子程序、搜索/排序子程序及其它子程序。
ANSI C	string.h	声明几个串操作和内存操作子程序。
C++	strstrea.h	声明和内存字节数组一道使用的 C++流类。
	sys\stat.h	用于打开和创建文件的符号常量
ANSI C	time.h	定义由时间转换子程序 asctime，localtime 和 gmtime 填充的结构以及子程序 ctime，deffrime，gmtime，localtime 和 stime 所用的类型，还提供了这些子程序的原型。
	sys\timeb.h	声明 ftime 函数和 ftime 返回的结构 timeb。
	sys\type.h	声明时间函数使用的 time_t 的类型
	values.h	声明 utime 函数和它返回的结构 utimbuf。
	utime.h	定义在 UNIX 系统 V 兼容的重要常量，包括机器依赖性。
	varargs.h	定义带有参变量的函数的存取参数。与 UNIX 兼容；对于新代码，用户可使用 stdarg.h。

1.4 库子程序分类

Borland C++库子程序执行各种任务。本节我们按其所执行的任务进行分类，并给出声明它们的包含文件。第三章将给出下表中函数的完整信息。

1.4.1 分类子程序

此类子程序将 ASCII 字符分为字母控制符，标点符号，大写字母等。

isalnum	(ctype.h)	isdigit	(ctype.h)	ispunct	(ctype.h)
isalpha	(ctype.h)	isgraph	(ctype.h)	isspace	(ctype.h)
isascii	(ctype.h)	islower	(ctype.h)	isupper	(ctype.h)
iscntrl	(ctype.h)	isprint	(ctype.h)	isxdight	(ctype.h)

1.4.2 转换子程序

此类子程序用于字符和串的字符形式的不同的数值表示(浮点数，整型，长整型)之间，大写与小写之间的转换。

atof	(stdlib.h)	itoa	(stdlib.h)	toascii	(ctype.h)

atoi	(stdlib.h)	_strdate	(stdlib.h)	_tolower	(ctype.h)
atol	(stdlib.h)	_strtime	(time.h)	tolower	(ctype.h)
ecvt	(stdlib.h)	strtod	(stdlib.h)	_toupper	(ctype.h)
fcvt	(stdlib.h)	strtol	(stdlib.h)	toupper	(ctype.h)
gcvt	(stdlib.h)	_strtold	(stdlib.h)	ultoa	(stdlib.h)
itoa	(stdlib.h)	strtoul	(stdlib.h)		

1.4.3　目录控制子程序

此类子程序用于对目录和路径名的操作

chdir	(dir.h)		
_chdrive	(direct.h)	_fullpath	(stdlib.h)
closedir	(direct.h)	getcurdir	(dir.h)
_dos_findfirst	(dos.h)	getcwd	(dir.h)
_dos_findnext	(dos.h)	_getdcwd	(direct.h)
_dos_getdiskfree	(dos.h)	getdisk	(dir.h)
_dos_getdrive	(dos.h)	_getdrive	(direct.h)
_dos_setdrive	(dos.h)	_makepath	(stdlib.h)
findfirst	(dir.h)	mkdir	(dir.h)
findnext	(dir.h)	mktemp	(dir.h)
fnmerge	(dir.h)	opendir	(dirent.h)
fnsplit	(dir.h)	readdir	(dirent.h)
rmdir	(dir.h)	rewinddir	(dirent.h)
_searchenv	(stdlib.h)	setdisk	(dir.h)
searchpath	(dir.h)	_splitpath	(stdlib.h)

1.4.5　诊断子程序

此类子程序提供内部故障检查能力。

assert	(assert.h)
matherr	(math.h)
_matherrl	(math.h)
perror	(errno.h)

1.4.6　图形子程序

此类子程序用于在屏幕上画出图形

arc	(graphics.h)	getpalettesize	(graphics.h)
bar	(graphics.h)	getpixel	(graphics.h)
bar3d	(graphics.h)	gettextsettings	(graphics.h)
circle	(graphics.h)	getviewsettings	(graphics.h)

cleardevice	(graphics. h)	getx	(graphics. h)
clearviewport	(graphics. h)	gety	(graphics. h)
closegraph	(graphics. h)	graphdefaults	(graphics. h)
detectgraph	(graphics. h)	grapherrormsg	(graphics. h)
drawpoly	(graphics. h)	_graphfreemem	(graphics. h)
ellipse	(graphics. h)	_graphgetmem	(graphics. h)
fillellipse	(grophics. h)	graphresult	(graphics. h)
fillpoly	(graphics. h)	imagesize	(graphics. h)
floodfill	(graphics. h)	initgraph	(graphics. h)
getarccoords	(graphics. h)	installuserdriver	(graphics. h)
getaspectratio	(graphics. h)	installuserfont	(graphics. h)
getbrcolor	(graphics. h)	line	(graphics. h)
getcolor	(graphics. h)	linerel	(graphics. h)
getdefaultpulette	(graphics. h)	lineto	(graphics. h)
getdrivername	(graphics. h)	moverel	(graphics. h)
getfillputtern	(graphics. h)	moveto	(graphics. h)
getfillsettings	(graphics. h)	outtext	(graphics. h)
getgraphmode	(graphics. h)	outtextxy	(graphics. h)
getimage	(graphics. h)	pieslice	(graphics. h)
getlinesettings	(graphics. h)	putimage	(graphics. h)
getmaxcolor	(graphics. h)	putpixel	(graphics. h)
getmaxmode	(graphics. h)	rectangle	(graphics. h)
getmaxx	(graphics. h)	registerbgidriver	(graphics. h)
getmaxy	(graphics. h)	registerbgifont	(graphics. h)
getmodename	(graphics. h)	restorecrtmode	(graphics. h)
getmoderange	(graphics. h)	sector	(graphics. h)
getpalette	(graphics. h)	setactivepage	(graphics. h)
setallpalette	(graphics. h)	setpalette	(graphics. h)
setaspectratio	(graphics. h)	setrgbpalette	(graphics. h)
setbkcolor	(graphics. h)	settextjustify	(graphics. h)
setcolor	(graphics. h)	settextstyle	(graphics. h)
setcursortype	(conio. h)	setusercharsize	(graphics. h)
setfillpattern	(graphics. h)	setviewport	(graphics. h)
setfillstyle	(graphics. h)	setvisualpage	(graphics. h)
setgraphbufsize	(graphics. h)	setwritemode	(graphics. h)
setlinestyle	(graphics. h)	textwidth	(graphics. h)

1.4.7 嵌入子程序

此类程序有嵌入版本。当用户使用#pragma intrinsic 或如果用户确定程序优化时，编译程序将产生直接插入版本的代码。

fabs	(math. h)	strcmp	(string. h)

memchr	(mem. h)	strcpy	(string. h)
memcmp	(mem. h)	strlen	(string. h)
memcpy	(mem. h)	strncat	(string. h)
_rotl	(stdlib. h)	strncmp	(string. h)
_rotr	(stdlib. h)	strncpy	(string. h)
stpcpy	(string. h)	strnset	(string. h)
strcat	(string. h)	strset	(string. h)

1.4.8　输入/输出子程序

此类子程序提供流和 DOS 的 I/O 操作功能。

access	(io. h)	_dos_creat	(dos. h)
cgets	(conio. h)	_dos_creatnew	(dos. h)
_chmod	(io. h)	_dos_getfileattr	(dos. h)
chmod	(io. h)	_dos_getftime	(dos. h)
chsize	(io. h)	_dos_open	(dos. h)
clearerr	(io. h)	_dos_read	(dos. h)
_close	(io. h)	_dos_setfileattr	(dos. h)
close	(io. h)	_dos_setftime	(dos. h)
cprintf	(conio. h)	_dos_write	(dos. h)
cputs	(conio. h)	dup	(io. h)
_creat	(io. h)	dup2	(io. h)
creat	(io. h)	eof	(io. h)
creatnew	(io. h)	fclose	(stdio. h)
creattemp	(io. h)	fcloseall	(stdio. h)
cscanf	(conio. h)	fdopen	(stdio. h)
_dos_close	(dos. h)	feof	(stdio. h)
ferror	(stdio. h)	printf	(stdio. h)
fflush	(stdio. h)	putc	(stdio. h)
fgetc	(stdio. h)	putch	(conio. h)
fgetchar	(stdio. h)	putchar	(stdio. h)
fgetpos	(stdio. h)	puts	(stdio. h)
fgets	(stdio. h)	putw	(stdio. h)
filelength	(io. h)	_read	(io. h)
fileno	(stdio. h)	read	(io. h)
flushall	(stdio. h)	remove	(stdio. h)
fopen	(stdio. h)	rename	(stdio. h)
fprintf	(stdio. h)	rewind	(stdio. h)
fputc	(stdio. h)	rmtmp	(stdio. h)
fputchar	(stdio. h)	scanf	(stdio. h)
fputs	(stdio. h)	setbuf	(stdio. h)
fread	(stdio. h)	setcursortype	(conio. h)

freopen	(stdio. h)	setftime	(io. h)
fscant	(stdio. h)	setmode	(io. h)
fseek	(stdio. h)	setvbuf	(stdio. h)
fsetpos	(stdio. h)	sopen	(io. h)
_fsopen	(stdio. h)	sprintf	(stdio. h)
fstat	(sys\stat. h)	sscanf	(stdio. h)
ftell	(stdio. h)	stat	(sys\stat. h)
fwrite	(stdio. h)	_strerror	(string. h, stdio. h)
getc	(stdio. h)	strerror	(stdio. h)
getch	(conio. h	tell	(io. h)
getchar	(stdio. h)	tempnam	(stdio. h)
getche	(conio. h)	tmpfile	(stdio. h)
getftime	(io. h)	tmpnam	(stdio. h)
getpass	(conio. h)	umask	(io. h)
gets	(stdio. h)	ungetc	(stdio. h)
getw	(stdio. h)	ungetch	(conio. h)
ioctl	(io. h)	unlock	(stdio. h)
isatty	(io. h)	utime	(stdio. h)
kbhit	(conio. h)	vfprintf	(stdio. h)
lock	(io. h)	vfscanf	(stdio. h)
locking	(io. h)	vprintf	(stdio. h)
iseek	(io. h)	vscanf	(stdio. h)
_open	(io. h)	vsprintf	(stdio. h)
open	(io. h)	vsscanf	(io. h)
perror	(stdio. h)	_write	(io. h)

1.4.9 接口子程序(DOS, 8086, BIOS)

此类子程序提供 DOS、BIOS 的功能调用以及使用机器硬件资源的功能

absread	(dos. h)	bdosptr	(dos. h)
abswrite	(dos. h)	bioscom	(bios. h)
bdos	(dos. h)	_bios_disk	(bios. h)
biosdisk	(bios. h)	harderr	(dos. h)
_bios_equiplist	(bios. h)	_hardresume	(dos. h)
biosequip	(bios. h)	hardresume	(dos. h)
_bios_keybrd	(bios. h)	_hardretn	(dos. h)
bioskey	(bios. h)	hardretn	(dos. h)
biosmemory	(bios. h)	inp	(conio. h)
biosprint	(bios. h)	inpw	(conio. h)
_bios_printer	(bios. h)	inport	(dos. h)
_bios_serialcom	(bios. h)	inportb	(dos. h)
biostime	(bios. h)	int86	(dos. h)

_chain_intr	(dos.h)	int86x	(dos.h)
country	(dos.h)	intdos	(dos.h)
ctrlbrk	(dos.h)	indosx	(dos.h)
_dissable	(dos.h)	intr	(dos.h)
disable	(dos.h)	keep	(dos.h)
dosexterr	(dos.h)	MK_FP	(dos.h)
_dos_getvect	(dos.h)	outp	(conio.h)
_dos_keep	(dos.h)	outpw	(conio.h)
_dos_setvect	(dos.h)	outport	(dos.h)
_enable	(dos.h)	outportb	(dos.h)
enable	(dos.h)	parsfnm	(dos.h)
FP_OFF	(dos.h)	peek	(dos.h)
FP_SEG	(dos.h)	peekb	(dos.h)
freemem	(dos.h)	poke	(dos.h)
geninterrupt	(dos.h)	pokeb	(dos.h)
getcbrk	(dos.h)	randbrd	(dos.h)
getdfree	(dos.h)	randbwr	(dos.h)
getdta	(dos.h)	segread	(dos.h)
getfat	(dos.h)	setcbrk	(dos.h)
getfatd	(dos.h)	setdta	(dos.h)
getpsp	(dos.h)	setvect	(dos.h)
getvect	(dos.h)	setverify	(dos.h)
getverify	(dos.h)	sleep	(dos.h)
_harderr	(dos.h)	unlink	(dos.h)

1.4.10　操作子程序

此类子程序处理串和内存块;拷贝、比较、转换和查找。

mblen	(stdlib.h)	memset	(mem.h, string.h)
mbstowcs	(stdlib.h)	movedata	(mem.h, string.h)
mbtowc	(stdlib.h)	movmem	(mem.h, string.h)
memccpy	(mem.h, string.h)	setmem	(mem.h)
memchr	(mem.h, string.h)	stpcpy	(string.h)
memcmp	(mem.h, string.h)	strcat	(string.h)
memcpy	(mem.h, string.h)	strchr	(string.h)
memicmp	(mem.h, string.h)	strcmp	(string.h)
memmove	(mem.h, string.h)	strcoll	(string.h)
strcpy	(string.h)	strnset	(string.h)
strcspn	(string.h)	strpbrk	(string.h)
strdup	(string.h)	strrchr	(string.h)
strerror	(string.h)	strrev	(string.h)
stricmp	(string.h)	strset	(string.h)

strlen	(string.h)	strspn	(string.h)
strlwr	(string.h)	strtok	(string.h)
strncat	(string.h)	strupr	(string.h)
strncmpi	(string.h)	strxfrm	(string.h)
strnicmpi	(string.h)	wcstombs	(stdlib.h)
strnicpy	(string.h)	wctomb	(stdlib.h)
strnicmp	(string.h)		

1.4.11 数学子程序

此类子程序执行数学计算和数值类型转换。

abs	(complex.h, stdlib.h)	expl	(math.h)
acos	(complex.h, math.h)	fabs	(math.h)
acosl	(math.h)	fabsl	(math.h)
arg	(complex.h)	fcvt	(stdlib.h)
asin	(complex.h, math.h)	floor	(math.h)
asinl	(math.h)	fmodl	(math.h)
atan2	(complex.h, math.h)	_fpreset	(float.h)
atan2l	(math.h)	frexp	(math.h)
atof	(stdlib.h, math.h)	frexpl	(math.h)
atoi	(stdlib.h)	gcvt	(stdlib.h)
atol	(stdlib.h)	hypot	(math.h)
_atold	(math.h)	hypotl	(math.h)
bcd	(bcd.h)	imag	(complex.h)
cabs	(math.h)	itoa	(stdlib.h)
cabsl	(math.h)	labs	(stdlib.h)
cabsl	(math.h)	labs	(stdlib.h)
ceil	(math.h)	ldexp	(math.h)
ceill	(math.h)	ldexpl	(math.h)
_clear87	(float.h)	ldiv	(math.h)
complex	(complex.h)	log	(complex.h, math.h)
conj	(complex.h)	logl	(math.h)
_control87	(float.h)	log10	(complex.h, math.h)
cos	(complex.h, math.h)	log100	(math.h)
cosl	(math.h)	_lrotl	(stdlib.h)
cosh	(complex.h, math.h)	_lrotr	(stdlib.h)
coshl	(math.h)	itoa	(stdlib.h)
div	(math.h)	matherr	(math.h)
ecvt	(stdlib.h)	_matherrl	(math.h)
exp	(complex.h, math.h)	modf	(math.h)
modfl	(math.h)	sinl	(math.h)
norm	(complex.h)	sinh	(complex.h, math.h)

polar	(complex.h)	sinhl	(math.h)
poly	(math.h)	sqrt	(complex.h, math.h)
polyl	(math.h)	sqrtl	(math.h)
pow	(complex.h, math.h)	srand	(stdlib.h)
powl	(math.h)	_status87	(float.h)
pow10	(math.h)	strtod	(stdlib.h)
pow10l	(math.h)	strtol	(stdlib.h)
rand	(stdlib.h)	_strtold	(stdlib.h)
random	(stdlib.h)	strtoul	(stdlib.h)
randomize	(stdlib.h)	tan	(complex.h, math.h)
real	(complex.h)	tanl	(math.h)
_rotl	(stdlib.h)	tanh	(complex.h, math.h)
rotr	(stdlib.h)	tanhl	(complex.h, math.h)
sin	(complex.h, math.h)	ultoa	(stdlib.h)

1.4.12　存储子程序

此类子程序提供大小模式的动态存储分配。

alloca	(malloch.)	farheapfillfree	(alloch.h)
allocmem	(dos.h)	farheapwalk	(alloc.h)
_bios_memsize	(bios.h)	farmalloc	(alloc.h)
brk	(alloc.h)	farrealloc	(alloc.h)
calloc	(alloc.h,	free	(alloc.h,
	stdlib.h)		stdlib.h)
coreleft	(alloc.h)	heapcheck	(alloc.h)
	stdlib.h)	heapcheckfree	(alloc.h)
_dos_allocmem	(dos.h)	heapchecknode	(alloc.h)
_dos_freemem	(dos.h)	heapwalk	(alloc.h)
_dos_setblock	(dos.h)	malloc	(alloc.h,
farcalloc	(alloc.h)		stdlib.h)
farcoreleft	(alloc.h)	realloc	(alloc.h,
farfree	(alloc.h)		stdlib.h)
farheapcheck	(alloc.h)	sbrk	(alloc.h)
farheapcheckfree	(alloc.h)	setblock	(dos.h)
farheapchecknode	(alloc.h)	set_new_handler	(new.h)

1.4.13　杂类子程序

delay	(dos.h)	setjmp	(setjmp.h)
localeconv	(locale.h)	setlocale	(locale.h)
longimp	(setjmp.h)	sound	(dos.h)
nosound	(dos.h)		

1.4.14 进程控制子程序

此类子程序从一个进程里请求和终止一个新进程。

abort	(process.h)	execve	(process.h)	spawnl	(process.h)
_c_exit	(process.h)	execvp	(process.h)	spawnle	(process.h)
_cexit	(process.h)	execvpe	(process.h)	spawnlp	(process.h)
exect	(process.h)	_exit	(process.h)	spawnlpe	(process.h)
execle	(process.h)	exit	(process.h)	spawnv	(process.h)
execlp	(process.h)	getpid	(process.h)	spawnve	(process.h)
execlpe	(process.h)	raise	(signal.h)	spawnvp	(process.h)
execv	(process.h)	signal	(signal.h)	spawnvpe	(process.h)
_beginthread	(process.h)	_beginthreadex	(process.h	_beginthread	(process.h)

1.4.15 文本窗口显示子程序

此类子程序用于在屏幕上输出文本。

clreol	(conio.h)	normvideo	(conio.h)
clrscr	(conio.h)	puttext	(conio.h)
delline	(conio.h)	setcursortype	(conio.h)
gettext	(conio.h)	textattr	(conio.h)
gettextinfo	(conio.h)	textbackground	(conio.h)
gotoxy	(conio.h)	textcolor	(conio.h)
highvideo	(conio.h)	textmode	(conio.h)
insline	(conio.h)	wherex	(conio.h)
lowvideo	(conio.h)	wherey	(conio.h)
movetext	(conio.h)	window	(conio.h)

1.4.16 时间和日期子程序

此类子程序用于时间的转换与操作。

asctime	(time.h)	gettime	(dos.h)
_bios_timeofday	(bios.h)	gmtime	(time.h)
ctime	(time.h)	localtime	(time.h)
difftime	(time.h)	mtime	(time.h)
_dos_getdate	(dos.h)	setdate	(dos.h)
_dos_gettime	(dos.h)	settime	(dos.h)
_dos_setdate	(dos.h)	stime	(time.h)
_dos_settime	(dos.h)	strftime	(time.h)
dostounix	(dos.h)	time	(time.h)
ftime	(sys\timeb.h)	tzset	(time.h)

getdate	(dos.h)	unixtodos	(dos.h)

1.4.17 变量参数表子程序

此类子程序用于存取可变参数表(如 vprintf 等)。

va_arg	(stdarg.h)
va_end	(stdarg.h)
va_start	(stdarg.h)

1.4.18 取消的定义

下面全局变量为了与 ANSI 命名需求相适应而重命名。最好使用新名。如果用这些在 Borland C++ 3.1 或以前中有的库函数,会产生错误信息:

Error: undefined external varname in module LIBNAME.LIB

导致上述错误的库模块应重新编译。然而,如果不能重编译库代码,可以与 OBSOLETE.LIB 库联接来确定外部变量名。

表 1.4 的全局变量被改名。

表 1.4 取消的全局变量

老式名字	新名字	头文件
daylight	_daylight	time.h
directvideo	_directvideo	conio.h
environ	_environ	stdlib.h
sys_errlist	_sys_errlist	errno.h
sys_nerr	_sys_nerr	errno.h
timezone	_timezone	time.h
tzname	_tzname	time.h

表 1.5 的老式函数还有。但编译器会产生一个警告,告知用户使用已取消的名字。将来的 Borland C++ 版本将不再支持老式函数名。

表 1.5 中的函数已被改名。

表 1.5 取消的函数名

老式名字	新名字	头文件
_chmod	_rtl_chmod	io.h
_close	_rtl_close	io.h
_creat	_rtl_creat	io.h
_heapwalk	_rtl_heapwalk	malloc.h
_open	_rtl_open	io.h
_read	_rtl_read	io.h
_write	_rtl_write	io.h

第二章 main 函数

每个 C 和 C++程序必须一个启动函数。基于控制台的程序调用 main 函数启动。Windows GUI 程序调用 WinMain 函数启动。将启动函数放在何处随程序员的意愿。一些程序员将 main 放在文件的开头,而另外的一些程序则将之放在程序的末尾。不管位置如何,下列规则都适用。

2.1 main 参数

Borland C++的启动代码将把 argc、argv 和三个参数传给 main.

argc 是一个整数,其值为传给 main 的命令行参数个数

argv 是一个指向字符串的指针数组:

- 在 DOS 3.X 版中,argv[0]是正在运行的程序的全路径名;
- 在 DOS 3.0 以前的版本,argv[0]指向空串;
- arg[1]指向 DOS 命令行中紧随着程序名键入的第一个字符串;
- arg[2]指向程序名后的第二个串;
- argv[argc-1]指向传给 main 的最后一个参数;
- argv[argc]是空串。

env 是一个字符串指针数组,env[]的每个元素包含一个字符串,每个字符串形式为:ENVVAR=value,其中

- ENVVAR 为环境变量名,如:PATH 或 87;
- value 是要设置的 ENVVAR 值,例如 C:\DOS;C:\BC(赋给 PATH)或 YES(赋给 87)。

如果要说明这些参数,必须按 argc,argv,env 的顺序来说明。例如,以下均为合法的 main 参数说明:

```
main()
main(int argc)                    /* 合法但很少见 */
main(int argc, char *argv[])
main(int argc, char *argv[], char *env[])
```

用 main(int argc)来说明 main 的参数是合法的,但这种在程序中只使用 argc 而不同时使用 argv 的情况是很少见的。

env 参数也可以通过全局变量 environ 得到,更详细的信息,请参见第三章的 environ 和

本章 putenv 与 getenv 中的有关条目。

2.1.1 范例程序

以下是一个范子程序 ARGS. EXE，该程序展示了如何将参数传给 main 的一种简单的方法。

```
/* Program ARGS.C */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *agrv[], char *env[])
{
  int i;
  printf("The value of argc is %d \n\n",argc);
  printf("There are the %d command-line arguments passed to main:\n\n",argc);
  for(i=0;i<argc;i++)
       printf("   argv[%d]: %s\n",i,argv[i]);
  printf("\nThe enviroment string(s) on this system are:\n\n");
  for(i=0;env[i]!=NULL;i++)
      printf("   env[%d]:%s\n",i,env[i]);
  return 0;
}
```

假定在 DOS 提示符下，用户输入以下命令来运行 ARGS. EXE

C>args first _arg "arg with blanks" 3 4 "last but one" stop!

注意：含有空格的串也可以作为参数传递，但必须用双引号将它引起来，就象命令行中的"arg with blanks"和"last but one"一样。输入以下命令后，ARGS. EXE 的输出结果如下：

```
The value of argc is 7

These are the 7 command-line arguments passed to main:

  argv[0]: C:\BC\ARGS.EXE
  argv[1]: first _arg
  argv[2]: arg with blanks
  argv[3]: 3
  argv[4]: 4
  argv[5]: last but one
  argv[6]: stop!

The enviroment string(s) on this system are:
  env[0]: COMSPEC=C:\COMMAND.COM
  env[1]: PROMPT=$p $g
  env[2]: PATH=C:\SPRINT;C:\DOS;C:\BC
```

传给 main 的各个参数的总长度(包括相邻参数间的空格和文件名本身)最大可以为