

学习和使用

Visual C++(下册)

李 蕾 朱志强 等编著

潘金贵 顾铁成 审 校



同济大学出版社

内 容 提 要

本书是《学习和使用 Visual C++》丛书的下册,主要介绍了 Visual C++ 程序设计中经常用到的函数,如文件和目录管理函数、流 I/O 函数、低级 I/O 函数、控制台和端口 I/O 函数、字符和数据转换函数、串和缓冲区处理函数、查找和排序函数、日期和时间管理函数、数学函数、进程控制函数、内存管理函数以及 DOS 和 BIOS 调用函数等。对 C++ (尤其是 Visual C++) 程序员而言,本书是一本很实用的技术指导书籍。本书可作为程序设计方法、通用程序设计技术、C++、Visual C++ 等方面的培训教材。

TC744 / 5201

责任编辑: 王建中
封面设计: 李志云

学习和使用 Visual C++ (下 册)

李 蕾 朱志强 等编

潘金贵 顾铁成 审校

同济大学出版社出版

(上海四平路 1239 号)

新华书店上海发行所发行

同济大学印刷厂印刷

开本: 787×1092 1/16 印张: 18 字数: 460 千字

1997 年 1 月第 1 版 1997 年 1 月第 1 次印刷

印数: 1—4000 定价: 30.00 元

ISBN7-5608-1692-4/TP·177

前 言

随着微软公司 Windows 95 的问世,Windows 在微机软件业的主流地位进一步巩固,作为 Windwos 环境下的优秀、正宗 C++ 编译器的 Visual C++ 进一步成为软件开发人员强有力的工具。Visual C++ 是汇集微软公司技术精华的主流产品,它不仅全面地贯彻了面向对象技术,而且在编译优化技术方面较其他同类产品具有明显的优势。《学习和使用 Visual C++》丛书共分三册。这套书从各个方面说明了应如何在 Visual C++ 环境下以 OOP 的概念设计 DOS 和 Windows 应用程序。

在 Windows 如此火热的时候,任何一个希望在微机平台上开发自己的软件产品的单位和个人都无法回避 Windows 环境。作为一个成熟的软件工程师,面对铺天盖地的 Word,Excel,PowerPoint,FoxPro,Visual Basic 等应用软件,有很多理由需要熟悉 Visual C++。笔者曾与美国微软公司一位资深业务主管在 Windows 95 研讨会上曾讨论过这个问题,他从美国软件产业的统计分析得出的结论是,附加在 Windwos 环境上的第三方应用软件的增值产品产值在过去五年中年平均递增约 70%。由此可见,借助 Visual C++ 开发工具,结合自己的专业技术,开发出各种专用应用软件的动态链接库,就能为广大用户提供各个方面的支持,其商业前景更是无可估量的。

Visual C++ 是一个面向对象、界面友好的 DOS 和 Windows 程序开发环境。在这个集成环境中开发应用程序时,不一定要手工设计用户界面,而只需选取菜单命令,Visual C++ 系统就会生成一个可实际运行的 Windows 应用程序框架。此后,程序员就可利用基于 Windows 的 C++ 编辑器,借助 AppWizard 建立面向对象的应用程序。

除了 AppWizard 外,Visual C++ 还包含 C++ 应用程序生成器、基于 Windows 的编辑器、基于 Windows 的面向图形的编程工具、使 C++ 代码和 Windows 消息及类成员函数相联系的交互式工具、可用来编写 Visual C++ 文本程序的 QuickWin 库以及预编译的头文件和源文件。当然,类库丰富更是 Visual C++ 的最大特色。

目前市面上虽然有不少 Visual C++ 方面的书籍,但学习和使用 Visual C++ 的人常常会觉得力不从心,主要问题是:

- (1) 对 Visual C++ 缺乏一致的认识,遇到问题总是缺这少那,缺乏一个值得信赖的良师益友。
- (2) 只了解 OOP 的直观语法,而对于 OOP 有何好处、为什么 OOP 可取代结构化语言并成为未来程序设计风格的主流,都不太明白。
- (3) 无法理解 OOP 对未来程序的管理与维护究竟有何重要性。
- (4) 在 Visual C++ 环境下设计 OOP 程序时,对 Visual C++ 提供的资源不太清楚。
- (5) 不知道如何用 Visual C++ 和 Windows 解决实际问题。

针对这些问题,这套书从各个方面介绍了 Visual C++ 集成环境、OOP 概念、类库以及基于 DOS 和 Windows 的 C++ 应用程序开发方法,并提供了丰富的类模板和大量的应用程序实例。

本套书的内容基本属于中等难度,适用于初、中级读者。不熟悉 OOP 技术和 C++ 语言的读者可参阅上册的内容,上册主要介绍 Visual C++ 集成环境和 OOP 概念;中册介绍经常从事应用程序开发的读者比较关心的 Visual C++ 类库和通用类的构造方法,并从我们所熟悉的数据结构着手,讨论如何设计和实现自己的通用类,这为进行大型程序开发的用户提供了设计开发环境的手段;下册侧重函数的介绍,包括文件输入输出、数据数理、进程控制及内存管理等内容。

本书是这套书的下册,参与编写的人员有:何亮、朱志强、覃牧、何国辉、李华清、李杨、魏志国、李蕾、李建、刘心海、刘崇福、李纪鸿、刘石华、赵越、陆静宜、李伯雄、陈楚南、梁友国、陈佳海。全书先由吴佳教授和陈忠敏研究员作了初审,由潘金贵和顾铁成终审。此外,朱闽虹为本书的编排付出了辛勤的劳动。在此对以上同志深表感谢。

限于水平和时间,书中的错误和不妥之处,敬请读者批评指正。

编 者

目 录

第一章 文件和目录管理函数	1	2.1.1 I/O 缓冲	22
1.1 MS-DOS 文件系统	1	2.1.2 FILE 类型	23
1.1.1 路径名	1	2.1.3 I/O 的格式化和非格式化	23
1.1.2 作为文件的设备	2	2.1.4 文件的当前位置	23
1.1.3 文件属性	2	2.1.5 预定义流	24
1.1.4 文件句柄	3	2.1.6 字符串输入/输出	24
1.2 基本文件和目录管理任务	3	2.2 基本流 I/O 任务	24
1.2.1 改变驱动器和目录	4	2.2.1 给文件添加行号	26
1.2.2 改变文件属性	6	2.2.2 简单的通信录	28
1.3 函数参考	7	2.2.3 显示错误信息的简单方法	34
1.3.1 _access	7	2.3 函数参考	35
1.3.2 _chdir	8	2.3.1 clearerr	35
1.3.3 _chdrive	8	2.3.2 fclose	35
1.3.4 _chmod	9	2.3.3 _fcloseall	36
1.3.5 _chsize	10	2.3.4 _fdopen	36
1.3.6 _filelength	10	2.3.5 feof	37
1.3.7 _fstat	11	2.3.6 ferror	38
1.3.8 _fullpath	11	2.3.7 fflush	38
1.3.9 _getcwd	12	2.3.8 fgetc	39
1.3.10 _getdcwd	13	2.3.9 _fgetchar	39
1.3.11 _getdrive	13	2.3.10 fgetpos	40
1.3.12 _isatty	14	2.3.11 fgets	40
1.3.13 _locking	14	2.3.12 _fileno	41
1.3.14 _makepath	15	2.3.13 _flushall	42
1.3.15 _mkdir	16	2.3.14 fopen	42
1.3.16 _mktemp	16	2.3.15 fprintf	43
1.3.17 remove	17	2.3.16 fputc	43
1.3.18 rename	17	2.3.17 _fputchar	44
1.3.19 _rmdir	17	2.3.18 fputs	44
1.3.20 _searchenv	18	2.3.19 fread	45
1.3.21 _setmode	18	2.3.20 freopen	45
1.3.22 _splitpath	19	2.3.21 fscanf	46
1.3.23 _stat	19	2.3.22 fseek	47
1.3.24 _umask	20	2.3.23 fsetpos	47
1.3.25 _unlink	21	2.3.24 _fsopen	48
第二章 流 I/O 函数	22	2.3.25 ftell	48
2.1 流	22	2.3.26 fwrite	49

2.3.27	getc	50	3.3.7	_lseek	75
2.3.28	getchar	50	3.3.8	_open	76
2.3.29	gets	50	3.3.9	_read	77
2.3.30	_getw	51	3.3.10	_sopen	78
2.3.31	perror	51	3.3.11	_tell	79
2.3.32	printf	52	3.3.12	_write	79
2.3.33	putc	54	第四章 控制台和端口 I/O 函数	81	
2.3.34	putchar	54	4.1 控制台和端口 I/O 基础	81	
2.3.35	puts	55	4.1.1 控制台和端口 I/O 的任务	81	
2.3.36	_putw	55	4.1.2 用_inp 和_outp 产生声音	82	
2.3.37	rewind	56	4.2 函数参考	83	
2.3.38	_rmtmp	56	4.2.1 _cgets	83	
2.3.39	scanf	57	4.2.2 _cprintf	84	
2.3.40	setbuf	58	4.2.3 _cputs	84	
2.3.41	setvbuf	59	4.2.4 _cscanf	85	
2.3.42	_snprintf	60	4.2.5 _getch	85	
2.3.43	sprintf	60	4.2.6 _getche	86	
2.3.44	sscanf	61	4.2.7 _inp	86	
2.3.45	_tempnam	61	4.2.8 _inpw	87	
2.3.46	trapfile	62	4.2.9 _kbhit	87	
2.3.47	tmpnam	62	4.2.10 _outp	88	
2.3.48	ungetc	63	4.2.11 _outpw	88	
2.3.49	vfprintf	63	4.2.12 _putch	89	
2.3.50	vprintf	64	4.2.13 _ungetch	89	
2.3.51	_vsnprintf	65	第五章 字符和数据转换函数	91	
2.3.52	vsprintf	66	5.1 字符和数据转换基础	91	
第三章 低级 I/O 函数	67		5.1.1 字符分类	91	
3.1 低级 I/O 函数基础	67		5.1.2 转换数据	92	
3.1.1 文本和二进制模式	67		5.2 字符和数据转换任务	92	
3.1.2 通过操作系统完成缓冲	67		5.2.1 将文本字符串转换为小写	94	
3.1.3 文件句柄	67		5.2.2 一个简单的计算器	95	
3.1.4 最大文件句柄数	68		5.3 函数参考	96	
3.2 基本的低级 I/O 任务	68		5.3.1 atof	96	
3.2.1 拷贝文件	69		5.3.2 atoi, atol	96	
3.2.2 用_dup 和_dup2 重定向			5.3.3 _atold	97	
stdout	71		5.3.4 _ecvt, _fcvt	97	
3.3 函数参考	72		5.3.5 _gcvt	98	
3.3.1 _close	72		5.3.6 isalnum	99	
3.3.2 _commit	73		5.3.7 isalpha	99	
3.3.3 _creat	73		5.3.8 _isascii	99	
3.3.4 _dup	74		5.3.9 isctrl, isdigit, isgraph,		
3.3.5 _dup2	74		islower, isprint, ispunct,		
3.3.6 _eof	75		isspace, isupper, isxdigit	100	

5.3.10	_iscsym, _iscsymf	101	6.3.23	strlen, _fstrlen	127
5.3.11	_itoa, _ltoa	101	6.3.24	_strlwr, _fstrlwr	127
5.3.12	strtod	102	6.3.25	strncat, _fstrncat	128
5.3.13	strtol	102	6.3.26	strncmp, _fstrncmp	128
5.3.14	_strtold	103	6.3.27	strncpy, _fstrncpy	129
5.3.15	strtoul	104	6.3.28	_strnicmp, _fstrnicmp	130
5.3.16	_toascii	104	6.3.29	_strnset, _fstrnset	130
5.3.17	_tolower, tolower	105	6.3.30	strpbrk, _fstrpbrk	131
5.3.18	_toupper, toupper	105	6.3.31	strrchr, _fstrchr	132
5.3.19	_ultoa	106	6.3.32	_strrev, _fstrrev	132
第六章 串和缓冲区处理函数		107	6.3.33	_strset, _fstrset	133
6.1	C 中的串和缓冲区	107	6.3.34	strspn, _fstrspn	133
6.1.1	声明串和缓冲区	107	6.3.35	strstr, _fstrstr	134
6.1.2	字典顺序	107	6.3.36	strtok, _fstrtok	135
6.1.3	多字节和宽位字符串	107	6.3.37	_strupr, _fstrupr	136
6.1.4	远程缓冲区和串	108	6.3.38	strxfrm	136
6.2	基本的串和缓冲区处理任务	108	6.3.39	_swab	137
6.2.1	拷贝视频内存	111	6.3.40	westombs, _fwestombs	137
6.2.2	分析文本行	112	6.3.41	wctomb, _fwctomb	138
6.3	函数参考	113	第七章 查找和排序函数		139
6.3.1	fmblen, _fmblen	113	7.1	查找和排序的基本任务	139
6.3.2	mbstowcs, _fmbstowcs	114	7.2	函数参考	141
6.3.3	mbtowc, _fmbtowc	114	7.2.1	bsearch	141
6.3.4	_memccpy, _fmemccpy	115	7.2.2	_lfind, _lsearch	142
6.3.5	memchr, _fmemchr	116	7.2.3	qsort	143
6.3.6	memcmp, _fmemcmp	116	第八章 日期和时间管理函数		144
6.3.7	memcpy, _fmemcpy	117	8.1	时间格式	144
6.3.8	_memicmp, _fmemicmp	118	8.1.1	当地时间、GMT 和 UCT	144
6.3.9	memmove, _fmemmove	118	8.1.2	时间转换	144
6.3.10	memset, _fmemset	119	8.2	基本的日期和时间管理任务	145
6.3.11	_movedata	120	8.2.1	获取并打印当前的日期和时间	146
6.3.12	strcat, _fstrcat	120	8.2.2	打印某个月的日历	146
6.3.13	strchr, _fstrchr	121	8.3	函数参考	150
6.3.14	strcmp, _fstrcmp	121	8.3.1	asctime	150
6.3.15	_strcmpi	122	8.3.2	clock	151
6.3.16	strcoll	123	8.3.3	ctime	151
6.3.17	strcpy, _fstrcpy	123	8.3.4	difftime	152
6.3.18	strcspn, _fstrcspn	124	8.3.5	_ftime	152
6.3.19	_strdup, _fstrdup, _nstrdup	124	8.3.6	gmtime, localtime	153
6.3.20	_strerror	125	8.3.7	mktime	154
6.3.21	strerror	126	8.3.8	_strdate	154
6.3.22	_stricmp, _fstricmp	126	8.3.9	strftime	155

8.3.10	strtime	156	9.3.28	_lmax, _lmin	180
8.3.11	time	156	9.3.29	modf, modfl	180
8.3.12	tzset	157	9.3.30	pow, powl	181
8.3.13	utime	158	9.3.31	rand	181
第九章 数学函数		159	9.3.32	_rotl, _rotr	182
9.1	浮点数概述	159	9.3.33	sin, sinl	182
9.1.1	浮点数的存储格式	159	9.3.34	sinh, sinhl	183
9.1.2	浮点变量的类型	159	9.3.35	sqrt, sqrtl	183
9.1.3	浮点库	160	9.3.36	srand	184
9.1.4	数学函数的错误处理	160	9.3.37	_status87	184
9.2	数学函数的类型	160	9.3.38	tan, tanl	185
9.2.1	计算借款	163	9.3.39	tanh, tanhl	185
9.2.2	计算正弦和余弦	164	第十章 进程控制函数		187
9.3	函数参考	165	10.1	进程管理	187
9.3.1	abs	165	10.1.1	环境	187
9.3.2	acos, acosl	165	10.1.2	信号	187
9.3.3	asin, asinl	166	10.1.3	可变长度的参数表	188
9.3.4	atan, atanl	166	10.1.4	现场	188
9.3.5	atan2, atan2l	167	10.2	进程控制的基本任务	188
9.3.6	贝塞耳函数	167	10.3	函数参考	194
9.3.7	_cabs, _cabsl	168	10.3.1	abort	194
9.3.8	ceil, ceil1	169	10.3.2	assert	194
9.3.9	_clear87, _control87	169	10.3.3	atexit, _fatexit	195
9.3.10	cos, cosl	170	10.3.4	_cexit, _c_exit	195
9.3.11	cosh, coshl	171	10.3.5	_execl, _execle, _execlp, _execlpe, _execv, _execve, _execvp, _execvpe	196
9.3.12	_dieetomshin, _dmsbintoieee	171	10.3.6	exit	197
9.3.13	div	172	10.3.7	_exit	197
9.3.14	exp, expl	172	10.3.8	getenv	198
9.3.15	fabs, fabsl	173	10.3.9	getpid	198
9.3.16	_fieetomshin, _fmsbintoieee	173	10.3.10	localeconv	199
9.3.17	floor, floorl	174	10.3.11	longjmp	200
9.3.18	fmod, fmodl	174	10.3.12	_onexit, _fonexit	200
9.3.19	_fpreset	175	10.3.13	_putenv	201
9.3.20	frexp, frexpl	175	10.3.14	raise	201
9.3.21	hypot, hypotl	176	10.3.15	setjmp	202
9.3.22	labs	176	10.3.16	setlocale	202
9.3.23	ldexp, ldexpl	177	10.3.17	signal	203
9.3.24	ldiv	177	10.3.18	_spawnl, _spawnle, _spawnlp, _spawnlpe, _spawnv, _spawnve, _spawnvp, _spawnvpe	204
9.3.25	log, log10, logl, log10l	178			
9.3.26	lrotl, lrotr	179			
9.3.27	_matherr, _matherrl	179			

10.3.19	system	205		_nmsize	227
10.3.20	va_arg, va_end, va_start(ANSI 版本)	206	11.3.17	realloc, _brealloc, _frealloc, _nrealloc	228
10.3.21	va_arg, va_end, va_start(UNIX 版本)	207	11.3.18	_stackavail	228
第十一章 内存管理函数			209	11.3.19	_vfree
11.1	内存管理基础	209	11.3.20	_vheapinit	229
11.1.1	分段内存寻址	209	11.3.21	_vheapterm	230
11.1.2	内存模式	209	11.3.22	_vload	231
11.1.3	near, far 和 huge 指针	210	11.3.23	_vlock	231
11.1.4	_near, _far 和 _huge 关键字	210	11.3.24	_vlockcnt	232
11.1.5	堆	210	11.3.25	_vmalloc	233
11.1.6	基堆	210	11.3.26	_vmsize	233
11.1.7	虚拟内存	212	11.3.27	_vrealloc	234
11.2	基本内存管理任务	212	11.3.28	_vunlock	235
11.3	函数参考	217	第十二章 DOS 和 BIOS 调用函数		
11.3.1	_alloca	217	12.1	访问 BIOS 和 DOS 服务	236
11.3.2	_bfreeseg	217	12.1.1	BIOS 服务	236
11.3.3	_bheapseg	218	12.1.2	MS-DOS 功能	237
11.3.4	calloc, _bcalloc, _fcalloc, _hallo, _ncallo	218	12.2	基本 DOS 和 BIOS 任务	238
11.3.5	_expand, _bexpand, _fexpand, _nexpand	219	12.2.1	目录清单	239
11.3.6	free, _bfree, _ffree, _hfree, _nfree	220	12.2.2	观察磁盘物理扇区	240
11.3.7	_freect	221	12.3	函数参考	244
11.3.8	_heapadd, _bheapadd	221	12.3.1	_bdos	244
11.3.9	_heapchk, _bheapchk, _fheapchk, _nheapchk	222	12.3.2	_bios_disk	244
11.3.10	_heapmin, _bheapmin, _fheapmin, _nheapmin	223	12.3.3	_bios_equiplist	246
11.3.11	_heapset, _bheapset, _fheapset, _nheapset	223	12.3.4	_bios_keybrd	247
11.3.12	_heapwalk, _bheapwalk, _fheapwalk, _nheapwalk	224	12.3.5	_bios_memsiz	248
11.3.13	malloc, _bmalloc, _fmalloc, _nmalloc	225	12.3.6	_bios_printer	249
11.3.14	_memavl	226	12.3.7	_bios_serialcom	250
11.3.15	_memmax	227	12.3.8	_bios_timeofday	252
11.3.16	_msize, _bmsize, _fmsize, _nmsize	227	12.3.9	_chain_intr	252
			12.3.10	_disable	253
			12.3.11	_dos_allocmem	253
			12.3.12	_dos_close	254
			12.3.13	_dos_commit	254
			12.3.14	_dos_creat, _dos_creatnew	255
			12.3.15	_dos_findfirst, _dos_findnext	256
			12.3.16	_dos_freemem	257
			12.3.17	_dos_getdate	258
			12.3.18	_dos_getdiskfree	258
			12.3.19	_dos_getdrive	259

12. 3. 20	_dos_getfileattr	260	12. 3. 32	_dos_settime	269
12. 3. 21	_dos_getftime	260	12. 3. 33	_dos_setvect	269
12. 3. 22	_dos_gettime	262	12. 3. 34	_dos_write	270
12. 3. 23	_dos_getvect	262	12. 3. 35	_dosexterr	271
12. 3. 24	_dos_keep	263	12. 3. 36	_enable	271
12. 3. 25	_dos_open	263	12. 3. 37	_FP_OFF, _FP_SEG	272
12. 3. 26	_dos_read	264	12. 3. 38	_harderr, _hardresume, _hardretn	272
12. 3. 27	_dos_setblock	265	12. 3. 39	_int86, _int86x	274
12. 3. 28	_dos_setdate	266	12. 3. 40	_intdos, _intdosx	275
12. 3. 29	_dos_setdrive	266	12. 3. 41	_segread	276
12. 3. 30	_dos_setfileattr	267			
12. 3. 31	_dos_setftime	268			

第一章 文件和目录管理函数

几乎所有的应用程序都要用文件来保存数据和结果,以便以后使用。要有效地使用文件,首先应该理解在 MS-DOS 中文件是如何按照目录层次进行组织的,以及 C 和 C++ 语言是如何操纵文件的。本章简要地介绍 MS-DOS 操作系统内的目录和文件组织,文件和目录的命名以及对文件内容的解释。

Microsoft Visual C++ 完全实现了负责文件和目录管理任务的函数,这些管理任务包括:

- (1) 创建和删除目录。
- (2) 查找一个文件。
- (3) 检查文件的状态。

本章也简要介绍了怎样使用这些函数。

通过阅读本章,用户可以为使用第二章介绍的流和第三章介绍的低级输入/输出(I/O)函数作好准备。利用本章介绍的函数也可以查找指定的文件或检查某个文件是否存在。

本章后面的部分是文件和目录管理函数的参考部分。

1.1 MS-DOS 文件系统

与大多数操作系统(如 UNIX 及 DEC 公司的 VMS)一样,MS-DOS 用树状层次来组织文件和目录。硬盘或软盘上的所有文件及目录都出现在根目录(由反斜杠“\”指示)下。根目录是由 MS-DOS 在格式化软盘或硬盘分区(物理硬盘媒介中的分区)时建立的,它不需要由用户自己建立。每个目录都可以包含子目录和文件,树状层次就是这样形成的。

1.1.1 路径名

由于文件系统可能在软盘或硬盘上,所以 MS-DOS 要求用户指定一个驱动器符号,以便完全确定文件。实际上,文件的全名(全路径名)标识了驱动器和从根目录开始的所有目录。全路径名以带有冒号(:)的驱动器符号开始,后跟单反斜杠(\)以指明是根目录,然后将子目录名依次列下去,中间以反斜杠隔开,而文件名则出现在路径名的末尾。

在 MS-DOS 中,文件名最多由 8 个字符组成,后面可跟最多由 3 个字符组成的扩展名,中间以点号(.)隔开。Microsoft Visual C++ 库中包含 `_fullpath`, `_makepath`, `_splitpath` 等函数,它们可对路径名进行操作。

程序员应注意非标准函数名的下划线前缀。Microsoft Visual C++ 在许多函数(如 `_access`, `_chdir` 和 `_chmod`)名前加了一个下划线前缀,这是为了与 ANSI 标准 C 语言中的非标准函数命名约定保持一致。在其他许多 C 编译器(包括早期版本的 Microsoft C 编译器)以及 UNIX 与 POSIX 中,这些函数都可通过不带下划线前缀的函数名来调用。注意,在 Microsoft Visual C++ 中也可使用不带下划线前缀的函数名。不带下划线前缀的函数定义于 `old-`

names. lib 库中,链接器在缺省方式下会自动查找这个库。

1. 当前驱动器

全路径名对于标识系统中的任意文件是必需的,但并不是在所有情况下都要提供全路径名。MS-DOS 标记了当前驱动器和当前工作目录。如果用户不指定驱动器名,MS-DOS 就认为文件位于当前驱动器中。同样,如果不提供目录名,则假定文件位于当前工作目录中。

在 MS-DOS 中,一个驱动器是指一台设备,即一个软盘或硬盘驱动器。驱动器符号的常规含义如下:

- (1) 驱动器 A:和 B:分别指第一个和第二个软盘驱动器。
- (2) 驱动器 C:~Z:标识硬盘驱动器,C:指第一个硬盘驱动器。

要在程序中确定当前驱动器或改变当前驱动器,可分别使用 `_getdrive` 和 `_chdrive` 函数。

2. 当前工作目录

正如 MS-DOS 记录着当前驱动器一样,它也对当前工作目录进行跟踪。在 C 程序中,通过调用 `_getcwd` 函数即可获取当前目录的全路径名。

可以用 DOS 命令 `CHDIR` 或 `CD` 来改变当前目录。若要创建新目录,则可使用 `MKDIR` 或 `MD` 命令,而 `RD` 命令则可删除某个空目录。Microsoft Visual C++ 提供了在程序中实现这些功能的 `_chdir`, `_mkdir` 及 `_rmdir` 函数。

1.1.2 作为文件的设备

虽然大多数文件是指保存在磁盘上的文件,但 DOS 中有些文件名却可以引用标准设备,如打印机或串行通信端口等。下面是一些这样的文件名:

- (1) AUX 指第一个串行通信口。
- (2) COM1 和 COM2 分别指第一个和第二个串行通信口。
- (3) COM3 和 COM4 分别指第三个和第四个串行通信口。由于串行通信口要共享中断,所以在 PC 机上使用多个串行通信口时可能会出现问題。
- (4) CON 指控制台,即键盘和屏幕。
- (5) PRN 指第一个并行打印机端口。
- (6) LPT1, LPT2 和 LPT3 分别指第一、第二和第三个并行打印机端口。
- (7) NUL 指空设备,即当将输出信息发送给 NUL 时什么也不产生。

即使提供了扩展名,也不能将这些名字(不管是小写还是大写)用作基于磁盘的程序文件或数据文件的名称。它们只能用于访问设备。

1.1.3 文件属性

除了名字之外,DOS 文件还具有如下重要属性,它们会影响文件 I/O 操作:

- (1) “权限设置”确定了文件的类型及其可访问性,即将文件操作限制为只读或读写。权限设置是 DOS 的一部分。
- (2) “转换模式”确定了在 I/O 操作过程中怎样解释文件的内容。转换是 C 语言的一部分。

1. 权限设置

文件的权限设置指出了允许对该文件所执行的操作。在 MS-DOS 中,可以把文件标识为只读文件或读写文件(MS-DOS 不支持只写文件,也不支持只执行文件)。可以用 `_chmod` 函数来设置文件的权限。

函数 `_access` 可以确定文件的访问类型。注意,当用 `fopen` 之类的函数打开文件时,必须用参数指定访问模式(希望打开文件的模式)。例如,可以用读写权限(只读、只写或读写)来打开文件。

2. 转换模式

当用 `fopen` 函数(请参阅第二章)打开文件时,除了可指定读写访问权限之外,还可指定怎样解释文件的内容。共有两种选择:文本模式和二进制模式。在二进制模式下,文件中的每个字节都不加任何转换地提供给程序。在文本模式下,当程序读写文件时要进行下面的转换和解释:

(1) 当读文件时,一个回车换行对被转换成一个换行符, `Ctrl+Z(0x1A)` 被解释成文件结束符(EOF)。

(2) 当写文件时,一个换行符被扩展成一个回车换行对,并被写入文件中。

在打开文件时可以指定转换模式,也可以用 `_setmode` 函数来改变已打开文件的转换模式。

1.1.4 文件句柄

尽管在 DOS 命令中文件名可能是标识文件最自然的方式,但在 C I/O 库中,文件是通过其他方式标识的。与将要在第二章中看到的一样,当用 `fopen` 函数打开某个文件时,它将返回一个指向 FILE 结构的指针,而且以后的 I/O 操作均利用此指针来标识这个文件。类似地,如果用 `_open`, `_sopen` 或 `_creat` 函数创建或打开了一个文件(细节见第三章),这些函数将返回一个称作文件句柄的整型文件标识符。后面介绍了几个用文件句柄作参数的函数,如 `_chsize`, `_filelength`, `_fstat`, `_isatty` 和 `_locking` 等。

1.2 基本文件和目录管理任务

表 1.1 是按字母顺序排列的文件和目录管理函数,而表 1.2 则按函数的功能进行组织。有关更多的信息,请参阅本章的后面部分。

表 1.1 按字母顺序排列的文件和目录管理函数

函 数	描 述
<code>_access</code>	检查文件是否存在及其读写权限设置
<code>_chdir</code>	改变当前工作目录
<code>_chdrive</code>	改变当前驱动器
<code>_chmod</code>	改变文件的读写权限设置

(续表 1.1)

函 数	描 述
<code>-chsize</code>	改变文件大小
<code>-filelength</code>	返回文件的字节长度
<code>-fstat</code>	提供一个用句柄标识的文件的有关信息
<code>-fullpath</code>	把部分路径名转换成全路径名
<code>-getcwd</code>	返回当前驱动器上的当前工作目录
<code>-getdcwd</code>	返回指定驱动器上的当前工作目录
<code>-getdrive</code>	返回一个标识着当前驱动器的整数(1=A, 2=B, 3=C)
<code>-isatty</code>	如果文件句柄指向字符设备, 则返回非零值
<code>-locking</code>	锁定或解锁文件中指定范围内的字节
<code>-makepath</code>	根据 DOS 路径成分建立 DOS 路径名
<code>-mkdir</code>	创建一个新目录
<code>-mktemp</code>	返回一个唯一的文件名
<code>remove</code>	删除由路径名标识的文件
<code>rename</code>	更改文件名
<code>-rmdir</code>	删除一个目录
<code>-searchenv</code>	在由环境变量指定的目录清单中查找某个文件
<code>-setmode</code>	设置已打开文件的转换模式
<code>-splitpath</code>	把 DOS 路径名分解成各组成部分
<code>-stat</code>	提供一个由文件名标识的文件的有关细节信息
<code>-umask</code>	设置新打开的文件所用的权限屏蔽位
<code>-unlink</code>	删除由路径名标识的文件

表 1.2 基本的文件和目录管理任务

任 务	函 数
设置或检验文件的访问权限	<code>-access, -chmod, -umask</code>
获取或设置当前驱动器	<code>-chdrive, -getdrive</code>
管理目录	<code>-chdir, -getcwd, -getdcwd, -mkdir, -rmdir</code>
定位文件	<code>-searchenv</code>
获取或设置文件属性	<code>-chsize, -filelength, -fstat, -isatty, -mktemp, -setmode, -stat</code>
删除文件	<code>-remove, -unlink</code>
更改文件名	<code>-rename</code>
锁定文件中某一范围内的字节	<code>-locking</code>
操纵 MS-DOS 路径名	<code>-fullpath, -makepath, -splitpath</code>

1.2.1 改变驱动器和目录

有时,可能需从程序中确定当前驱动器和当前工作目录。例如,假定要编写一个菜单驱动的实用程序,以使用户能够改变当前驱动器和当前目录。清单 1.1 给出了这样一个程序(`dirutil.c`),它利用函数`-getdrive`,`-getcwd`,`-chdrive`和`-chdir`来完成所要求的任务。菜单驱动的用户界面是利用文本模式下的输出函数(见`graph.doc`文件)生成的。

注意,像`dirutil.c`这样的程序在经过扩展后,可以提供其他许多功能,如创建目录、删除目录、更改文件名或删除文件等。对于这样的程序,可以通过调用本章描述的函数来实现。

清单 1.1 用来改变当前驱动器或当前目录的菜单驱动程序 dirutil.c

```
// -----
// dirutil.c: 允许用户改变当前驱动器或当前目录的菜单驱动程序
// -----
#include <stdio.h>
#include <graph.h>
#include <direct.h>
#include <process.h>
#include <conio.h>
#include <ctype.h>

static int curdrive;
static char dirname[80], drivename[8];
// -----
void main(void)
{
    int done = 0, c;
    char outbuf[80];

    while (! done)
    {
        // 显示当前驱动器及当前目录
        curdrive = _getdrive();
        _getcwd(dirname, 80);
        sprintf(drivename, "%c", curdrive + 'A' - 1);

        _clearscreen(_GCLEARSCREEN);
        _settextwindow(10, 10, 19, 70);
        _settextposition(1, 1);

        // 显示驱动器字母
        sprintf(outbuf, "Current Drive:    %s\n", drivename);
        _outtext(outbuf);

        // 显示当前目录
        sprintf(outbuf, "Current Directory: %s\n", dirname);
        _outtext(outbuf);

        _settextposition(5, 1);
        _outtext(" 1  Change Drive\n");
        _outtext(" 2  Change Directory\n");
        _outtext(" 0  Exit");

        _settextposition(9, 1);
        _outtext("Enter selection:");

        // 读取按键
        c = _getche();

        // 处理按键代表的命令
        switch(c)
        {
            case '0':
                exit(0);

            case '1':
                // 改变驱动器
                _settextposition(9, 1);
                _outtext("Enter drive letter:");
                c = _getche();
                _chdrive(toupper(c) - 'A' + 1);
        }
    }
}
```

```

        break;

        case '2':
            // 改变当前工作目录
            _settextposition(9,1);
            _outtext("Enter directory name:");
            gets(dirname);
            if(_chdir(dirname) != 0)
                _outtext("\nError changing directory!");
            break;
    }
}

```

要建立这个 `dirutil` 程序, 可以用 Visual Workbench 创建一个工程, 此工程应该是一个 DOS 可执行程序(不需要包含 MFC 支持, 但如果已经建立了 DOS MFC 库, 那么包含 MFC 也不会发生错误)。在工程中要包含 `dirutil.c` 程序, 必须使此程序为 DOS 可执行程序; 如果使此程序为 QuickWin 程序, 它也能运行, 但是用户将看不到运行结果。

当运行 `dirutil` 程序时将显示如下画面:

```

Current Drive:      D
Current Directory:  D:\MCPB\EX\C\CHI

1 Change Drive
2 Change Directory
0 Exit

Enter selection: 0

```

可以通过按“0”键退出此程序。选项 1 和选项 2 可以改变当前驱动器和当前目录。当任何一项被改变后, 程序将更新在菜单上显示的当前驱动器和目录名。

1.2.2 改变文件属性

作为使用文件和目录管理函数的另一个例子, 考虑如何使文件变成只读文件, 以防被意外删除或覆盖。清单 1.2 给出了程序 `protect.c`, 此程序利用 `_chmod` 函数使文件变为只读文件。一旦编译并建立了 `PROTECT.EXE` 文件, 就可以用如下命令使某个文件(此例中是 `protect.obj`)变为只读文件:

```
protect protect.obj
```

清单 1.2 使文件变成只读文件以防意外删除的程序 `protect.c`

```

// -----
// protect.c: 使文件变成只读文件, 以防意外删除或覆盖
// 用法: protect 文件名
// -----
#include <stdio.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>
// -----

void main(int argc, char * * argv)
{
    if(argc < 2)
    {
        printf("Usage: %s <pathname>\n", argv[0]);
    }
    else

```

```

    {
        if(_chmod(argv[1],S_IREAD) == -1)
            perror("Error in _chmod");
        else
            printf("%s protected\n",argv[1]);
    }
}

```

一旦某个文件被 PROTECT.EXE“保护”起来,只要用户想删除此文件,就会看到以下错误信息:

Access denied

要恢复保护前的状态,需要另外一个叫作 unprot 的应用程序,它允许重新对文件执行读写操作。清单 1.3 给出了完成此任务的程序 unprot.c。

清单 1.3 允许对文件进行读写的程序 unprot.c

```

// -----
// unprot.c: 允许对被“保护”起来的文件执行读写操作
// 用法: unprot 文件名
// -----
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
// -----
void main(int argc,char * * argv)
{
    if(argc < 2)
    {
        printf("Usage: %s <pathname>\n",argv[0]);
    }
    else
    {
        if(_chmod(argv[1],S_IREAD|S_IWRITE) == -1)
            perror("Error in _chmod");
        else
            printf("%s unprotected\n",argv[1]);
    }
}

```

1.3 函数参考

1.3.1 _access

MSC6	MSC7	VC++	QC2.5	QC-WIN	TC2	TC++	BC++2	BC++3	ANSI	POSIX	UNIX V	DOS	QWIN	WIN	WINDLL
x	x	x	x	x	x	x	x	x		x	x	x	x	x	x

关于函数名前下划线的含义,见本章前面的有关注释。

概要

```

#include <io.h>

int _access(
    const char * name,    // 要检查的文件的路径名
    int mode);           // 文件是否可用此模式访问

```