

# HOPE

6.0 版

## Turbo C TOOLS 源程序剖析

## Turbo C 高级程序设计实例

(共二册)

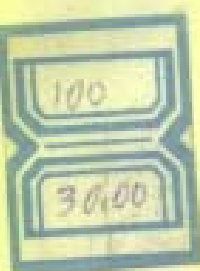


北京希望电脑公司

1

# HOPE

6.0 版



## Turbo C TOOLS 源程序剖析

## Turbo C 高级程序设计实例

(共二册)



北京希望电脑公司

2

# Turbo C TOOLS 6.0 源程序剖析

## Turbo C 高级程序设计实例(上)

李 文 编译

- 域编辑编程扩展
- 扩展的文件操作
- 帮助系统程序设计
- 用 C 进行中断服务子程序的设计
- 内存驻留程序设计的插入码
- 高级键盘管理程序设计
- 内存管理高级程序设计
- 选单程序设计
- 鼠标器编程
- 打印机编程
- 屏幕(视频)编程
- 字符串处理
- 实用函数和宏
- 窗口系统高级程序设计
- 创建扩展函数库



中国科学院希望高级电脑技术公司

一九九一年七月

# Turbo C TOOLS 6.0 源程序剖析

## Turbo C 高级程序设计实例(下)

李 文 编译

- 域编辑编程扩展
- 扩展的文件操作
- 帮助系统程序设计
- 用 C 进行中断服务子程序的设计
- 内存驻留程序设计的插入码
- 高级键盘管理程序设计
- 内存管理高级程序设计
- 选单程序设计
- 鼠标器编程
- 打印机编程
- 屏幕(视频)编程
- 字符串处理
- 实用函数和宏
- 窗口系统高级程序设计
- 创建扩展函数库



中国科学院希望高级电脑技术公司

一九九一年七月



# 前言

Turbo C TOOLS 6.0 在 5.0 版本的基础上, 提供了更丰富的函数。本书详细介绍了 Turbo C Tools 6.0 中每一个函数的使用方法, 介绍的函数比在手册中介绍的函数还要多。并提供了字符串转换、屏幕操作、窗口、选单、编辑器、帮助系统、鼠标器、键盘、文件、打印机、内存管理、中断服务、插入码等所有函数的源程序, 其中包括珍贵的虚拟窗口、虚拟选单、编辑器、帮助系统和鼠标器几类重要的函数的源程序。

通过函数包, 本书还介绍了用 Turbo C 进行字符串转换、屏幕操作、窗口、选单、编辑器、帮助系统、鼠标器、键盘、文件、打印机、内存管理、中断服务、插入码和内存驻留程序(TSR)程序设计的要点和技巧。

函数之多, 俯首拾来, 皆成程序。这些函数能够帮助用户设计出爽心悦目, 而且功能强大的程序。源程序能让您进一步探索这些函数或对它进行有效的开发, 仔细研读这些子程序, 能更好地使用这些函数, 开发自己的实用函数包, 极大地提高实际的编程水平。

## 本书的组织

本书分成十三章, 和 A、B 和 C 三个附录。

第一章 Turbo C 工具扩展源程序的使用说明

第二章 域编辑编程扩展

第三章 文件操作

第四章 帮助系统程序设计

第五章 用 C 进行中断服务子程序的设计

第六章 内存驻留程序设计的插入码

第七章 高级键盘管理程序设计

第八章 内存管理高级程序设计

第九章 选单程序设计

第十章 鼠标器编程

第十一章 打印机编程

第十二章 屏幕(视频)编程

第十三章 字符串处理

第十四章 实用函数和宏

第十五章 窗口系统高级程序设计

附录 A 使用的汇编头文件

附录 B 创建功能强大的函数库的批命令

附录 C 函数包使用的头文件

在本书的编译、整理过程中, 得到了多方面的关心和帮助, 特别是得到了希望公司经理秦人华同志的大力支持和热情鼓励, 在此编者表示衷心的感谢。

# 目 录

## 前言

第一章 Turbo C 工具源程序的使用说明 .....	1
函数包介绍 .....	1
函数的分类 .....	1
软件需求 .....	2
硬件需求 .....	2
支持的编译器版本和内存模式 .....	2
警告和使用注意事项 .....	3
异常错误及解决方法 .....	3
提取源代码 .....	3
打印源代码 .....	4
从源代码中取消 TAB .....	4
重建函数库 .....	4
重建示例程序 .....	5
C 和汇编模块及宏 .....	5
函数库建造说明 .....	7
利用数包的程序设计 .....	7
编程风格、编译警告和头文件 .....	7
修改 Turbo C TOOLS .....	8
源代码约定 .....	8
插入码函数程序设计示例 .....	8
第二章 域编辑编程扩展 .....	12
域编辑函数的种类 .....	12
数据项和编辑 .....	12
编辑键定义 .....	12
编辑操作和缺省的按键分配 .....	12
域编辑函数源程序 .....	14
EDBASE      编辑和返回一个用户响应 .....	14
EDBUFFER     对编辑缓冲区执行编辑动作 .....	22
EDCHGKEY     加入或修改 EDFIELD 和 WNFIELD 所认可的一个按键 .....	30
EDFIELD      对屏幕上的一个域进行编辑 .....	32
EDINITKY     安装 EDFIELD 和 WNFIELD 所接受的缺省按键 .....	34
EDREDUCE     把连续的空白转换成单个空格 .....	36
EDREMKEY     删去 EDFIELD 和 WNFIELD 所接受的一个按键 .....	37
EDRETINF     返回当前光标和视屏信息 .....	38
EDRETKEY     报告一个按键的编辑动作 .....	39
EDSETCUR     设置光标的尺寸和位置 .....	40
EDWRAP       向编辑区写入字符时, 带有整字换行 .....	41
EDWRRECT     向屏幕写入一长方形图形 .....	42
EDZAPKEY     删去 EDFIELD 和 WNFIELD 所接受的按键队列 .....	43
编辑函数程序设计示例 .....	43
KEYCTRL.C 键控制函数的样例 .....	44
ENTRYEDT.C 是面向窗口多编辑域的程序 .....	45

<b>第三章 文件操作</b>	<b>52</b>
文件管理函数(FL)	52
文件管理函数的种类	52
维护卷标	52
杂类	52
文件操作函数源程序和使用参考	52
FLDOLOCK 对已打开文件一个文件段上锁或解锁,必要时等待	52
FLFLUSH 迫使挂起的文件输出写到磁盘上	54
FLGETDTA 返回磁盘传送地址	55
FLLOCK 对已打开文件的一个文件段上锁或解锁	55
FLNORM 验证一个文件名,把它转化成标准形式	56
FLPROMPT 从标准输入中返回一行,可以使用提示信息	66
FLPUTDTA 设置磁盘传送地址(DTA)	67
FLREMOVOL 从给定的磁盘上删除卷标(如果有的话)	67
FLRETVOL 报告给定磁盘驱动器上的卷标	69
FLSETVOL 建立或修改给定磁盘上的卷标	71
文件操作程序设计示例	73
<b>第四章 帮助系统程序设计</b>	<b>75</b>
帮助系统(HL)	75
帮助函数的功能	75
设计帮助窗口	75
编写帮助源文件	75
控制内存分配	77
帮助函数源程序	77
HLCLOSE 释放二叉帮助文件的已存索引	78
HLDISP 从帮助文件读取一段帮助信息,显示在屏幕上,供用户浏览	78
HLFRINDX 释放在给定节点以下的所有帮助索引节点	79
HLLOOKUP 从二叉帮助文件中读取一段帮助信息	80
HLPAS2C 转换一个 Pascal 的字符串为 C 的格式	88
HLREAD 在视口中显示帮助信息,供用户浏览	89
帮助程序示例	94
<b>第五章 用 C 进行中断服务子程序的设计</b>	<b>128</b>
概述	128
ISR 的一般用法	128
建立 ISR	128
驻留程序	128
调用步骤	128
删除	129
按惯例过滤中断	129
从 ISR 中的特殊退出	129
扩展函数源程序	130
ISCALL 对软件中断调用中断服务例程进行模拟	130
ISCURPRC 返回或设置当前执行的进程	134
ISGETVEC 返回一个中断向量	134
isdispat 中断服务子程序调度程序	135
ISINSTAL 安装一个中断服务例程(ISR)	144
ISPREP 预备一个 ISR 控制块	145
ISPUTVEC 设置一个中断向量	147
ISREMOVE 去除一个驻留程序	148

ISRESERV	保留 ISR 所需的动态内存 .....	149
ISRESEXT	中止一个程序但保持驻留 .....	150
ISSENSE	检测一个已安装的中断服务例程(ISR) .....	151
使用中断子程序的程序设计 .....		152
CTLBRK.C 说明 control-break 处理子程序的实现和使用 .....		156
CRITERR.C 说明严重错误处理子程序和使用 .....		157
<b>第六章 内存驻留程序设计的插入码 .....</b>		<b>161</b>
概述 .....		161
插入码实用函数 .....		161
建立一个插入函数 .....		161
调度 .....		161
调用步骤 .....		162
重新调度 .....		162
摘除 .....		162
高级用法 .....		162
使用扩展键盘 BIOS .....		162
暂时使一个插入函数失效 .....		163
选择插入过滤程序 .....		163
防止异步通讯拥塞 .....		163
插入码扩展码源程序 .....		163
IVCTRL	报告本程序中插入控制块的地址 .....	163
IVDETECT	检测已安装的插入函数,即使它被部分覆盖 .....	176
IVDISABL	使一个插入函数失效 .....	181
IVINSTAL	安装一个插入函数 .....	182
IVSENSE	检测一个已安装的插入函数是否是可删除的 .....	192
IVVECS	设置或返回插入过滤程序所使用的中断向量 .....	192
<b>第七章 高级键盘管理程序设计 .....</b>		<b>194</b>
增强键盘 .....		194
键盘函数的功能 .....		194
键盘输入 .....		194
处理超前键入缓冲区 .....		194
处理移位键 .....		194
使用增强键盘 .....		194
使用键控制函数 .....		194
取得键码 .....		195
高级特性:键控制函数 .....		195
调用步骤 .....		195
高级键盘管理函数源程序和使用参考 .....		195
KBEQUIP	检测键盘环境 .....	196
KBEXTEND	选用扩展的或一般的 BIOS 键盘服务 .....	197
KBFLUSH	废弃所有在键盘缓冲区中等待的按键 .....	198
KBGETKEY	等待读入下一个按键 .....	198
KBKCFLSH	通过键控制函数废弃所有的等待按键 .....	199
KBPLACE	在键盘缓冲区中放置一个按键 .....	200
KB POLL	通过一个键控制函数查看下一个等待按键 .....	202
KBQUERY	从标准 IBM 控制台读邓用户的响应 .....	203
KBQUEUE	报告键盘缓冲区总容量及剩余容量 .....	206
KBREADY	检查下一个等待按键 .....	207
KBSCANOF	返回一个字符的键码 .....	209

KBSET	设置移位键的当前状态 .....	210
KBSTATUS	报告移位键的当前状态 .....	211
KBSTUFF	强行将一个字符串送入 BIOS 超前键入缓冲区 .....	212
KBWAIT	等待并通过键控制函数读取下一个按键 .....	214
CMKEY.C 键盘宏程序示例 .....		215
<b>第八章 内存管理高级程序设计 .....</b>		<b>220</b>
内存管理函数 .....		220
内存管理函数源程序 .....		220
MMCTRL	读取 DOS 内存控制块 .....	220
MMFIRST	报告第一个内存块的地址 .....	221
MMSIZE	报告一个程序的尺寸 .....	222
<b>第九章 选单程序设计 .....</b>		<b>223</b>
选单函数功能概述 .....		223
选单函数的种类 .....		223
建立、显示和释放选单 .....		223
定义标准选项和按键 .....		223
定义 Lotus 形式的选项 .....		223
使用鼠标器的准备工作 .....		223
读取用户的反应 .....		223
亮条操作 .....		223
高级选单特性 .....		223
按键动作和亮条移动 .....		224
缺省的键分配 .....		224
控制内存分配 .....		225
选单扩展函数源程序和使用参考 .....		225
MNATR	改变菜单在项的属性 .....	225
MNCREATE	建立一个包含单信息的选单结构和窗口 .....	227
MNCREATO	分配并创建一个菜单结构 .....	227
MNDEFKEY	增加约束菜单的缺省键 .....	228
b_mndefkey	包含 MNDEFKEY 使用的缺省键表 .....	229
MNDLITMS	释放选项表中所有项的内存 .....	231
MNDLKEYS	释放键表中使用的所有的键的内存 .....	232
MNDSPY	在同尺寸视口中显示一个选单 .....	232
MNDSTROY	从屏幕上取消一个选单, 废弃其数据结构 .....	233
MNFINDSL	找到给出起始坐标的菜单的第一可选项 .....	234
MNHILITO	移动或删除菜单的高亮条或项描述 .....	236
MNHILITE	移动或取消选单亮条及选项说明 .....	238
MNITEM	插入、修改或删除一个选项 .....	239
MNITMKEY	向选单加入一个选项, 为它分配选择字符 .....	244
MNKEY	加入、修改或取消一个选单的键分配 .....	245
MNLITEM	加入、修改或取消一个 Lotus 形式的选项 .....	249
MNLITKEY	加入一个 LOTUS 形式的选项, 为它分配选择字符 .....	250
MNLREAD	通过 Lotus 形式的选单读入一个用户响应 .....	252
MNREAD	从给定的开始行和列使用亮条读取菜单 .....	253
MNMCHITM	匹配一说明条件的菜单项 .....	274
MNMCHKEY	匹配一说明的键 .....	275
MNMOUSE	加入、修改或删除一个选单认可的鼠标器事件 .....	277
MNMSTYLE	设立一个标准选单鼠标器格式 .....	281
MNREAD	读取来自选单的用户响应 .....	284

MNVALMN0	检查 BMENU 结构的有效性 .....	306
MNVDISP	在视口中显示一个虚拟选单 .....	307
示例程序	.....	308
源程序(MENU.C)	.....	308
源程序(PULLMENU.C)	.....	321
<b>第十章 鼠标器编程</b>	.....	<b>333</b>
鼠标器事件的种类	.....	333
访问鼠标器状态	.....	333
控制鼠标器位置	.....	333
控制鼠标的外观	.....	333
对鼠标器硬件中断的反应	.....	333
其它鼠标器操作	.....	333
处理鼠标器中断	.....	334
调用屏蔽: 相关事件组	.....	334
调用步骤	.....	334
鼠标器编程函数源程序函数使用参考	.....	334
MOAVOID	在指定区域中隐藏鼠标 .....	335
MOBUTTON	报告鼠标器按钮的按下/释放历史 .....	336
mocatch	捕获鼠标器按钮的按下与释放 .....	338
MOCHECK	检查最近发生的鼠标器事件 .....	342
MOCURMOV	移动鼠标 .....	348
MOEQUIP	检查鼠标器驱动程序的存在 .....	349
MOGATE	鼠标器驱动程序的人口 .....	350
MOGETMOV	报告自上次查询以来物理鼠标器的移动 .....	351
MOGRAPH	设置鼠标器图形方式光标 .....	351
MOHANDLR	安装或摘除中断处理程序 .....	352
MOHARD	设置鼠标器硬件字符方式光标 .....	355
MOHIDE	隐藏或显示鼠标 .....	356
moinst	无条件安装鼠标器中断处理子程序 .....	357
MOJUMP	设置鼠标器加速阈值 .....	357
MOLITYPEN	使鼠标器光笔模拟有效或失效 .....	358
MOPRECLK	安装或删除 MOCHECK 所有的内部例程 .....	359
MORANGE	设置鼠标器范围界限 .....	360
MORESET	重置鼠标器驱动程序 .....	361
MOSOFT	设置鼠标器软件字符方式光标 .....	361
MOSPEED	设置鼠标器速度 .....	362
MOSTAT	报告鼠标器位置和按钮状态 .....	363
movars	一些 MO 函数使用的全局变量:	
	b_mocatch, b_momask, b_modispat, b_mohanmask .....	363
MOUSEHAN.C	鼠标中断处理子程序演示程序 .....	364
<b>第十一章 打印机编程</b>	.....	<b>368</b>
BIOS 打印机接口	.....	368
与 PRINT 程序的接口	.....	368
打印机控制函数源程序	.....	368
PRCANCEL	删除假脱机打印队列中一个或全部文件 .....	368
PRCHAR	通过 BIOS 向打印机发送一个字符 .....	369
PRERROR	返回解释错误代码的字符串, 这些错误代码来自 PR(打印机)函数 .....	370
PRGETQ	报告假脱机打印队列中的一个文件名 .....	371
PRINTT	通过 BIOS 初始化一个打印口 .....	373

PRINSTLD	检查驻留式假脱机打印系统 PRINT 是否已安装 .....	373
PRSPOOL	将一个文件提交给假脱机打印系统 .....	374
PRSTATUS	通过 BIOS 报告打印机的状态 .....	375
<b>第十二章 屏幕(视频)编程 .....</b>		<b>376</b>
屏幕操作函数的种类 .....		376
读取屏幕方式信息 .....		376
选择显示设备和方式 .....		376
在显示页之间切换 .....		376
控制/读取光标形状和位置 .....		376
清除和滚动 .....		376
常规的屏幕写入 .....		376
写入一个矩形区域 .....		377
屏幕读取 .....		377
调色板支持 .....		377
对直接视频访问的支持 .....		377
保存和恢复整个显示状态 .....		377
强制快速屏幕访问 .....		378
屏幕(视频)控制函数源程序 .....		378
SCAPAGE	显示(激活)一个显示页 .....	378
SCATTRIB	用指定的显示属性显示一个字符的拷贝 .....	379
SCBLINK	选择前景闪烁或背景亮度 .....	379
SCBORDER	设置当前显示屏幕的边界颜色 .....	380
SCBOX	用图形字符在屏幕上画一个方框 .....	381
SCCHGDEV	切换至彩色或单色显示 .....	384
SCCLRMSG	清除屏幕上的消息 .....	386
SCCURSET	移动当前显示页上的光标 .....	387
SCCURST	返回当前显示页上的光标位置和尺寸 .....	387
SCEQUIP	检测显示硬件环境 .....	388
SCGETVID	记录整个显示状态 .....	397
SCMODE	返回屏幕的显示方式 .....	397
SCMODE4	设置方式 4 色板和背景颜色 .....	399
SCNEWDEV	选择并设置显示设备, 设置字符行数 .....	400
SCPAGE	设置当前显示页 .....	408
SCPAGES	返回显示页的数目 .....	408
SCPALI	定义一个 EGA、VGA、或 MCGA 色板颜色 .....	410
SCPALETT	定义 EGA、VGA 或 MCGA 颜色的整个色板 .....	411
SCPCLR	清除当前显示页 .....	413
SCPGCUR	设置当前页的光标尺寸 .....	414
SCREAD	从屏幕读取一个显示字符及其属性 .....	416
SCRESTPG	恢复一个显示页 .....	417
SCROWS	返回屏幕的字符行数 .....	418
SCSAVEPG	保存一个显示页 .....	419
SCSETVID	恢复整个显示状态 .....	420
SCTTYWIN	以 TTY 方式向矩形区域写入一个字符 .....	422
SCTTYWRT	以 TTY 方式向屏幕写一个字符 .....	425
SCWRAP	以 TTY 方式向一个矩形中写入一个字符串, 带有整字换行 .....	426
SCWRITE	在屏幕上显示一个字符的多个拷贝 .....	430
VIATRECT	改变屏幕上一个矩形的属性 .....	430
VIDSPMSG	显示一条消息 .....	432

vidirec0	直接从或向视屏适配读取或写入长方形的字符 .....	432
VIHORIZ	在当前显示页上水平滚动正文列 .....	445
VIPTIR	将屏幕位置转换成内存地址 .....	447
VIRDRECT	读取屏幕上一个矩形区域中的内容 .....	448
VIRDSECT	将屏幕的一个区域读入内存中更大的矩形区域 .....	449
VISCROLL	垂直滚动当前显示页上的正文行 .....	450
VIWRRECT	向当前显示页上一个矩形区域中写入数据 .....	450
VIWRSECT	显示矩形缓冲区中的一个矩形段 .....	451
<b>第十三章 字符串处理 .....</b>		<b>453</b>
字符串函数的种类 .....		453
对字符串的一部分进行填充 .....		453
查找 .....		453
字符转换 .....		453
处理 tab 字符 .....		453
字符串处理扩展函数源程序 .....		453
STIEXPAN	将 tab 字符转换为空格 .....	456
STPJUS	在域中将一个字符串左右对齐或居中 .....	458
STPTABFY	用 tab 字符替换空格 .....	459
STPXLT	用翻译表翻译一个字符串 .....	461
STSCHIND	查找字符串中的一个字符, 返回它的位置 .....	462
<b>第十四章 实用函数和宏 .....</b>		<b>463</b>
指针和地址 .....		463
内存传送 .....		463
探测指针错误 .....		463
数据压缩 .....		463
ANSI.SYS 支持 .....		463
程序环境信息 .....		463
口 I/O .....		464
时钟访问 .....		464
关闭或开放中断 .....		464
扬声器控制 .....		464
算术计算 .....		464
在长字节、字、字节和半字节之间转换 .....		464
设置范围界限 .....		465
数据类型操作 .....		465
各种输出 .....		465
实用函数源程序 .....		465
UTANSI	检测、关闭或重新开放 ANSI.SYS .....	466
UTCHKNIL	报告无效指针赋值, 使程序夭折 .....	469
UTCRT	取得 DOS 临界段标志的地址 .....	470
UTCTLBRK	设置或返回 Ctrl-Break 检查的状态 .....	470
UTDOSRDY	报告 DOS 服务是否可用 .....	471
UTGETCLK	报告自午夜以来 BIOS 计时脉冲的个数 .....	471
UTINTFLG	打开或关闭硬件中断 .....	472
UTMODEL	报告 IMB 型号和子型号及 BIOS 版本 .....	473
UTMOVME	不受限制地从内存或向内存任何位置拷贝数据 .....	475
UTNORM	使一个指针具有最小的偏移值 .....	477
UTNULCHK	检测无效的指针赋值 .....	477
UTOFF	返回一个地址的偏移部分 .....	478



UTPEEKB	从任意地址读取一个字节 .....	479
UTPEEKN	从任意地址读取多个字节的数据 .....	479
UTPEEKW	从任意地址读取一个字 .....	479
UTPLONG	将一个指针转换为指向 20 位物理地址的指针 .....	480
UTPOKEB	在任意地址存放一个字节的的数据 .....	480
UTPOKEN	在任意地址存放多个字节的数据 .....	480
UTPOKEW	在任意位置写入一个字的数据 .....	481
UTSAFCPY	以确保不跨越段界的方式拷贝数据 .....	481
UTSEG	返回任意地址的段部分 .....	482
UTSLEEP	暂停处理直至经过几个计时脉冲之后 .....	482
UTSPKR	打开或关闭扬声器 .....	483
UTSQZSCN	压缩一个屏幕图象 .....	484
UTTIM2TK	将时间转换为计时脉冲计数 .....	489
UTTK2TIM	将计时脉冲计数转换为 24 小时制时间 .....	489
UTTOFAR	用段和偏移构造一个双字指针 .....	490
UTTOFARU	用一个段和偏移构造一个泛双字指针 .....	491
UTUNSQZ	还原一个压缩屏幕图象 .....	491
实用程序函数设计示例	.....	493
<b>第十五章 窗口系统高级程序设计</b>	.....	<b>498</b>
窗口功能概述	.....	498
窗口函数的种类	.....	498
建立和释放窗口结构	.....	498
显示和删除窗口	.....	498
虚拟窗口	.....	498
窗口输出	.....	498
窗口输入	.....	499
滚动和清除	.....	499
控制属性	.....	499
控制光标	.....	499
控制窗口任选项	.....	499
使用 Turbo C 的字符窗口	.....	500
为 WNREAD 进行窗口输入做准备	.....	500
使用高级窗口特性	.....	500
显示和更新窗口	.....	500
虚拟窗口	.....	500
不可删除的窗口	.....	500
控制内存分配	.....	500
窗口扩展函数源程序	.....	501
WNATRBLK	修改窗口中一个矩形块的属性 .....	501
WNATRSTR	改变窗口中一片连续位置的属性 .....	502
WNATTR	改变当前窗口的属性 .....	504
WNCHGEVN	加入或修改 WNREAD 认可的一个用户响应 .....	504
WNCREATE	建立一个窗口结构 .....	505
WNCOVER	如果被一给定的长方形区域遮盖, 则标为低层窗口 .....	506
WNCREATU	分配和创建一窗口结构 .....	507
WNCURMOV	移动当前窗口的光标 .....	509
WNCUPRPOS	返回当前窗口的光标位置 .....	510
WNCURSOR	激活一个窗口光标 .....	511
WNDSPRAY	在同尺寸视口中显示一个窗口 .....	512

WNCURTRK	移动窗口的光标, 如果必要的活调整数据区源 .....	513
WNDSTROY	废弃一个窗口结构 .....	515
WNERORR	记录窗口或选单的系统错误 .....	516
WNFIELD	对窗口中的一个域进行编辑 .....	516
WNFORGET	从屏幕位置脱离窗口, 但并不清除屏幕 .....	518
WNGETIMC	读取屏幕长方形区域的图象 .....	519
WNGETOPI	读取窗口信息项或状态 .....	521
WNHIDE	删除窗口但仍与视频显示设备和显示页联连 .....	524
undisp	暂时删除(不显示)屏幕上由一长方形区域遮盖的窗口 .....	527
cocovr	标志非覆盖的窗口 .....	528
upcovr	检查高层窗口是否覆盖长方形区域 .....	529
redisp	重新显示指定窗口以上的被暂时隐藏删除的窗口 .....	530
WNHORIZ	水平滚动当前窗口 .....	531
WNINITEV	为 WNREAD 安装缺省的窗口事件 .....	532
WNCMOVE	移动窗口的光标 .....	535
WNSETWIN	设置编译器 native 文本窗口的尺寸 .....	536
WNCHKDM	检查编译器的 native 文本窗口的尺寸和大小 .....	537
WNGETATR	得到当前 native 文本窗口的属性 .....	537
WNSETATR	设置当前 native 文本窗口的属性 .....	538
WNNEEDUP	标志一窗口为无用, 如果可能进行更新 .....	538
WNNUPBLK	标志一窗口为无用, 如果可能更新一部分 .....	539
WNORIGIN	在视口中称动窗口 .....	540
WNOVRLAP	报告两个长方形区域是否覆盖 .....	542
WNPGADD	增加窗口到在一设备一或显示页上显示的窗口的链表上 .....	542
WNPGREM	从显示在一设备或显示页上的窗口链表上删除 .....	543
WNPIMBLK	输出窗口的一部分到屏幕 .....	544
WNPOLL	查询属于特定集的键盘或鼠标事件一次 .....	546
WNPRINTF	向当前窗口写入一个格式化的字符串 .....	548
WNPUTBOR	沿着长方形区域显示一边界 .....	549
WNPUTSEN	显示窗口的 sensor 表 .....	552
WNQUERY	返回经窗口得到的来自用户的字符串 .....	555
WNRDBUF	读取当前窗口中一片连续位置的内容 .....	558
WNREAD	允许用户在虚拟窗口中浏览 .....	560
POLL_EVENT	查询窗口的鼠标或键盘事件一次 .....	566
WNREDRAW	重现显示在当前显示页上的全部窗口 .....	571
WNREMEVN	删去 WNREAD 接受的一个用户响应 .....	573
WNREMOVE	从屏幕上取消一个窗口 .....	575
WNRESPRV	恢复窗口先前的屏幕内容 .....	575
WNRETEVN	从窗口列表中返回一窗口事件记录 .....	576
WNRETINF	返回窗口当前光标和尺寸的信息 .....	578
WNREVUPD	用显示的数据更新已保存的窗口图象 .....	579
WNSCRBLK	在窗口中以任意方向滚动一个矩形区域 .....	582
WNSCRLBR	向窗口加入一个滚动箭头 .....	584
WNSCROLL	垂直滚动当前窗口 .....	589
WNSELECT	选择用于 I/O 的窗口 .....	589
WNSETBUF	为 WNPRINTF 分配内部缓冲区 .....	590
WNSETCUR	设置当前窗口的状态、尺寸和位置 .....	591
WNSETOPT	设置窗口控制项 .....	592
WNSHOBLK	在视口间隙中显示一个窗口数据块 .....	597
WNSHOBLK	在窗口视区内显示一长方形区域的数据 .....	600

WNUNHIDE	重新显示隐藏的窗口 .....	603
WNUPDATE	将挂起的输出写入窗口 .....	606
WNVALEVO	验证窗口事件 .....	607
WNVALNOO	验证窗口节点 .....	607
WNVALWIO	验证 BIWINDOW 结构 .....	608
WNVDISP	在视口中显示一个虚拟窗口 .....	608
WNWRAP	以 TTY 方式向当前窗口写入一个字符串, 常有整字换行 .....	612
WNWRBUF	向当前视口中的一片连续位置写入字符 .....	614
WNWRRECT	写入窗口的一个矩形区域 .....	616
WNWRSTR	以 TTY 方式向当前窗口写入一个字符串 .....	618
WNWRSTRN	以 TTY 方式向窗口写入一个字符串, 带有任选项 .....	619
WNWRTTY	以 TTY 方式向当前窗口写入一个字符 .....	623
WNWRTTYX	写一字符到一 TTY 方式的窗口 .....	623
WNZAPEVN	删除 WNREAD 认可的窗口事件表 .....	624
WNZAPSEN	释放窗口的 sensor 列表 .....	625
附录 A 使用的汇编头文件 .....		626
COMP T2S.MAC .....		626
COMP T2M.MAC .....		626
COMP T2C.MAC .....		627
COMP T2L.MAC .....		627
COMP T2H.MAC .....		628
BEGINASM.MAC 编译依赖的符号和宏 .....		628
附录 B 创建功能强大的函数库批命令 .....		634
创建库文件的 LIB 程序的响应文件(LIBRESP) .....		636
附录 C 函数包使用的头文件 .....		639
BEDIT.H	域编辑函数的头文件 .....	639
BFILES.H	文件和目录函数的头文件 .....	642
BCENWIN.H	禁止提供编译器本地文本窗口支持的头文件 .....	644
BHELP.H	帮助函数的头文件 .....	644
BINTERV.H	插入函数的头文件 .....	649
BINTRUPT.H	中断服务函数的头文件 .....	653
BKEYBRD.H	BIOS 键盘函数的头文件 .....	656
BKEYS.H	定义键码的头文件 .....	659
BLAISE.H	所有函数的头文件 .....	676
BMEM.H	内存管理函数的头文件 .....	676
BMENU.H	菜单函数的头文件 .....	677
BMOUSE.H	鼠标函数的头文件 .....	684
BNATVWN.H	支持编译器本地文本窗口的头文件 .....	688
BPRINT.H	支持打印机管理的头文件 .....	688
BSCREENS.H	屏幕函数的头文件 .....	689
BSTRINGS.H	字符串函数的文件 .....	694
BUTIL.H	实用函数的头文件 .....	694
BVIDEO.H	直接存取视屏硬件函数的头文件 .....	702
BWINDOW.H	窗口函数的头文件 .....	703

# 第一章 Turbo C 工具源程序的使用说明

## 本章的主要内容

- 函数包介绍
- 软件和硬件要求
- 支持的编译器版本和内存模式
- 警告和使用注意事项
- 错误及解决方法提取源代码
- 打印源代码
- 从源代码中消除 tab
- 重建函数库
- 重建示例程序
- C 和汇编模块及宏函数库构造说明

## 函数包介绍

函数包中的函数能用于 IBM PC 系列的 Turbo C 程序中。它们提供:

- 屏幕和键盘的快速、多功能控制;
- 创建、显示和使用多窗口的窗口管理;
- 使用移动高亮条的用户界面的菜单;
- 帮助系统使产生正文的过程自动化;
- 支持与标准 Microsoft 兼容的鼠标器驱动器模块接口的函数;
- 完全用 C 语言编写中断服务程序的能力。这些程序处理硬件和软件中断,并调用 DOS 服务;
- 插入码:按一热键,或在给定的时间间隔里调度 C 函数执行的能力;
- 创建和删除驻留在内存中的程序(也叫 TSR 应用)的能力;
- 常驻伪脱打印机制的界面;
- 具有强大功能的常规字符串函数。

Turbo C TOOLS 经过精心设计,支持许多易使用的重要的功能。整个源代码组装在一起,使用户能调整此程序包适应自己的需要。另一方面,用户使用 Turbo C TOOLS 时不必考察其源代码,除非有意这么做。若用户需要调整其中的功能以适应自己的特殊需要,将会看到 Turbo C TOOLS 的源代码是易于理解、修改和安装的。

Turbo C 是软件开发的优秀编译器。写得好的 Turbo C 程序是可靠的、有效的、可读性好的、易于维护的。并且 Turbo C 代码量少,运行速度快。然而,许多功能需要利用 IBM PC 环境中的高级硬件和软件特性才能实现,标准 Turbo C 库不支持这些高级功能。而 Turbo C TOOLS 支持许多附加的高级功能。

## 函数的分类

函数根据各工具解决的问题或相关问题集分类。共有 15 类:

前缀	类别
ST	字符串函数
SC	屏幕处理
VI	直接视频访问
GR	图形
WN	窗口管理
MN	菜单
HL	帮助系统
MO	鼠标器支持

KB 键盘控制  
 FL 文件和目录管理  
 PR 打印机和 PRINT.COM 伪机打印机界面  
 MM 存贮管理  
 IS 中断服务支持  
 UT 实用函数和宏

每个函数的名字以分类字母为前缀。例如，屏幕处理类(前缀 SC)包括在屏幕上面框的函数 scbox()。象所有 Turbo C TOOLS 函数一样，调用它时使用它的小写字母名字，它在手册中和其他文档中的名字为大写 SCBOX()。

虽然，每类函数解决一个特定领域的问题，但 Turbo C TOOLS 中有些函数依赖于别的函数。我们力图使各类函数自包含，以便你不用研究整个产品，就可以学习使用其中的一类函数。然而，有些领域的问题不可避免的需要别的领域的知识。例如，为了充分发掘窗口管理函数的功能，有必要了解屏幕处理函数的功能和限制。

## 软件需求

有些 Turbo C TOOLS 函数是用汇编语言写的。一般不需汇编器，除非修改这些特殊函数。若一定要这样做，建议用 Microsoft 的 3.00 以上版本的宏汇编包。我们编译 Turbo C TOOLS 汇编语言时也使用它。

为了推广和维护目标模块库(.lib)，需要有与 Microsoft 库管理软件 LIB 兼容的库管理程序。这也由 Microsoft 宏汇编包提供。

为了推广可执行.exe 文件，我们用到了 Turbo C 编译中的 TLINK 程序。

还需要 DOS 2.00 或更高版本，尽管有些函数只适用于 3.00 以上版本，但大部分函数在 DOS 的所有版本中的行为相同。需要 DOS 3.00 以上版的函数在说明中特加以强调。

## 硬件需求

许多 Turbo C TOOLS 程序能在运行 Microsoft 操作系统(MS-DOS) 2.00 以上版本的机器上正确执行。(IBM PC-DOS 是 Microsoft MS-DOS 的改写版。)事实上，ST、FL、MM 或 IS 类中的程序都不依赖于任何特定的 IBM 硬件特性。

## 支持的编译器版本和内存模式

这个版本的 Turbo C TOOLS 支持 Turbo C 版本 1.00、1.50 和 2.00。

**Tiny(T) 模式：**Turbo C TOOLS 不正式支持 T 模式，然而 S 模块库(TCT\_T2S.LIB)中的很多模块在 T 模式程序中可以正确地工作。

**Huge(H) 模式：**许多使用 H 模式的程序可以使用 L 模式库(TCT\_T2L.OBJ)，特别是如果这些程序不包含中断服务例程(ISR)或插入码。然而，如果支持大于 65,535 字节的数据对象或者程序工作失常，则应使用 H 内存模式重新编译这个库。参见下面的“重建函数库”。

下面的每个文件都针对于特定的内存模式，同一个文件持 Turbo C 1.00、1.50 和 2.00 几种版本，但在下面的“警告和使用注意事项”中列出了几个例外。下表的栏目解释如下：

内存模式	函数库	“原始窗口”	COMPILER.MAC	“不支持鼠标器”
S	TCT_T2S.LIB	NW_T2S.OBJ	COMP_T2S.MAC	BNOMOUSE.OBJ
M	TCT_T2M.LIB	NW_T2M.OBJ	COMP_T2M.MAC	BNOMOUSE.OBJ
C	TCT_T2C.LIB	NW_T2C.OBJ	COMP_T2C.MAC	BNOMOUSE.OBJ
L	TCT_T2L.LIB	NW_T2L.OBJ	COM_T2L.MAC	BNOMOUSE.OBJ

**内存模式：**它影响程序中数据和代码的最大长度以及数据和代码指针的特性。在 Turbo C 集成环境(TC.EXE)中，通过指明 Options/Compiler/Model 开关可以指定内存模式。缺省的模式是 S，除非您保存了一个不同的配置。对于命令行版本(TCC.EXE)，利用 /m 命令行开关能够指定内存模式。缺省值是 S 模式。

函数库：它们是所有 Turbo C TOOLS 函数的已编译的版本。要想把它们与您的程序相链接，需在编译器、链接器命令行或 project 文件中指明适当的库文件。

口操作与 Turbo C 字符窗口结合起来。这个目标文件需要 Turbo C 版本 1.50 或更高版本。

COMPILER.MAC 文件：它为用汇编语言写的 Turbo C 模块指明内存模式。如果重新汇编任何汇编模块，首先应将所需的文件拷贝到 COMPILER.MAC。

“不支持鼠标器”文件：与 BNOMOUSE.OBJ 相链接的程序不包含鼠标器函数 MOCHECK、MOHIDE 和 MOPRECLK。如果您的程序使用选单、WNREAD 或帮助系统并且您不希望使用鼠标器，则用 BNOMOUSE.OBJ 链接可以使程序小一些。请注意，同一个 BNOMOUSE.OBJ 文件支持所有的内存模式。

## 警告和使用注意事项

ISR 中的栈检查——对于 Turbo C 版本 2.00 和更高版本，在 T、S 和 M 内存模式下的中断服务例程 (ISR) 或插入函数中不允许做栈溢出检查，在所有受到影响的 ISR 和插入函数所调用的函数中也禁止做栈检查。（注意：仅当对程序进行编译时设置了 0/C/Code/Test Stack 选项（在集成环境中）或指明了 -N 命令行任选项时才会出现栈检查。）

TURBO C 1.0 与 FLPROMPT 及 MOCHECK——如果在 Turbo C 1.0 下程序使用 FLPROMPT 或 MOCHECK，那么您应该在版本 1.0 下重新编译 FLPROMPT.C 或 MOCHECK.C，将它们与程序的 OBJ 文件相链接。（MOCHECK 能够检测鼠标器的点按动作，用于 HLDISP、HLREAD、MNREAD 和 WNREAD。如果程序与 BNOMOUSE.OBJ 链接，则 MOCHECK 被抑制。）

在 DOS 版本 2 下的插入码——在 DOS 版本 2 下插入调度程序不设置当前进程，因而在 DOS 版本下，后台程序中的插入函数应该避免内存分配。另外，这些插入函数在每次被激活时应该打开和重新关闭它们的文件。

在非 IBM DOS 上的插入码——如果在安装插入函数时指明了 IV\_DOS\_NEED，则调用程序在中断 0x24（严重错误）正在进行的情况下将延迟激活插入函数。中断 0x24 忙标志的位置在不同 DOS 版本下有所不同。

滚动单色显示页——BIOS 滚动功能通常检查光标原位置来取得新空行的属性。Enhanced Graphics Adapter 在 Monochrome Display 下支持多个显示页。对于超过页 0 的单色显示页，BIOS 滚动操作可能会出现不正常的情况，新空行的属性将从页 0 而不是从活动页取得，这会影响 SCTTYWIN、SCTTYWRT 和 VISCROLL。

## 异常错误及解决方法

程序异常中止：如果启动模块不能为堆栈和 near 堆分配足够的空间，这时会出现程序异常中止的情况。建议在对全局变量 “\_stklen” 和 “\_heaplen” 进行初始化时试用一个小一些的值。

字符显现的颜色不正确：当 MNDSTROY 和 WNREMOVE 将 Turbo C 字符窗口设置为全屏时，它们并不改变字符的颜色，这样，以后通过 Turbo C 字符窗口函数输出的字符所显示的颜色可能是以前所选窗口的颜色。

鼠标器操作期间出现异步通讯口拥塞：由于 Microsoft mouse 通过硬件中断进行工作，所以鼠标器的移动和按钮的按下及释放有可能在数据到达串口时引起拥塞错误。在这种情况下首先应测试一下不用 Turbo C TOOLS 时是否会出现拥塞。在使用 HLDISP、HLREAD、MOCHECK、MNREAD 或 WNREAD 时将安装 MOCATCH，这个内部函数在一段相当长的时间内将关闭中断。

## 提取源代码

库的源代码以两个自展开的文档程序 TCTSRC1.EXE 和 TCTSRC2.EXE 的形式提供。后面列出了这两个文件的全部内容。

要想提取一个文件的内容，首先应将当前驱动器 and 目录设置为待放置展开文件的驱动器和目录，然后执行自展开文档文件，如下例所示：

```
a:tctsrc1
```

要想有选择地提取文档中的文件，可以使用多义文件名，如下例所示。它提取出所有窗口 (WN) 函数的源代码：

## 打印源代码

Turbo C TOOLS 经过了很好的设计和文档说明, 容易理解, 你不必参阅源代码。然而, 你也许希望能很容易地打印代码。

为了打印源代码, 我们提供了一个叫 **CONCAT** 的程序, 它产生格式化的打印文件。你会发现它是对你的软件库的有益补充。格式化打印文件的每页的顶部是文件名、最后修改日期和时间以及页号。每页的第二行显示 **CONCAT** 格式化此文件的时间。每个输出行以一个换页字符(\14)结束。

为了执行 **CONCAT**, 键入命令:

```
Concat /oFile /ppsize /mmarg /npnum /ddebug
```

其参数的意义如下:

**/oFile** 是由 **oFile** 产生的格式化打印文件的名称。若此文件已存在, 则被冲掉, 若此文件不存在, 则创建之。若没有给出输出文件名, 则在当前目录上打开 **concat.prt** 文件。

**/pPsize** 是每页打印的行数。每页的头部需要 4 行(一个空白行, 2 个头信息行, 和另一个空白行); 因此最小的可接受值是 5。缺省值是 64, 对标准 66 行来说, 允许在每页的顶部有两个空白行。

**/mMarg** 是每个打印行前的空格数。缺省值是 0。最大值是 10。

**/nPnum** 是开始页号。文件的各页顺序编号。每个文件的开始页号的缺省值是 1, 如果 **/n** 没有指定值, 则页编号顺序从 1 开始, 但是再也不为每个文件重新设置开始页号。

**/dDebug** 是纠错信息送到其中的某文件的名称。如果此开关没有指定一文件名, 则自动创建文件 **concat.dbg**。若没有指定此开关, 则不产生纠错文件。

所有的参数都是任选项。如果用到了开关, 则参数可以任何次序出现。如果没有用到开关, 则参数以上述顺序解释。开始字符的斜杠(/)可以用减号(-)代替。

例如, 下面两命令中的每个都能产生输出文件 **foo.prt**, 其格式是每页 64 个输出行, 左边的空边为 10 个字符宽, 首页号为 1:

```
concat foo.prt 64 10 1
```

```
conct foo.prt /n /m 10
```

执行 **CONCAT** 时, 屏幕上显示一欢迎信息, 然后出现提示符 > >。这时你可以键入一个文件的全路径名。此步骤可以重复任意次。一空反应(仅敲 **ENTER** 键)结束 **CONCAT**。

输出文件构造出来后, 可以用 **DOS PRINT** 命令打印之, 或拷贝到 **prn**(打印机)设备上。

**CONCAT** 展开 **tab** 字符成 8 个字符宽的空格, 此约定在 **DOS PRINT** 和 **TYPE** 命令中也用到。

找不到输入文件时显示一警告信息, 但是整个过程仍继续下去。**CONCAT** 在输出文件打不开或有错误写到输出文件中时才结束。

## 从源代码中取消 TAB

源模块将连续的空格压缩为 **tab** 字符, 每隔八个字符一个 **tab** 站位, 并以这种形式保存起来。这里提供的 **EXPAND.COM** 工具程序可以将 **tab** 扩展回复为空格, 这对于从帮助源文件中取消 **tab** 是很有用的。

**EXPAND** 是一个过滤程序, 它从标准输入设备中取得输入, 将输出写至标准输出。然而, 您使用它时, 可以重新定向标准输入和输出, 如下例所示:

```
expand <wnexampl.c>wnexampl.ntb
```

结果文件 **WNEXAMP.NTB** 中的 **tab** 字符被扩展为空格串。

## 重建函数库

系统提供了一个批文件 **BUILDLIB.BAT** 和一个库管理程序响应文件 **LIBRESP**, 使得重建 Turbo C TOOLS 函数库的工作更为容易。批文件作了一些假设。如果这些假设不适合您的环境, 您应该修改这个批文件, 使之适合您的系统。这些假设是:

所有源文件(.C 和 .ASM)及 **LIBRESP** 文件在当前目录中;

**BEGINASM.MAC** 和 **COMP\_T2?.MAC** 在 **MAC** 子目录中;

C 语言头文件(.H)在 **\TURBOC\INCLUDE** 目录中;

Turbo C 命令行编译器(**TCC.EXE**)、Turbo Assembler (**TASM.EXE**) 和 Turbo Librarian

(TLIB.EXE) 在当前目录中或者在 PATH=环境变量所指定的目录中。

下面的命令激活 BUILDLIB

```
buildlib model
```

这里“model”是 s、m、c、l 或 h(小写字母), 用来指明内存模式。批文件扫描类别来编译和汇编源文件(为使重新编译更为容易, 可以只编译某类源文件), 然后激活库管理程序建立该库。例如, 要重建小内存模式库, 可以发出下面的命令

```
buildlib s
```

## 重建示例程序

MAKEXPLS 文件与 Turbo MAKE 工具程序结合使用, 用来重新建立十八个示例程序。它作如下的假设:

示例程序源文件在当前目录中;

所有 Turbo C 和 Turbo C TOOLS 头文件在 \TURBOC\INCLUDE 目录中。

所有的库在 \TURBOC\LIB 目录中。

Turbo C 命令行编译器(TCC.EXE)和 Turbo Linker (TLINK.EXE)在当前目录中或者在 PATH=环境变量所列的目录中。

要重建 TCWIN.EXE, 批文件 MAKTCWIN.BAT 必须在当前目录或者在 PATH=环境变量所列的目录中。另外, 适当的 NW.T2?OBJ 文件必须在 \TURBOC\LIB 目录中。

通过发出下面的命令激活 make 文件可以重建一个示例程序

```
make-fmakexpls mnexampl.exe
```

或者用下面的命令重建全部十八个程序:

```
make -fmakexpls
```

## C 和汇编模块及宏

下表列出了哪些函数是用 C 实现的, 哪些是用汇编语言写的, 哪些是定义在头文件中的宏。

表中还列出了某些特殊源文件。

下面的函数是可常规的 C 函数实现的, 每个函数在其对应的 .C 文件中:

edbase	isprep	mnhilite0	mospeed	scattywin	wncdgevn	wnremevn
edbuffer	iaremove	mnhilite	mostat	scattywrt	wncover	unremove
edchkey	isreserv	mnitem	prcancel	scwrap	wncreat0	wnresprv
edinitky	issense	mnitmkey	prerror	scwrite	wncurmov	wnretevn
edreduce	ivdetect	makey	prgeto	stpcvt	wncurpos	wnretinf
edremky	ivdisabl	mnlittem	pristld	stpexpan	wncurset	wnrevupd
edretinf	ivinstal	mnlitkey	prspool	stpjust	wncursor	wnscribk
edretkey	ivsense	mnmmchitm	scapage	stptabfy	wncurtrk	wnscribr
edsetcur	ivvecs	mnmmchkey	scattrit	stpxlate	wndstroy	wnscroll
edwrap	kbequip	mnmouse	scblink	stschind	wnerror	wnseldew
edwrect	kbextend	mnmmstyle	scborder	utansi	wnforget	wnselect
edzapkey	kbflush	mnread	scbox	utcrit	wngetimg	wnsetbuf
fdolock	kbgetkey	mnvalmn0	scchgdev	utctfbrk	wngetopt	wnsetcur
flflush	kbkcflsh	mnvdisp	scclrmgs	utgetclk	wnhide	wnsetopt
flgetdta	kbpc!!	moavoid	sccurset	utmoveime	wnhoriz	wnshoblk
fllock	kbquery	mobutton	scurst	utnulchk	wninitcv	wnunhide
flnorm	kbqueue	mocheck	scequip	utsafcpy	wnneedup	wnupdate
flprompt	kbset	mocurmov	scgetvid	utsleep	wnnupblk	wnvalev0
flputda	kbstuff	moequip	scmode	utspkr	wnorigin	wnvalno0
flremovl	kbwait	mogate	scmode4	utsqzscn	wnovrlap	wnvalwi0
flretvol	mmctri	mogetmov	scnewdev	uttim2tk	wnpgadd	wnvdisp
flsetvol	mmnfirst	mograph	scpages	utt2tim	unpgrem	wnwrap



hlclose	mmsize	monadlr	scpali	utunsqz	wnpimhbk	wnwrbuf
hldisp	mnatr	mohard	scpalett	viatrect	wnpoll	wnwrrect
hlfrindx	mncreat0	mohide	scpclr	vihoriz	wnprintf	wnwrstrn
hllookup	mndefkey	mojust	scpgcur	viptr	wnputhor	wnwrttyx
hlopen	mndisabl	mojump	scread	virdsect	wnputsen	wnzapenv
hlpas2c	mndlitms	politpen	screstpg	viscroll	wnquery	wnzapsen
hlread	mndlkeys	mopreclk	scrows	viwrsect	wnrdbuf	
iscurprc	mndstroy	morange	scsavepg	wnatrbk	wnread	
isinstal	mndfindal	mosoft	scsetvid	wnatrstr	wnredraw	

下面的函数是用汇编语言实现的，每个函数在其对应的.ASM 文件中：

iscall	ivctrl	kbreedy	utamove	utmodel
isdispat	kbplace	mocatch	utintflg	vidirec0

下面的例程是用宏实现的，它们被定义在下面所展示的头文件中：

宏	定义文件	宏	定义文件	宏	定义文件
---	------	---	------	---	------

edfield	bedit.h	utdosmajor	butil.h	utpokeb	butil.h
isgetvec	bintrupt.h	utdosminor	butil.h	utpoken	butil.h
isputvec	bintrupt.h	utdosrdy	butil.h	utpokew	butil.h
isresext	bintrupt.h	utdosver	butil.h	utpspseg	butil.h
kbscanof	bkeybrd.h	uthibyte	butil.h	utrange	butil.h
kbstatus	bkeybrd.h	uthinyb	butil.h	utseg	butil.h
max	butil.h	uthiword	butil.h	utsign	butil.h
min	butil.h	utinp	butil.h	utskip	butil.h
mcreate	bmenu.h	utintoff	butil.h	utsound	butil.h
mndsplay	bmenu.h	utinton	butil.h	utspkoff	butil.h
mnlread	bmenu.h	utlobyte	butil.h	utspkon	butil.h
moreset	bmouse.h	utlonyb	butil.h	uttofar	butil.h
prchar	bpriat.h	utlowlim	butil.h	uttofaru	butil.h
prinit	bprint.h	utloword	butil.h	utuplim	butil.h
prstatus	bprint.h	utmax	butil.h	utwdlong	butil.h
scclmsg	bscreens.h	utmin	butil.h	vidspmsg	bvideo.h
scpage	bscreens.h	utnorm	butil.h	virdirect	video.h
utabs	butil.h	utnybbyt	butil.h	viwrrect	bvideo.h
utalarm	butil.h	utoff	butil.h	wnattr	bwindow.h
utalloc	butil.h	utoutp	butil.h	wcreate	bwindow.h
utbound	butil.h	utpeekb	butil.h	wndsplay	bwindow.h
utbywor	butil.h	utpeekn	butil.h	wnfield	bwindow.h
utchknil	butil.h	utpeekw	butil.h	wnwrstr	bwindow.h
utcopy	butil.h	utplong	butil.h	wnwrtty	bwindow.h

下面的函数没有它们自己的源文件，但被定义在这里展示的源文件中：

函数	定义文件	函数	定义文件
----	------	----	------

b_vidcpy	scnewdev.c	ivdisk	ivctrl.asm
ivbiosky	ivctrl.asm	ivdos	ivctrl.asm
ivcbreak	ivctrl.asm	ividle	ivctrl.asm
ivcom1	ivctrl.asm	ivkeybd	ivctrl.asm
ivcom2	ivctrl.asm	ivtimer	ivctrl.asm

下面的源文件并非用于单独的某一个函数：

**BNOMOUSE.ASM** 定义 **MOCHCEC**、**MOHIDE** 和 **MOPRECLK** 的替换版本。

**NW.C** 定义 **WNCHKDM**、**WNCMOVE**、**WNGETATR**、**WNSETATR** 和 **WNSETWIN** 的版本，这些

函数与 Turbo C 字符窗口打交道。NW.C 是 NW-T2?.OBJ 文件的源模块。

WNNATVWN.C 定 WCHKDM、WNCMOVE、WNGETATR、WNSETATR 和 WNSETWIN 的版本, 这些函数不使用 Turbo C 字符窗口。WNNATVMN.C 是库中 WNNATVWN.OBJ 模块的源文件。

下面的源文件不定义函数:

KBSCANOF.C	定义 b_keycod[] 数组, 该数组被宏 KBSCANOF 所引用。
MOVARS.C	定义全局变量 b_mocatch、b_modispat、b_mohanmask 和 b_momask。
SCPAGE.C	定义被宏 SCPAGE 所引用的全局变量 b_curpage。

## 函数库建造说明

所有 C 模块和示例程序都是用 Turbo C 2.0 命令行编译器(TCC)编译的, 使用了下面的开关: -c(只编译)、-m(内存模式)、-O(对跳转优化)和 -w(允许所有警告)。所有汇编模块都是用 Turbo Assembler (TASM) 1.0 汇编的, 使用了 /mx 和 /w+ 开关。函数库由 Turbo Librarian 示例程序是用 TLINK 版本 2.0 链接, 带有开关。

## 利用数包的程序设计

Turbo C TOOLS 通过精心设计, 不需要特殊的编译和连接任选项。需要关心的只是:

- 包括适当的头文件;
- 通过适当的编码风格消除所有的编译警告;
- 与适当的 Turbo C TOOLS 库(tct\_t1?.lib)连接。在用 Turbo C TOOLS 创建第一个程序之前, 应该首先创建和执行一个或多个没有使用 Turbo C TOOLS 的简单程序, 以此来练习编译。试一下传统的测试程序:

```
#include <stdio.h>
int main()
{
    printf("Hello, World!\n");
    return 0; /* Return code 0 indicates success. */
}
```

用 Turbo C TOOLS 创建程序时, 唯一附加的步骤是指明适当的 Turbo C TOOLS 库(tct\_t1?.lib)。对 Turbo C 的命令行版本(tcc.exe)来说, 仅仅指明库名即可, 如下例:

```
tcc wnexampl tct_t1s.lib
```

在集成环境中, 你必须用 Project menu 指明一工程文件, 以保证 Turbo C TOOLS 库与你的程序连接。可以试试下面的简单工程文件:

```
wnexampl.c
tct_t1s.lib
```

作为开始的练习, 从 Turbo C TOOLS 盘中将 wnexampl.c 拷到工作目录或盘中。编译, 连接, 运行之。它能测试是否安装好了 Turbo C 和 Turbo C TOOLS。如果此过程进行得很顺利, 可以继续创建自己的程序了。

## 编程风格、编译警告和头文件

现代 C 编译如 Turbo C 提供强数据类型的任选项。它们能以象 UNIX 的 lint 机制那样的方式辨别不当的数据类型转换错误。另外, 还能接受函数参数类型的说明, 并用此信息鉴别函数的数量和类型的不匹配错误。

强数据类型为检查和避免程序错误提供有力的帮助。因此, Turbo C TOOLS 大量使用头文件来定义数据类型, 并声明函数的参数和返回值的数据类型。

建议你也这样做。置所有的编译警告消息为能使。包含所有有关的头文件, 这些头文件为你调用的 Turbo C 和 Turbo C TOOLS 库中函数声明参数。用自己的头文件声明自己定义的函数的参数。如果编译警告良性的类型冲突, 则仔细修改代码以避免或抑制警告。

如果你依靠头文件定义模块间共享的所有定义, 并且避免代码产生特殊编译错误, 那么你将从某些难以捉摸的错误的早期诊断中获益。(强类型甚至能诊断出还没有对你构成威胁但将来会对你有害的错

误)。

## 修改 Turbo C TOOLS

重新编译或重新汇编 Turbo C TOOLS 可以使它们适用于不同的编译版本和存贮模式。这在 readme.doc 文件中有详尽说明。(例如:在头文件中被当作宏处理的函数)。

为了修改一 Turbo C TOOLS 函数,只需重新编译或重新汇编相应的源代码。所有源文件的名字都是函数名加上.c 或.asm 的扩展部分。readme.doc 文件指出哪些函数是用 C 语言写的,哪些是用汇编语言写的,哪些被当作宏来处理。

重新编译和汇编后,再在编译和连接你的测试程序时显式指定目标文件。例如,如果你修改了 STPCVT 并且创建了 stpcvt.obj,用下面的命令将测试程序与 Turbo C 的 S 模式连接起来:

```
tcc test stpcvt.obj tct tla.lib
```

这样显式地命名 stpcvt.obj,使连接器使用新版本代替库(tct\_tla.lib)中原来的版本。为了在集成环境中达到同样的目的,在你的项目文件中加入下列行:

```
stpcvt.obj
```

若你对这种修改满意,可以用它取代库中的目标模块,这一工作需要一库管理程序完成,象 Microsoft 的 LIB 工具,它是由 Microsoft 宏汇编包提供的。

建议你不要改变任何 Turbo C TOOLS 函数的调用顺序。如果真的这样做,你会混淆可能调用这些函数的其它 Turbo C TOOLS 函数。相反,你可以自己命名定义新函数。

## 源代码约定

为了提高代码的易读性和可理解性,Turbo C TOOLS 中用到了一些约定。

每个函数名以一个两字母的前缀打头,它指明此函数的类属。每个函数的源文件的开头是一个文档头。调用序列、参数定义和说明信息都在源文件的开始用注释表达出来。

源代码遵从类似于 Kernighan 和 Ritchie 的著作里推荐的风格规范。所有的标识符都有声明,即使是整数也如此。指针变量以字母 P 打头;指针的指针以 PP 打头。缩进的风格也是标准的。编码风格虽是个人的事情,但是我们力图采用相对的标准,以提供一个可以让 Turbo C TOOLS 用户模仿的模式。

## 插入码函数程序设计示例

IVCLOCK.C 用来在屏幕上显示不断更新的 时间。

程序在屏幕的右上角显示时钟。运行该程序的所有支持皆由插入码子程序(IV)提供。

IVCLOCK 从每次计时中断中取得控制,但每十八次显示一次时间。(稍微比一秒少一些)

IVCLICK 从 BIOS 中以时钟为单位获得时间。

命令行格式:

```
ivclock [-r]
```

IVCLOCK 是自删除服务程序。如果调用 IVCLOCK 时没有参数则安装 IVCLOCK;如果调用时有“-r”参数,则从内存中删除。

IVCLOCK 用来展示插入码的功能,并提供一个插入代码的结构和使用的例子。

```
#include <dos.h>
#include <stdio.h>
#include <binterv.h>
#include <bintrupt.h>
#include <bscreens.h>
#include <bvideo.h>

/* The name of the function which will be called */
/* approximately once per second. */

void intime (IV_EVENT *);
#define STKSIZE 2000
#define OK      0
#define FALL_OUT 101
```

```

#define NOT_FOUND      1
#define ERROR_DISABLING2
#define ERROR_REMOVING3
#define ALREADY_INSTALLED 4
        /* Definitive signature for this version of IVCLOCK.*/
#define IVCLOCK_SIGN "IVCLOCK 03/31/89"
        /* Allocation of stack space for the interrupting. */
        /* scheduler and user function. */
char schedstk [STKSIZE];
        /* This is the data structure to pass to the */
        /* scheduler so that it will give control every 18 */
        /* clock ticks. */
IV_TIME timetab = {18, IV_TM_INTERVAL};
        /* Internal functions -- install & remove ISR */
int install_iv (void);
int remove_iv (void);
void main(int, char **);
void main (argc, argv)
int  argc;
char *argv [];
{
    if (argc == 1)
        exit (install_iv ());
    if ((argc == 2)
        &&
        (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
        ((argv [1][1] == 'r') || (argv [1][1] == 'R'))))
        exit (remove_iv ());
    printf ("usage: ivclock [-r]\n");
    exit (0);
}
/**
INSTALL_IV -- 安装IVCLOCK的中断矢量，终止并驻留。
        ret = install_iv ();
        int ret          当安装中断服务子程序时出现问题 }从IVINSTAL中返回错误码。
        如果没有安装IVCLOCK则安装之。
返回    ret    ALREADY_INSTALLED (4)
                已安装IVCLOCK。
                FALL_OUT (101)
                ISRESEXT()失败。
                IV_INSTALLED (5)
                IVCLOCK已安装。
**/
int install_iv ()
{
    int          ercode;
    IV_VECTORS vecs;
        /* Check to see if IVCLOCK already installed. */
    ivvecs (IV_RETVEC, &vecs);
    if (ivsense (&vecs, IVCLOCK_SIGN) != FARNIL)
    {

```

```

    puts ("IVCLOCK already installed.");
    return (ALREADY_INSTALLED);
}

    /* Install the interrupt service routine--i.e. tell */
    /* the scheduler about our clock routine.          */
if (0 !=
    (rcode =
        ivinstal (intime, IVCLOCK_SIGN, schedstk, STKSIZE,
            (IV_KEY *) NIL, 0, &timetab,
            sizeof(timetab) / sizeof(IV_TIME),
            /* IVCLOCK uses TURBO C floating point support. */
            IV_FLOAT_NEED))))
{
    /* Error!                                          */
    printf ("IVINSTAL error %d.\n", rcode);
    return (rcode);
}

    /* Terminate and stay resident.                  */
isresext (OK);
    /* Should never get here.                         */
return (FALL_OUT);
}

/**
    REMOVE_IV -- 删除先前安装的IVCLOCK。
        ret = remove_iv ();
        int ret      返回码
                        NO_ERROR(0)-
                        没有发现代码。
                        NOT_FOUND (1)-
                        IVCLOCK未找到。
                        ERROR_REMOVING (3)--
                        IVCLOCK不能被删除(由于MALLOC指针被重写)。
    函数删除内存中活动的IVCLOCK，清除中断矢量，释放内存。
    返回 ret (如果错误为非零)。
**/
int remove_iv ()
{
    IV_VECTORS vecs;
    IV_CTRL far *pivctrl;
    ivvecs (IV_RETVEC, &vecs);
    if ((pivctrl = ivsense (&vecs, IVCLOCK_SIGN)) == FARNIL)
    {
        puts ("IVCLOCK not found.");
        return (NOT_FOUND);
    }
    if (ivdisabl (pivctrl))
    {
        puts ("Error disabling IVCLOCK.");
        return (ERROR_DISABLING);
    }
}

```

```

    if (isremove (pivctrl->psp))
    {
        puts ("Error removing IVCLOCK.");
        return (ERROR REMOVING);
    }
    return (0);
}

/**
    INTIME -- 在屏幕上显示当前时间的数字。
    (只由中断服务调用程序调用)。
    IV_EVENT *pevent    指向插入事件结构。结构含有当前时间(从午夜开始以时
                        钟走时为单位)以及其它数据。
    函数每次18.2次的调度程序中接收控制, 显示当前时间的数字(每调用18次显示一次), 并退出。
    返回          无
**/
void intime (pevent)
IV_EVENT *pevent;
{
    char time_str[10];
    int  mode, act_page, columns;
    int  hours, mins, secs;
    long elpsec;

        /* Get the current mode and active display page */
        /* from BIOS. */
    scmode (&mode,&columns,&act_page);
        /* Direct our display output to the current display */
        /* page. */
    scpage (act_page);
        /* Figure out number of seconds past midnight from */
        /* (given) number of timer ticks since midnight. */
    elpsec = (long) (0.0549254 * (double) pevent->time);
        /* Figure out hours, minutes, seconds from seconds */
        /* past midnight. */
    hours = (int) (elpsec / 3600L);
    mins  = (int) ((elpsec / 60L) % 60L);
    secs  = (int) (elpsec % 60L);
        /* Display the time. */
    sprintf (time_str, "%02d:%02d:%02d", hours, mins, secs);
    vidspmsg (0, columns - strlen (time_str),
        NORMAL | INTENSITY, SC_BLACK, time_str);
}

```

## 第二章 域编辑编程扩展

这种类型的函数提供了一个小型的编辑器，可用于屏幕或窗口上的一个矩形区域。这个编辑器足以进行一般的数据输入工作，也可加入数据检验或其它功能。它的可选项及特性包括：

输入区域中有多个字符行。

输入时有整字换行。

允许对齐、居中、用零填充。

可以控制插入和替换方式，这些方式可以由不同的光标形状来指示。

通过WNFIELD能够完全支持Turbo C TOOLS窗口。利用虚拟窗口可以编辑一个比显示数据的视口更大的缓冲区。

可以用初始值预选装载缓冲区。

支持键控制函数，这就可以完全控制每一个按键，也使您能够以任何方式、在任何时候修改输入的数据。

完全控制每一个按键的定义和按键执行的动作。

有两个示例程序展示了使用域编辑的技术。DEMOEDT介绍了一个具有多行域的简单编辑器，ENTRYEDT的编辑域则具有几个数据项并使用键控制函数进行数据验证。

### 域编辑函数的种类

#### 数据项和编辑

EDFIELD 允许用户编辑屏幕上的一个域。

WNFIELD 允许用户编辑窗口中的一个域。

EDBUFFER 对内存中的一个缓冲区执行一次或多次编辑操作，不与屏幕或用户打交道。

#### 编辑键定义

EDINITKY 建立一个由EDFIELD和WNFIELD接受的缺省按键表。(用户不必使用EDINITKY，因为若有必要EDFIELD和WNFIELD在第一次被调用时使自动激活它。)

EDRETKEY 读取由EDFIELD和WNFIELD接受的按键表中的一项。

EDCHGKEY 修改或向表中加入一项。

EDREMKEY 从表中删去一个按键。

EDZAPKEY 删除整个表。

#### 编辑操作和缺省的按键分配

ED\_ACTION枚举数据类型列出了可以分配给按键的每一个编辑操作，该数据类型在头文件bedit.h中定义。

```
typedef enum          /* ED ACTION: 编辑操作. */
{
    ED_NULL,          /* 无动作. */
    ED_ABORT,         /* 不编辑, 退出. */
    ED_BEEP,          /* 使系统扬声器发声. */
    ED_ATTR,          /* 改变属性. */
    ED_ASCII,         /* 显示字符. */
    ED_LEFT,          /* 左移. */
    ED_RIGHT,         /* 右移. */
    ED_UP,            /* 上移一行. */
}
```

```

ED_DOWN,          /* 下移一行 */
ED_FRONT,         /* 移到缓冲区的开头 */
ED_END,           /* 移到缓冲区的末尾。 */
ED_TEXT_END,      /* 移到最后一个非空白位置。 */
ED_WRAP,          /* 使缓冲区整字换行。 */
ED_REDUCE,        /* 压缩空白。 */
ED_NEXT_WORD,     /* 移到下一个字。 */
ED_PREV_WORD,     /* 移到前一个字。 */
ED_BEGIN_WORD,    /* 移到字的开始位置。 */
ED_END_WORD,      /* 移到字的末尾。 */
ED_NEXT_WHITE,    /* 移到下一个空白位置。 */
ED_PREV_WHITE,    /* 移到前一个空白位置。 */
ED_NEXT_NONWHITE, /* 移到下一个非空白字符。 */
ED_PREV_NONWHITE, /* 移到前一个非空白字符。 */
ED_DELETE,        /* 删除当前字符。 */
ED RUBOUT         /* 左移，删去字符。 */
ED DELLEFT,       /* 删去左边的字符。 */
ED DEL_WORD,      /* 删去一个字。 */
ED_CLEAR,         /* 删除整个缓冲区。 */
ED_CLEAREOL,      /* 清空至行尾。 */
ED_CLEAREOB,      /* 清空至缓冲区末尾 */
ED_INSERT,        /* 插入方式。 */
ED_REPLACE,       /* 替换(非插入)方式。 */
ED_INSTOGGLE,     /* 转置插入/替换方式。 */
ED_UNDO,          /* 回收前一个缓冲区。 */
ED_TRANSMIT       /* 完成编辑并返回。 */

```

} ED\_ACTION;

一个按键可以执行一个或一系列上表所列的操作。下面是EDINITKY设置的缺省键分配。按照缺省情况，每个键仅执行一个动作。

#### 缺省的编辑键分配

键	ASCII值	键码	动作
Esc		0x1b 0x01	ED_ABORT
Up	(数字小键盘)	0x00 0x48	ED_UP
Up	(增强键)	0xe0 0x48	ED_UP
Down	(数字小键盘)	0x00 0x50	ED_DOWN
Down	(增强键)	0xe0 0x50	ED_DOWN
Right	(数字小键盘)	0x00 0x4d	ED_RIGHT
Right	(增强键)	0xe0 0x4d	ED_RIGHT
Left	(数字小键盘)	0x00 0x4b	ED_LEFT
Left	(增强键)	0xe0 0x4b	ED_LEFT
Home	(数字小键盘)	0x00 0x47	ED_FRONT
Home	(增强键)	0xe0 0x47	ED_FRONT
End	(数字小键盘)	0x00 0x4f	ED_END
End	(增强键)	0xe0 0x4f	ED_END
Ctrl-End		0x00 0x75	ED_TEXT_END
Ctrl-F		0x06 0x21	ED_NEXT_WORD
Ctrl-A		0x01 0x1e	ED_PREV_WORD
Ctrl-T		0x14 0x14	ED_DEL_WORD
-	(数字小键盘)	0x2d 0x4a	ED_BEGIN_WORD
+	(数字小键盘)	0x2b 0x4c	ED_END_WORD



F2		0x00	0x3c	ED_NEXT_WHITE
F1		0x00	0x3B	ED_PREV_WORD
F4		0x00	0x3E	ED_NEXT_NONWHITE
F3		0x00	0x3D	ED_NEXT_NONWHITE
Del	(数字小键盘)	0x00	0x53	ED_DELETE
Del	(增强键)	0xe0	0x53	ED_DELETE
Backspace		0x08	0x0e	ED RUBOUT
Ctrl-H		0x08	0x23	ED_DELETE
Ctrl-Home	(数字小键盘)	0x00	0x77	ED_CLEAR
Ctrl-Home	(增强键)	0xe0	0x77	ED_CLEAR
Ctrl-Right	(数字小键盘)	0x00	0x74	ED_CLEAREOL
Ctrl-Right	(增强键)	0xe0	0x74	ED_CLEAREOL
Alt-E		0x00	0x12	ED_INSERT
Alt-R		0x00	0x15	ED_REPLACE
Ins	(数字小键盘)	0x00	0x52	ED_INSTOGGLE
Ins	(增强键)	0xe0	0x52	ED_INSTOGGLE
Alt-X		0x00	0x2d	ED_UNDO
Enter		0x0d	0x1c	ED_TRANSMIT
Enter	(数字小键盘)	0x0d	0xe0	ED_TRANSMIT

请注意，这些分配使得增强键盘(101或102键)上的新键与PC或AT键盘(83和84键上对应的旧键表现相同，这样，通过KBEXTEND选择扩展BIOS键盘功能不影响缺省的键动作。

未定义的按键具有下列动作：

字符值非零的按键执行ED\_ASCII，即插入或替换缓冲区中的字符。

字符值为零的按键执行ED\_NULL，也就是说，它们被忽略。

## 域编辑函数源程序

下面罗列了所有的域编辑扩展函数的源程序：

### EDBASE 编辑和返回一个用户响应。

```
#include <butil.h>
#include <bedit.h>
#include <bstrings.h>
#include <bvideo.h>
#include <bwindow.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

int edbase(pdata, pinitstr, pretstr, target_size, pfield_control,
           pfinal_key, return_info, set_cursor, write_rect)
void      *pdata;
const char *pinitstr;
char      *pretstr;
int       target_size;
const ED_CONTROL *pfield_control;
KEY_SEQUENCE *pfinal_key;
PED_RET_FUNC return_info;
PED_SET_FUNC set_cursor;
PED_WRITE_FUNC write_rect;
*pdato: 指向将传给return_info、set_cursor和write_rect数据的指针。
```

**pinitstr** 缓冲区的初始值。  
**pretstr** 返回的用户响应。  
**target size** 包括结尾的'\0'在内的缓冲区最大尺寸。  
**ED CONTROL** 被显示和编辑的域。  
**pfield control** 控制结构。  
**pfinal\_key** 指向用户最后击键的指针。  
**return\_info** 指向将后回光标信息的函数。  
**set cursor** 指向设置光标状态的函数。  
**write\_rect** 指向输出一矩形屏幕数据的函数。

EDBASE为从屏幕域获取信息和以格式化的文本字符串返回数据提供一个灵活的方法。

**field control**参数用来控制域在屏幕上的位置, 显示缓冲区的长度, 光标的尺寸以及一般的编辑控制。

**pinitstr**中如果包函值, 该值用作域的初始值, 初始值允许用户编辑。包括结尾'\0'文本长度不能比**target size**大。

EDBASE用来定位光标, 显示缓冲器的内容, 以及格式化最后转换的缓冲区内容。

由于EDBASE接受进行物理屏幕I/O的函数指针, 意在提供给宏EDFIELD和WNFIELD调用。

宏EDFIELD提供进行屏幕I/O的函数指针, 而WNFIELD提供进行窗口I/O的函数。

**error(返回)**      **EDFIELD**返回:  
                   **ED\_NO\_ERROR** 成功返回数据。  
                   **ED\_ILL\_DIM**    域尺寸不合法。  
                   **ED\_NO\_MEMORY** 没有临时存贮的内存。  
                   **ED\_USER\_ABORT** 用户放弃编辑, 无数据返回。  
                   **WNFIELD**返回:  
                   **WN\_NO\_ERROR** 数据成功返回。  
                   **WN\_ILL\_DIM**    域尺寸不合法。  
                   **WN\_NO\_MEMORY** 没有临时存贮内存。  
                   **WN\_BAD\_WIN**    \*pdata 指向无效的窗口结构。  
                   **WN\_ILL\_VALUE** 光标尺寸非法。  
                   **ED\_USER\_ABORT** 用户放弃编辑, 没有数据返回。  
                   **h\_werr**        如果没有错误, 值不改变; 否则与**jerror**相等。

源程序(EDBASE.C):

```

#include <butil.h>
#include <bedit.h>
#include <bstrings.h>
#include <bvideo.h>
#include <bwindow.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

int edbase(pdata, pinitstr, pretstr, target size, pfield_control,
           pfinal_key, return_info, set_cursor, write_rect)
void      *pdata;
const char *pinitstr;
char      *pretstr;
int       target_size;
const ED CONTROL *pfield_control;
KEY SEQUENCE *pfinal_key;
PED RET FUNC  return_info;
PED SET FUNC  set_cursor;
  
```

```
PED_WRITE_FUNC write_rect;
```

```
{
    int            row, col;
    ED_BUFFER      edit_buffer;
    int            buffer_length;
    int            field_width, field_height;
    int            cur_edit_row, cur_edit_col;
    int            fore_attr, back_attr;
    CUR_TYPE       replace_curs, insert_curs;
    KEY_SEQUENCE   Round_keyseq;
    ED_KEY          edit_key;
    ED_KEY          *pedit_key;
    char           fill_char;
    ED_ACTION       edit_action;
    unsigned char   beep_on, done;
    int            final_action;
    int            error_code;
    char           *ptemp_buffer;
    int            just_code;
    CUR_INFO        info;
    int            error;
    char           *ptemp_string = NIL;
    int            nul_index;
    field_height = pfield_control->dimensions.h;
    field_width  = pfield_control->dimensions.w;
    error = return_info(pdata, &info, &(pfield_control->ul_corner),
                      field_height, field_width);
    if (error)
        return(error);
        /* Set the initial field coordinates. */
    row = pfield_control->ul_corner.row;
    col = pfield_control->ul_corner.col;
        /* Default attributes are the existing attributes */
        /* if foreground and background are both zero. */
    if (pfield_control->attribute == 0)
    {
        fore_attr = -1;
        back_attr = -1;
    }
    else
    {
        fore_attr = uthinyb(pfield_control->attribute);
        back_attr = utlonyb(pfield_control->attribute);
    }
    if ((pfield_control->replace_cursor.high == -1) &&
        (pfield_control->replace_cursor.low == -1))
    {
        replace_curs = info.size;
    }
    else
```

```

    replace_curs = pfield_control->replace_cursor;
    if ((pfield_control->insert_cursor.high == -1) &&
        (pfield_control->insert_cursor.low == -1))
    {
        insert_curs = info.size;
    }
    else
        insert_curs = pfield_control->insert_cursor;

    /* Now initialize the edit buffer, edit_buffer. The */
    /* previous values are set to the same values as      */
    /* the current values. The buffers are set to the    */
    /* initial string, and length is the length of the   */
    /* edit field.                                         */
    buffer_length = field_width * field_height;
    edit_buffer.pbuffer = malloc(buffer_length);
    if (edit_buffer.pbuffer == NIL)
        return(ED_NO_MEMORY);
    memset(edit_buffer.pbuffer, ' ', buffer_length);
    edit_buffer.pbuffer[buffer_length - 1] = '\0';
    utuplim(buffer_length, target_size - 1);
    /* Set the edit buffer control flags so that          */
    /* EDBUFFER will know what to do.                     */
    edit_buffer.control_flags = pfield_control->control_flags &
                               (ED_INSERT_MODE | ED_WORD_WRAP);
    edit_buffer.buffer_size = buffer_length;
    edit_buffer.cursor_pos = 0;
    edit_buffer.attribute = utnybbyt(back_attr, fore_attr);
    edit_buffer.dimensions = pfield_control->dimensions;
    if (pinitstr != NIL)
    {
        /* If word wrap is specified, remove all leading */
        /* blanks from the buffer.                          */
        if ((pfield_control->control_flags & ED_WORD_WRAP) &&
            isspace(pinitstr[0]))
        {
            ptemp_string = malloc(strlen(pinitstr));
            if (ptemp_string == NIL)
            {
                free(edit_buffer.pbuffer);
                return(ED_NO_MEMORY);
            }
            strcpy(ptemp_string, pinitstr);
            stpcvt(ptemp_string, RLWHITE);
            pinitstr = ptemp_string;
        }
        if (strlen(pinitstr) <= buffer_length)
            edit_buffer.data_end = strlen(pinitstr);
        else
            edit_buffer.data_end = buffer_length;
        memmove(edit_buffer.pbuffer, pinitstr, edit_buffer.data_end);
    }

```

```

}
else
    edit_buffer.data_end = 0;
    /* Begin by applying word wrapping to the buffer, */
    /* if requested. */
if (pfield_control->control_flags & ED_WORD_WRAP)
{
    edreduce(&edit_buffer);
    edwrap(&edit_buffer);
}

/* The main loop of EDBASE displays the current */
/* buffer, and positions the cursor to the */
/* appropriate position on the screen. The current */
/* edit mode determines the size of the cursor. The */
/* function KBPOLL returns the keystroke by polling */
/* the keyboard. The address of the key control */
/* function is passed to KBPOLL, and the address of */
/* the edit buffer is passed as the function data. */
/* The application key control function can */
/* therefore inspect and alter the edit buffer if */
/* it wishes. */
beep_on = 0;
done = 0;
do
{
    cur_edit_row = edit_buffer.cursor_pos / field_width;
    cur_edit_col = edit_buffer.cursor_pos -
        (field_width * cur_edit_row);
    /* Set the size & position of the cursor. */
if (edit_buffer.control_flags & ED_INSERT_MODE)
    error = set_cursor(pdata, 0, &insert_curs,
        row + cur_edit_row, col + cur_edit_col);
else
    error = set_cursor(pdata, 0, &replace_curs,
        row + cur_edit_row, col + cur_edit_col);
if (error)
{
    free(edit_buffer.pbuffer);
    if (ptemp_string != NIL)
        free(ptemp_string);
    return(error);
}

/* Display the data contained in the buffer. */
write_rect(pdata, row, col,
    row + field_height - 1,
    col + field_width - 1,
    edit_buffer.pbuffer, fore_attr,
    back_attr, CHARS_ONLY);
/* Now return the next keystroke. If it is not a */
/* recognized edit key (that is, it does not appear */

```

```

        /* in the edit key list), the action code taken is */
        /* ED_ASCII (display as standard character), if the */
        /* character code is nonzero. All other keys are */
        /* ignored (ED_NULL). */
while (!kbpoll(pfield_control->control_function, &edit_buffer,
              &found_keyseq, KB_REMOVE_KEY))
;
pedit_key = edretkey(&found_keyseq);
if (pedit_key == NIL)
{
    edit_action = (found_keyseq.character_code ? ED_ASCII
                  : ED_NULL);

    edit_key.key_sequence = found_keyseq;
    edit_key.edit_actions.num_actions = 1;
    edit_key.edit_actions.pactions = &edit_action;
    pedit_key = &edit_key;
}
final_action = edbuffer(&edit_buffer, pedit_key);
/* If at the end of the buffer, check to see if */
/* transmission should occur (ED_AUTOSKIP) or if */
/* the system speaker should sound (ED_BEEP_END). */
if (edit_buffer.cursor_pos == (buffer_length - 1))
{
    if ((pfield_control->control_flags & ED_BEEP_END) != 0)
        beep_on = 1;
    if ((pfield_control->control_flags & ED_AUTOSKIP) != 0)
    {
        done = 1;
        error_code = ED_NO_ERROR;
    }
}
if (beep_on)
{
    beep_on = 0;
    scattywrt("\a", 0);
}
switch (final_action)
{
case ED_ABORT:
    done = 1;
    error_code = ED_USER_ABORT;
    memset(edit_buffer.pbuffer, ' ', buffer_length);
    edit_buffer.data_end = strlen(pinitstr);
    utuplim(edit_buffer.data_end, buffer_length);
    memmove(edit_buffer.pbuffer, pinitstr,
            edit_buffer.data_end);
    break;
case ED_ATTR:
    if (edit_buffer.attribute == 0)
    {

```

```

        fore_attr = -1;
        back_attr = -1;
    }
    else
    {
        fore_attr = uthinyb(edit_buffer.attribute);
        back_attr = utlonyb(edit_buffer.attribute);
    }
    break;
case ED_UNDO:
    metaset(edit_buffer.pbuffer, ' ', buffer_length);
    edit_buffer.cursor_pos = 0;
    edit_buffer.data_end = strlen(pinitstr);
    utuplim(edit_buffer.data_end, buffer_length - 1);
    memmove(edit_buffer.pbuffer, pinitstr,
            edit_buffer.data_end);
    edit_buffer.attribute = utnybbyt(back_attr, fore_attr);
    edit_buffer.control_flags = pfield_control->control_flags &
        (ED_INSERT_MODE | ED_WORD_WRAP);
    break;
case ED_TRANSMIT:
    done = 1;
    error_code = ED_NO_ERROR;
    break;
}
while (!done);
if (pfield_control->control_flags & ED_WORD_WRAP)
    edreduce(&edit_buffer);
if (pfinal_key != NIL)
    *pfinal_key = found_keyseq;
utuplim(edit_buffer.data_end, target_size - 1);
edit_buffer.pbuffer[edit_buffer.data_end] = '\0';
/* Alter the buffer based on the control_flags */
/* member of the pfield_control structure. The low */
/* order byte is the conversion code to pass to */
/* stpcvt(). The low order nybble of the high */
/* order byte determines justification and fill */
/* character. Note that if justification is */
/* specified, blank fill is the default. Moreover, */
/* left justification takes precedence over right */
/* justification.
if (error_code == ED_NO_ERROR) /* Only alter the buffer if */
/* normal transmission. */
    stpcvt(edit_buffer.pbuffer,
        pfield_control->control_flags & 0xff);
/* Now justify the string if requested. Right */
/* justification takes place within the entire edit */
/* buffer field, but left justification just within */
/* the string length.

```

```

3A
if ((pfield_control->control_flags &
    (ED_LEFTJUST | ED_RIGHTJUST | ED_CENTERJUST)) != 0)
{
    if ((pfield_control->control_flags & ED_ZERO_FILL) != 0)
        fill_char = '0';
    else
        fill_char = ' ';
    if ((pfield_control->control_flags & ED_LEFTJUST) != 0)
        just_code = -1;
    else
        if ((pfield_control->control_flags & ED_RIGHTJUST) != 0)
            just_code = 1;
        else
            just_code = 0;
    ptemp_buffer = malloc(buffer_length + 1);
    if (ptemp_buffer == NIL)
    {
        /* Error allocating temporary storage. */
        free(edit_buffer.pbuffer);
        if (ptemp_string != NIL)
            free(ptemp_string);
        return(ED_NO_MEMORY);
    }
    stpjjust(ptemp_buffer, edit_buffer.pbuffer, fill_char,
        buffer_length, just_code);
    memmove(edit_buffer.pbuffer, ptemp_buffer, buffer_length + 1);
    free(ptemp_buffer);
}

/* Now display any changes that were made to the */
/* buffer. */
nul_index = strlen(edit_buffer.pbuffer);
memset(edit_buffer.pbuffer + nul_index, ' ',
    buffer_length - nul_index);
write_rect(pdata, row, col,
    row + field_height - 1,
    col + field_width - 1,
    edit_buffer.pbuffer, fore_attr,
    back_attr, CHARS_ONLY);
edit_buffer.pbuffer[nul_index] = '\0';

/* Restore the original state of the cursor. */
error = set_cursor(pdata, info.off, &info.size,
    info.row, info.col);

if (error)
{
    free(edit_buffer.pbuffer);
    if (ptemp_string != NIL)
        free(ptemp_string);
    return(error);
}

```



```

}
strcpy(pretstr, edit_buffer.pbuffer);
free(edit_buffer.pbuffer);
if (ptemp_string != NIL)
    free(ptemp_string);
return(error code);
}

```

## EDBUFFER 对编辑缓冲区执行编辑动作

```
#include <bedit.h>
```

```
int edbuffer(    ED_BUFFER *pbuffer,
                const ED_KEY *pcommand);
```

**pbuffer** 待修改的编辑缓冲区的地址

**pcommand** 结构的地址，该结构列出了待执行的操作

(返回) 来自最后处理的一个键的动作码。下列动作引起立即返回，其后的键不作处理。

ED_ABORT	(1)	不做修改即结束。
ED_ATTR	(3)	属性(颜色)被改变
ED_UNDO	(22)	恢复原缓冲区内容
ED_TRANSMIT	(23)	编辑过程正常结束

EDBUFFER对缓冲区进行一个或多个编辑操作。

**pbuffer**所指向的结构用数据、高度、宽度和其它信息描述缓冲区的当前状态。ED\_BUFFER在bedit.h中定义如下：

```

typedef struct          /* DIM结构:  */
{
    int h,w;            /* 矩形区域的高度和宽度  */
} DIM;
typedef struct          /* ED_BUFFER结构:  */
{
    /* 传递给edbuffer()的编辑  */
    /* 缓冲区  */
    char *pbuffer;      /* 编辑缓冲区  */
    DIM dimensions;     /* 域尺寸  */
    int attribute;      /* 显示属性  */
    int buffer_size;    /* 缓冲区最大尺寸  */
    int edit_pos;       /* 缓冲区中的编辑位置  */
    int data_end;       /* 缓冲区数据的末尾  */
    int control_flags;  /* 插入方式?  */
} ED_BUFFER;

```

下面是ED\_BUFFER结构一些成员的说明：

**pbuffer**包含一个字符数组的地址，该数组正是屏幕上编辑域中字符的内容。数组必须至少是(dimensions.h\*dimensions.w+1)字节长，这样才有地方放置尾随的NUL字节("\0")。

**buffer\_size**是EDBUFFER可返回的最长字符串的长度。

**edit\_pos**表示相对于缓冲区开始处0的光标位置。

**data\_end**是当前缓冲区中数据的字节数(如果是空字符串，则为0)。

**control\_flags**与在EDFIELD中描述的ED\_CONTROL结构的control\_flags成员相同，但有效的两位是ED\_INSERT\_MODE和ED\_WORD\_WRAP位。

**pcommand**所指向的ED\_KEY结构指明编辑操作，该结构在bedit.h中定义如下：

```

typedef struct
{
    /* ED ACTION_LIST结构:  */
    int num_actions;        /* 操作码的数量。  */
}

```

```

    ED_ACTION *pactions;    /* 操作码数组。 */
} ED_ACTION_LIST;
typedef struct              /* KEY_SEQUENCE结构: */
{                          /* 一个键的字符和键码。 */
    unsigned char    character_code;
    unsigned char    key_code;
} KEY_SEQUENCE;
typedef struct
{                          /* ED_KEY结构: */
    ED_ACTION_LIST edit_actions; /* 编辑动作、编辑键的键序列 */
    KEY_SEQUENCE key_sequence; /* 和属性。 */
    int attribute; /* */
} ED_KEY;

```

用户以下列方式装载ED\_KEY结构的成员:

edit\_actions包含待执行操作的操作表, 见下面的示例。每个操作是枚举数据类型ED\_ACTION的成员。

当动作是ED\_ASCII时, key\_sequence.character\_code是加到缓冲区的字符; 当动作是ED\_ATTR时, attribute是缓冲区的新属性。

源程序(edbuffer.c):

```

#include <bedit.h>
typedef struct
{
    ED_ACTION action;
    unsigned reduce : 1;
    unsigned wrap : 1;
    unsigned reserved : 14;
} WRAP_ACTION;

/* The array wrap_actions specifies which edit
/* actions, when word wrap is specified, require
/* the buffer to have whitespace reduced prior to
/* execution, and/or that the buffer be word
/* wrapped after execution. */

static WRAP_ACTION wrap_actions[] =
{
    {ED_ASCII, 1, 1},
    {ED_LEFT, 1, 1},
    {ED_RIGHT, 1, 1},
    {ED_NEXT_WHITE, 1, 1},
    {ED_PREV_WHITE, 1, 1},
    {ED_DELETE, 1, 1},
    {ED_RUBOUT, 1, 1},
    {ED_DELEFT, 1, 1},
    {ED_DEL_WORD, 1, 1},
    {ED_CLEAREOL, 1, 1},
};

ED_ACTION edbuffer(pedit_buffer, pedit_key)
ED_BUFFER *pedit_buffer;
const ED_KEY *pedit_key;
{

```

```

int buffer_length;
int current_action;
int new_cursor_pos;
int num_chars_deleted;
int found, index;
int line_end;
if (pedit_key->edit_actions.num_actions <= 0)
    return(ED_NULL);
buffer_length = pedit_buffer->buffer_size;
for (current_action = 0;
     current_action < pedit_key->edit_actions.num_actions;
     current_action++)
{
    if (pedit_buffer->control_flags & ED_WORD_WRAP)
    {
        found = 0;
        for (index = 0;
             index < sizeof(wrap_actions) / sizeof(wrap_actions[0]);
             index++)
        {
            if (pedit_key->edit_actions.pactions[current_action] ==
                wrap_actions[index].action)
            {
                found = 1;
                break;
            }
        }
        if (found && wrap_actions[index].reduce)
            edreduce(pedit_buffer);
    }
    switch(pedit_key->edit_actions.pactions[current_action])
    {
        case ED_NULL:
            break;
        case ED_ABORT:
            return(ED_ABORT);
        case ED_BEEP:
            scattywrt("\a", 0);
            break;
        case ED_ATTR:
            pedit_buffer->attribute = pedit_key->attribute;
            return(ED_ATTR);
        case ED_ASCII:
            if (pedit_buffer->control_flags & ED_INSERT_MODE)
                memmove(&pedit_buffer->pbuffer[pedit_buffer->cursor_pos + 1],
                        &pedit_buffer->pbuffer[pedit_buffer->cursor_pos],
                        buffer_length - pedit_buffer->cursor_pos - 1);
            pedit_buffer->pbuffer[pedit_buffer->cursor_pos] =
                pedit_key->key_sequence.character_code;
            if (pedit_buffer->cursor_pos == (buffer_length - 1))

```

```

        pedit_buffer->data_end = buffer_length;
    if (pedit_buffer->cursor_pos < (buffer_length - 1))
        pedit_buffer->cursor_pos++;
    if (pedit_buffer->cursor_pos > pedit_buffer->data_end)
        pedit_buffer->data_end = pedit_buffer->cursor_pos;
    else
        if ((pedit_buffer->control_flags & ED_INSERT_MODE) &&
            (pedit_buffer->data_end < buffer_length))
        {
            pedit_buffer->data_end++;
        }
        /* Since the character could have been past the end */
        /* of the existing data, we may need to reduce the */
        /* buffer again. */
    if (pedit_buffer->control_flags & ED_WORD_WRAP)
        edreduce(pedit_buffer);
    break;
case ED_LEFT:
    if (pedit_buffer->cursor_pos > 0)
        pedit_buffer->cursor_pos--;
    break;
case ED_RIGHT:
    if (pedit_buffer->cursor_pos < (buffer_length - 1))
        pedit_buffer->cursor_pos++;
    break;
case ED_UP:
    if (pedit_buffer->cursor_pos -
        (pedit_buffer->dimensions.w - 1) > 0)
    {
        pedit_buffer->cursor_pos -=
            pedit_buffer->dimensions.w;
    }
    if (pedit_buffer->control_flags & ED_WORD_WRAP)
    {
        edreduce(pedit_buffer);
        edwrap(pedit_buffer);
    }
    break;
case ED_DOWN:
    if ((pedit_buffer->cursor_pos +
        (pedit_buffer->dimensions.w - 1)) < (buffer_length - 1))
    {
        pedit_buffer->cursor_pos +=
            pedit_buffer->dimensions.w;
    }
    if (pedit_buffer->control_flags & ED_WORD_WRAP)
    {
        edreduce(pedit_buffer);
        edwrap(pedit_buffer);
    }
}

```

```

        break;
case ED_FRONT:
    pedit_buffer->cursor_pos = 0;
    break;
case ED_END:
    if (pedit_buffer->data_end < buffer_length)
        pedit_buffer->cursor_pos = pedit_buffer->data_end;
    else
        pedit_buffer->cursor_pos = pedit_buffer->data_end - 1;
    break;
case ED_TEXT_END:
    for (new_cursor_pos = buffer_length - 1;
        (new_cursor_pos > 0) &&
        (isspace(pedit_buffer->pbuffer[new_cursor_pos]));
        new_cursor_pos--)
        ;
    if (!isspace(pedit_buffer->pbuffer[new_cursor_pos]))
        pedit_buffer->cursor_pos = new_cursor_pos;
    break;
case ED_WRAP:
    edwrap(pedit_buffer);
    break;
case ED_REDUCE:
    edreduce(pedit_buffer);
    break;
case ED_NEXT_WORD:
    if (isspace(pedit_buffer->pbuffer[pedit_buffer->cursor_pos]))
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos;
            (new_cursor_pos < pedit_buffer->data_end) &&
            (new_cursor_pos < (buffer_length - 1)) &&
            (isspace(pedit_buffer->pbuffer[new_cursor_pos]));
            new_cursor_pos++)
            ;
        pedit_buffer->cursor_pos = new_cursor_pos;
    }
    for (new_cursor_pos = pedit_buffer->cursor_pos;
        (new_cursor_pos < pedit_buffer->data_end) &&
        (new_cursor_pos < (buffer_length - 1)) &&
        (isspace(pedit_buffer->pbuffer[new_cursor_pos]));
        new_cursor_pos++)
        ;
    pedit_buffer->cursor_pos = new_cursor_pos;
    break;
case ED_PREV_WORD:
    if (isspace(pedit_buffer->pbuffer[pedit_buffer->cursor_pos]))
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos;
            (new_cursor_pos > 0) &&
            (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
            new_cursor_pos--)
            ;
        pedit_buffer->cursor_pos = new_cursor_pos;
    }
    for (new_cursor_pos = pedit_buffer->cursor_pos;
        (new_cursor_pos > 0) &&
        (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
        new_cursor_pos--)
        ;
    pedit_buffer->cursor_pos = new_cursor_pos;
    break;

```

```

        new_cursor_pos--)
    ;
    pedit_buffer->cursor_pos = new_cursor_pos;
}
for (new_cursor_pos = pedit_buffer->cursor_pos;
     (new_cursor_pos > 0) &&
     (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
     new_cursor_pos--)
    ;
pedit_buffer->cursor_pos = new_cursor_pos;
for (new_cursor_pos = pedit_buffer->cursor_pos;
     (new_cursor_pos > 0) &&
     (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
     new_cursor_pos--)
    ;
if (new_cursor_pos != 0)
    new_cursor_pos++;
pedit_buffer->cursor_pos = new_cursor_pos;
break;
case ED_BEGIN_WORD:
    if (!isspace(pedit_buffer->pbuffer[pedit_buffer->cursor_pos]))
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos;
             (new_cursor_pos > 0) &&
             (!isspace(pedit_buffer->pbuffer[new_cursor_pos - 1]));
             new_cursor_pos--)
            ;
        pedit_buffer->cursor_pos = new_cursor_pos;
    }
    break;
case ED_END_WORD:
    if (!isspace(pedit_buffer->pbuffer[pedit_buffer->cursor_pos]))
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos;
             (new_cursor_pos < pedit_buffer->data_end) &&
             (new_cursor_pos < (buffer_length - 1)) &&
             (!isspace(pedit_buffer->pbuffer[new_cursor_pos + 1]));
             new_cursor_pos++)
            ;
        pedit_buffer->cursor_pos = new_cursor_pos;
    }
    break;
case ED_NEXT_WHITE:
    new_cursor_pos = pedit_buffer->cursor_pos + 1;
    if ((new_cursor_pos < pedit_buffer->data_end) &&
        (new_cursor_pos < buffer_length))
    {
        for (;
             (new_cursor_pos < pedit_buffer->data_end) &&
             (new_cursor_pos < buffer_length) &&

```

```

        (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
        new_cursor_pos++)
    ;
    if ((new_cursor_pos < pedit_buffer->data_end) &&
        (new_cursor_pos < buffer_length) &&
        (!isspace(pedit_buffer->pbuffer[new_cursor_pos])))
    {
        pedit_buffer->cursor_pos = new_cursor_pos;
    }
}
break;
case ED_PREV_WHITE:
    if (pedit_buffer->cursor_pos > 0)
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos - 1;
            (new_cursor_pos > 0) &&
            (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
            new_cursor_pos--)
        ;
        if (isspace(pedit_buffer->pbuffer[new_cursor_pos]))
            pedit_buffer->cursor_pos = new_cursor_pos;
    }
    break;
case ED_NEXT_NONWHITE:
    new_cursor_pos = pedit_buffer->cursor_pos + 1;
    if ((new_cursor_pos < pedit_buffer->data_end) &&
        (new_cursor_pos < buffer_length))
    {
        for (;
            (new_cursor_pos < pedit_buffer->data_end) &&
            (new_cursor_pos < buffer_length) &&
            (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
            new_cursor_pos++)
        ;
        if ((new_cursor_pos < pedit_buffer->data_end) &&
            (new_cursor_pos < buffer_length) &&
            (!isspace(pedit_buffer->pbuffer[new_cursor_pos])))
        {
            pedit_buffer->cursor_pos = new_cursor_pos;
        }
    }
    break;
case ED_PREV_NONWHITE:
    if (pedit_buffer->cursor_pos > 0)
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos - 1;
            (new_cursor_pos > 0) &&
            (!isspace(pedit_buffer->pbuffer[new_cursor_pos]));
            new_cursor_pos--)
        ;

```

```

        if (lisspace(pedit_buffer->pbuffer[new_cursor_pos]))
            pedit_buffer->cursor_pos = new_cursor_pos;
    }
    break;
case ED_DELEFT: /* Delete left if not at start of line. */
    if (pedit_buffer->cursor_pos == 0)
        break;
case ED_RUBOUT: /* Move left if possible & delete char. */
    if (pedit_buffer->cursor_pos > 0)
        pedit_buffer->cursor_pos--;
case ED_DELETE: /* Delete char under cursor. */
    if (pedit_buffer->cursor_pos < (buffer_length - 1))
    {
        memmove(&pedit_buffer->pbuffer[pedit_buffer->cursor_pos],
                &pedit_buffer->pbuffer[pedit_buffer->cursor_pos + 1],
                buffer_length - pedit_buffer->cursor_pos - 1);
        if (pedit_buffer->cursor_pos < pedit_buffer->data_end)
            pedit_buffer->data_end--;
    }
    pedit_buffer->pbuffer[buffer_length - 1] = ' ';
    break;
case ED_DEL_WORD:
/* Find the start of the next word. */
    if (lisspace(pedit_buffer->pbuffer[pedit_buffer->cursor_pos]))
    {
        for (new_cursor_pos = pedit_buffer->cursor_pos;
             (new_cursor_pos < pedit_buffer->data_end) &&
             (new_cursor_pos < buffer_length) &&
             (lisspace(pedit_buffer->pbuffer[new_cursor_pos]));
             new_cursor_pos++)
            ;
    }
    else
        new_cursor_pos = pedit_buffer->cursor_pos;
    for (;
        (new_cursor_pos < pedit_buffer->data_end) &&
        (new_cursor_pos < buffer_length) &&
        (lisspace(pedit_buffer->pbuffer[new_cursor_pos]));
        new_cursor_pos++)
        ;
/* Now move the buffer left by the required amount. */
    memmove(&pedit_buffer->pbuffer[pedit_buffer->cursor_pos],
            &pedit_buffer->pbuffer[new_cursor_pos],
            buffer_length - new_cursor_pos);
/* Now clear the excess at the end of the buffer and set the new */
/* buffer end position. */
    num_chars_deleted = new_cursor_pos -
                        pedit_buffer->cursor_pos;
    memset(&pedit_buffer->pbuffer[pedit_buffer->data_end -
                                num_chars_deleted], ' ', num_chars_deleted);

```



```

    pedit_buffer->data_end -= num_chars deleted;
    break;
case ED_CLEAR:
    pedit_buffer->cursor_pos = 0;
    pedit_buffer->data_end = 0;
    memset(pedit_buffer->pbuffer, ' ', buffer length);
    break;
case ED_CLEAREOL:
    line_end = ((pedit_buffer->cursor_pos /
        pedit_buffer->dimensions.w) + 1) *
        pedit_buffer->dimensions.w - 1;
    memset(&pedit_buffer->pbuffer[pedit_buffer->cursor_pos],
        ' ', line_end - pedit_buffer->cursor_pos + 1);
    if (!trange(pedit_buffer->data_end,
        pedit_buffer->cursor_pos, line_end))
        pedit_buffer->data_end = pedit_buffer->cursor_pos;
    break;
case ED_CLEAREOB:
    memset(&pedit_buffer->pbuffer[pedit_buffer->cursor_pos],
        ' ', buffer length - pedit_buffer->cursor_pos + 1);
    if (pedit_buffer->cursor_pos < pedit_buffer->data_end)
        pedit_buffer->data_end = pedit_buffer->cursor_pos;
    break;
case ED_INSERT:
    pedit_buffer->control_flags |= ED_INSERT_MODE;
    break;
case ED_REPLACE:
    pedit_buffer->control_flags &= ~ED_INSERT_MODE;
    break;
case ED_INSTOGGLE:
    if (pedit_buffer->control_flags & ED_INSERT_MODE)
        pedit_buffer->control_flags &= ~ED_INSERT_MODE;
    else
        pedit_buffer->control_flags |= ED_INSERT_MODE;
    break;
case ED_UNDO:
    return(ED_UNDO);
case ED_TRANSMIT:
    return(ED_TRANSMIT);
}
if ((pedit_buffer->control_flags & ED_WORD_WRAP) &&
    found && wrap actions[index].wrap)
{
    edwrap(pedit_buffer);
}
}
return(pedit_key->edit_actions.pactions[current_action - 1]);
}

```

**EDCHGKEY**      加入或修改EDFIELD和WNFIELD所认可的一个按键

```
#include <bedit.h>
```

```
int edchgkey( const KEY_SEQUENCE *pkey_seq,
              const ED_ACTION_LIST *pedits);
```

**pkey\_seq** 结构的地址, 该结构指明键序列。

**pedits** 结构的地址, 该结构说明按键所执行的编辑操作的新操作表。

(返回) 错误代码。可能值包括:

**ED\_NO\_ERROR** (0) 成功。

**ED\_NO\_MEMORY** (1) 无足够动态内存来分配操作表的新成员。

**EDCHGKEY**加入或修改**EDFIELD**和**WNFIELD**所接受的按键队列中的一个按键。如果按键队列中的一个按键与**\*pkey\_seq**匹配, 则它的操作表改为**\*pedits**, 否则新按键加入到表中。

**pkey\_seq**所指向的**KEY\_SEQUENCE**结构指明这个按键。

```
typedef struct /* KEY_SEQUENCE 结构: */
{ /* 键的字符和键码。 */
```

```
    unsigned char    character_code;
```

```
    unsigned char    key_code;
```

```
} KEY_SEQUENCE;
```

**pedits**所指向的**ED\_ACTION\_LIST**结构指明编辑操作的操作表。

```
typedef struct
{ /* ED_ACTION_LIST结构: */
    int    num_actions; /* 操作码的数量, */
    ED_ACTION *pactions; /* 操作码数组。 */
} ED_ACTION_LIST;
```

数组中的每个操作是列在“域编辑(ED)”中的枚举数据类型**ED\_ACTION**的一个成员。如果按键队列未经**EDINITKY**初始化, 则**EDCHGKEY**在开始时调用**EDINITKY**。

### 源程序(EDCHG.KEY.C):

```
#include <bedit.h>
```

```
#include <bkeys.h>
```

```
#include <butil.h>
```

```
#include <string.h>
```

```
int edchgkey(pkey_sequence, paction_list)
```

```
const KEY_SEQUENCE *pkey_sequence;
```

```
const ED_ACTION_LIST *paction_list;
```

```
{
```

```
    ED_KEY    *pfound_key;
```

```
    ED_LIST    *pedit_node;
```

```
    ED_ACTION *pnew_llst;
```

```
    int    action_list_size;
```

```
    pfound_key = edretkey(pkey_sequence);
```

```
    action_list_size = paction_list->num_actions *
                      sizeof(ED_ACTION);
```

```
    if (pfound_key == NIL)
```

```
    {
```

```
        /* The specified key is not in the list, so add it. */
```

```
        pedit_node = malloc(sizeof(ED_KEY));
```

```
        if (pedit_node == NIL)
```

```
            return(ED_NO_MEMORY);
```

```
        pedit_node->edit_key.key_sequence = *pkey_sequence;
```

```
        pedit_node->edit_key.edit_actions.num_actions =
```

```

        paction_list->num_actions;
pnew_list = malloc(action_list_size);
if (pnew_list == NIL)
    return(ED_NO_MEMORY);
memmove(pnew_list, paction_list->pactions,
        action_list_size);
pedit_node->edit_key.edit_actions.pactions = pnew_list;
b_pkey_root->pprev = pedit_node;
pedit_node->pnext = b_pkey_root;
pedit_node->pprev = NIL;
b_pkey_root = pedit_node;
}
else
{
    /* Change the action list for the specified key.      */
    pfound_key->edit_actions.num_actions =
        paction_list->num_actions;
    pnew_list =
        realloc(pfound_key->edit_actions.pactions,
                action_list_size);
    if (pnew_list == NIL)
        return(ED_NO_MEMORY);
    memmove(pnew_list, paction_list->pactions, action_list_size);
    pfound_key->edit_actions.pactions = pnew_list;
}
return(ED_NO_ERROR);
}

```

## EDFIELD 对屏幕上的一个域进行编辑

```

#include <bstrings.h> /* 用于STPCVT常量。 */
#include <bedit.h>

```

```

int edfield(const char    *pinitstr,
            char    *pretstr,
            int    retsize,
            const ED_CONTROL*pctrl,
            KEY_SEQUENCE*pfinal);

```

pinitstr 显示在域中的初始字符串的地址，该字符串以NUL字符结尾。

pretstr 由用户的编辑数据所填充的缓冲区的地址。

retsize pretstr所指向的缓冲区的尺寸，结尾NUL ('\0')的一个字节也计算在内。

pctrl 决定编辑过程的结构体的地址。

pfinal 报告最后按键的结构体的址。如果不需要报告，返回NIL。

(返回) 成功代码，可能值包括：

ED_NO_ERROR	(0)	成功。
ED_NO_MEMORY	(1)	动态内存不够。
ED_ILL_DIM	(2)	pctrl->dimensions中的域尺寸非法。
ED_USER_ABORT	(302)	用户放弃编辑，不返回数据。

EDFIELD是一个宏，它允许用户编辑当前显示页上的一个域。

ED\_CONTROL结构说明各种任选项，包括屏幕上显示数据的区域。该区域在头文件bedit.h中定义为：

```

typedef struct                                /* LOC结构:  */
{                                              /* 相对于一个矩形区域的行和列  */
    int row,col;
} LOC;
typedef struct                                /* DIM 结构:  */
{                                              /* 矩形区域的高和宽。  */
    int h,w;
} DIM;
typedef struct                                /* CUR_TYPE 结构:  */
{                                              /* 光标的高低扫描行。  */
    int high, low;
} CUR_TYPE;
typedef struct                                /* ED_CONTROL 结构: */
{                                              /* 编辑缓冲区的位置和其它信息: */
    LOC ul_corne                               /* 左上角  */
    DIM dimensions;                             /* 域尺寸  */
    int attribute;                             /* 显示属性  */
    CUR_TYPE replace_cursor;                   /* 替换及插入方式下的光标尺寸  */
    CUR_TYPE insert_cursor;
    PKEY_CONTROL control_function;             /* 键控制函数  */
    unsigned int control_flags;                /* edfield ( ) 控制标志  */
} ED_CONTROL;

```

用户应按下列方式装载ED\_CONTROL的域:

用ul\_cowner和dimensions说明域的位置和大小, 域高可以是1行或多行。resize限定了用户可输入字符的数量。

attribute说明前景和背景显示属性。

设置attribute为0即使用屏幕上的现有属性。

replace\_cursor和insert\_cursor说明用户在替换方式及插入方式时显示的光标尺寸, -1使得high和low使用屏幕上现有的光标参数, 扫描行界限根据当前屏幕方式作相应的调整。参见SCPGCUR。

control\_function是一个键控制函数的地址, 每次查询键盘后该函数被调用。键控制函数接受ED\_BUFFER结构的地址作为它的KB\_DATA结构的pfunction\_data成员。参见EDBUFFER有关ED\_BUFFER结构的说明。

对control\_flags可使用下面的任选位(其中一些借自于STPCVT), 对于不需要的特性可使用零位。

符号	值	意义
ED_LEFTJUST	0x8000	退出时左对齐
ED_CENTERJUST	0x4000	退出时居中
ED_RIGHTJUST	0x2000	退出时右对齐
ED_ZERO_FILL	0x1000	退出时填充零
ED_BEEP_END	0x0800	在域的末端鸣叫
ED_AUTOSKIP	0x0400	在域的末端传送
ED_INSERT_MODE	0x0200	开始时为插入方式
ED_WORD_WRAP	0x0100	连续整字换行; 不允许两个邻接的空白。
RCONTROL	0x0080	退出时废弃全部控制字符
TOLOW	0x0040	退出时将大写字母转换为小写字母
TOUP	0x0020	退出时将小写字母改为大写字母
NOQUOTE	0x0010	退出时不修改括在单引号(')或双引号(")中的字符。
REDUCE	0x0008	退出时将全部空白压缩成一个空格
RTWHITE	0x0004	退出时废弃全部的尾随空白
RLWHITE	0x0002	退出时删去全部前导空白
RWHITE	0x0001	退出时删去全部空白

除非

tfinal

是NIL指针，否则最后的按键在\*

tfinal

中返回。

tfinal

是KEY\_SEQUENCE结构的地  
址，该结构在bkeybrd.h中定义如下：

```
typedef struct
```

```
{
    unsigned char character_code
    unsigned char key_code;
} KEY_SEQUENCE;
```

用EDINITKY、EDRETKEY、EDCHGKEY、EDREMKEY和EDZAPKEY可以维护已接受的按键命  
令队列，影响按键命令的效果。

### 源程序(BEDIT.H):

该函数以宏的准函数出现，定义在BEDIT.H中。请见附录B。

**EDINITKY**      安装EDFIELD和WNFIELD所接受的缺省按键。

```
#include <bedit.h>
```

```
int edinitky(void);
```

(返回)      错误代码。可能值包括：

ED\_NO\_ERROR      (0)      成功。

ED\_NO\_MEMORY      (1)      无足够的动态内存来分配所有的操作表成员。

EDINITKY设置EDFIELD和WNFIELD所接受的按键的缺省队列以及这些按键所执行的动作。

EDFIELD和WNFIELD自动执行EDINITKY，因此除非用户想修改缺省按键和动作的队列，否则  
并不需要调用EDINITKY。用户可以通过EDRETKEY、EDCHGKEY、EDREMKEY和EDZAPKEY来  
维护这个队列。

### 源程序(EDINITKY.C):

```
#include <butil.h>
```

```
#include <bkeys.h>
```

```
#include <bedit.h>
```

```
/* Default edit key records.      They are placed in the      */
/* linked list of edit keys at initialization.      */
```

```
ED_DEF_KEY b_defkeys[] =
```

```
{
    {ED_ABORT,      {0x1B, 0x01}},      /* Quit with no edit - ESC.      */
    {ED_LEFT,      {0x00, 0x4B}},      /* Move left - pad 4.      */
    {ED_LEFT,      {0xE0, 0x4B}},      /* Move left - left arrow.      */
    {ED_RIGHT,      {0x00, 0x4D}},      /* Move right - pad 6.      */
    {ED_RIGHT,      {0xE0, 0x4D}},      /* Move right - right arrow.      */
    {ED_UP,      {0x00, 0x48}},      /* Move up - pad 6.      */
    {ED_UP,      {0x38, 0x48}},      /* Move up - up arrow.      */
    {ED_DOWN,      {0x00, 0x50}},      /* Move down - pad 6.      */
    {ED_DOWN,      {0x32, 0x50}},      /* Move down - down arrow.      */
    {ED_FRONT,      {0x00, 0x47}},      /* Move to front - pad 7.      */
    {ED_FRONT,      {0xE0, 0x47}},      /* Move to front - Home.      */
    {ED_END,      {0x00, 0x4F}},      /* Move to end - pad 1.      */
    {ED_END,      {0xE0, 0x4F}},      /* Move to end - End.      */
    {ED_TEXT_END, {0x00, 0x75}},      /* Move to text end - C/End.      */
    {ED_NEXT_WORD, {0x06, 0x21}},      /* Forward word - C/F.      */
    {ED_PREV_WORD, {0x01, 0x1E}},      /* Back word - C/A.      */
}
```

```

{ED_DEL_WORD,{0x14, 0x14}}, /* Delete word - C/T. */
{ED_BEGIN_WORD, {0x2D, 0x4A}}, /* Begin word - pad '-'. */
{ED_END_WORD,{0x2B, 0x4E}}, /* End word - pad '+'. */
{ED_NEXT_WHITE, {0x00, 0x3C}}, /* Next whitespace - F2. */
{ED_PREV_WHITE, {0x00, 0x3B}}, /* Previous whitespace - F1. */
{ED_NEXT_NONWHITE,{0x00, 0x3E}}, /* Next non whitespace - F4. */
{ED_PREV_NONWHITE,{0x00, 0x3D}}, /* Previous non whitespace - F3.*/
{ED_DELETE, {0x00, 0x53}}, /* Delete char - pad period. */
{ED_DELETE, {0xE0, 0x53}}, /* Delete character - Delete. */
{ED_RUBOUT, {0x08, 0x0E}}, /* Move left delete - Backspace.*/
{ED_DELEFT, {0x08, 0x23}}, /* Delete char left - C/H. */
{ED_CLEAR, {0x00, 0x77}}, /* Clear buffer - C/pad 7. */
{ED_CLEAR, {0xE0, 0x77}}, /* Clear buffer - C/Home. */
{ED_CLEAREOL, {0x00, 0x74}}, /* Clear to end - C/pad 6. */
{ED_CLEAREOL, {0xE0, 0x74}}, /* Clear to end - C/Right. */
{ED_INSERT, {0x00, 0x12}}, /* Insert mode - Alt/E. */
{ED_REPLACE, {0x00, 0x13}}, /* Replace mode - Alt/R. */
{ED_INSTOGGLE{0x00, 0x52}}, /* Toggle INS mode - pad 0. */
{ED_INSTOGGLE{0xE0, 0x52}}, /* Toggle INS mode - Insert. */
{ED_UNDO, {0x00, 0x2D}}, /* Restore buffer - Alt/X. */
{ED_TRANSMIT, {0x0D, 0x1C}}, /* Finish edit - Enter. */
{ED_TRANSMIT, {0x0D, 0xE0}}, /* Finish edit - pad Enter. */
{ED_INVALID_ACTION, /* End of edit key entries. */
{KBNDEF,KBNDEF}}
};

ED_LIST *b_pkey_root = NIL; /* Edit key list head. */
int b_pkeys_init = 0; /* 0 = key list uninitialized. */
int edinitky()
{
    int i;
    ED_LIST *new_key_node;
    if (b_pkey_root != NIL)
        edzapkey();
        /* Initialize the list of default edit keys. For */
        /* each key in the default key table, a new record */
        /* is allocated, initialized and placed at the */
        /* beginning of the list. */
    i = 0;
    while (b_defkeys[i].action_code != ED_INVALID_ACTION)
    {
        new_key_node = malloc(sizeof(ED_LIST));
        if (new_key_node == NIL)
            return(ED_NO_MEMORY);
        new_key_node->edit_key.edit_actions.num_actions = 1;
        new_key_node->edit_key.edit_actions.pactions =
            malloc(sizeof(ED_ACTION));
        *new_key_node->edit_key.edit_actions.pactions =
            b_defkeys[i].action_code;
        new_key_node->edit_key.key_sequence =
            b_defkeys[i].key_sequence;
    }
}

```

```

    if (b_pkey_root != NIL)
    {
        b_pkey_root->pprev = new_key_node;
        new_key_node->pnext = b_pkey_root;
        new_key_node->pprev = NIL;
        b_pkey_root = new_key_node;
    }
    else
    {
        new_key_node->pnext = NIL;
        new_key_node->pprev = NIL;
        b_pkey_root = new_key_node;
    }
    i++;
}
b_pkeys_init = 1;          /* 1 = key list initialized. */
return(ED_NO_ERROR);
}

```

**EDREDUCE** 把连续的空白转换成单个空格。

#include <bedit.h>

edreduce(pedit\_buffer);

pedit\_buffer 指向转换的缓冲区。

EDREDUCE扫描\*pedit\_buffer的pbuffer成员，把连续的空白转换成单个空格。相对调整data\_end和cursor\_pos域。

返回 无

**源程序(EDREDUCE.C):**

#include <bedit.h>

void edreduce(pedit\_buffer)

ED\_BUFFER \*pedit\_buffer;

```

{
    int current_char;
    int in_white = 0;
    int data_end = pedit_buffer->data_end;
    int cursor_pos = pedit_buffer->cursor_pos;
    char *pdest;
    pdest = pedit_buffer->pbuffer;
    for (current_char = 0; current_char < pedit_buffer->data_end;
        current_char++)
    {
        if (isspace(pedit_buffer->pbuffer[current_char]))
        {
            if (!in_white)
            {
                in_white = 1;
                *pdest++ = ' ';
            }
            else

```

```

        {
            if (pedit_buffer->data_end >= current_char)
                data_end--;
            if (pedit_buffer->cursor_pos >= current_char)
                cursor_pos--;
        }
    }
    else
    {
        in_white = 0;
        *pdest++ = pedit_buffer->pbuffer[current_char];
    }
}
while (pdest < (pedit_buffer->pbuffer + pedit_buffer->buffer_size))
{
    *pdest++ = ' ';
}
pedit_buffer->data_end = data_end;
pedit_buffer->cursor_pos = cursor_pos;
}

```

**EDREMKEY** 删去EDFIELD和WNFIELD所接受的一个按键。

#include <bedit.h>

int edremkey(const KEY\_SEQUENCE \*pkey\_seq);

pkey\_seq 结构的地址, 该结构说明待删去的键序列。

(返回) 错误代码。可能值包括:

ED\_NO\_ERROR (0) 成功。

ED\_NO\_KEY (300) 在队列中未发现对应的按键

EDREMKEY从EDFIELD和WNFIELD所接受的按键队列中删去一个按键。

如果按键队列还未被EDINITKY初始化, 则EDREMKEY在开始时调用EDINITKY。

**源程序(EDREMKEY.C):**

#include <bedit.h>

int edremkey(pkey\_sequence)

const KEY\_SEQUENCE \*pkey\_sequence;

```

{
    ED_KEY *pfound_key;
    ED_LIST *pfound_node;
    pfound_key = edretkey(pkey_sequence);
    if (pfound_key == NIL)
        /* The key sequence is not in the edit list. */
        return(ED_NO_KEY);
    /* Locate the edit key list node corresponding to the edit
     * key given. */
    for (pfound_node = b_pkey_root;
        (pfound_node != NIL) && (&pfound_node->edit_key != pfound_key);
        pfound_node = pfound_node->pnext)
        ;
    /* If the member is the first element of the list (that is, */

```



```

/* pointed to by the root), just after the root to point to */
/* the next element of the list. If it is not the first */
/* element of the list, surgically remove it from the edit */
/* key list by altering the pnext and pprev pointers of the */
/* nodes surrounding it. */
if (b_pkey_root == pfound_node)
{
    b_pkey_root = b_pkey_root->pnext;
    b_pkey_root->pprev = NIL;
    pfound_node->pnext = pfound_node->pprev = NIL;
}
else
{
    if (pfound_node->pprev != NIL)
        pfound_node->pprev->pnext = pfound_node->pnext;
    if (pfound_node->pnext != NIL)
        pfound_node->pnext->pprev = pfound_node->pprev;
    pfound_node->pnext = pfound_node->pprev = NIL;
}
free(pfound_key->edit_actions.pactions);
free(pfound_key);
free(pfound_node);
return(ED_NO_ERROR);
}

```

**EDRETINF** 返回当前光标和视屏信息。

```

#include <bscreen.h>
#include <bedit.h>
error = edretinf( pdata, pinfo, pfield_loc,
                  field height, field width)
error      返回的错误代码。
pdata      无用；参见下面的内容。
pinfo      指向被填充的结构。
pfield_loc 指向建议的域位置的指针。
field height 显示的域高
field width 显示的域宽

```

**EDRETINF**返回编辑域显示的光标和视屏页。信息放在pinfo指向的区域。pdata参数从不使用；包括它是为了与**WNRETINF**的调用序列兼容。

返回           int error           **ED\_NO\_ERROR** - 没有发生错误。  
                                   **ED\_ILL\_DIM**- 域不放在活动显示页上。

**源程序(EDRETINF.C):**

```

#include <bcreens.h>
#include <bedit.h>
int edretinf(pdata, pinfo, pfield_loc, field height, field width)
void        *pdata;
CUR_INFO* pinfo;
const LOC *pfield_loc;
int         field height;

```

```

int      field_width;
{
    int mode, columns, act_page;
        /* Supress unused parameter compiler warnings.      */
    ((char *) pdata)++;
    mode = columns = act_page = 0;
    scmode(&mode, &columns, &act_page);
    pinfo->off = sccurst(&(pinfo->row),
                        &(pinfo->col),
                        &(pinfo->size.high),
                        &(pinfo->size.low));
    if (utrange(pfield_loc->row, 0, scrows() - field_height) ||
        utrange(pfield_loc->col, 0, columns - field_width))
    {
        return(ED_ILL_DIM);
    }
    return(ED_NO_ERROR);
}

```

## EDRETKEY 报告一个按键的编辑动作

#include <bedit.h>

ED\_KEY \*edretkey(const KEY\_SEQUENCE \*pkey\_seq);

pkey\_seq 结构的地址，该结构说明需报告的键序列。

(返回) 包含键动作的结构地址。如果未发现，返回NIL。

EDRETKEY报告与一个按键关联的编辑动作，这个按键来自于EDFIELD和WNFIELD所接受的按键队列。

如果在队列中找到了这个按键，则EDRETKEY返回一个ED\_KEY结构的地址，该结构在bedit.h中定义如下：

```

typedef struct
{
    /* ED_ACTION_LIST 结构: */
    int      num_actions;      /* 动作码的数量 */
    ED_ACTION *pactions;      /* 动作码数组 */
} ED_ACTION_LIST;

typedef struct
{
    /* KEY_SEQUENCE 结构: */
    /* 一个键的字符和键码。 */
    unsigned char  character_code;
    unsigned char  key_code;
} KEY_SEQUENCE;

typedef struct
{
    /* ED_KEY 结构: */
    ED_ACTION_LIST edit_actions; /* 编辑键的编辑动作、键序列 */
    KEY_SEQUENCE key_sequence; /* 和属性。 */
    int      attribute;
} ED_KEY;

```

ED\_KEY 结构的成员解释如下：

edit\_actions包含着待执行的动作表(请参看下面的示例)，每个操作是列举在“域编辑(ED)”中的枚举数据类型ED\_ACTION的一个成员。

当动作为ED\_ASCII时，key\_sequence.character\_code是加入到缓冲区中的字符。

当动作为ED\_ATTR时，attribute是缓冲区的新属性。

当队列未用EDINITKY初始化时，EDRETKEY自动调用EDINITKY。

源程序(EDRETKEY.C):

```
#include <butil.h>
#include <bedit.h>
ED_KEY *edretkey(pkey_sequence)
const KEY_SEQUENCE *pkey_sequence;
{
    ED_LIST *pcurrent;
    int error_code;
    if (b_pkeys_init == 0)
    {
        error_code = edinitky();
        if (error_code != ED_NO_ERROR)
            return(NIL);
    }
    pcurrent = b_pkey_root;
    /* Do a sequential search of the linked list until the */
    /* key is found, or the list is exhausted. */
    while (pcurrent != NIL)
        if ((pcurrent->edit_key.key_sequence.character_code ==
            pkey_sequence->character_code) &&
            (pcurrent->edit_key.key_sequence.key_code ==
            pkey_sequence->key_code))
            break;
        else
            pcurrent = pcurrent->pnext;
    /* If the edit key was found, pcurrent points to it; */
    /* otherwise, pcurrent is NIL. */
    if (pcurrent != NIL)
        return(&pcurrent->edit_key);
    else
        return(NIL);
}
```

**EDSETCUR** 设置光标的尺寸和位置。

#include <bedit.h>

error = edsetcur(pdata, off, pcursor, row, col);

error 返回的错误代码。

pdata 无用；参见下面的内容。

off 光标的新状态。(0 = on, 1 = off).

pcursor 指向包含光标尺寸的结构。

row,col 新光标位置相对于屏幕左上角的行和列。

移动了光标，光标的尺寸被设置成在\*pcursor中的值。pdata参数无用，用户与WNRETINF的调用序列兼容。

返回 error ED\_NO\_ERROR

源程序(EDSETCUR.C):

#include <bedit.h>

```

int edsetcur(pdata, off, pcursor, row, col)
void      *pdata;
int       off;
CUR_TYPE *pcursor;
int       row;
int       col;
{
    /* Suppress unused parameter compiler warnings. */
    ((char *) pdata)++;
    sccurset(row, col);
    scpgcur(off, pcursor->high, pcursor->low, CUR_ADJUST);
    return(ED_NO_ERROR);
}

```

**EDWRAP** 向编辑区写入字符时，带有整字换行。

```
#include <bedit.h>
```

```
edwrap(pedit_buffer)
```

**pedit\_buffer** 编辑缓冲区结构。

EDWRAP接受指向编辑缓冲区的指针，并进行整字换行处理。字定义成由空白包围其中没有空白的一系列字符。当字遇到域的边界时，在其前插入空格，强迫字写入下一行。如果字大于一行长，则折开它。整字换行可能使数据超出缓冲区末，以至于数据丢失。

返回 无

**源程序(EDWRAP.C):**

```
#include <bedit.h>
```

```
void edwrap(pedit_buffer)
```

```
ED_BUFFER *pedit_buffer;
```

```

{
    int current_row;
    int last_char_on_row;
    int current_index;
    /* If the displayed field is only one row, there's nothing to do. */
    if (pedit_buffer->dimensions.h == 1)
        return;
    for (current_row = 0; current_row < (pedit_buffer->dimensions.h - 1);
        current_row++)
    {
        last_char_on_row = (pedit_buffer->dimensions.w *
                           (current_row + 1)) - 1;
        if (last_char_on_row >= pedit_buffer->buffer_size)
            return;
        /* Find the last non-space on the current line. */
        for (current_index = last_char_on_row;
            (current_index > last_char_on_row -
             pedit_buffer->dimensions.w) &&
            (lisspace(pedit_buffer->pbuffer[current_index]));
            current_index--);
        /* Check to see if we should wrap here. */
    }
}

```

```

if (!inrange(current_index, last_char_on_row -
    pedit_buffer->dimensions.w + 1,
    last_char_on_row - 1) &&
    !isspace(pedit_buffer->pbuffer[current_index + 1]))
{
    current_index++;
    memmove(&(pedit_buffer->pbuffer[last_char_on_row + 1]),
        &pedit_buffer->pbuffer[current_index],
        pedit_buffer->buffer_size - last_char_on_row - 1);
    memset(&pedit_buffer->pbuffer[current_index], ' ',
        (last_char_on_row - current_index) + 1);
    /* Update current end of data, if necessary. */
    if (pedit_buffer->data_end >= current_index)
    {
        pedit_buffer->data_end += (last_char_on_row -
            current_index) + 1;
        utuplim(pedit_buffer->data_end,
            pedit_buffer->buffer_size);
    }
    /* Update current edit position, if necessary. */
    if (pedit_buffer->cursor_pos >= current_index)
    {
        pedit_buffer->cursor_pos += (last_char_on_row -
            current_index) + 1;
        utuplim(pedit_buffer->cursor_pos,
            pedit_buffer->buffer_size - 1);
    }
}
}
}
}

```

**EDWRRECT** 向屏幕写入一长方形图形。

```
#include <bedit.h>
```

```
edwrrect( pdata, u_row, u_col, l_row, l_col, buffer, gap,
    fore, back, option);
```

**pdata** 无用，用来与传递给VIWRRECT的参数一致。

**EDWRRECT**简单地丢弃**pdata**参数，并以其余的参数调用**VIWRRECT**，提供给具有相同的调用序列但使用**VIWRRECT**的**WNWRRECT**作为子程序。

返回 无

源程序(EDWRRECT.C):

```
#include <bvideo.h>
```

```
#include <bedit.h>
```

```
void edwrrect(pdata, u_row, u_col, l_row, l_col, buffer, fore, back,
    option)
```

```
void *pdata;
```

```
int u_row, u_col, l_row, l_col;
```

```
const char *buffer;
```

```
int fore, back, option;
```

```

{
    /* Suppress unused parameter compiler warnings. */
    ((char *) pdata)++;
    vwrtrect(u_row, u_col, l_row, l_col, buffer, fore, back, option);
}

```

## EDZAPKEY 删去EDFIELD和WNFIELD所接受的按键队列

```
#include <bedit.h>
```

```
void edzapkey(void);
```

EDZAPKEY 删去EDFIELD和WNFIELD所接受的整个按键队列。

用EDREMKEY删去队列中个别的按键。

## 源程序(EDZAPKEY.C):

```
#include <butil.h>
```

```
#include <bedit.h>
```

```
void edzapkey()
```

```

{
    ED_LIST *p_current_item, *p_next_item;
    p_current_item = b_pkey_root;
    while (p_current_item != NIL)
    {
        p_next_item = p_current_item->pnext;
        p_current_item->pnext = NIL;
        p_current_item->pprev = NIL;
        free(p_current_item->edit_key.edit_actions.pactions);
        free(p_current_item);
        p_current_item = p_next_item;
    }
    b_pkey_root = NIL;
}

```

## 编辑函数程序设计示例

DEMOEDT.C 编辑函数用法演示程序。

```
#include <bedit.h>
```

```
#include <bstrings.h>
```

```
#include <stdio.h>
```

```
#define FIELD_WIDTH 30
```

```
#define FIELD_HEIGHT 5
```

```
void main(void);
```

```
char returned_string[FIELD_WIDTH * FIELD_HEIGHT + 1];
```

```
void main()
```

```

{
    int          error_code;
    ED_CONTROL   field_control;
    int          cursor_was_off,row,col,high,low;
    char         *pinitial_string =
        "This is a demonstration of the Turbo C TOOLS edit functions.";
    KEY_SEQUENCE final_key;

```

```

cursor was_off = securst(&row,&col,&high,&low);
field_control.ul_corner.row      = 0;
field_control.ul_corner.col      = 0;
field_control.dimensions.h       = FIELD_HEIGHT;
field_control.dimensions.w       = FIELD_WIDTH;
field_control.attribute          = 0x0007; /* Reverse video. */
field_control.replace_cursor.high = 6;    /* Underline    */
field_control.replace_cursor.low  = 7;    /* cursor.      */
field_control.insert_cursor.high  = 0;    /* Solid cursor. */
field_control.insert_cursor.low   = 7;
field_control.control_function    = (PKEY_CONTROL) NIL;
field_control.control_flags       = 0;
sepcr();
error_code = edfield(pinitial_string, returned_string,
                    sizeof(returned_string),
                    &field_control, &final_key);
secursor(FIELD_HEIGHT + 1, 0);
sepgcur(cursor was_off, high, low, CUR_NO_ADJUST);
if ((error_code != ED_NO_ERROR) && (error_code != ED_USER_ABORT))
{
    printf("Error %d editing.\n", error_code);
    exit(-1);
}
printf("Final key: char %02x, key %02x.\n",
       final_key.character_code, final_key.key_code);
printf("Returned string:\n\"%s\"\n", returned_string);
}

```

## KEYCTRL.C 键控制函数的样例。

```

#include <bkeybrd.h>
#include <butil.h>
#include <bvideo.h>
#include <ctype.h>
#include <stdio.h>

/* Function prototypes. */
void main(void);
void key_control(KB_DATA *);

/* Global variables. */
int columns;
void main()
{
    int final_key;
    int num_scrolled;
    int mode, active_page;
    char user_response[41];

    /* Remember the number of columns so that the key */
    /* control function will know where to display the */
    /* time. */
}

```

```

    scmode(&mode, &columns, &active_page);
    puts("Please enter your message (Press Enter to terminate):");
        /* Now install the key control function and get a */
        /* response from the user. */
    b_key_ctrl = key_control;
    kbquery(user_response, sizeof(user_response), &final_key,
        &num_scrolled);
        /* Echo the user's response. */
    puts("\nYour message:");
    puts(user_response);
}
/**

```

**KEY CONTROL** -- 函数把所有字符串换成大写，并显示在当前时间。  
 由KBQUERY作为键控制函数调用。  
 函数从KBQUERY每次搜寻键盘时获得控制。每当键入一字母键时，转换成大小，并返回给调用者。无论是否击键，当前时间显示于屏幕右上角。  
 返回 无  
 \*\*/

```

void key_control(pkey_data)
KB_DATA *pkey_data;
{
    long tick_count;
    int hours, minutes, seconds, hundredths;
    char time[9];
        /* If a key was struck and it was alphabetic, */
        /* convert it to upper case. Note that the key */
        /* code is changed to match the new character code. */
    if (pkey_data->key_found)
    {
        if (isalpha(pkey_data->key_seq.character_code))
        {
            pkey_data->key_seq.character_code =
                toupper(pkey_data->key_seq.character_code);
            pkey_data->key_seq.key_code =
                kbscanof(pkey_data->key_seq.character_code);
        }
    }
        /* Now get the current time of day and display it. */
    utgetclk(&tick_count);
    uttk2tim(tick_count, &hours, &minutes, &seconds, &hundredths);
    sprintf(time, "%d:%02d:%02d", hours, minutes, seconds);
    vidspmsg(0, columns - strlen(time), SC_BLACK, SC_WHITE, time);
}

```

## ENTRYEDT.C 是面向窗口多编辑域的程序。

程序显示包含几个由简单的用户信息记录组成的编辑域的窗口，并请求输入。  
 “state”域有一个与之相联的键存取子程序，以确保用户只键入某些状态名；“zip”域也有一以确保用户只键入数字的键存取子程序。按Enter接收每个域，当按下F10时接收整屏。



命令行格式:

entryedt

ENTRYEDT提供一个在一窗口中正确使用多个编辑域的例子。

```
#include <bedit.h>
#include <hwindow.h>
#include <hstrings.h>
#include <bkeys.h>
#include <stdio.h>

/* Internal function prototypes. */
void main(void);
void state_validation(KB_DATA *);
void zip_validation(KB_DATA *);

/* These constants define the number of columns
/* each field will have when it is displayed on the */
/* screen. */
#define NAME_WIDTH 40
#define ADDRESS_WIDTH 40
#define CITY_WIDTH 20
#define STATE_WIDTH 2
#define ZIP_WIDTH 5

/* Allocate storage for character buffers in which
/* the data for the fields will be returned. */
char name[NAME_WIDTH + 1] = "";
char address[ADDRESS_WIDTH + 1] = "";
char city[CITY_WIDTH + 1] = "";
char state[STATE_WIDTH + 1] = "";
char zip[ZIP_WIDTH + 1] = "";
typedef struct /* FIELD_DESCRIPTOR structure: */
{ /* field info and data buffer. */
    ED_CONTROL control; /* Field control structure. */
    int buffer_size; /* Size of pbuffer in bytes. */
    char *pbuffer; /* Buffer in which to return data. */
} FIELD_DESCRIPTOR;

/* screen_fields is an array of FIELD_DESCRIPTOR
/* structures, each of which describes one field on */
/* the screen. */
FIELD_DESCRIPTOR screen_fields[] =
{
    {{1, 14}, {1, NAME_WIDTH}, 0x74, /* Name field. */
     {6, 7}, {0, 7}, 0, 0},
    sizeof(name), name},
    {{2, 14}, {1, ADDRESS_WIDTH}, 0x74, /* Address field. */
     {6, 7}, {0, 7}, 0, 0},
    sizeof(address), address},
    {{3, 14}, {1, CITY_WIDTH}, 0x74, /* City field. */
     {6, 7}, {0, 7}, 0, 0},
    sizeof(city), city},
    {{4, 14}, {1, STATE_WIDTH}, 0x74, /* State field. */
     {6, 7}, {0, 7}, state_validation, TOUP},
    sizeof(state), state},
```

```

        {{5, 14}, {1, ZIP_WIDTH}, 0x74,          /* Zip field.      */
        {6, 7}, {0, 7}, zip_validation, 0},
        sizeof(zip), zip)
};
typedef struct          /* FIELD_LABEL structure: location & text */
{                      /* of one screen field label.      */
    int    row;         /* The row at which to place the text.    */
    int    col;         /* The column at which to place the text. */
    char *ptext;        /* The text to use as the field label.    */
} FIELD_LABEL;

/* field_labels is an array of FIELD_LABEL */
/* structures, each of which contains the location */
/* and text for one the label for one field. */

FIELD_LABEL field_labels[] =
{
    {1, 8, "Name:"},
    {2, 5, "Address:"},
    {3, 8, "City:"},
    {4, 7, "State: (AZ, CA, OR, WA or NV only)"},
    {5, 9, "ZIP:"}
};

BWINDOW *pwin;
void main()
{
    int          error_code;
    BORDER       bord;
    WHERE        location;
    int          mode,column,act_page;
    int          cursor_was_off,row,col,high,low;
    KEY_SEQUENCE key_sequence;
    ED_ACTION     action;
    ED_ACTION_LISTkey_action_list;
    int          index;

        /* First, create the window and write the field */
        /* labels to their appropriate locations.      */

    pwin = wncreate(8, 60, SC_CYAN);
    if (pwin == NIL)
        exit(-1);
    wnselect(pwin);
    for (index = 0;
        index < sizeof(field_labels) / sizeof(field_labels[0]);
        index++)
    {
        if (wnwrbuf(field_labels[index].row,
                    field_labels[index].col, 0,
                    field_labels[index].ptext,
                    -1, -1, CHARS_ONLY) == 0)
        {
            printf("Error %d writing to window.\n", b_werr);
            exit(-1);
        }
    }
}

```

```

    }
}

    /* Install the F10 key as a transmission key, which */
    /* will be used as a signal that the user wishes to */
    /* transmit the screen. */
key_sequence.character_code = KB_C_N_F10;
key_sequence.key_code       = KB_S_N_F10;
key_action_list.num_actions = 1;
action                     = ED_TRANSMIT;
key_action_list.pactions   = &action;
if (ED_NO_ERROR != edchgkey(&key_sequence, &key_action_list))
{
    printf("Error %d adding keystroke.\n", b_werr);
    exit(-1);
}

    /* Now set up the BORDER and WHERE structures. */
bord.type       = BBRD_SSSS | BBRD_TCT;
bord.lattr      = SC_MAGENTA;
bord.ttattr     = SC_CYAN;
bord.pttitle    = " Press f10 to transmit... ";
location.dev    = scmode(&mode,&columns,&act_page);
location.page   = act_page;
location.corner.row = 10;
location.corner.col = 5;

    /* Remember the cursor size and state to restore */
    /* later, and display the window. */
cursor_was_off = scurst(&row,&col,&high,&low);
if (NIL == wndsplay(pwin, &location, &bord))
{
    printf("Error %d displaying window.\n", b_werr);
    exit(-1);
}

    /* Now read the fields on the screen in sequence */
    /* until the user presses F10. */
index = 0;
do
{
    error_code = wnfield(pwin,
                        screen_fields[index].pbuffer,
                        screen_fields[index].pbuffer,
                        screen_fields[index].buffer_size,
                        &(screen_fields[index].control),
                        &key_sequence);

    index++;
    if (index >= (sizeof(screen_fields) / sizeof(screen_fields[0])))
        index = 0;
}
while ((key_sequence.character_code != KB_C_N_F10) ||
      (key_sequence.key_code      != KB_S_N_F10));
if (error_code != WN_NO_ERROR)

```

```

    {
        printf("Error %d editing in window.\n", error_code);
        exit(-1);
    }
    wnremove(pwin);
    wndstroy(pwin);
    if (b_wnerr)
    {
        printf("Error %d removing or destroying window.\n", b_wnerr);
        exit(-1);
    }
    scurset(row, col);
    scpgcur(cursor_was_off, high, low, CUR_NO_ADJUST);
    printf("Name:      \\\n", name);
    printf("Address: \\\n", address);
    printf("City:      \\\n", city);
    printf("State:     \\\n", state);
    printf("Zip:       \\\n", zip);
}

```

/\*\*

**VALIDATE STATE** - 验证数据录入屏幕的“state”域的数据。

只作为键存取函数的域编辑函数调用。

函数从域编辑函数的每次取键中取得控制，检查是否已按键，并检查是否是接收键(Enter或F10)。如果是接收键，则检查在编辑缓冲区的数据(由pkey\_data中的pdata指出) 否有效，如果无效则显示一信息，信息在击下一次键时消除。

返回 无

\*/

void state\_validation(pkey\_data)

KB\_DATA \*pkey\_data;

```

{
    static int    message_displayed = 0;
    static char *pmessage = "Please enter a west coast state.";
    static char *legal_states[] = {"AZ", "CA", "OR", "WA", "NV"};
    char          *phbuffer;
    int            found;
    int            current_state;
    int            num_states =
        sizeof(legal_states) / sizeof(legal_states[0]);
    if (pkey_data->key_found)
    {
        if (message_displayed)
        {
            wnscribk(pwin, 7, 0, 7, strlen(pmessage) - 1,
                -1, -1, WNSCR_UP, 0, WN_UPDATE);
            message_displayed = 0;
        }

        /* Check to see if the key was an Enter or F10. */
        if (((pkey_data->key_seq.character_code == KB_C_N_ENTER) &&

```

```

        (pkey_data->key_seq.key_code == KB_S_N_ENTER))    ||
        ((pkey_data->key_seq.character_code == KB_C_N_F10)  &&
        (pkey_data->key_seq.key_code == KB_S_N_F10)))
    {
        found = 0;
        num_states = sizeof(legal_states) / sizeof(legal_states[0]);
        pbuffer = ((ED_BUFFER *)
                    pkey_data->pfunction_data->pbuffer;
        /* Validate the data. */
        for (current_state = 0; (current_state < num_states) &&
            !found; current_state++)
        {
            if (!strnicmp(pbuffer, legal_states[current_state],
                strlen(legal_states[current_state])))
            {
                found = 1;
            }
        }
        /* If the data was not valid, display a message and */
        /* ignore the transmission key. */
        if (!found)
        {
            wnwrrect(pwin, 7, 0, 7, strlen(pmessage) - 1,
                    pmessage, -1, -1, CHARS_ONLY);
            scttywrt('\a', 0);
            message displayed = 1;
            pkey_data->key_found = 0;
        }
    }
}

```

/\*\*

VLEIDATE\_ZIP - 验数据录入屏幕上的“zip”域。

只由域编辑函数作为键存取子程序调用。

在域编辑函数和每次取键时获得控制。检查是否已按键，并判别是否为“0”-“9”的ASCII数字。如果不是，则扬声器以表示键入无效数据。

返回 无

\*/

void zip\_validation(pkey\_data)

KB\_DATA \*pkey\_data;

```

{
    if (pkey_data->key_found)
    {
        /* Check to see if the key was Enter, F10, or ESC. */
        if (!(((pkey_data->key_seq.character_code == KB_C_N_ENTER) &&
            (pkey_data->key_seq.key_code == KB_S_N_ENTER)) ||
            ((pkey_data->key_seq.character_code == KB_C_N_F10) &&
            (pkey_data->key_seq.key_code == KB_S_N_F10)) ||
            ((pkey_data->key_seq.character_code == KB_C_N_ESC) &&

```

```

    (pkey_data->key_seq.key_code == KB_S_N_ESC))))
{
    /* Check to see if the key was a number. */
    if (utrange(pkey_data->key_seq.character_code,
                KB_C_N_0, KB_C_N_9)
        ||
        utrange(pkey_data->key_seq.key_code,
                KB_S_N_1, KB_S_N_0))
    {
        /* Key is non-numeric, so beep and swallow the key. */
        scttywrtr("\a", 0);
        pkey_data->key_found = 0;
    }
}
}
}

```

## 第三章 文件操作

### 文件管理函数(FILE)

从版本2.00开始, DOS提供了一套灵活方便的功能用于管理文件和目录。Turbo C所带的库提供了访问大部分DOS文件服务的手段。文件管理函数的目的是提供访问一些有用的而Turbo C库未包含进来的DOS文件服务。

文件管理函数独立于硬件, 因此它可以运行在装有DOS 2.00或更高版本的系统上。

当DOS打开一个文件时, 它返回文件的句柄。文件柄是一个用于引用已打开文件的整数值, 并不对应于标准fopen()函数返回的FILE指针。Turbo C将文件柄作为FILE结构的fd成员提供给用户。标准open()和fileno()函数的Turbo C版本将文件柄作为函数值返回。

### 文件管理函数的种类

#### 维护卷标

FLRETVOL	返回给定驱动器的卷标(如果有的话)。
FLSETVOL	建立或修改给定驱动器的卷标。
FLREMOVOL	删除一个卷标。

#### 杂类

FLFLUSH	迫使一个已打开文件的修改内容写入磁盘。
FLPROMPT	在当前控制台上显示一个提示信息(除非标准输出已被定向到一个磁盘文件)并从标准输入返回一行输入。(请与KBQUERY相比较。)
FLNORM	检查一个文件名的文法正确性, 将它转换为标准形式(使之标准化), 以便于同其它文件名相比较。
FLLOCK	在网络环境中(DOS 3.10或更高版本)对一个已打开文件的一个区域上锁或解锁。
FLDOLOCK	在网络环境中(DOS 3.10或更高版本)对一个已打开文件的一个区域上锁或解锁。如果该区域已被其它程序上锁, FLDOLOCK则以智能方式等待。
FLEGTDTA	报告DOS磁盘传送地址(DTA)。
FLPUTDTA	设置DOS磁盘传送地址(DTA)。

### 文件操作函数源程序和使用参考

以下列出文件操作的扩展函数源程序。

**FLDOLOCK** 对已打开文件一个文件段上锁或解锁, 必要时等待

```
#include <bfiles.h>
```

```
int fldolock(    int handle,  
                unsigned long offset,  
                unsigned long length,  
                int seconds,  
                int option);
```

handle 已打开文件的句柄。

offset, length

待上锁区域开始位置的偏移和长度(以字节计)。

seconds 当区域已被锁住时等待成功的时间。

option FL\_LOCK (0) 或 FL\_UNLOCK (1)。

(返回) 返回的DOS的错误代码。可能值包括：

- 0 成功。
- 1 未知功能：非法的任选项值或DOS版本小于3.10或SHARE未安装。
- 6 不是一个已打开文件的柄。
- 33 上锁违例(见下面)。

FLDOLOCK对与指定的柄关联的已打开文件的一个文件段上锁或解锁，防止其它程序(甚至是子进程)访问这个被上锁的文件段。如果该段已被上锁，FLDOLOCK可以等待一段指定的时间，看看锁是否已被解除。

用户需要DOS 3.10或更高版本，也必须安装DOS的SHARE程序。对于早期版本的DOS来说，返回值1表示功能号未知。参见DOS参考手册中有关SHARE的安装说明。

指定的文件区域可以包括整个文件乃至延伸越过文件尾。

建议按下述的技术使用文件锁：

- 1) 用FLDOLOCK尝试锁住一个区域。
- 2) 如果由于区域已被锁住而使上锁的尝试失败(返回值33)，则不断尝试，或者报告一个错误。
- 3) 锁住该区域后，读或写数据。
- 4) 用FLLOCK对该区域解锁。

如果(上锁时)区域的某部分已被锁住，或(解锁时)指定的区域目前未被该程序锁住，则出现锁操作违例。

如果未打开所有的锁，无论如何也不要关闭文件或让程序上止(即使是Ctrl-Break或DOS严重错误)。

源程序(FLDOLOCK.C)：

```
#include <dos.h>
#include <errno.h>
#include <io.h>
#include <time.h>
#include <bfiles.h>
#define ALREADY_LOCKED 0x21
int fdlock (handle, offset, length, seconds, option)
int      handle, option, seconds;
unsigned long offset, length;
{
    int    result;
    long   start_second, now_second;
           /* DOS 2.x does not know about file locking. */
    if (utdosmajor < 3)
        return (1);
    time (&start_second);
    do
    {
        /* (Note:  DOS 3.00 does not support function 0x5c */
        /* but at least it reports a valid error code & */
        /* extended error information.) */
        /* If there was an error, and it was not a "Lock */
        /* Violation" error, return immediately -- If there */
        /* wasn't an error, return. */
    }
```



```

    if ((result = flock (handle, option, offset, length)) == 0)
        return (0);
    else
        if (result != ALREADY_LOCKED)
            return ((int) result);
    time (&now_second);
} while ((now_second - start_second) < (long) seconds);
    /* If we got an error on our last call (and we got
    /* to here so it must be error 0x21), return it.
return (result);
}

```

## FLFLUSH 迫使挂起的文件输出写到磁盘上

```
#include <bfiles.h>
```

```
int flflush(int handle);
```

handle 已打开文件的柄

(返回) DOS错误代码。可能值包括:

0 没有错误。

4 打开的柄太多。

6 不是已打开文件的柄。

FLFLUSH清空与一个给定的文件柄关联的DOS缓冲区, 它只对做写操作的文件有用, 保证数据确实放在磁盘上而不只是放在内存缓冲区中。

如果通过可移植的流函数执行文件输出(fwrite()、printf()、puts()和同类函数), 一定要首先利用fflush()清空标准库的缓冲区。

源程序(FLFLUSH.C):

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

```
#include <bfiles.h>
```

```
int flflush (handle)
```

```
int handle;
```

```

{
    union REGS regs;
        /* Try to duplicate the file handle.
regs.x.ax = 0x4500;
regs.x.bx = handle;
int86 (FL_DOS_INT, &regs, &regs);
        /* If we didn't get an error, now we will close the
        /* new handle to effect the flush operation.
if (! (regs.x.cflag))
{
    /* The new handle was returned to us in AX. Put it
    /* in BX for the close.
regs.x.bx = regs.x.ax;
regs.x.ax = 0x3e00;
int86 (FL_DOS_INT, &regs, &regs);
}
        /* Return the DOS error code if there was one.
return ((regs.x.cflag)

```

```

        / (regs.halt)
        : (0));
}

```

## FLGETDTA 返回磁盘传送地址

```
#include <bfiles.h>
```

```
void far # flgetda(void);
```

(返回) 一个双字长指针的磁盘传送地址。

磁盘传送地址(DTA)是一个数据区的地址, 该数据区被一些DOS功能包括DOS 1.10的文件I/O功能所使用。当一个程序开始执行时, DTA被置为程序段前缀的偏移0x80。一些函数利用它在功能调用之间进行信息交换。

FLGETDTA把磁盘传送地址(DTA)作为函数值返回。

FLPUTDTA指明一个新的DTA地址。

源程序(FLGETDTA.C):

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

```
#include <bfiles.h>
```

```
void far *flgetda ()
```

```

{
    union REGS regs;
    struct SREGS sregs;

        /* Set up and do "return DTA" function call.          */
    regs.x.ax = 0x2f00;
    int86x (FL_DOS_INT, &regs, &sregs, &sregs);
        /* Return far pointer to DTA.                          */
    return uttfaru(sregs.es, regs.x.bx);
}

```

## FLLOCK 对已打开文件的一个文件段上锁或解锁

```
#include<bfiles.h>
```

```
int flock(    int      handle,
              int      option,
              unsigned long offset,
              unsigned long length);
```

handle 已打开文件句柄。

option FL\_LOCK (0) 或FL\_UNLOCK (1)。

offset, length 待上锁区域的起始偏移和长度(以字节计)。

(返回) 返回DOS错误代码。可能值包括:

- 0 成功。
- 1 未知功能: 非法任选项值或DOS版本小于3.10或SHARE未安装。
- 6 不是一个已打开文件的句柄。
- 33 锁操作违例(见下面)。

FLLOCK对与指定的句柄关联的已打开文件的一个文件段上锁或解锁, 防止其它程序(甚至子进程)访问这个被锁的文件段。

用户需要DOS 3.10 或更高版本并且必须安装DOS的SHARE程序。DOS早期版本返回一个值1, 表示功能号未知。参见DOS参考手册有关SHARE的安装说明。

文件的指定区域可以包括整个文件乃至延伸到文件末尾之外。

建议按下述技术使用文件锁：

- 1) 用FLLOCK 尝试锁住一个区域。
- 2) 如果由于区域已被锁住而使得上锁的尝试失败，则继续尝试。如果区域仍被锁住超过10秒钟，显示错误信息。
- 3) 锁住一个区域之后，读或写一些数据。
- 4) 用FLLOCK对该区域解锁。

若要重复进行步骤(2)的上锁尝试，请用FLDOLOCK而不要用FLLOCK。

(上锁时)如果区域的某一部分已被锁住或(开锁时)指定区域未被该程序上锁，则引起锁操作违例。

如果未解除所有的锁，无论如何也不要关闭文件或让程序中止运行(即使是Ctrl-Break或DOS严重错误)。

源程序(FLLOCK.C)：

```
#include <dos.h>
#include <bfiles.h>
#define LOCK_FUNC 0x5c
int flock (handle, option, offset, length)
int          handle, option;
unsigned long offset, length;
{
    union REGS regs;
    if (utdosmajor < 3)                /* DOS version 2.x:          */
        return 1;                    /* This function is unknown. */
    /* (Note:  DOS 3.00 does not support function 0x5c but at least */
    /* it reports a valid error code & extended error information.) */
    regs.x.ax = utbyword (LOCK_FUNC, option);
    regs.x.bx = handle;
    regs.x.cx = (int) uthiword (offset);
    regs.x.dx = (int) utloword (offset);
    regs.x.si = (int) uthiword (length);
    regs.x.di = (int) utloword (length);
    int86 (FL_DOS_INT, &regs, &regs);
    return ((regs.x.cflag)
            ? (regs.h.al)
            : (0));
}
```

**FLNORM**          验证一个文件名，把它转化成标准形式

```
#include<bfiles.h>
```

```
int flnorm(  const char *pfile,
             char      *pnorm,
             int       *plast);
```

pfile          需标准化的路径名。

pnorm          用标准化路径名填充的字符缓冲区的地址，它必须至少为MAX\_FLEN(67)字节长。

plast          整数变量的地址，该变量返回\*pnorm中标准化路径名最后一部分的首字符的位置。

(返回)          0    没有错误；

                1    有错误。

DOS文件系统提供了多种方法对给定的磁盘文件命名。例如，下面所有的文件名都是等效的(当前驱动器是C:，C:上的当前目录是\foo):

```
bar
BAR
C:bar
\foo\bar
c:\foo\bar      (最标准的形式)
c:\foo\BAR
.\bar
.\.\bar
..\foo\bar
```

有时在所有这些可能的形式中知道“真正”的路径名是很重要的。(例如，DOS COPY命令需要知道保时正在把一个文件拷贝到它本身)。FLNORM通过把一个文件转化成上面所注的“最标准”的形式来提供这种服务：把路径名变成小写字母，包括驱动器名和从根目录开始的全路径。另外，FLNORM检查所给的路径名，查看非法字符和语法。

这些操作包括：

- 检查所有非法字符，包括多余的句点('.')、斜杠('/')和反斜杠('\')等；
- 将所有字母转换为小写字母；
- 将所有斜杠转换为反斜杠；
- 删去可能来自字符设备名的尾随冒号(':')；
- 加入当前磁盘驱动器名(如果未指明磁盘驱动器)；
- 如果省缺了从根目录(\)开始的全路径名，则加入之；
- 在这种情况下将"."和".."转换成它们的实际意义；
- 将路径名的每一部分截取成八个字符+点+三个字符；
- 从路径名中没有文件扩展名的每一部分中删去点。

这些操作不包括检查路径及指定文件的字符设备是否存在(如果发现尾随的冒号(':') )。

注意，除非路径名已经是根目录(\)开始的，否则实际结果将(部分地)依赖于有关驱动器上的当前目录。

\*plast中返回的值是标准路径名中最后一个文件名在\*pnorm中的位置。例如，如果标准路径名是c:\blaise\fnorm.c

则\*plast将是10('P'的位置)。如果没有最后一个文件名(也就是说，路径名是某驱动器上的根目录)，则\*plast将是尾随NUL('\0')的位置。例如，如果标准化的路径名是

d:\

则\*plast将是3。

如果路径名包含非法字符或非法语法，或者经标准化后的路径名太长，则1 作为函数值返回，

\*pnorm 和\*plast中的结果无效。

这个函数并不知道磁盘驱动器或目录名是否已被DOS的ASSIGN JOIN 或SUBST命令重新分配。如果这些命令中的某一个生效，则用FLNORM有可能使实际上属于同一个磁盘文件的两个路径名被标准化为不同的结果。

调用函数必须为返回的全路径名至少分配MAX\_FLEN(67)字节。

源程序(FLNORM.C):

```
#include <dos.h>
#include <ctype.h>
#include <stdio.h>          /* For NULL.          */
#include <string.h>
#include <bfiles.h>
```

```

#include <bstrings.h>
#include <butil.h>
#define MAXNAME (MAX_FLEN - 1) /* Maximum length of result */
#define BACKSLASH02
static int curdir(int, char *), /* Internal functions (see */
static int norm(char *), /* below). */
static int dodot(char *);
static int isfc(char);
static int retdrv(void);
int flnorm(pfile, pnorm, plast)
const register char *pfile;
char *pnorm;
int *plast;
{
    int result;
    register int j;
    int drive;
    /* Macro to add trailing NUL to pnorm.
#define ENDNORM {pnorm[j] = '\0';}
    /* Macro to quit immediately (perhaps because of error). */
#define EXIT {ENDNORM;} return result;
    /* Macro to add one character to pnorm. */
#define ADDCHAR(c) {if (j >= MAXNAME) {EXIT;} else pnorm[j++] = (c);}
    /* Macro to step to next character in pfile. */
#define NEXT {pfile++;}
    result = 1; /* In case of premature exit. */
    j = 0;
    *plast = 0;
    if (strlen(pfile) <= 0)
        EXIT
    if (isalpha(*pfile) && *(pfile+1) == ':')
    {
        /* A disk drive was named. */
        drive = tolower(*pfile) - (int) 'a';
        NEXT /* Step past disk drive. */
        NEXT
    }
    else
    {
        drive = retdrv(); /* Infer current drive. */
    }
    ADDCHAR((char) (drive + 'a'))
    ADDCHAR(':')
    if (*pfile == BACKSLASH || *pfile == '/')
    {
        /* Name starts from root. */
        ADDCHAR(BACKSLASH)
        NEXT
    }
}

```

```

else
{
    /* Get start of path from system */
    if ( curdir(drive + 1, pnorm + 2)
        || norm(pnorm + 2))
        EXIT
    *pnorm = (char) tolower(*pnorm);
    j = (int) strlen(pnorm);
    if (pnorm[j-1] != BACKSLASH)
        ADDCHAR(BACKSLASH)
}
while (*pfile != '\0') /* Copy remainder of pfile */
{ /* to pnorm. */
    ADDCHAR(*pfile)
    NEXT
}
ENDNORM
if ( dodot(pnorm + 2) /* Resolve dot and double dot. */
    || norm(pnorm + 2)) /* Remove extra chars, fold to */
                        /* lower case, etc. */
                        /* Quit if error. */
    return 1;
for (*plast = (int) strlen(pnorm);
    *plast > 3 && pnorm[*plast-1] != BACKSLASH;
    (*plast)--);
result = 0; /* Success. */
EXIT
}
#undef EXIT
#undef ADDCHAR
#undef ENDNORM
#undef NEXT
/**
* Name      curdir -- Report current directory on specified drive.
* Synopsis  ercode = curdir(drive, pdir);
*           int  ercode      0 if okay, nonzero if error.
*           int  drive       Disk drive (0 = default, 1 = A:, etc.)
*           char *pdir       Directory path of current directory
* Description This function returns the full path name (starting from
*           the root directory) of the current directory for the
*           specified drive. The returned path name begins with
*           a backslash character ('\\').
*           The calling function must allocate sufficient space for
*           the returned full path name. Because the path can be 64
*           characters (including the trailing null byte), at least
*           (MAX_FLEN-2) bytes should be allocated by the calling
*           function for pdir. (MAX_FLEN is defined in BFILES.H.)
*           The string returned in pdir may lack a trailing NUL

```

```

*          ('\0') if DOS reports an error.
* Returns   ercode          DOS function error code
*          *pdir            The full path name of the current
*                          directory
**/
static int curdir (drive, pdir)
int      drive;
char *pdir;
{
    union REGS regs;
    struct SREGS sregs;
    *pdir++ = '\\';
        /* Set up to query DOS for the current directory on */
        /* a specified drive.                                */
    regs.x.ax = 0x4700;
    regs.h.dl = (unsigned char)drive;
    sregs.ds = utseg (pdir);
    regs.x.si = utoff (pdir);
        /* Do the call.                                     */
    int86x (FL DOS INT, &regs, &regs, &sregs);
        /* If error, return it, otherwise return zero.      */
    return (regs.x.cflag ? regs.x.ax : 0);
}

```

```

/**
* Name      norm -- Check and normalize a path name (convert to
*                standard form)
* Synopsis  ercode = norm(pfile);
*          int  ercode      0 if okay, 1 if error.
*          char *pfile      Path name to normalize
* Description This function validates a path name and partially
*                normalizes it for comparison with other path names.
*                This includes
*                1) Detecting all illegal characters, including
*                   extra periods ('.'), slashes ('/'), backslashes
*                   ('\'), etc.;
*                2) Converting all letters to lower case;
*                3) Converting all slashes to backslashes;
*                4) Removing the trailing colon (':') from a possible
*                   device name;
*                5) Truncating each portion of the path to eight
*                   characters + dot + three characters; and
*                6) Removing the dot from each portion of the path
*                   where it is redundant.
*                This does not include
*                1) Checking the existence of any portion of the path;
*                2) Checking for the presence of a possible device
*                   (if a trailing colon is found);

```

```

*           3) Adding the current disk drive name;
*           4) Adding the full path from the root directory
*              ("\\"); or
*           5) Resolving "." and "..".
*           The path name is assumed NOT to specify a disk drive.
*           If the path name is found to contain illegal characters
*           or syntax, 1 is returned as the value of the function
*           and the results in *pfile are invalid.
*           The result is written back to *pfile without increasing
*           the length of the string.
* Returns      ercode          0 if okay, 1 if error.
*              *pfile          Character buffer filled with the
*                               normalized path name.
**/
static int norm(pfile)
register char *pfile;
{
    int size;
    int result;
    register char *ptarget = pfile;
    /* Macro to quit immediately (perhaps because of error). */
#define EXIT      (*ptarget = '\0'; return result;)
    /* Macro to add one character to ptarget. */
#define ADDCHAR(c) (*ptarget++ = (c);)
    /* Macro to back up one character in ptarget. */
#define BACKCHAR {ptarget--;}
    /* Macro to step to next character in source. */
#define NEXT      (pfile++;)
    result = 1; /* In case of early exit. */
    if (*pfile == BACKSLASH || *pfile == '/')
    {
        ADDCHAR(BACKSLASH)
        NEXT
    }
    while (*pfile != 0)
    {
        size = 0;
        while (
            *pfile != '.' /* Step through filename, */
            && *pfile != BACKSLASH /* saving up to 8 characters. */
            && *pfile != '/'
            && *pfile != ':'
            && *pfile != '\0')
        {
            if (!isfc(*pfile))
                EXIT
            if (size < 8)
            {

```



```

        ADDCHAR((char) tolower(*pfile))
        size++;
    }
    NEXT
}
if (size <= 0)
    EXIT /* No filename was found. */
if (*pfile == '.')
{
    /* There may be an extension. */
    ADDCHAR('.')
    NEXT
    size = 0;
    while ( *pfile != BACKSLASH /* Save first 3 chars of */
            && *pfile != '/' /* extension. */
            && *pfile != ':'
            && *pfile != '\0')
    {
        if (!isfc(*pfile))
            EXIT
        if (size < 3)
        {
            ADDCHAR((char) tolower(*pfile))
            size++;
        }
        NEXT
    }
    if (size <= 0)
        BACKCHAR /* Drop unneeded period (.). */
}
if (*pfile == BACKSLASH || *pfile == '/')
{
    NEXT
    if (*pfile == '\0')
        EXIT /* Error: trailing slash or */
            /* backslash after filename. */
    ADDCHAR(BACKSLASH)
}
else if (*pfile == ':')
{
    NEXT
    if (*pfile == '\0')
        result = 0; /* OK -- this may be a device. */
    EXIT
}
}
result = 0; /* Success. */
EXIT

```

```

}
#undef EXIT
#undef ADDCHAR
#undef BACKCHAR
#undef NEXT
/**
 * Name          dodot -- Resolve dot (".") and double dot ("..")
 *                in a path name.
 * Synopsis      ercode = dodot(pfile);
 *                int  ercode      0 if okay, 1 if error.
 *                char *pfile      Path name to modify
 * Description    This function removes the special file names "." and
 *                ".." from a full path name (which begins from the root
 *                directory of a drive). Each ".." removes the previous
 *                subdirectory from the path; each "." is just removed.
 *                This function does not
 *                1) Detect illegal characters, such as
 *                   extra periods (','), slashes ('/'), backslashes
 *                   ('\'), etc.;
 *                2) Convert letters to lower case;
 *                3) Remove the trailing colon (':') from a possible
 *                   device name;
 *                4) Truncate any portion of the path to eight
 *                   characters + dot + three characters;
 *                5) Remove any redundant dot from a directory name
 *                   in the path;
 *                6) Check the existence of any portion of the path;
 *                7) Check for the presence of a possible device
 *                   (if a trailing colon is found);
 *                8) Add the current disk drive name; or
 *                9) Add the full path from the root directory ("\\").
 *                The path name is assumed NOT to specify a disk drive.
 *                The result is written back to *pfile without increasing
 *                the length of the string.
 *                Slashes are converted to backslashes.
 *                This function detects when too many double dots ("..")
 *                have consumed the entire path or when a dot begins an
 *                illegal filename. If an error occurs then 1 is returned
 *                as the value of the function and the result in *pfile is
 *                invalid.
 * Returns       ercode          0 if okay, 1 if error.
 *                *pfile         Character buffer filled with the
 *                               modified path name.
 */
static int dodot(pfile)
register char *pfile;
{

```

```

int result,start;
register char *pfrom = pfile;
register int    j;
/* Macro to quit immediately (perhaps because of error). */
#define EXIT      {pfile[j] = '\0'; return result;}
/* Macro to add one character to target. */
#define ADDCHAR(c) {pfile[j++] = (c);}
/* Macro to back up one character in target. */
#define BACKCHAR(j--;)
/* Macro to step to next character in source. */
#define NEXT      {pfrom++;}
result = 1; /* In case of early exit. */
j      = 0;
if (*pfrom == BACKSLASH || *pfrom == '/')
{
    ADDCHAR(BACKSLASH)
    NEXT
}
start = j;
while (*pfrom != '\0')
{
    if (*pfrom == '.')
    {
        NEXT
        if (*pfrom == '.')
        {
            /* Double dot: */
            NEXT
            do /* Strip off last directory */
            { /* name. */
                BACKCHAR
            } while (j > 0 && pfile[j-1] != BACKSLASH);
            if (j < start)
                EXIT /* Error: went too far. */
        }
        else
        {
            /* Do nothing: just discard the lone dot. */
        }
        if (*pfrom == BACKSLASH || *pfrom == '/')
        {
            NEXT
        }
        else if (*pfrom == '\0')
        {
            /* Do nothing. */
        }
        else

```

```

EXIT                                /* Only slash or backslash may */
                                    /* follow one or two periods. */
}
else
{
                                    /* Just copy to next slash or */
                                    /* backslash. */
while ( *pfrom != BACKSLASH
        && *pfrom != '/'
        && *pfrom != '\0')
{
    ADDCHAR(*pfrom)
    NEXT
}
if (*pfrom == BACKSLASH || *pfrom == '/')
{
    ADDCHAR(BACKSLASH)
    NEXT
}
}
}
if (j > start && pfile[j-1] == BACKSLASH)
    BACKCHAR                        /* Remove extra trailing */
                                    /* backslash. */

result = 0;
EXIT
}
#undef EXIT
#undef ADDCHAR
#undef BACKCHAR
#undef NEXT
/**
* Name      isfc -- Return whether a character is a valid DOS
*            filename character.
* Synopsis  isokay = isfc(ch);
*            int isokay      1 if okay, 0 if illegal.
*            char ch         Character to check.
* Description This function checks whether a character is a valid DOS
*            filename character and returns 1 if so, 0 if not.
* Returns    isokay          1 if okay, 0 if illegal.
**/
static int isfc(ch)
char ch;
{
    return ( isalpha(ch)
            || isdigit(ch)
            || (-1 != stschind(ch,"$&#@!%\"()_~^~")));
}

```

```

**
* Name      retdrv -- Return default disk drive number.
* Synopsis  drive = retdrv ();
*           int  drive      0 for drive A, 1 for B, etc.
* Description This function returns the current drive.
* Returns    drive          Current drive number.
**/
static int retdrv()
{
    union REGS regs;
    regs.x.ax = 0x1900;
    int86 (FL_DOS_INT, &regs, &regs);
    return (regs.h.al);
}

```

## FLPROMPT 从标准输入中返回一行，可以使用提示信息

```

#include <bfiles.h>
int flpromt(    const char *pprompt,
                char *presp,
                int resp_size);
pprompt        如果标准输入通道是控制台，则显示提示信息。
presp          放置输入行的字符缓冲区的地址。
resp_size      输入缓冲区的尺寸(包括用于换行('\n') (如果有的话)及尾随的NUL('\0')的空间)。
(返回)         0 正常;
                1 文件末尾;
                2 内部错误。

```

FLPROMPT从标准输入设备返回一行正文。如果标准输入是控制台，则FLPROMPT首先在当前控制台上显示提示信息。

如果正文行以换行字符结尾，则换行符与其它正文一齐在缓冲区中返回。

KBQUERY 接受来自标准IBM控制台的用户响应。

源程序(FLPROMPT.C):

```

#include <io.h>
#include <stdio.h>
#include <bfiles.h>
#define OK      0
#define END_OF_FILE1
#define ERROR   2
int flprompt (pprompt, presp, resp_size)
const char *pprompt;
char      *presp;
int       resp_size;
{
    /* Display a prompt if the standard input device is */
    /* a "TTY".                                           */
    if (isatty (fileno (stdin)))

```

```

{
    /* First flush standard output channel.          */
    fflush (stdout);
    /* Put the prompt out to standard error channel. */
    fputs (pprompt, stderr);
    /* Flush standard error channel.                  */
    fflush (stderr);
}

/* Get response from standard input channel.          */
if (fgets (presp, resp_size, stdin) == NULL)
{
    if (feof (stdin))
        return (END_OF_FILE);
    else
        return (ERROR);
}
return (OK);
}

```

## FLPUTDTA 设置磁盘传送地址(DTA)

#include <bfiles.h>

void flputdta(void far \*pdta);

pdta 指向新的磁盘传送地址的far指针。

磁盘传送地址(DTA)是一个数据区的地址，该数据区被一些DOS功能包括DOS 1.10的文件I/O功能所使用。当一个程序执行时，DTA被设置为程序段前缀的偏移0x8。一些函数利用它在功能调用之间交换信息。

FLPUTDTA将磁盘传送地址(DTA)设置在由pdta所指定的位置。

用FLGETDTA可以获取当前的DTA。

源程序(FLPUTDTA.C):

#include <dos.h>

#include <bfiles.h>

void flputdta (pdta)

void far \*pdta;

```

{
    union REGS regs;
    struct SREGS sregs;
    /* Set up registers for "set DTA" function call.          */
    regs.x.ax = 0x1a00;
    sregs.ds = utseg (pdta);
    regs.x.dx = utoff (pdta);
    /* Go set the DTA.                                          */
    int86x (FL_DOS_INT, &regs, &sregs, &sregs);
}

```

## FLREMOVOL 从给定的磁盘上删除卷标(如果有的话)

#include <bfiles.h>

int flremvol(int drive);

drive        磁盘驱动器(0=缺省, 1=A:, 2=B:, 等等。)

(返回)      0    成功

            1    错误。

**FLREMOVOL** 从给定磁盘驱动器上删除卷标(如果有的话)。如果卷标不存在, 并不发生错误。当发现一个卷标而不能删除时, 则发生错误。

源程序(FLREMOVOL.C):

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

```
#include <files.h>
```

```
#define fcb (xpcb + 7)
```

```
int flremvol (drive)
```

```
{
```

```
    union REGS  regs;
```

```
    struct SREGS sregs;
```

```
    unsigned char local_dta[44], xpcb[44];
```

```
    void far      *pold_dta;
```

```
    int           i, result;
```

```
        /* Save pointer to old DTA. */
```

```
    pold_dta = flgetdta ();
```

```
        /* Set new DTA. */
```

```
    flputdta ((void far *) local_dta);
```

```
        /* Zero entire extended FCB. */
```

```
    for (i = 0; i < 44; i++)
```

```
        xpcb[i] = 0;
```

```
        /* Mark this as an Extended FCB. */
```

```
    fcb[-7] = 0xff;
```

```
        /* Select volume attribute. */
```

```
    fcb[-1] = AT_VOLUME;
```

```
    fcb[0]  = (char) drive;
```

```
        /* Search for "???????????" */
```

```
    for (i = 1; i <= 11; i++)
```

```
        fcb[i] = '?';
```

```
        /* Set up for DOS function 0x11: search directory. */
```

```
    regs.x.ax = 0x1100;
```

```
    sregs.ds  = utseg (xpcb);
```

```
    regs.x.dx = utoff (xpcb);
```

```
        /* Do the search. */
```

```
    int86x (FL_DOS_INT, &regs, &regs, &sregs);
```

```
    if (regs.h.al == 0)
```

```
    {
```

```
        /* Found a volume label, so delete it. */
```

```
        regs.x.ax = 0x1300;
```

```
        sregs.ds  = utseg (local_dta);
```

```
        regs.x.dx = utoff (local_dta);
```

```
        int86x (FL_DOS_INT, &regs, &regs, &sregs);
```

```
        result = (regs.h.al != 0);
```

```

    }
    else          /* No volume label found, so quit.          */
        result = 0;
                /* Restore former DTA.                        */
    flputdta (pold_dta);
    return (result);
}

```

## FLRETVOL 报告给定磁盘驱动器上的卷标

```
#include <bfiles.h>
```

```
int flretvol(    int drive,
                char *pvol,
                unsigned *pfdate, unsigned *pftime);
```

drive 磁盘驱动器(0=缺省, 1=A:, 2=B:, 等等。)

pvol 放置卷标名(如果发现)的字符缓冲区的地址, 该缓冲区必须有11字节加上尾随字符NUL(“\0”)的空间。

pfdate, pftime

变量的地址, 该变量返回卷标日期和时间记录。

(返回) 0 发现卷标

1 未发现卷标。

FLRETVOL 返回给定驱动器上的卷标(如果有的话)以及卷标的建立日期和时间。卷标(如果发现)总是以11个字符的大写字符串返回, 它可能包含某些空白或其它通常在DOS文件名中非法的字符。

卷标的日期和时间记录以下面的压缩16位表示:

```

*****位*****
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
y y y y y y y M M M M d d d d =*pfdate
h h h h h m m m m m m s s s s =*pftime

```

这里yyyyyy是年0-119(1980到2099)

MMMM 是月1-12

dddd 是日1-31hhhhh是小时0-23

mmmmmm是分钟0-59

sssss 是以两秒为增量的增量数

这与DOS文件目录所使用的紧缩16位格式相同。

## 源程序(FLRETVOL.C):

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

```
#include <bfiles.h>
```

```
#define fcb      (xfcb      + 7)
```

```
#define found_fcb (local_dta + 7)
```

```
int flretvol (drive, pvol, pfdate, pftime)
```

```
int      drive;
```

```
char     *pvol;
```

```
unsigned *pfdate, *pftime;
```

```
{
```

```
    union REGS  regs;
```

```
    struct SREGS sregs;
```

```
    unsigned char local_dta[44], xfcb[44];
```



```

register int    i;
int            result;
void far       *pold_dta;

        /* Save former Data Transfer Address. */
pold_dta = flgetdta ();
        /* Select local DTA. */
flputdta ((void far *) local_dta);
        /* Clear entire Extended FCB */
for (i = 0; i < 44; i++)
    xfcf[i] = 0;
        /* Mark this as an Extended FCB */
fcf[-7] = 0xff;
        /* Select volume attribute. */
fcf[-1] = AT_VOLUME;
fcf[0] = (char) drive;
        /* Search for "????????.???" */
for (i = 1; i <= 11; i++)
    fcf[i] = '?';
        /* Set up for DOS function 0x11: search for first
        /* matching entry. */
regs.eax = 0x1100;
sregs.ds = utseg (xfcf);
regs.edi = utoff (xfcf);
        /* Go do the search. */
int86x (FL DOS INT, &regs,&regs,&sregs);
if (regs.h.al == 0)
{
    /* Found a volume label -- copy name to *pvol. */
    for (i = 1; i <= 11; i++)
        *pvol++ = found_fcf[i];
    *pvol = '\0';
        /* Copy time and date stamps. */
    *pftime = *((unsigned *) &(found_fcf[23]));
    *pfdate = *((unsigned *) &(found_fcf[25]));
    result = 0;
}
else
{
    /* No volume label found. */
    *pvol = '\0';
    *pfdate = 0;
    *pftime = 0;
    result = 1;
}
        /* Restore former DTA. */
flputdta (pold_dta);
return (result);
}

```

## FLSETVOL 建立或修改给定磁盘上的卷标

#include <bfiles.h>

```
int flsetvol( int drive,
              const char *pvol);
```

drive 磁盘驱动器(0=缺省、1=A:, 2=B:, 等等。)

pvol 卷标名(至多11个字符长)。

(返回) 0 没有错误

1 有错误。

FLSETVOL修改给定磁盘上的卷标(如果已存在)或建立卷标(如果原来不存在)。

如果卷标被建立, 就赋予它当前的时间和日期; 如果只是换名, 卷标的日期和时间记录不变。

卷标可以包含空白或其它通常在DOS文件名中认为是非法的字符。FLSETVOL仅使用\*pvol的前11个字符(最多11), 卷标以大写字母存放。

源程序(FLSETVOL.C):

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

```
#include <string.h>
```

```
#include <bfiles.h>
```

```
#define fcb (xfcb + 7)
```

```
#define found_fcb (local_dta + 7)
```

```
int flsetvol (drive, pvol)
```

```
int drive;
```

```
const char *pvol;
```

```
{
```

```
    union REGS regs;
```

```
    struct SREGS sregs;
```

```
    unsigned char local_dta[44], xfcb[44];
```

```
    register int i;
```

```
    int length;
```

```
    void far *pold_dta;
```

```
        /* Save former Data Transfer Address. */
```

```
    pold_dta = fgetdta ();
```

```
        /* Select local DTA. */
```

```
    fputdta ((void far *) local_dta);
```

```
        /* Clear entire Extended FCB */
```

```
    for (i = 0; i < 44; i++)
```

```
        xfcb[i] = 0;
```

```
        /* Mark this as an Extended FCB */
```

```
    fcb[-7] = 0xff;
```

```
        /* Select volume attribute. */
```

```
    fcb[-1] = AT_VOLUME;
```

```
    fcb[0] = (char) drive;
```

```
        /* Search for "????????.???" */
```

```
    for (i = 1; i <= 11; i++)
```

```
        fcb[i] = '?';
```

```
        /* Set up for DOS function 0x11: search for first */
```

```
        /* matching entry. */
```

```
    regs.x.ax = 0x1100;
```

```

sregs.ds = utseg (xfcb);
regs.x.dx = utoff (xfcb);
/* Go do the search. */
int86x (FL_DOS_INT, &regs,&regs,&sregs);
length = (int) strlen (pvol);
utuplim (length, 11)
if (regs.h.al == 0)
{
/* Found a volume label, so rename it. */
/* Zero parts of FCB. */
for (i = 12; i <= 36; i++)
found_fcb[i] = 0;
/* Copy over new volume name. */
for (i = 0; i < length; i++)
found_fcb[17 + i] = *pvol++;
/* Pad volume name with blanks. */
for (; i < 12; i++)
found_fcb[17 + i] = ' ';
/* DOS function 0x17: rename. */
regs.x.ax = 0x1700;
sregs.ds = utseg (local_dta);
regs.x.dx = utoff (local_dta);
/* Go rename it. */
int86x (FL_DOS_INT, &regs, &regs, &sregs);
}
else
{
/* No volume label found, so create one. */
/* Zero parts of FCB. */
for (i = 12; i <= 36; i++)
fcb[i] = 0;
/* Copy over new volume name. */
for (i = 0; i < length; i++)
fcb[1 + i] = *pvol++;
/* Pad volume name with blanks. */
for (; i < 12; i++)
fcb[1 + i] = ' ';
/* DOS function 0x16: create file. */
regs.x.ax = 0x1600;
/* Go create the volume label. */
int86x(FL_DOS_INT,&regs,&regs,&sregs);
if (regs.h.al == 0)
{
/* Successfully created the label, so close the
/* newly-created file. */
regs.x.ax = 0x1000;
int86x(FL_DOS_INT,&regs,&regs,&sregs);
}
}
/* Restore former DTA. */

```

```

    flputdta (pold_dta);
    return (regs.h.al != 0);
}

```

## 文件操作程序设计示例

NORMFILE.C进行文件名的标准化。

```

/**
 * The purpose of NORMFILE is to show just exactly what is meant
 * by the normalization of a file name.
 * This program displays a prompt on the screen, asking for a
 * filename to normalize. If the user presses just ENTER, he is
 * returned to DOS. If the user types a string, it is normalized
 * into a filename, and then the user is prompted for another
 * filename.
 * The command line format is as follows:
 *      normfile
 */
#include <stdio.h>
#include <files.h>
#define NUL '\0'
static char *descrip[] =
{
    "NORMFILE normalizes file names. Normalization means:", "",
    "  1) Detecting all illegal characters, including extra periods ('.').",
    "    slashes ('/'), backslashes ('\'), etc.;",
    "  2) Converting all letters to lower case;",
    "  3) Converting all slashes to backslashes;",
    "  4) Removing the trailing colon (':') from a possible device name;",
    "  5) Adding the current disk drive name (if no disk drive is named);",
    "  6) Adding the full path from the root directory if it is absent;",
    "  7) Resolving \".\" and \"..\";",
    "  8) Truncating each portion of the path name to eight characters",
    "    + dot + three characters; and",
    "  9) Removing the dot from each portion of the path name where it",
    "    is redundant.", "",
    "This does not include:", "",
    "  1) Checking the existence of any portion of the path, or",
    "  2) Checking for the presence of a possible device (if a trailing",
    "    colon is found).",
    "",
    NIL
};

void main (void)
{
    char filename [80];
    char normname [80];
    char *pdescrip;

```

```

int    i = 0;
int    lastpart;
int    errcode;

        /* Print out description                                */
for (i = 0, pdescrip = descrip[0];
    pdescrip != NIL;
    i++, pdescrip = descrip [i])
    puts (pdescrip);
printf ("press ENTER for some examples: ");
gets (filename);

        /* Show some examples of file normalization.          */
errcode = flnorm ("a:\z", normname, &lastpart);
printf ("    original      = <%s>\n", "a:\z.");
printf ("    normalization = <%s>\n", normname);
printf ("    filename       = <%s>\n", &(normname [lastpart]));
printf ("    errcode        = %d\n\n", errcode);

errcode = flnorm ("..\zap\v.1234", normname, &lastpart);
printf ("    original      = <%s>\n", "..\zap\v.1234");
printf ("    normalization = <%s>\n", normname);
printf ("    filename       = <%s>\n", &(normname [lastpart]));
printf ("    errcode        = %d\n\n", errcode);

printf ("\nfile to normalize -- ENTER to quit?\n\n");

do
{
        /* Get a filename, or ENTER to terminate.            */
printf ("original          = ");
gets (filename);

        /* If we were not asked to terminate, go do the      */
        /* normalization.                                     */
if (filename [0] != NUL)
{
    errcode = flnorm (filename, normname, &lastpart);
    /* Show what FLNORM returned to the user.                */
printf ("    normalization = <%s>\n", normname);
printf ("    filename       = <%s>\n", &(normname [lastpart]));
printf ("    errcode        = %d\n\n", errcode);
}
} while (filename [0] != NUL);
}

```

## 第四章 帮助系统程序设计

### 帮助系统(HL)

Turbo C TOOLS中的帮助系统使产生正文的过程自动化,这些正文将显示在窗口。您可以生成并显示用于用户帮助及其它用途的正文,这些帮助信息可多可少,您完全能够控制在何时何处显示这些正文。用户可以利用键盘或鼠标器浏览正文的各个部分。

帮助信息以压缩的形式存放在磁盘上的一个二进制帮助文件中,因此独立地维护帮助正文和应用程序是很容易的。BUILDHLP工具程序能够从用户写的帮助源文件产生一个二进制帮助文件。

### 帮助函数的功能

HLOPEN	读入二进制帮助文件的一部分,为快速访问该文件建立一个索引。
HLCLOSE	废弃当前二进制帮助文件的索引。
HLLOOKUP	从一个二进制帮助文件读入一段帮助信息,包括帮助信息的全部正文和用于显示信息的视口的描述。
HLREAD	在视口中(如果必要)显示一段帮助信息,使用户能够用键盘或鼠标器浏览这段信息。
HLDISP	将整个过程合为一步:从二进制帮助文件读取帮助信息,显示这段信息,让用户浏览。

### 设计帮助窗口

利用帮助源文件可以指定将出现帮助信息的窗口的尺寸、位置、属性、边界和标题,每段帮助信息可以具有它自己的窗口,也可以几段信息具有同一个窗口。然而,您的程序能够在任何时候取代窗口的任意一个控制项。

如果未指定窗口,系统将使用下面的窗口和视口:

#### 缺省帮助窗口

窗口控制项	值
左上角	行6,列10
右下角	行20,列70
数据区中的行数	(与帮助信息中的行数相匹配)
数据区中的列数	61
数据区属性	蓝绿背景、增白前景
边界类型	单线
边界属性	蓝绿背景、增白前景
标题	(无)
标题属性	蓝色背景、增白前景

上表中,行列值均相对于显示器的左上角(1,1),“角”指的是视口数据区而不是边界。

### 编写帮助源文件

帮助源文件是一个一般的ASCII正文文件,我们建议给它一个文件扩展名.txt。每一行必须以回车和换行结束,也可以任选地用Ctrl-Z字符(ASCII 0x1a)结束一行。

在帮助源文件中允许有tab字符,但它被作为显示字符(象一个圆圈)而不作为控制字符看待。在帮助源文件中最好不要用tab。

在源文件中可以出现三种正文行:

- 注释 以一个点和星号(\*)开头的行。BUILDHLP忽略并废弃这一行的其它正文。
- 点命令 以一个点(.)和一个命令开头的行。点必须在第一列出现,其后跟随一个字,其中没有空白。如果点之后的字不是一个认可的命令字,BUILDHLP将放弃处理,输出一个错误信息。检查命令时不关心字母的大小写。

源行 第一列不以点开头的行被认为是帮助信息的正文。

点命令有两种：标识命令决定以后源行的上下文，窗口命令指明用于显示帮助正文的窗口和视口的特性。

#### BUILDHELP标识点命令

命令	意义
id	定义一个标识帮助信息的字符串
describe	保留用于以后使用

#### BUILDHELP窗口点命令

命令	意义
xupper	视口数据区的最左列
yupper	视口数据区的顶行
xlower	视口数据区的最右列
ylower	视口数据区的底行
datawidth	窗口数据区的列数
forecolor	窗口数据区的前景属性
backcolor	窗口数据区的背景属性
bordertype	视口边界类型
borderfore	视口边界的前景属性
borderback	视口边界的背景属性
title	帮助视口标题
titlefore	视口标题的前景属性
titleback	视口标题的背景属性

列值和行值指的是相对于显示屏幕左上角的极端位置，它们不包括边界。

每一个点命令在同一行上必须跟随有其它信息：

.title命令之后必须跟随有标题的正文。您可以把标题正文用省略号(?)或双引号(")括起来。如果省略号或双引号仅出现在标题的开头或末尾，则它将被剔除。用空字符串("")可以消除标题。

.id命令之后必须跟随一个一到十二个字符长的标识字符串。字符大小写、前导或尾随的空白(空格和tab)无关紧要，但嵌入的空白具有一定的意义。例如，下面这些标识字符串被认为是相同的：

```
"MYIDENT"  
"myident"  
"MyIdent"  
" MyIdent "
```

但"My Ident"与它们都不相同，因为它有嵌入的空白。然而，

```
"My 2nd Id"  
"My 2nd Id"
```

被认为是相同的，因为每个嵌入的空白被看作一个空格。

颜色命令(.forecolor、.backcolor、.borderfore、.borderback、.titlefore和.titleback)之后必须跟随有下列颜色命令字之一：

颜色字	四位属性
黑色	0
蓝色	1
绿色	2
蓝绿	3
红色	4
紫红	5
棕色	6
淡灰	7
暗灰	8
淡蓝	9

淡绿	10
淡青	11
淡红	12
淡紫	13
黄色	14
白色	15

另一个命令必须跟随一个0到255之间的数，这个数可以用十进制或十六进制表达。要想表达为十六进制，需把一个美元符号(\$)作为该数字的前缀。

用一个或多个窗口点命令可以改变显示帮助信息的窗口或视口。窗口命令对于其后的帮助信息保持有效，直至被另一个同类命令取代。

用帮助源文件开始定义一段帮助信息时，首先通过`id`命令标识这个信息，然后在下面的连续几行中给出整个帮助信息的正文，中间不能插有点命令。(对于这个版本的BUILDHELP，您可以在上面未列出的正文中嵌入点命令，然而，这将引发警告信息。)在帮助信息中允许有注释。

帮助信息的源正文在窗口中的显示形式与在帮助源文件中的出现形式相同，但帮助信息的每一行在必要时被截断以合于窗口的数据区。如果帮助正文的尺寸比视口大，用户可以按箭头键来查看信息的各个部分。一条信息的行数有一个限制：窗口的行数乘以列数不能超过32,767。在一个帮助源文件中的帮助信息数目没有限制。

帮助正文的显示属性可以被改变。按照缺省，帮助信息在显示时使用窗口数据区的缺省属性。要改变属性，可以嵌入A(补字符加上大写的A)后随两个定义新属性的十六进制字符。先定义背景属性，然后定义前景属性。例如，A07是通常的黑色背景、浅灰前景，A>7是反显状态，而A1F则为蓝色背景、增白前景。新属性保持有效，直至到达帮助信息的末尾或属性被重新定义。

要想在帮助信息中嵌入特殊的IBM ASCII字符，可以嵌入C(补字符加上大写的C)后随两个十六进制字符，这两个字符定义了特殊字符的ASCII值。例如，CEO指明小写的希腊字母alpha。

## 控制内存分配

如果想建立一个中止并驻留的程序，请阅读“窗口函数(WN)”中相应的控制内存分配一节。

由于帮助函数利用了Turbo C TOOLS窗口，所以，对于帮助系统来说也需要象窗口那样注意内存分配这个问题。帮助函数也将标准`malloc()`函数用于帮助文件的索引和帮助信息的正文。要想不实际显示信息而执行所有这些内存分配操作，可以参照下面的方法：

用`HLOPEN`对二叉帮助文件建立一个索引。(应该只使用一个二叉帮助文件。`HLOPEN`只能保持一个索引，所以使用帮助文件的数目多于一个时意味着只要使用一个新的帮助文件就要分配一个新的索引。)

建立一个窗口来容纳帮助信息。用户可以调用`HLLOOKUP`来取得帮助信息窗口的特性。

对该窗口执行下面这个语句

```
wnsetopt(pwin, WN_PREV_ALLOC, 1);
```

用`WNINITEV`和`WNSCRLBR`为`HLREAD`内部使用的`WNREAD`设立窗口，通过`HLLOOKUP`取得帮助信息的正文。

执行完上述步骤之后就可以显示帮助信息，让用户浏览而不必担心其它内存分配问题。作为一个选择，您可以用下面的方法节省更多的空间和时间：

用`HLCLOSE`释放内部帮助索引。

通过`WNWRITE`或其它窗口输出函数在窗口中显示帮助信息。

释放`HLLOOKUP`用于返回帮助信息正文的缓冲区。

用`WNREAD`来代替`HLREAD`，因为窗口已经建立，帮助信息的正文已经从帮助文件中读出。

实际上，您可能希望通过`ISRESEXT`使程序中中止之后再利用`HLLOOKUP`取得帮助信息的正文。为此，请参照“中断服务支持(IS)”一章中“在硬件ISR中使用`malloc()`”一节。

## 帮助函数源程序

为了方便研习，给出帮助扩展函数源程序。



## HLCLOSE      释放二叉帮助文件的已存索引

```
#include <bhelp.>
int hlclose(void);
(返回)          错误代码。
HLCLOSE          释放由HLOPEN建立的索引, 把全局变量b_phelp_index置为NIL。
```

源程序(HLCLOSE.C):

```
#include <bhelp.h>
int hlclose()
{
    if (b_phelp_index == NIL)
        hlerror(HL_NO_INDEX);
    hlfrindx(b_phelp_index);
    b_help_path[0] = '\0';
    return(HL_NO_ERROR);
}
```

## HLDISP          从帮助文件读取一段帮助信息, 显示在屏幕上, 供用户浏览

```
#include <bhelp.h>
WN_EVENT *hldisp( const char          *ppath,
                  const char          *pid_string,
                  unsigned long        offset,
                  WN_EVENT            *pfinal,
                  int                  option);
```

ppath          待打开的二叉帮助文件的文件名, 以NUL('\0')结尾。可以任选地包含驱动器名和全路径名。

pid\_string      读取信息所用标识字符串的地址(见下面)。如果要使用偏移量, 可以指明NIL或空串("")。

offset          如果pid\_string为NIL或指向空字符串(""), 这就是BUILDHLP所报告的二叉帮助文件中帮助信息的位置。

pfinal          结构的地址, 该结构接受用户触发的最后一个事件的拷贝。 如果不需要报告, 则为NIL。见WNREAD中有关WN\_EVENT结构的说明。

option          下列值的任意组合:

HL_KBIGNORE (4)	忽略并废弃所有的按键。
HL_USE_MOUSE (0)	如果有鼠标器, 则使用它。
HL_NO_MOUSE (8)	不使用鼠标器。

(返回)          pfinal的拷贝。如果失败, 返回NIL。

HLDISP从二叉帮助文件中读取一段帮助信息, 在视口中显示它, 让用户利用键盘或鼠标器进行浏览。HLDISP使用在二叉帮助文件中指定的窗口和视口。

源程序(HLDISP.C):

```
#include <bhelp.h>
WN_EVENT *hldisp(pdatabase_path, pid_string, offset, pfinal,
                 option)
const char      *pdatabase_path;
const char      *pid_string;
unsigned long    offset;
WN_EVENT        *pfinal;
```

```

int          option;
{
    HL_WINDOW *help_window;
    char      *phelp_text;
    int       text_length;
    WN_EVENT *presult;
    int       error;

    error = hllookup(pdatabase_path, pid_string, offset, &help_window,
                    &phelp_text, &text_length, HL_COMPRESSED);
    if (error != HL_NO_ERROR)
        return(NIL);
    presult = hlread(NIL, &help_window, phelp_text, text_length,
                    pfinal, HL_COMPRESSED | (option & ~HL_TEXT_TYPE));
    free(phelp_text);
    return(presult);
}

```

**HLFRINDX**    释放在给定节点以下的所有帮助索引节点。

```

#include <bhelp.h>
#include <stdlib.h>
hlfrindx(pindex_node);
pindex_node    指向要释放的帮助索引子树的指针。
               函数接受一个指向帮助索引节点的指针，并释放在该节点以下(包括该节点)的子树。
返回          无
#include <bhelp.h>
#include <stdlib.h>

void hlfrindx(pindex_node)
HL_INDEX_NODE *pindex_node;
{
    if (pindex_node == NIL)
        return;

    if (pindex_node->pleft != NIL)
    {
        hlfrindx(pindex_node->pleft);
        pindex_node->pleft = NIL;
    }

    if (pindex_node->pright != NIL)
    {
        hlfrindx(pindex_node->pright);
        pindex_node->pright = NIL;
    }

    free(pindex_node);
}

```

## HLLOOKUP 从二叉帮助文件中读取一段帮助信息

```
#include <bhelp.h>
```

```
int hllookup(    const char    *ppath,
                const char    *pid_string,
                unsigned long   offset,
                HL_WINDOW      *phelp_win,
                char            **pptext,
                int             *plength,
                int             option);
```

**ppath** 等打开的二叉帮助文件的文件名, 以NUL('\0')结尾。可以任选地包含驱动器名和全路径名。

**pid\_string** 帮助信息的标识字符串的地址(见下面)。如果要使用偏移值, 可以指明NIL或空字符串("")。

**offset** 如果pid\_string为NIL或指向一个空字符串(""), 这就是BUILDHLP所报告的二叉帮助文件中帮助信息的位置。

**phelp\_win** 结构的地址, 该结构中存放有帮助信息的窗口和视口的描述。

**pptext** 字符指针的地址, 该指针返回帮助信息正文的地址。

**plength** 变量的地址, 该变量返回包含帮助信息正文的缓冲区的长度。

**option** 帮助信息正文的存放格式:

HL_CHARS_ONLY	(0)	返回的正文已被展开但属性信息被废弃。
HL_CHAR_ATTR	(2)	返回的正文和属性信息均被展开。
HL COMPRESSED	(64)	返回的正文保持UTUNSQZ所支持的压缩格式。

(返回) 错误代码。

**HLLOOKUP** 从二叉帮助文件中读取帮助信息, 包括帮助信息的全部正文和它的窗口及视口的描述。

帮助正文的窗口描述在HL\_WINDOW 结构中返回。见HLREAD中有关该结构的说明。

返回的正文保存在用标准malloc() 函数分配的缓冲区中, 缓冲区的地址由\*pptext返回。

必要时HLLOOKUP调用HLOPEN为二叉帮助文件建立一个索引。

源程序(HLLOOKUP.C):

```
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <errno.h>
#include <bhelp.h>
#include <bstrings.h>
#include <bfiles.h>
#define close_error(error) {close(help_fd); hlerror(error);}
static HL_INDEX_NODE *search_index(HL_INDEX_NODE *, char *);
int hllookup(pdatabase_path, pid_string, offset, phelp_win, pptext,
            ptext_length, option)
const char    *pdatabase_path;
const char    *pid_string;
unsigned long   offset;
HL_WINDOW      *phelp_win;
char            **pptext;
int            *ptext_length;
int            option;
```

```

{
    int            error;
    HL_INDEX_NODE *index_node;
    HL_RECORD_DATA record_data;
    extern int     errno;
    int            help_fd;
    int            buffer_index;
    const char     *ppath;
    char           *puncompressed_text;
    char           *pchars_only_text;
    int            uncompressed_size;
    int            chars_only_size;
    char           uppercase_id[HL_ID_SIZE];
    char           normalized_path[MAX_FLEN];
    int            name_index;
    unsigned long  search_offset;
    int            id_specified = 0;
    errno = 0;
    if (fnorm(pdatabase_path, normalized_path, &name_index) != 0)
        ppath = pdatabase_path;
    else
        ppath = normalized_path;
        /* First check to see if we need to build the index.*/
    if (strcmp(ppath, b_help_path) != 0)
    {
        hlclose();
        error = hlopen(ppath);
        if (error != HL_NO_ERROR)
            return(error);
    }
        /* Now search the index if necessary. */
    if ((pid_string != NIL) && (*pid_string != '\0'))
    {
        id_specified = 1;
        strncpy(uppercase_id, pid_string, HL_ID_SIZE - 1);
        uppercase_id[HL_ID_SIZE - 1] = '\0';
        stpcvt(uppercase_id, RLWHITE | RTWHITE | TOUP);
        pindex_node = search_index(b_phelp_index, uppercase_id);
        if (pindex_node == NIL)
            hlerror(HL_NOT_FOUND);
        search_offset = pindex_node->index_data.help_record;
    }
    else
        search_offset = offset;
        /* The index has been found, so open the file and
        /* read the data.
    help_fd = open(ppath, O_RDONLY | O_BINARY);
    if (help_fd == -1)
        switch (errno)
        {

```

```

    case EACCES:
        hlerror(HL_BAD_DATABASE);
    case EMFILE:
        hlerror(HL_NO_HANDLES);
    case ENOENT:
        hlerror(HL_NO_PATH);
    default:
        hlerror(HL_SYSTEM);
}
if (lseek(help_fd, search_offset, SEEK_SET) != search_offset)
    close_error(HL_BAD_DATABASE);
    /* First, read the record header and window data. */
if ((read(help_fd, &record_data, sizeof(HL_RECORD_DATA)) !=
    sizeof(HL_RECORD_DATA)) ||
    (record_data.record_sign.type != HL_WINDOW_RECORD) ||
    (record_data.record_sign.length != sizeof(HL_RECORD_DATA)))
{
    close_error(HL_BAD_DATABASE);
}

hlpas2c(record_data.id_string, record_data.id_string,
    sizeof(record_data.id_string));

if (id_specified &&
    (strcmp(record_data.id_string, uppercase_id) != 0))
{
    close_error(HL_BAD_DATABASE);
}

    /* Convert the help record from disk format to */
    /* memory format. */
*phelp_win = *((HL_WINDOW *) &record_data.view_top);
phelp_win->view_top--;
phelp_win->view_left--;
phelp_win->view_bottom--;
phelp_win->view_right--;
phelp_win->origin_row--;
phelp_win->origin_col--;
    /* Now read in the help window's text. */
*pptext = malloc(record_data.image_length);
if (*pptext == NIL)
    close_error(HL_NO_MEMORY);
if (read(help_fd, *pptext, record_data.image_length) !=
    record_data.image_length)
{
    close_error(HL_BAD_DATABASE);
}

    /* Now check to see if the window has a title. */
if (record_data.title_length != 0)
{

```

```

    phelp_win->pwindow_title = malloc(record_data.title_length + 1);
    if (phelp_win->pwindow_title == NIL)
        close_error(HL_NO_MEMORY);
    if (read(help_fd, phelp_win->pwindow_title,
            record_data.title_length) != record_data.title_length)
        close_error(HL_BAD_DATABASE);
    phelp_win->pwindow_title[record_data.title_length] = '\0';
}
else
    phelp_win->pwindow_title = NIL;

    /* Now check to see how the text should be returned.*/
switch(option)
{
case HL_COMPRESSED:
    *ptext_length = record_data.image_length;
    break;

case HL_CHAR_ATTR:
case HL_CHARS_ONLY:
    uncompressed_size = *((int *) ((*pptext) + 2));
    puncompressed_text = malloc(uncompressed_size);
    if (puncompressed_text == NIL)
        close_error(HL_NO_MEMORY);
    if (utunsqz(*pptext, puncompressed_text,
            record_data.image_length, uncompressed_size) !=
        uncompressed_size)
    {
        free(puncompressed_text);
        close_error(HL_BAD_DATABASE);
    }

    if (option == HL_CHAR_ATTR)
    {
        free(*pptext);
        *pptext = puncompressed_text;
        *ptext_length = uncompressed_size;
    }
    else
    {
        chars_only_size = uncompressed_size / 2;
        pchars_only_text = malloc(chars_only_size);
        if (pchars_only_text == NIL)
        {
            free(puncompressed_text);
            close_error(HL_NO_MEMORY);
        }
        for (buffer_index = 0; buffer_index < chars_only_size;
            buffer_index++)
        {

```

```

        {chars_only_text[buffer_index] =
          puncompressed_text[buffer_index * 2];
        }
        free(*pptext);
        *pptext = pchars_only_text;
        free(puncompressed_text);
        *ptext_length = chars_only_size;
    }
    break;
default:
    return(HL_SYSTEM);
}
return(HL_NO_ERROR);
}

```

```

/**
 * Name          SEARCH INDEX -- Recursively search a help index tree for
 *                a specified ID string.
 * Synopsis      pindex_node = search_index(pindex_root, pid_string);
 *                HL_INDEX_NODE *pindex_node Pointer to found index node,
 *                or NIL if ID is not in the
 *                index.
 *                HL_INDEX_NODE *pindex_root Pointer to root of subtree
 *                to search.
 *                char *pid_string          The help ID string to search
 *                for.
 * Description    This function will recursively search the specified
 *                help index subtree for the given id string.      The return
 *                value is a pointer to the help index node for the ID,
 *                or NIL if the ID cannot be found in the subtree.
 * Returns       pindex_node      Found index node or NIL if ID does not
 *                                exist.
 **/

```

```

HL_INDEX_NODE *search_index(pindex_root, pid_string)
HL_INDEX_NODE *pindex_root;
char *pid_string;
{
    int result;
    if (pindex_root == NIL)
        return(NIL);
    result = strcmp(pid_string, pindex_root->index_data.id_string);
    if (result == 0)
        return(pindex_root);
    if (result < 0)
        return(search_index(pindex_root->pleft, pid_string));
    return(search_index(pindex_root->pright, pid_string));
}

```

**HLOPEN** 对一个二叉帮助文件建立索引

#include <bhelp.h>

int hlopen (const char \*ppath);

ppath 待打开的帮助文件的文件名, 以NUL('\0') 结尾。 可以任选地包含驱动器名和全路径名。用NIL或空字符串("")指明最近打开的二叉帮助文件。

(返回) 错误代码。

HLOPEN建立一个内存索引以便快速访问二叉帮助文件。它暂时打开该文件, 然后又关闭它, 并不修改任何磁盘文件。

如果已经为另一个二叉帮助文件建立了索引, 则HLOPEN在开始执行之前废弃前一个索引。

如果成功, HLOPEN在全局变量b\_help\_path中记录下数据库文件的名字, 在b\_phelp\_index中记录下索引的初始结点。 两个全局变量都在bhelp.h中声明。

源程序(FLOPEN.C):

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <io.h>

#include <errno.h>

#include <bhelp.h>

#include <bfiles.h>

HL\_INDEX\_NODE \*b\_phelp\_index = NIL;

char b\_help\_path[MAX\_FLEN] = {'\0'};

#define close\_error(error)

```
{
    close(help_fd);
    b_help_path[0] = '\0';
    hlfrindx(b_phelp_index);
    hlerror(error);
    return(b_wnerr);
}
```

static void cdecl insert\_index(HL\_INDEX\_NODE \*\*, HL\_INDEX\_NODE\*);

int hlopen(pdatabase\_path)

const char \*pdatabase\_path;

```
{
    int help_fd;
    int name_index = 0;
    int bytes_read;
    unsigned long file_size;
    HL_FILE_HEADER file_header;
    HL_INDEX_NODE *new_node;
```

errno = 0;

```
if ((pdatabase_path == NIL) || (*pdatabase_path == '\0'))
    pdatabase_path = b_help_path;
```



```

        /* Try to open and read the help database. */
help_fd = open(pdatabase_path, O_RDONLY | O_BINARY);
if (help_fd == -1)
{
    switch (errno)
    {
        case EACCES:
            hlerror(HL_BAD_DATABASE);
        case EMFILE:
            hlerror(HL_NO_HANDLES);
        case ENOENT:
            hlerror(HL_NO_PATH);
        default:
            hlerror(HL_SYSTEM);
    }
    return(b_werrr);
}

        /* Now that the database has been opened, copy the */
        /* pathname into b_help_path. */
if (pdatabase_path != b_help_path)
    if (fnorm(pdatabase_path, b_help_path, &name_index) != 0)
    {
        strncpy(b_help_path, pdatabase_path, sizeof(b_help_path));
        b_help_path[sizeof(b_help_path) - 1] = '\0';
    }
if (b_phelp_index != NIL)
{
    hlfrindx(b_phelp_index);
    b_phelp_index = NIL;
}
file_size = lseek(help_fd, 0, SEEK_END);
lseek(help_fd, 0, SEEK_SET);
if ((read(help_fd, &file_header,
        sizeof(file_header)) != sizeof(file_header)) ||
    (file_size < file_header.file_size) ||
    (file_header.index_offset > file_size) ||
    ((file_header.index_offset != 0) &&
     (file_header.index_offset < sizeof(file_header))) ||
    (strcmp(file_header.file_sign, HL_FILE_SIGN,
        strlen(HL_FILE_SIGN)) != 0))
{
    close_error(HL_BAD_DATABASE);
}

if (file_header.version != HL_VERSION)
{
    close_error(HL_VER_MISMATCH);
}

if (lseek(help_fd, file_header.index_offset, SEEK_SET) == 0)

```

```

{
    close(help_fd);
    return(HL_NO_ERROR);
}

/* Now begin reading the database index. */
do
{
    pnew_node = calloc(sizeof(HL_INDEX_NODE), 1);
    if (pnew_node == NIL)
    {
        close_error(HL_NO_MEMORY);
    }
    bytes_read = read(help_fd, &(pnew_node->index_data),
                      sizeof(pnew_node->index_data));
    if (pnew_node->index_data.description_len != 0)
    {
        pnew_node->pdescription =
            malloc(pnew_node->index_data.description_len + 1);
        if (pnew_node->pdescription == NIL)
        {
            close_error(HL_NO_MEMORY);
        }
        bytes_read += read(help_fd, pnew_node->pdescription,
                          pnew_node->index_data.description_len);
    }
    else
        pnew_node->pdescription = NIL;
    if (bytes_read != 0)
    {
        if (bytes_read != (sizeof(pnew_node->index_data) +
                          pnew_node->index_data.description_len))
        {
            close_error(HL_BAD_DATABASE);
        }
        else
        {
            hlpas2c(pnew_node->index_data.id_string,
                   pnew_node->index_data.id_string,
                   sizeof(pnew_node->index_data.id_string));

            pnew_node->pdescription[
                pnew_node->index_data.description_len] = '\0';
            insert_index(&b_phelp_index, pnew_node);
        }
        lseek(help_fd, pnew_node->index_data.next_index,
              SEEK_SET);
    }
}
while ((pnew_node->index_data.next_index != 0) &&

```

```

        (bytes read != 0));
close(help_fd);
return(HL_NO_ERROR);
}

/**
 * Name          INSERT_INDEX Insert a help index node into the given
 *                help index subtree.
 * Synopsis      insert_index(ppindex_node, pnew_node);
 *                HL_INDEX_NODE *ppindex_node Pointer to pointer to
 *                subtree into which the new
 *                node is to be inserted.
 *                HL_INDEX_NODE *pnew_node Pointer to the new help
 *                index node to insert.
 * Description    This function accepts a pointer to a pointer to a help
 *                index subtree and a pointer to a new help index node
 *                and inserts the given node into the subtree.      If
 *                *ppindex node is NIL, *ppindex node is set to pnew_node.
 * Returns       *ppindex_node If NIL, it is set to pnew_node.
 */

```

```

void insert_index(ppindex_node, pnew_node)
HL_INDEX_NODE **ppindex_node;
HL_INDEX_NODE *pnew_node;
{
    if (*ppindex_node == NIL)
    {
        *ppindex_node = pnew_node;
        return;
    }

    if (strcmp(pnew_node->index_data.id_string,
               (*ppindex_node)->index_data.id_string) <= 0)
        insert_index(&((*ppindex_node)->pleft), pnew_node);
    else
        insert_index(&((*ppindex_node)->pright), pnew_node);

    return;
}

```

**HLPAS2C** 转换一个Pascal的字符串为C的格式。

```

#include <hhelp.h>
#include <string.h>
length = hlpas2c( ppascal_string, pc_string,
                  max_length);
length          包括终止'\0'的字符串长度。
ppascal_string  要转换的Pascal字符串。
pc_string       存放C字符串的缓冲区。
max_length      包括'\0'的C字符串最大长度。

```

HLPAS2C转换前导长度字节的Pascal字符串为以'\0'为尾的C字符串。最大长度不能超过max\_length。

返回                      length        包括尾符'\0'的C字符串长度。

源程序(HLPAS2C.C):

```
#include <bhelp.h>
#include <string.h>
int hlpas2c(ppascal_string, pc_string, max_length)
const char *ppascal_string;
char *pc_string;
int max_length;
{
    int c_length;
    int temp_index;
    c_length = (int) ppascal_string[0] + 1;
    c_length = (max_length > c_length) ? c_length
                                         : max_length;

    ppascal_string++;
    for (temp_index = 0; temp_index < (c_length - 1); temp_index++)
        pc_string[temp_index] = ppascal_string[temp_index];
    pc_string[c_length - 1] = '\0';
    return(c_length);
}
```

**HLREAD**                      在视口中显示帮助信息，供用户浏览

```
#include <bhelp.h>
WN_EVENT *hlread( BWINDOW                      *pwin,
                  const HL_WINDOW              *phelp_win,
                  const char                    *ptext,
                  int                           length,
                  WN_EVENT                      *pfinal,
                  int                           option);
```

**pwin**                      用于显示帮助信息的原已存在的窗口。若为NIL，则暂时建立一个窗口，事后取消该窗口。

**phelp\_win**                结构的地址，该结构说明帮助信息的窗口和视口。若为NIL，则使用缺省窗口。

**ptext**                    帮助信息正文的地址，它的格式由option描述。

**length**                   信息正文所占用的内存字节数。

**pfinal**                   结构的地址，该结构接受用户触发的最后一个事件的拷贝。如果不需要报告，则为NIL。

见WNREAD中有关WN\_EVENT结构的说明。

**option**                   下列位值的任意组合：

符号	值	意义
HL_CHARS_ONLY	0x00	正文已展开，仅包含字符。
HL_CHAR_ATTR	0x02	正文和属性信息均已展开。
HL_COMPRESSED	0x40	正文以压缩的形式提供。
HL_KEIGNORE	0x40	忽略并废弃所有的按键。
HL_USE_MOUSE	0x00	如果有一个鼠标器，则使用它。
HL_NO_MOUSE	0x08	不使用鼠标器。
HL_REMOVE_WIN	0x10	即使窗口以前存在，事后取消之。
HL_DESTROY_WIN	0x30	即使窗口以前存在，事后取消并废弃之。

(返回)                    pfinal的拷贝。如果失败，返回NIL。

HLREAD在视口中显示帮助信息，让用户使用键盘或鼠标器浏览这些信息。必要时它还暂时建立一

个窗口来放置这些信息。

正文可以按压缩或非压缩的形式传递。如果是压缩的，格式与UTSQZSCN所接受的格式相同；如果是非压缩的，则正文的传递可以带有也可以不带有属性信息。

用户可以通过HL\_WINDOW结构指明帮助窗口，这个结构在bhelp.h中定义如下：

```
typedef struct /*HL_WINDOW: */
{
    int view_top;           /*视口上沿。 */
    int view_left;          /*视口左沿。 */
    int view_bottom;        /*视口下沿。 */
    int view_right;         /*视口右沿。 */
    int origin_row;         /*最初显示在视口原点的数据区坐标。*/
    int origin_col;
    int data_fore;          /*数据区前景属性。 */
    int data_back;          /*数据区背景属性。 */
    int border_type;        /*视口边界类型。 */
    int border_fore;        /*边界前景属性。 */
    int border_back;        /*边界背景属性。 */
    int title_type;         /*窗口标题。 */
    int title_fore;         /*标题前景属性。 */
    int title_back;         /*标题背景属性。 */
    int data_rows;          /*图象的行数。 */
    int data_columns;       /*图象的列数。 */
    int xref_fore;          /*保留用于以后使用。 */
    int xref_back;          /*保留用于以后使用。 */
    int highlight_fore;     /*用于亮条的前景/背景属性*/
    int highlight_back;     /*用于亮条。 */
    char *pwindow_title;    /*窗口标题的正文。 */
} HL_WINDOW;
```

下面是HL\_WINDOW 结构中一些成员的说明：

view\_top、view\_left、view\_bottom和view\_right成员的值均相对于显示左上角的0行、0列。

origin\_row和origin\_col是相对于窗口数据区左上角(0, 0)的值。

如果data\_columns比帮助源文件中的.datawidth值小，则帮助信息将被部分隐去。

border\_type包含下列值：

值	意义
0-15	SCBOX使用的边界类型值。
255	没有边界。
其它	整个边界所用的字符。

title\_type包含下列值：

值	意义
0	标题在上方中部
1	标题在上方右部
2	标题在上主左部
3	标题在下方中部
4	标题在下方右部
5	标题在下方左部

源程序(HLREAD.C):

```
#include <bhelp.h>
#include <bwindow.h>
#include <butil.h>
```

```

HL_WINDOW b_def_help_win =      /* Default help record window.  */
{
    5, 9,                        /* Viewport top left corner.  */
    19, 69,                      /* Viewport bottom right corner.*/
    0, 0,                        /* Data area origin coordinates.*/
    NORMAL | INTENSITY, SC_CYAN/* Window foreground/background.*/
    0,                            /* Default border (BBRD_SSSS). */
    NORMAL | INTENSITY, SC_CYAN/* Border foreground/background.*/
    BBRD_NO_TITLE,               /* Title position.            */
    NORMAL | INTENSITY, SC_BLUE/* Title foreground/background. */
    15,                          /* Number of rows in data area.*/
    61,                          /* # of columns in data area.  */
    SC_BLUE, SC_CYAN,            /* Cross reference attributes. */
    SC_BLUE | INTENSITY, NORMAL/* Highlighted cross reference. */
    {'\0'})                      /* # of columns in data area.  */
};

const static int title_table[] = /* Table of title positions.  */
{
    BBRD_TCT,
    BBRD_TRT,
    BBRD_TLT,
    BBRD_BCT,
    BBRD_BRT,
    BBRD_BLT
};

WN_EVENT *hread(pwin, phelp_win, ptext, text_length, pfinal,
                option)
BWINDOW      *pwin;
const HL_WINDOW*phelp_win;
const char    *ptext;
int           text_length;
WN_EVENT      *pfinal;
int           option;
{
    int         save_error;
    int         made_window = 0;
    char        *pwindow_data;
    int         dest_length;
    int         mode, columns;
    BORDER      border;
    WHERE       where;
    const HL_WINDOW *phelp_window;
    if (phelp_win == NIL)
        phelp_window = &b_def_help_win;
    else
        phelp_window = phelp_win;
    if (pwin == NIL)
    {
        pwin = wncreate(phelp_window->data_rows,
                        phelp_window->data_columns,

```

```

        utnybbyt(phelp_window->data_back,
                phelp_window->data_fore));
if (pwin == NIL)
    return(NIL);
if (wnsetopt(pwin, WN_CUR OFF, 1) == NIL)
{
    save_error = b_wnerr;
    wndstroy(pwin);
    hlreterr(save_error);
}
made_window = 1;
}

/* If the window is not displayed, set up a WHERE */
/* and a BORDER for it based on the default help */
/* window. Otherwise, just make a copy of the */
/* values stored in the window. */
if ((pwin->where_shown.dev != SC_COLOR) &&
    (pwin->where_shown.dev != SC_MONO))
{
    where.dev = scmode(&mode, &columns, &where.page);
    where.corner.row = phelp_window->view_top;
    where.corner.col = phelp_window->view_left;
    if (!utrange(phelp_window->border_type, 0, 15))
        border.type = phelp_window->border_type + 1;
    else
        if (phelp_window->border_type == 255)
            border.type = BBRD_NO_BORDER;
        else
        {
            border.type = 31;
            border.ch = phelp_window->border_type;
        }
    border.attr = utnybbyt(phelp_window->border_back,
                          phelp_window->border_fore);
    if (utrange(phelp_window->title_type, 0,
                (sizeof(title_table) / sizeof(title_table[0])) - 1) ||
        (phelp_window->pwindow_title == NIL) ||
        (phelp_window->pwindow_title[0] == '\0'))
    {
        border.type |= BBRD_NO_TITLE;
    }
    else
        border.type |= title_table[phelp_window->title_type];
    if (border.type != BBRD_NO_BORDER)
        if (border.type & (BBRD_TLT | BBRD_TCT | BBRD_TRT))
        {
            border.pttitle = phelp_window->pwindow_title;
            border.ttattr = utnybbyt(phelp_window->title_back,
                                    phelp_window->title_fore);
        }
}

```

```

        else
            if (border.type & (BBRD_BLT | BBRD_BCT | BBRD_BRT))
            {
                border.pbttitle = phelp_window->pwindow_title;
                border.btattr = utnybbyt(phelp_window->title_back,
                                         phelp_window->title_fore);
            }
    }
    else
    {
        memcpy(&where, &(pwin->where_shown), sizeof(where));
        memcpy(&border, &(pwin->bord), sizeof(border));
    }

    if (wnselect(pwin) == NIL)
    {
        save_error = b_wmerr;
        if (made_window)
            wndstroy(pwin);
        hlreterr(save_error);
    }
    switch (option & HL_TEXT_TYPE)
    {
    case HL_CHARS_ONLY:
        wnwrrect(pwin, 0, 0, wndata_h(pwin) - 1, wndata_w(pwin) - 1,
                 ptext, -1, -1, CHARS_ONLY);
        break;

    case HL_CHAR_ATTR:
        wnwrrect(pwin, 0, 0, wndata_h(pwin) - 1, wndata_w(pwin) - 1,
                 ptext, -1, -1, CHAR_ATTR);
        break;

    case HL_COMPRESSED:
        pwindow_data = (char *) pwin->img.pdata;
        dest_length   = phelp_window->data_columns *
                        phelp_window->data_rows * 2;
        utunsqz(ptext, pwindow_data, text_length, dest_length);
        break;

    default:
        hlreterr(HL_BAD_OPT);
    }

    /* Now read user responses from the window.  Note */
    /* that if the window is already displayed, the */
    /* viewport dimensions, starting origin, where and */
    /* border structures are all ignored. */
    wnread(pwin, &where,
           phelp_window->view_bottom - phelp_window->view_top + 1,
           phelp_window->view_right - phelp_window->view_left + 1,
           phelp_window->origin_row, phelp_window->origin_col,

```



```

        &border, pfinal, (option & (HL_NO_MOUSE | HL_KBIGNORE)));
if (made_window || (option & HL_REMOVE_WIN))
    if (wnremove(pwin) == NIL)
        return(NIL);

if (made_window || (option & HL_DESTROY_WIN))
{
    if (wndstroy(pwin) != WN_NO_ERROR)
        return(NIL);
}
return(pfinal);
}

```

## 帮助程序示例

**BUILDHLP**从文本文件转换成帮助记录。

该程序读取一个包含嵌入命令的ASCII文件和帮助窗口的文本，生成被帮助函数在运行时读取的二进制文件。

命令行的用法为：

```

buildhlp [/i] infile [/o outfile]      [/l lines] /p /q
/i infile      处理的文本文件名。
/o outfile     输出文件的可选名。如果没有提供名，那么使用输入名加上“HLP”。
/l lines       以十进制或十六进制(以$开头)的每页行数。最小数为8。缺省值为66。
/p            预显方式。在处理过程中显示页，以检查页是否对。
/q            静方式。复制提示和统计不显示。然而显示任何使BUILDHLP终止的错误。

```

**BUILDHLP**读取文本文件查找帮助信息记录的命令和源行。源行被变成屏幕图象，并以压缩的形成作为输出文件存盘。记录号、记录ID字符串和记录的文件位置被写向标准输出。

当**BUILDHLP**收集一个记录时，它创建一字符属性图象，把该图象和记录头一起输出到输出文件。不管是位置还是记录名都传给**HLREAD**，**HLREAD**从文件取得记录，创建一帮助窗口，扩展帮助信息映象，并在窗口中显示。

处理开始时，用在**BHELPH**的缺省的帮助记录格式化文本行。如果没有修改帮助记录，则用在运行时可修改的缺省记录即 `ef_help_rec`显示文本。缺省值可以用源文件中嵌入的命令改变。每当遇到点命令，已有的记录写向数据库并且后续的帮助图象用新的帮助记录特点在一窗口中显示。这些命令大小写无关除 `title`参数以外命令参数的大小写也不敏感。下面是合法的命令和有效的帮助记录：

<b>id</b>	建立唯一的帮助信息图象标识。每一图象皆以小于或等于12个的ID字符串标识。命令不修改帮助记录或帮助文本显示的窗口。														
<b>title</b>	在显示时窗口的边界上把它作为标题。														
<b>titlepos</b>	窗口标题的位置。接受的值和其意义如下：														
	<table border="0"> <tr> <th>值</th> <th>标题类型</th> </tr> <tr> <td>0</td> <td>顶中</td> </tr> <tr> <td>1</td> <td>顶右</td> </tr> <tr> <td>2</td> <td>顶左</td> </tr> <tr> <td>3</td> <td>底中</td> </tr> <tr> <td>4</td> <td>底右</td> </tr> <tr> <td>5</td> <td>底左</td> </tr> </table>	值	标题类型	0	顶中	1	顶右	2	顶左	3	底中	4	底右	5	底左
值	标题类型														
0	顶中														
1	顶右														
2	顶左														
3	底中														
4	底右														
5	底左														
<b>xupper,</b>															
<b>yupper</b>	命令修改帮助图象显示的视口的左上角。														
<b>xlower,</b>															
<b>ylower</b>	命令修改帮助图象显示的视口的左上角。														

**datawidth** 设置帮助窗口数据区的宽度。  
**forecolor** 设置窗口数据区的前景颜色属性。  
**backcolor** 设置窗口数据区的背景颜色属性。  
**border type** 设置帮助窗口的边界类型。接受的值和其意义如下：

值	相应的边界类型
0	BBRD_SSSS
1	BBRD_SSSD
2	BBRD_DSSS
3	BBRD_DSSD
4	BBRD_SDSS
5	BBRD_SDSD
6	BBRD_DDSS
7	BBRD_DDSD
8	BBRD_SSDS
9	BBRD_SSDD
10	BBRD_DSDD
11	BBRD_DSDD
12	BBRD_SDDS
13	BBRD_SDDD
14	BBRD_DDDS
15	BBRD_DDDD
16-254	用该ASCII代码组成边界。
255	无边界。

**borderfore** 设置边界的前景颜色属性。  
**borderback** 设置边界的背景颜色属性。  
**titlefore** 设置标题的前景颜色属性。  
**titleback** 设置标题的背景颜色属性。

上述命令用下列颜色名改变颜色属性：

**black, blue, green, cyan, red, magenta, brown, lightgray,**  
**darkgray, lightblue, lightgreen, lightcyan, lightred, lightmagenta,**  
**yellow, white.**

另外，可以用下述代码嵌在帮助文本中控制颜色属性和显示任何IBM字符。

**^A** 下两个字符解释成新颜色属性的十六进制值。比如。**^A60**将在棕色背景下显示黑色字符该命令可以用来加亮某一帮助文本。

**^C** 下两个字符解释成ASCII字符的十六进制值。比如**^C02**将值为2的ASCII码。

如果想在帮助文本在有尖号(^)，要么用两个“^^”，要么用**^C%E**。后不跟A或C的尖号将产生一警告但从不放在帮助文本中。

程序接受的文文件有某些限制：

- 1) 不以“.”开始的文本被认为是当前创建的帮助的文本。所有文本行必须与一帮助记录ID相联系。因此帮助源文件在命令后而“**.id**”之前有文本是一个错误。以“.”开始的行被认为是命令。如果在帮助行中的第一个字符必须是“.”那么以**^C2E**开始。
- 2) 以点号和关键字组成的点命令在关键字后不能有尾随的点号。
- 3) 在帮助文本之后出现的点命令皆被认为是新帮助记录的开始。因此在文本的点命令不能嵌套。未知的命令不结束当前帮助记录。

```
#include <bhelp.h>
#include <blaise.h>
#include <stdio.h>
#include <conio.h>
#include <fcntl.h>
```

```

#include <sys\stat.h>
#include <io.h>
#include <time.h>
#include <dir.h>
#include <string.h>
#define COMMENT      '*'          /* The asterisk signals a comment. */
#define SQUOTE       '\"'         /* Single quotation mark. */
#define DQUOTE       '"'         /* Double quotation mark. */
#define KEYWORDLEN   13           /* Length of a key word. */
#define INITIAL_SIZE 0x8000       /* Initial size of image buffer. */
#define CHUNK_SIZE   0x0800       /* Size to add when growing image. */
#define LAST_TITLE_POS 6
#define MAX_VIEW_COLS 80
#define MAX_VIEW_ROWS 80
#define MIN_LINES     8
/* Error codes from parse_command(). */
#define PARSE_NO_ERROR      0
#define PARSE_BAD_SWITCH   1
#define PARSE_BAD_ARG      2
#define PARSE_NO_ARG       3
#define PARSE_BAD_FILE_NAME 4
#define PARSE_NO_FILE_NAME 5
#define COMPRESS_SIZE 10000
typedef enum
{
    comment text,          /* Source file line contains a comment. */
    command text,          /* Line contains a command keyword. */
    help text              /* Line contains help information. */
} SOURCE_TEXT_TYPE;
/* color_words is an initialized array of text strings whose
/* index corresponds to the value for the attribute.
const char *color_words[] =
{
    "BLACK",
    "BLUE",
    "GREEN",
    "CYAN",
    "RED",
    "MAGENTA",
    "BROWN",
    "LIGHTGRAY",
    "DARKGRAY",
    "LIGHTBLUE",
    "LIGHTGREEN",
    "LIGHTCYAN",
    "LIGHTRED",
    "LIGHTMAGENTA",
    "YELLOW",
    "WHITE"
};

```

```

        /* The array key_words is a typed constant which      */
        /* contains the keywords used as command signals      */
        /* when processing the help source file.      In the  */
        /* text source file each command line begins with a */
        /* period followed by a parameter word, for          */
        /* instance a color name.                            */
const char *key_words[] =
{
    "BACKCOLOR",
    "BORDERBACK",
    "BORDERFORE",
    "BORDERTYPE",
    "DATAWIDTH",
    "DESCRIBE",
    "FORECOLOR",
    "ID",
    "TITLE",
    "TITLEBACK",
    "TITLEFORE",
    "TITLEPOS",
    "XLOWER",
    "XUPPER",
    "YLOWER",
    "YUPPER"
};

/* Function prototypes. */
void      main(int, char **);
void      print_copyright(void);
void      print_usage(void);
int       parse_command(int, char **, char **, char **, int *,
                        int *, int *, int *);
void      split_path(char *, char *, char *, char *);
unsigned char hex_to_byte(char *);
void      abort_error(char *);
HL_INDEX_NODE *insert_index(HL_INDEX_NODE **, char *, char *, long);
void      init_header(char *, char *);
void      open_files(char *, char *, FILE **, int *);
int       keyword_search(char *);
SOURCE_TEXT_TYPE parse_line(char *, int *, char *);
char      get_next_char(char *, int *);
void      text_to_image(CELL **, int *, CELL**, char *);
void      write_file(int, void *, int);
void      print_warning(char *);
void      print_string(char *);
void      fill_image(CELL *, unsigned);
void      write_record(int, HL_INDEX_NODE **, CELL *, int);
int       color_search(char *);
int       convert_number(char *);
void      process_command(int, char *);
int       cstr2pas(const char *, char *, int);

```

```

/* Global variables. */
char *pparse_errors[] =
{
    "", /* PARSE_NO_ERROR */
    "An invalid switch was specified.\n", /* PARSE_BAD_SWITCH */
    "An invalid argument was specified.\n", /* PARSE_BAD_ARG */
    "Switch missing argument.\n", /* PARSE_NO_ARG */
    "An invalid file name was specified.\n", /* PARSE_BAD_FILE_NAME */
    "No input file name was specified.\n" /* PARSE_NO_FILE_NAME */
};

CELL *pcurrent_image_line; /* Current image line. */
unsigned char current_attribute; /* Current screen attribute.*/
HL_INDEX_NODE *pindex_root; /* Root of the index. */
HL_RECORD_DATA def_help_rec;
char help_description[255]; /* Help page description. */
char window_title[255]; /* Window title text. */
int source_file_line = 0; /* Source file line number. */
char page_header[81]; /* The page header. */
int report_line = 0; /* Line # of output report. */
int datawidth_found = 0; /* Found a .datawidth. */
int quiet_mode = 0; /* Quiet mode specified. */
int lines_per_page = 66; /* The number of lines/page.*/

HL_WINDOW default_help_win = /* Default help record window. */
{
    9, 5, /* Viewport top left corner. */
    70, 20, /* Viewport bottom right corner.*/
    NORMAL | INTENSITY, SC_CYAN, /* Window foreground/background.*/
    BBRD_SSSS, /* Default border type. */
    NORMAL | INTENSITY, SC_CYAN, /* Border foreground/background.*/
    BBRD_TCT, /* Title position. */
    NORMAL | INTENSITY, SC_BLUE, /* Title foreground/background. */
    0, /* Number of rows in data area. */
    61 /* # of columns in data area. */
};

void main(argc, argv)
int argc;
char **argv;
{
    char *pinput_file_name = NIL;
    char *poutput_file_name = NIL;
    char made_output_file_name[67] = {"\0"};
    FILE *pinput_file = NIL;
    int output_file_descriptor;
    int preview = 0;
    int help_requested = 0;
    int error_code;
    int id_found = 0;
    int screen_dirty = 0;
    int cursor_row, cursor_column;

```

```

int          cursor_high, cursor_low;
int          cursor_off;
unsigned     found_data_rows;    /* # help lines in record.*/
CELL        *pscreen_image;     /* Allocated screen image.*/
int          image_size;
char         input_line[256];    /* Input file source line.*/
int          screen_count = 0;   /* # of help screens.    */
SOURCE_TEXT_TYR *input_line_type; /* Info in source line.  */
char         command_params[256]; /* Parameters to command. */
int          command_index;      /* Index of command      */
HL_FILE_HEADER file_header;      /* File header record    */
HL_INDEX_NODE *pindex_node;      /* Pointer to index node. */
WN_EVENT     final;

error_code = parse_command(argc, argv, &pinput_file_name,
                           &poutput_file_name, &lines_per_page,
                           &quiet_mode, &preview, &help_requested);
if (error_code != PARSE_NO_ERROR)
{
    print_copyright();
    printf("%s\n", pparse_errors[error_code]);
    print_usage();
    exit(-1);
}
if (!quiet_mode)
    print_copyright();
if (help_requested)
{
    printf("\n");
    print_usage();
    exit(0);
}

/* Open the input and output files. */
if (poutput_file_name == NIL)
    poutput_file_name = made_output_file_name;
open_files(pinput_file_name, poutput_file_name, &pinput_file,
           &output_file_descriptor);
/* Allocate memory for the help page's screen image.*/
image_size = INITIAL_SIZE;
pscreen_image = (CELL *) calloc(image_size, 1);
if (pscreen_image == NIL)
    abort_error("Error: Insufficient memory.");
pcurrent_image_line = pscreen_image;

/* Initialize the page heading string. */
init_header(page_header, pinput_file_name);

/* Initialize the default help record. */
*((HL_WINDOW *) &(def_help_rec.view_top)) = b_def_help_win;
b_def_help_win.pwindow_title = NIL;
def_help_rec.view_top++;
def_help_rec.view_left++;

```

```

def_help_rec.view_bottom++;
def_help_rec.view_right++;
def_help_rec.origin_row++;
def_help_rec.origin_col++;
def_help_rec.record_sign.type      = HL_WINDOW_RECORD;
def_help_rec.record_sign.length = sizeof(def_help_rec);
def_help_rec.reserved              = 0;
def_help_rec.title_length          = 0;
def_help_rec.image_length          = 0;

        /* Initialize the file header and write it to the      */
        /* beginning of the file.                                */
strcpy(file_header.file_sign, HL_FILE_SIGN,
        sizeof(file_header.file_sign));
file_header.text_terminator      = 0x1a; /* Control/Z.          */
file_header.reserved              = 0;
file_header.version              = HL_VERSION;
file_header.index_offset         = 0;
file_header.file_size            = 0;

        /* Write the file header to the database.              */
write_file(output_file_descriptor, &file_header,
        sizeof(file_header));

help_description[0] = '\0';
found_data_rows     = 0;
pindex_root         = NIL;
current_attribute    = utnybbyt(def_help_rec.data_back,
                                def_help_rec.data_fore);

        /* Read a line at a time from the help file, and      */
        /* process it.                                          */
while (!feof(pinput_file))
{
    fgets(input_line, sizeof(input_line), pinput_file);

    if (feof(pinput_file))
        break;
    if (input_line[strlen(input_line) - 1] == '\n')
        input_line[strlen(input_line) - 1] = '\0';
    input_line_type = parse_line(input_line, &command_index,
                                command_params);

    switch(input_line_type)
    {
    case comment_text:
        break;
    case command_text:
        if (screen_dirty)
        {
            fill_image(pscreen_image, found_data_rows);

```

```

if (preview)
{
    cursor_off = accurst(&cursor_row,
                        &cursor_column,
                        &cursor_high,
                        &cursor_low);
    b_def_help_win = *((HL_WINDOW *)
                    &def_help_rec.view_top);
    b_def_help_win.pwindow_title = window_title;
    b_def_help_win.origin_row--;
    b_def_help_win.origin_col--;
    b_def_help_win.view_top--;
    b_def_help_win.view_left--;
    b_def_help_win.view_bottom--;
    b_def_help_win.view_right--;

    hread(NIL, &b_def_help_win,
        (char *) pscreen_image,
        b_def_help_win.data_rows *
        b_def_help_win.data_columns * 2,
        &final, HL_CHAR_ATTR);

    accurset(cursor_row, cursor_column);
    scpgcur(cursor_off, cursor_high, cursor_low,
        CUR_NO_ADJUST);
}
write_record(output_file_descriptor, &pinde_x_root,
            pscreen_image, screen_count++);
pcurrent_image_line = pscreen_image;
id_found = 0;
datawidth_found = 0;
screen_dirty = 0;
found_data_rows = 0;
def_help_rec.data_rows = 0;
current_attribute = utnybbyt(def_help_rec.data_back,
                            def_help_rec.data_fore);
/* Now clear all the strings to all zeroes. */
memset(help_description, '\0',
        sizeof(help_description));
memset(def_help_rec.id_string, '\0',
        sizeof(def_help_rec.id_string));
}
if (strcmp(key_words[command_index], "ID") == 0)
    id_found = 1;
process_command(command_index, command_params);
break;
case help_text:
    /* If there is no ID, or a new ID has not been */
    /* specified, then abort. */
    if ((def_help_rec.id_string[0] == '\0') || !id_found)

```



```

        abort_error("Error: Missing record ID.");
    if (iscreen_dirty)
        if (datawidth_found)
        {
            if (def_help_rec.data_columns <
                (def_help_rec.view_right -
                 def_help_rec.view_left + 1))
            {
                abort_error(
                    "Error: Data area too small for viewport.");
            }
        }
        else
            def_help_rec.data_columns =
                def_help_rec.view_right -
                def_help_rec.view_left + 1;
    text_to_image(&pscreen_image, &image_size,
                &pcurrent_image_line, &put_line);
    found_data_rows++;
    screen_dirty = 1;
    break;
}
source_file_line++;

/* If the last lines of the file were help */
/* information lines, then a record exists to write */
/* to disk. */
if (screen_dirty)
{
    fill_image(pscreen_image, found_data_rows);
    if (preview)
    {
        cursor_off = scurst(&cursor_row, &cursor_column,
                           &cursor_high, &cursor_low);
        b_def_help_win = *((HL_WINDOW *) &def_help_rec.view_top);
        b_def_help_win.pwindow_title = window_title;
        b_def_help_win.origin_row--;
        b_def_help_win.origin_col--;
        b_def_help_win.view_top--;
        b_def_help_win.view_left--;
        b_def_help_win.view_bottom--;
        b_def_help_win.view_right--;
        hhread(NIL, &b_def_help_win, (char *) pscreen_image,
               b_def_help_win.data_rows *
               b_def_help_win.data_columns * 2,
               &final, HL_CHAR_ATTR);
        scurset(cursor_row, cursor_column);
        scpgcur(cursor_off, cursor_high, cursor_low, CUR_NO_ADJUST);
    }
    write_record(output_file_descriptor, &pindex_root,

```

```

        pscreen_image, screen_count++);
    }

    /* Write the index records at the end of the file. */
    /* Record the file's length in the file header, and */
    /* update the file header. */
    file_header.index_offset = lseek(output_file_descriptor, (long) 0,
                                    SEEK_CUR);
    for (pindex_node = pindex_root; pindex_node != NIL;
         pindex_node = pindex_node->pleft)
    {
        cstr2pas(pindex_node->index_data.id_string,
                pindex_node->index_data.id_string,
                sizeof(pindex_node->index_data.id_string));
        if (pindex_node->pleft != NIL)
            pindex_node->index_data.next_index =
                lseek(output_file_descriptor, (long) 0, SEEK_CUR) +
                sizeof(pindex_node->index_data) +
                pindex_node->index_data.description_len;
        else
            pindex_node->index_data.next_index = 0L;
        write_file(output_file_descriptor,
                  &pindex_node->index_data,
                  sizeof(pindex_node->index_data));
        if (pindex_node->index_data.description_len != 0)
            write_file(output_file_descriptor,
                      pindex_node->pdescription,
                      pindex_node->index_data.description_len);
    }
    file_header.file_size = lseek(output_file_descriptor, (long) 0,
                                    SEEK_END);
    lseek(output_file_descriptor, (long) 0, SEEK_SET);
    write_file(output_file_descriptor, &file_header,
                sizeof(file_header));
    fclose(pinput_file);
    close(output_file_descriptor);
}

/**
 * Name          PRINT_COPYRIGHT Print copyright and version notice.
 * Synopsis      print_copyright();
 * Description    This function prints a short copyright notice and
 *               the version number of the program.
 * Returns       nothing.
 */
void print_copyright()
{
    printf("Turbo C TOOLS Help File Utility\n");
    printf("Version      6.00  (C)Copyright Blaise Computing Inc.
          "1989\n");
}

/**

```

```

* Name      PRINT_USAGE- Print a message describing valid
*
*              command line options.
* Synopsis  print_usage();
* Description This function prints a short usage description to
*              the standard output which describes all the valid
*              command line switches and their arguments.
* Returns   nothing.
**/
void print_usage()
{
    printf("Command line syntax:\n");
    printf("buildhlp [/i] <inputfile> [/o] <outputfile> [/p] [/q] [/?] "
        "[/h]\n\n");
    printf("    /i inputfile    Name of text source file\n");
    printf("    /o outputfile  Name of compiled output file\n");
    printf("    /l lines       Number of lines per page on output\n");
    printf("    /q             Suppress version, copyright, and "
        "statistics output\n");
    printf("    /p             Preview mode. Help pages are displayed "
        "during processing\n");
    printf("    /? or /h      Display this message\n");
}
/**
* Name      PARSE_COMMAND Parse the command line for valid
*
*              options.
* Synopsis  error = parse_command(argc, argv, pinput_file_name,
*              poutput_file_name, pquiet_mode, ppreview,
*              phelp_requested);
*
*      int    error          Error encountered during
*
*                          parse - PARSE_NO_ERROR if
*                          none.
*
*      int    argc           Number of command line
*
*                          arguments.
*
*      char **argv           Array of strings containing
*
*                          command line arguments.
*
*      char **ppinput_file_name Upon return, this points to a
*
*                          string indicating the name of
*                          the input file.
*
*      char **ppoutput_file_name Upon return, this points to a
*
*                          string indicating the name of
*                          the output file.
*
*      int    *plines_per_page The number specified for the
*
*                          /l option (unchanged if not
*                          present).
*
*      int    *pquiet_mode    Whether or not the /q option
*
*                          was specified.
*
*      int    *ppreview       Whether or not the /p option
*
*                          was specified.
*
*      int    *phelp_requested Whether or not the /? or /h
*
*                          option was specified.

```

\* **Description**      This function accepts the number of arguments passed to  
 \*                      `main()` and a pointer to a NULL terminated array of  
 \*                      strings containing the command line arguments passed to  
 \*                      `main()`, and parses it for correctness.      Each valid  
 \*                      option encountered causes the corresponding argument to  
 \*                      be set, indicating that the option was found.

\* **Returns**            `error`                      Possible values:  
 \*                      `PARSE_NO_ERROR`: no errors.  
 \*                      `PARSE_BAD_SWITCH`: an invalid switch  
 \*                      was specified.  
 \*                      `PARSE_BAD_ARG`: an argument which  
 \*                      was neither a switch nor a  
 \*                      filename was encountered.  
 \*                      `PARSE_BAD_FILE_NAME`: an invalid file  
 \*                      name was specified.  
 \*                      `PARSE_NO_FILE_NAME`: no input file  
 \*                      name was specified.

\*                      `ppinput_file_name`      A string indicating the name of the  
 \*                      input file; unchanged if no name  
 \*                      found.

\*                      `ppoutput_file_name`      A string indicating the name of the  
 \*                      output file; unchanged if no name  
 \*                      found.

\*                      `plines_per_page`            A number indicating the parameter  
 \*                      after the `/l` option; unchanged if  
 \*                      option not specified.

\*                      `pquiet_mode`                Non-zero if the `/q` option was  
 \*                      specified; unchanged otherwise.

\*                      `ppreview`                    Non-zero if the `/p` option was  
 \*                      specified; unchanged otherwise.

\*                      `phelp_requested`          Non-zero if the `/?` or `/h` option was  
 \*                      specified; unchanged otherwise.

\*\*/

```
int parse_command(argc, argv, ppinput_file_name, ppoutput_file_name,
                  plines_per_page, pquiet_mode, ppreview,
                  phelp_requested)

int argc;
char **argv;
char **ppinput_file_name;
char **ppoutput_file_name;
int *plines_per_page;
int *pquiet_mode;
int *ppreview;
int *phelp_requested;
{
    int current_argument;
    int input_file_found = 0;
    int output_file_found = 0;
    for (current_argument = 1; current_argument < argc;
         current_argument++)
```

```

{
    if (argv[current_argument][0] == '/' ||
        argv[current_argument][0] == '-')
    {
        switch(toupper(argv[current_argument][1]))
        {
            case 'I':
                if (current_argument == (argc - 1))
                    return(PARSE_NO_ARG);
                *ppinput_file_name = argv[++current_argument];
                if ((*ppinput_file_name[0] == '/') ||
                    (*ppinput_file_name[0] == '-'))
                {
                    return(PARSE_BAD_FILE_NAME);
                }
                input_file_found = 1;
                break;
            case 'O':
                if (current_argument == (argc - 1))
                    return(PARSE_NO_ARG);
                *ppoutput_file_name = argv[++current_argument];
                if ((*ppoutput_file_name[0] == '/') ||
                    (*ppoutput_file_name[0] == '-'))
                {
                    return(PARSE_BAD_FILE_NAME);
                }
                output_file_found = 1;
                break;
            case 'L':
                if (current_argument == (argc - 1))
                    return(PARSE_NO_ARG);
                *plines_per_page =
                    convert_number(argv[++current_argument]);
                if (*plines_per_page < MIN_LINES)
                    return(PARSE_BAD_ARG);
                break;
            case 'Q':
                *pquiet_mode = 1;
                break;
            case 'P':
                *ppreview = 1;
                break;
            case 'H':
            case '?':
                *phelp_requested = 1;
                break;
            default:
                return(PARSE_BAD_SWITCH);
        }
    }
}

```

```

else
    /* We have encountered an argument which has no */
    /* leading switch character. The input and output */
    /* file names are the only arguments for which this */
    /* is permissible. If we already have an input file */
    /* name, assume this argument is the output file; */
    /* if we have both names already, it is an error. */
    if (!input_file_found)
    {
        *ppinput_file_name = argv[current_argument];
        input_file_found = 1;
    }
    else
        if (!output_file_found)
        {
            *ppoutput_file_name = argv[current_argument];
            output_file_found = 1;
        }
        else
            return(PARSE_BAD_ARG);
}
if (!input_file_found)
    return(PARSE_NO_FILE_NAME);
return(PARSE_NO_ERROR);
}
/**
 * Name          SPLIT_PATH - Parse a pathname into a directory name,
 *                  file name, and extension.
 * Synopsis      split_path(ppath_name, pdirectory_name, pfile_name,
 *                  pfile_extension);
 *
 * char *ppath_name      The pathname to parse.
 * char *pdirectory_name  The parsed directory name.
 * char *pfile_name       The parsed file name.
 * char *pfile_extension  The parsed extension name.
 * Description      This function will accept a pointer to a path name with
 *                  optional drive specifier, and will separate it into a
 *                  directory name, file name, and extension.
 * Returns          *pdirectory_name  The parsed directory name.
 *                  *pfile_name       The parsed file name.
 *                  *pfile_extension  The parsed extension name.
 **/
void split_path(ppath_name, pdirectory_name, pfile_name, pfile_extension)
char *ppath_name;
char *pdirectory_name;
char *pfile_name;
char *pfile_extension;
{
    int index;
    char temp_string[256];
    /* Search from the end of ppath_name to the */

```

```

        /* beginning to find the end of the path component. */
for (index = strlen(ppath_name) - 1; (index > 0) &&
    (ppath_name[index] != '\\') && (ppath_name[index] != ':');
    index--)
    ;

    /* Store the path, and put the remainder of the      */
    /* string in temporary storage.      The file name and */
    /* extension will be extracted from it.                */
strcpy(pdirectory_name, ppath_name, index);
pdirectory_name[index] = '\0';
strcpy(temp_string, ppath_name + index);
    /* Like the path delimiter, the extension delimiter */
    /* is searched for starting at the end of the      */
    /* string.                                          */
for (index = strlen(temp_string) - 1; (index > 0) &&
    (temp_string[index] != '.')); index--)
    ;
strcpy(pfile_extension, temp_string + index + 1);
if (index > 8)
    index = 8;
strcpy(pfile_name, temp_string, index);
pfile_name[index] = '\0';
}
/**
 * Name          ABORT_ERROR Issue an error message and halt the
 *                program.
 * Synopsis      abort_error(pmessage);
 *                char *pmessage Pointer to error message string.
 * Description    This function will accept a pointer to an error message
 *                string, prints it to the standard output followed by a
 *                message indicating the line of the input file being
 *                processed when the error was encountered, and terminates
 *                the program.
 * Returns       nothing.
 **/
void abort_error(pmessage)
char *pmessage;
{
    printf("\n%s\n", pmessage);
    if (source_file_line > 0)
        printf("BUILDHLP: terminating at line %d of input file.\n",
            source_file_line);
    exit(-1);
}
/**
 * Name          INSERT_INDEX Insert a help record ID into the linked
 *                list.
 * Synopsis      presult = insert_index(ppid_list, pid_string,
 *                pdescription, file_position)
 *                HL_INDEX_NODE **ppid_list Pointer to a pointer to the

```

```

*                                     head of the linked list of
*                                     index nodes.
*      char *pid_string               The id of the new help record
*                                     to be inserted in the index.
*      char *pdescription             Pointer to the help record's
*                                     description string.
*      long file_position             The offset in the file of the
*                                     help record.
* Description      This function will create a new help record index node
*                  and place it in the linked list of index nodes.      This
*                  list will eventually be written to the end of the
*                  binary help file.      The program will be aborted if the
*                  given id string is a duplicate of one that already
*                  exists or if there is not enough memory to create a new
*                  index node.
* Returns          presult  Pointer to the newly created index node.
**/
HL_INDEX_NODE *insert_index(ppid_list, pid_string, pdescription,
                           file_position)

HL_INDEX_NODE **ppid_list;
char          *pid_string;
char          *pdescription;
long          file_position;
{
    HL_INDEX_NODE *pcurrent;
    HL_INDEX_NODE *pnode;
    char          message[81];
    pnode = (HL_INDEX_NODE *) calloc(sizeof(HL_INDEX_NODE), 1);
    if (pnode == NIL)
        abort_error("Error: Insufficient memory.");
    pnode->index_data.record_sign.type      = HL_INDEX_RECORD;
    pnode->index_data.record_sign.length = sizeof(HL_INDEX_DATA);
    pnode->index_data.reserved              = 0;
    pnode->index_data.help_record           = file_position;
    strcpy(pnode->index_data.id_string, pid_string);
    /* Determine the length of the description string.      */
    pnode->index_data.description_len =
        (pdescription != NIL) ? strlen(pdescription) : 0;
    if (pnode->index_data.description_len != 0)
    {
        pnode->pdescription =
            malloc(pnode->index_data.description_len + 1);
        strcpy(pnode->pdescription, pdescription);
    }
    if (*ppid_list == NIL)
    {
        *ppid_list = pnode;
        return(pnode);
    }

    /* If there is a matching ID, then exit.      */

```



```

for(pcurrent = *ppid_list; pcurrent->pleft != NIL;
    pcurrent = pcurrent->pleft)
{
    if (strcmp(pcurrent->index_data.id_string, pid_string) == 0)
    {
        sprintf(message, "Error: Duplicate ID \"%s\".",
            pid_string);
        abort_error(message);
    }
}
pcurrent->pleft = pnode; /* Use the left node pointer. */
return(pnode);

```

```

/**
 * Name          INIT HEADER- Initialize the report page header
 *               string.
 * Synopsis      init_header(pheader_string, pfile_path);
 *               char *pheader_string  Pointer to page header string.
 *               char *pfile_path      Pointer to pathname of help
 *               source file.
 * Description    This function accepts a pointer to the pathname of the
 *               input help source file, and forms a page header string
 *               consisting of the file name and the current date and
 *               time, which will be printed at the top of each page
 *               (assumed to be 66 lines long) of the output report.
 * Returns       *pheader_string  Page header string.
 */

```

```

void init_header(pheader_string, pfile_path)
char *pheader_string;
char *pfile_path;
{
    int      source_index, target_index;
    time_t   current_time;
    char      directory_name[MAXDIR];
    char      file_name[MAXFILE];
    char      file_extension[MAXEXT];
    char      full_name[MAXFILE + MAXEXT + 1];
    char      *ptime_string;
    memset(pheader_string, ' ', 80);
    pheader_string[80] = '\0';
    split_path(pfile_path, directory_name, file_name, file_extension);
    strcpy(full_name, file_name);
    if (file_extension[0] != '\0')
    {
        strcat(full_name, ".");
        strcat(full_name, file_extension);
    }

    /* First insert the file name. */
    for (source_index = 0; target_index = 0;
        source_index < strlen(full_name);

```

\*/

```

        source_index++, target_index += 2)
    {
        pheader_string[target_index] = toupper(full_name[source_index]);
        pheader_string[target_index + 1] = ' ';
    }

    /* Now insert the current date and time string. */
    time(&current_time);
    ptime_string = ctime(&current_time);
    source_index = strlen(ptime_string);
    memmove(&pheader_string[80 - source_index], ptime_string,
            source_index);
}
/**
 * Name      OPEN_FILES- Open the input and output files.
 * Synopsis  open_files(pinput_path, poutput_path, ppinput_file,
 *                    poutput_file);
 *
 *          char *pinput_path   Pointer to the input file name.
 *          char *poutput_path  Pointer to the output file name.
 *          FILE **ppinput_file Pointer to the returned open
 *                               input file pointer.
 *          int  *poutput_file  Pointer to the returned open
 *                               output file descriptor.
 *
 * Description This function accepts pointers to the pathnames of the
 *             input and output files and attempts to open them both.
 *             It returns the file stream pointer to the open help
 *             text input file, and the file descriptor of the binary
 *             output help database.
 *
 * Returns    *ppinput_file  Pointer to the file stream pointer for
 *                               the input text file.
 *            *poutput_file  Pointer to the file descriptor for the
 *                               output file.
 */
void open_files(pinput_path, poutput_path, ppinput_file, poutput_file)
char *pinput_path;
char *poutput_path;
FILE **ppinput_file;
int *poutput_file;
{
    char directory_name[MAXDIR];
    char file_name[MAXFILE];
    char file_extension[MAXEXT];
    char response_character;
    char message[81];
    if (access(pinput_path, 4) != 0)
    {
        sprintf(message,
                "Error: Cannot open input file \"%s\" for reading.",
                pinput_path);
        abort_error(message);
    }
}

```

```

split_path(pinput_path, directory_name, file_name, file_extension);
if (poutput_path[0] == '\0')
    sprintf(poutput_path, "%s%s.hlp", directory_name, file_name);
split_path(poutput_path, directory_name, file_name, file_extension);
if (file_extension[0] == '\0')
    strcat(poutput_path, ".hlp");
if (access(poutput_path, 0) == 0)
{
    printf("Output file %s already exists; it ", poutput_path);
    printf("will be overwritten.\n");
    printf("Continue (Y/N)? ");
    response_character = getch();
    printf("%c\n", response_character);
    if (toupper(response_character) != 'Y')
        exit(0);
}

/* Open the text source file and create the help */
/* file. */
*ppinput_file = fopen(pinput_path, "r");
*poutput_file = open(poutput_path, (int) (O_RDWR | O_CREAT |
    O_TRUNC | O_BINARY), S_IRREAD | S_IWRITE);
}
/**
* Name      KEYWORD_SEARCH Search the array of known keywords
*           for a match.
* Synopsis  index = keyword_search(pkey_word);
*           int index      Returned index of the requested
*                           key word in the array key_words.
*           char *pkey_word Pointer to the key word to look for.
* Description This function accepts a pointer to a potential key word
*           and searches the global array key_words to see if it is
*           a legitimate key word. If it is, its index in the array
*           key_words is returned, otherwise a -1 is returned.
* Returns   index Index of *pkey_word in key_words if found, or -1.
**/
int keyword_search(pkey_word)
char *pkey_word;
{
    int index;
    if (pkey_word == NIL)
        return(-1);
    for (index = 0; index < (sizeof(key_words) / sizeof(key_words[0]));
        index++)
    {
        if (strcmp(pkey_word, key_words[index]) == 0)
            return(index);
    }
    return(-1);
}
/**

```

```

* Name      PARSE_LINE- Parse a line of text from the help source
*
*           file.
* Synopsis  type = parse_line(psource_line, pcommand_index,
*                               pcommand_parameters)
*
*           char *psource_line      Pointer to the line of input
*                                   to be parsed.
*
*           int  *pcommand_index    If *psource_line contains a
*                                   known command, this is its
*                                   index; -1 if an unrecognized
*                                   command.
*
*           char *pcommand_parameters Pointer to first character
*                                   after command in psource_line;
*                                   unchanged if *psource_line is
*                                   not a command.
*
* Description This function accepts a pointer to a line of input from
* the help source file and determines whether it is a
* comment, a command, or help text. If it is a command,
* *pcommand_index will contain the index of the command if
* it is a known command, or -1 if it is not, and
* *pcommand_parameters will point to the first character
* after the command, which may contain extra parameters
* for the command.
*
* Returns   type                    Type of text contained in
*                                   *psource_line.
*
*           *pcommand_index        If *psource_line is a command,
*                                   this is its index, or -1 if it is
*                                   an unknown command.
*
*           *pcommand_parameters  Pointer to first character after
*                                   command in psource_line; unchanged
*                                   if *psource_line is not a command.
**/
SOURCE_TEXT_TYPE parse_line(psource_line, pcommand_index,
                             pcommand_parameters)

char  *psource_line;
int    *pcommand_index;
char  *pcommand_parameters;
{
    int  source_index, target_index;
    char keyword[KEYWORDLEN];
    char message[81];

    /* Look for a period in the first position of each text */
    /* line. If one is found, then the line is either a */
    /* comment or a command. If no period is found, or the */
    /* line is zero characters long, then the type of */
    /* text defaults to HelpTxt. */
    if ((strlen(psource_line) == 0) || (psource_line[0] != '.'))
        return(help_text);
    /* If the next character is an asterisk, it's a comment.*/
    if (psource_line[1] == COMMENT)
        return(comment_text);

```

```

        /* Attempt to create a command word by processing no      */
        /* more than KEYWORDLEN characters.                        */
for (source_index = 1, target_index = 0;
    (target_index < sizeof(keyword) - 1) &&
    (psource_line[source_index] != '\0') &&
    (psource_line[source_index] != ' ');
    source_index++, target_index++)
{
    keyword[target_index] = toupper(psource_line[source_index]);
}
keyword[target_index] = '\0';
*pcommand_index = keyword_search(keyword);
if (*pcommand_index == -1)
{
    /* Could not match command word.                               */
    sprintf(message, "Warning: Unknown command word \"%s\".",
        keyword);
    print_warning(message);
    return(comment_text);
}
strcpy(pcommand_parameters, psource_line + source_index);
return(command_text);
}

```

```

/**
 * Name          HEX_TO_BYTE  Convert a hexadecimal string to a byte.
 * Synopsis      byte = hex_to_byte(phex_string);
 *               unsigned char byte          The converted byte.
 *               char *psource_line          Pointer to the string to be
 *               converted.
 * Description    This function accepts a pointer to a string of two
 *               hexadecimal characters and converts it into a byte.
 *               If any digit in the string is not a valid hex digit,
 *               an error message is displayed and the program is
 *               aborted.
 * Returns       byte          The converted byte.
 */

```

```

unsigned char hex_to_byte(phex_string)
char *phex_string;
{
    char message[81];
    unsigned return_value;
    if (!isxdigit(*phex_string))
    {
        sprintf(message, "Error: Invalid hexadecimal digit %c.",
            *phex_string);
        abort_error(message);
    }
    if (!isxdigit(*(phex_string + 1)))
    {
        sprintf(message, "Error: Invalid hexadecimal digit %c.",

```

```

        *(phex_string + i));
    abort_error(message);
}
sscanf(phex_string, "%02x", &return_value);
return((unsigned char) return_value);
}
/**
* Name      GET_NEXT_CHAR Return the next character from a
*            string.
* Synopsis  character = get_next_char(pstring, pindex);
*            char character    The next character.
*            char *pstring     Pointer to the string from which a
*                               character is to be returned.
*            int *pindex       Pointer to the current index into
*                               the string.
* Description This function accepts a pointer to a string and a
*            pointer to the current index into the string, and will
*            return the next character value from the string and
*            update *pindex so that it references the next available
*            character. The following embedded controls (prefixed by
*            a '^') are recognized and interpreted:
*            ^A: Change the current attribute. The next two
*                characters are taken to be a hexadecimal string, and
*                are converted to a byte representing their value
*                which is then placed in the global variable
*                current_attribute.
*            ^C: Special character. The next two characters are
*                taken to be a hexadecimal string, and are converted
*                to a byte representing their value which is then
*                returned. This is useful for entering control
*                characters or IBM graphic characters.
*            ^^: Caret. The first caret is ignored; the second is
*                returned.
*            Unrecognized commands will generate a warning message,
*            but will not terminate the program. Invalid arguments
*            for ^A or ^C will generate an error message and will
*            terminate the program.
* Returns    byte The converted byte.
**/
char get_next_char(pstring, pindex)
char *pstring;
int *pindex;
{
    char return_value;
    char message[81];
    if (*pindex >= strlen(pstring))
        return(' ');
    if (pstring[*pindex] != '^')
        return(pstring[(*pindex)++]);
    (*pindex)++;

```

```

switch (toupper(pstring[*pindex]))
{
case 'A':
    /* Specified attribute. */
    current_attribute = hex_to_byte(&pstring[*pindex + 1]);
    *pindex += 3;
    return(get_next_char(pstring, pindex));
case 'C':
    /* Special character. */
    return_value = (char ) hex_to_byte(&pstring[*pindex + 1]);
    *pindex += 3;
    return(return_value);
case '^':
    /* Display the caret. */
    return(pstring[*pindex] + + );
}

/* Unknown character after the caret. */
sprintf(message, "Warning: Unknown control character %c.",
        pstring[*pindex]);
print_warning(message);
return(pstring[*pindex] - 1);
}

```

/\*\*

* Name	TEXT_TO_IMAGE	Convert a line of help source to a help image.
* Synopsis	text_to_image(ppscreen_image, pimage_size, ppimage_line, ptext_line);	
* CELL **ppscreen_image	Pointer to a pointer to the screen image being built.	
* int *pimage_size	Pointer to the number of bytes allocated for *ppscreen_image.	
* CELL **ppimage_line	Pointer to a pointer to the beginning of the current line within *ppscreen_image.	
* char *ptext_line	Pointer to the help source line to be converted.	
* Description	<p>This function will convert a line of text from the help source file into a screen image in character/attribute format. If the screen image is not large enough to hold the help text, an attempt will be made to re-allocate it. If this attempt fails, an error message will be printed and the program will be aborted. If it succeeds, the new pointer will be returned in *ppscreen_image and the new size will be returned in *pimage_size. In either case, a pointer to the start of the next screen image line is returned in *ppimage_line.</p>	
* Returns	*ppscreen_image	Pointer to the screen image buffer (possibly reallocated).
	*pimage_size	Pointer to the screen image buffer

```

*                                     size (possibly increased).
*                                     *ppimage_line   Pointer to the start of the next
*                                     screen image line.
**/
void text_to_image(ppscreen_image, pimage_size, ppimage_line,
                  ptext_line)
CELL **ppscreen_image;
int   *pimage_size;
CELL **ppimage_line;
char  *ptext_line;
{
    int      source_index, target_index;
    unsigned line_offset;
                /* Determine that the image will not overwrite */
                /* memory. */
    line_offset = ((unsigned) (*ppimage_line - *ppscreen_image)) * 2;
    if (line_offset +
        ((unsigned) def_help_rec.data_columns * 2) >=
        ((unsigned) *pimage_size))
    {
        *pimage_size += CHUNK_SIZE;
        *ppscreen_image = realloc(*ppscreen_image, *pimage_size);
        if (*ppscreen_image == NIL)
            abort_error("Error: Insufficient memory.");
        *ppimage_line = *ppscreen_image + (line_offset / 2);
    }
                /* Place the character attribute pairs in the image */
                /* buffer. */
    source_index = 0;
    for (target_index = 0;
        target_index < def_help_rec.data_columns;
        target_index++)
    {
        (*ppimage_line)[target_index].ch =
            get_next_char(ptext_line, &source_index);
        (*ppimage_line)[target_index].attr = current_attribute;
    }
    *ppimage_line += def_help_rec.data_columns;
}
/**
* Name      WRITE_FILE - Write a block of data to the
*                help database.
* Synopsis  write_file(output_file_descriptor, pdata, data_size);
*                int output_file_descriptor    The file descriptor to write
*                to.
*                void *pdata                  Pointer to the data to be
*                written.
*                int data_size                 Size of the data to write.
* Description This function will write the specified number of bytes
*                from the output buffer to the specified file.      On error,

```



```

*          a message is displayed and the program terminated.
* Returns   Nothing.
**/
void write_file(output file descriptor, pdata, data_size)
int    output file descriptor;
void *pdata;
int    data_size;
{
    if (write(output file descriptor, pdata, data_size) != data_size)
        abort error("Error: Cannot write to output file.");
}
/**
*
* Name      PRINT STRING  Print a line of text to standard
*                  output.
* Synopsis   print_string(pstring);
*            char *pstring  Pointer to the string to be printed.
* Description This function accepts a pointer to a string and prints
*            it on the standard output.  If the end of a page has
*            been reached (66 lines per page is assumed) then a page
*            skip is generated and several lines of header
*            information are printed at the top of the next page
*            before the current line is printed.
* Returns    Nothing.
**/
void print_string(pstring)
char *pstring;
{
    if (quiet mode)
        return;
    if (report_line >= (lines_per_page - 3))
    {
        printf("\n\n");
        report_line = 0;
    }
    if (report_line == 0)
    {
        printf("%s\n", page_header);
        printf(" Item Number      ID String  Position\n");
        printf(" -----      -----  -----\n");
        report_line = 4;
    }
    printf(" %s\n", pstring);
    report_line++;
}
/**
* Name      PRINT WARNING  Print a warning to the standard
*                  error.
* Synopsis   print_warning(pstring);
*            char *pstring  Pointer to the string to be printed.

```

```

* Description   This function accepts a pointer to a string and prints
*               it on the standard error.   The string will be prefixed
*               by a space and postfixed by a newline ("\n").
* Returns      Nothing.
**/

```

```

void print_warning(pstring)
char *pstring;
{
    fprintf(stderr, " %s\n", pstring);
}

```

```

/**
* Name          FILL IMAGE- Fill in unused space in the screen image
*               if necessary.
* Synopsis      fill_image(pimage, data_rows);
*               CELL *pimage      Pointer to the screen image to be
*                               filled.
*               unsigned data_rows Number of data rows contained in
*                               *pimage.
* Description   This function will fill all the rows from the end of
*               the constructed screen image up to the height of the
*               viewport with blanks having the current attribute if
*               the number of rows in the screen image is less than
*               the specified height of the viewport.
* Returns      *pimage      Possibly padded with blanks having the
*                               current attribute.
**/

```

```

void fill_image(pimage, data_rows)
CELL *pimage;
unsigned data_rows;
{
    int win_view_rows;
    CELL temp_cell;
    int temp_index, start_index, end_index;
    int win_data_columns = def_help_rec.data_columns;
    /* Determine the height of the help viewport. */
    win_view_rows = def_help_rec.view_bottom -
        def_help_rec.view_top + 1;
    /* Record the number of rows and columns of the */
    /* image in the help record. */
    def_help_rec.data_rows = (data_rows > win_view_rows)
        ? data_rows : win_view_rows;
    /* The display routine requires that the image be */
    /* at least as many lines long as the window it is */
    /* to be displayed in. If this is not the case, */
    /* fill the remaining lines with spaces having the */
    /* current display attribute. */
    if (win_view_rows > data_rows)
    {

```

```

temp_cell.ch      = ' ';
temp_cell.attr = current_attribute;
start_index      = (win_data_columns * data_rows);
end_index        = (win_data_columns * win_view_rows);
for (temp_index = start_index; temp_index <= end_index;
    temp_index++)
{
    pimage[temp_index] = temp_cell;
}
}
}
/**
 * Name          WRITE_RECORD Write a help record to a database.
 * Synopsis      write_record(output_file_descriptor, ppindex_root,
 *                          pimage, screen_number);
 *
 *      int output_file_descriptor      File descriptor
 *                                      of database to write
 *                                      to.
 *
 *      HL_INDEX_NODE **ppindex_root  Pointer to a pointer
 *                                      to the root of the
 *                                      help database index.
 *
 *      CELL *pimage                  Pointer to the image
 *                                      of the help screen to
 *                                      write.
 *
 *      int screen_number              Number of screen being
 *                                      written.
 *
 * Description    This function will write out a complete help database
 *                record, including the record header and compressed
 *                screen image (which it constructs based on the
 *                uncompressed image in *pimage). It also places a
 *                new entry in the in memory database index for the help
 *                record.
 *
 * Returns        *ppindex_root      If NIL on entry, *ppindex root points
 *                                     to the newly created help index record.
 */
void write_record(output_file_descriptor, ppindex_root, pimage,
                  screen_number)
int                output_file_descriptor;
HL_INDEX_NODE **ppindex_root;
CELL               *pimage;
int                screen_number;
{
    char *pcompressed_image;
    int   required_size;
    char  message[81];
    int   image_size;
    long  record_position;
    image_size = def_help_rec.data_rows *
                  def_help_rec.data_columns * 2;
    /* If the preview option has been selected (/P)          */

```

```

        /* then display the help page in its prescribed */
        /* window. */
        /* Allocate a compressed image buffer of 10000 */
        /* bytes. If it is not large enough, then use the */
        /* returned value from utsqzscn() to determine the */
        /* actual number of bytes. */
pcompressed_image = (char *) malloc(COMPRESS_SIZE);
if (pcompressed_image == NIL)
    abort_error("Error: Insufficient memory.");
required_size = utsqzscn((char *) pimage, pcompressed_image,
                        image_size, COMPRESS_SIZE);
if (required_size > COMPRESS_SIZE)
{
    pcompressed_image = realloc(pcompressed_image, required_size);
    if (pcompressed_image == NIL)
        abort_error("Error: Insufficient memory.");
        /* If the screen compression fails again, then there */
        /* is some problem. */
    if (required_size != utsqzscn((char *) pimage,
                                pcompressed_image,
                                image_size, required_size))
        abort_error("Error: Internal BUILDHLP error "
                    "while compressing screen.");
}
def_help_rec.image_length = required_size;
record_position = lseek(output_file_descriptor, (long) 0,
                        SEEK_CUR);
        /* Add the ID and position to the index. */
insert_index(ppindex_root, def_help_rec.id_string,
            help_description, record_position);
help_description[0] = '\0';
sprintf(message, "%11d %13s %9ld", screen_number,
        def_help_rec.id_string, record_position);
print_string(message);
        /* Write the help record, compressed image, and */
        /* window title to the file. */
def_help_rec.title_length = strlen(window_title);
        /* Convert the record's id string to pascal format. */
cstr2pas(def_help_rec.id_string, def_help_rec.id_string,
        sizeof(def_help_rec.id_string));
write_file(output_file_descriptor, &def_help_rec,
        sizeof(def_help_rec));
write_file(output_file_descriptor, pcompressed_image,
        required_size);
if (def_help_rec.title_length != 0)
    write_file(output_file_descriptor, window_title,
        def_help_rec.title_length);
free(pcompressed_image);
}
/**

```

- \* Name            **COLOR SEARCH** Return a color value for the named color.
- \* Synopsis        `color = color_search(pstring);`
- \*                `int color`            The color value of the color.
- \*                `char *pstring`        Name of the color to convert.
- \* Description     This function accepts a pointer to a string which contains the name of a color, searches for it in the global array `color_words[]`, and returns its index (which is also its value if used as part of a screen attribute).    If the color is not in `color_words[]` an error message is displayed and the program is aborted.
- \* Returns         `color`    the color value of the named color.

```

/**/
int color_search(pstring)
char *pstring;
{
    int index;
    int found = 0;
    char message[81];
    stpcvt(pstring, TOUP | RWHITE);
        /* Attempt to find a match in the array of color */
        /* words. */
    for (index = 0;
        (index < sizeof(color_words) / sizeof(color_words[0]));
        index++)
    {
        found = (strcmp(pstring, color_words[index]) == 0);
        if (found)
            break;
    }
    if (!found)
    {
        sprintf(message, "Error: Unrecognized color \"%s\".", pstring);
        abort_error(message);
    }
    return(index);
}
/**/

```

- \* Name            **CONVERT NUMBER** Convert a string to a number.
- \* Synopsis        `number = convert_number(pstring);`
- \*                `int number`            The converted number.
- \*                `char *pstring`        Pointer to the string to be converted.
- \* Description     This function accepts a pointer to a string of characters and converts it into an int.    If the first character is a '\$', then the string is assumed to contain hexadecimal digits; otherwise it is assumed to be decimal. If any digit in the string is not a valid hex or decimal digit (as appropriate), an error message is displayed and the program is aborted.
- \* Returns         `number` The converted number.

```

*/
int convert_number(pstring)
char *pstring;
{
    int index;
    char temp[12];
    char message[81];
    int value;
    strncpy(temp, pstring, sizeof(temp));
    temp[sizeof(temp)] = '\0';
    stpcvt(temp, RLWHITE | RTWHITE);
    if (temp[0] != '$')
    {
        /* Scan the string for non-decimal characters. */
        for (index = 0; temp[index] != '\0'; index++)
            if (!isdigit(temp[index]))
            {
                sprintf(message, "Error: Invalid decimal digit %c.",
                    temp[index]);
                abort_error(message);
            }
        return(atoi(temp));
    }
    else
    {
        /* Scan the string for non-hex characters. */
        for (index = 1; temp[index] != '\0'; index++)
            if (!isxdigit(temp[index]))
            {
                sprintf(message, "Error: Invalid hexadecimal digit %c.",
                    temp[index]);
                abort_error(message);
            }
        scanf(&temp[1], "%x", &value);
        return(value);
    }
}

```

**/\*\***

- \* **Name**            **PROCESS\_COMMAND** Process a command keyword.
- \* **Synopsis**        **process\_command(command\_index, pcommand\_params);**
- \*                    int command\_index        The index of the command.
- \*                    char \*pcommand\_params Pointer to the string of
- \*    parameters for the command.
- \* **Description**    This function accepts the index of a command and a
- \*                    pointer to a string to be used as parameters to the
- \*                    command, and performs the command.
- \* **Returns**        b\_def help rec    Various fields altered.
- \*                    Various global variables affected.

**\*/**

```
void process_command(command_index, pcommand_params)
```

```

int          command_index;
char         *pcommand_params;
{
    int length;
    char message[81];
    switch(command_index)
    {
    case 0:
        /* Window background color. */
        def_help_rec.data_back = color_search(pcommand_params);
        current_attribute = utnybbyt(def_help_rec.data_back,
                                     def_help_rec.data_fore);

    break;
    case 1:
        /* Border background color. */
        def_help_rec.border_back = color_search(pcommand_params);
    break;
    case 2:
        /* Border foreground color, */
        def_help_rec.border_fore = color_search(pcommand_params);
    break;
    case 3:
        /* Border type. */
        def_help_rec.border_type = convert_number(pcommand_params);
    break;

    case 4:
        /* Width of help window's data area. */
        def_help_rec.data_columns = convert_number(pcommand_params);
        datawidth_found = 1;
    break;

    case 5:
        /* Help description string. */
        stpcvt(pcommand_params, RLWHITE | RTWHITE);
        strncpy(help_description, pcommand_params,
                sizeof(help_description));
    break;

    case 6:
        /* Window foreground color. */
        def_help_rec.data_fore = color_search(pcommand_params);
        current_attribute = utnybbyt(def_help_rec.data_back,
                                     def_help_rec.data_fore);
    break;

    case 7:
        /* Help screen id. */
        stpcvt(pcommand_params, RLWHITE | RTWHITE | TOUP);
        strncpy(def_help_rec.id_string, pcommand_params,

```

```

        HL_ID_SIZE);
    def_help_rec.id_string[HL_ID_SIZE - 1] = '\0';
break;

case 8:
    /* Window title. */
    /* Remove leading and trailing white space. */
    stpcvt(pcommand_params, RLWHITE | RTWHITE);
    /* Quotes can be used to add beginning or trailing */
    /* blanks, and are always removed. */

    if ((pcommand_params[0] == SQUOTE) ||
        (pcommand_params[0] == DQUOTE))
    {
        utmovmem(pcommand_params + 1, pcommand_params,
            strlen(pcommand_params));
    }
    length = strlen(pcommand_params) - 1;
    if ((pcommand_params[length] == SQUOTE) ||
        (pcommand_params[length] == DQUOTE))
    {
        pcommand_params[length] = '\0';
    }

    strcpy(window_title, pcommand_params, sizeof(window_title) - 1);
    window_title[sizeof(window_title) - 1] = '\0';
break;

case 9:
    /* Title background color. */
    def_help_rec.title_back = color_search(pcommand_params);
break;

case 10:
    /* Title foreground color. */
    def_help_rec.title_fore = color_search(pcommand_params);
break;

case 11:
    /* Title position and type. */
    def_help_rec.title_type = convert_number(pcommand_params);
    if (def_help_rec.title_type > LAST_TITLE_POS)
    {
        sprintf(message, "Error: Invalid title position %d.",
            def_help_rec.title_type);
        abort_error(message);
    }
break;

case 12:

```



```

        /* X coordinate of right edge of viewport.          */
        def_help_rec.view_right = convert_number(pcommand_params);
        if (utrange(def_help_rec.view_right,
                    def_help_rec.view_left, MAX_VIEW_COLS))
            abort_error("Error: Illegal viewport dimensions.");
        break;

    case 13:
        /* X coordinate of left edge of viewport.          */
        def_help_rec.view_left = convert_number(pcommand_params);
        if (utrange(def_help_rec.view_left, 1, def_help_rec.view_right))
            abort_error("Error: Illegal viewport dimensions.");
        break;

    case 14:
        /* Y coordinate of bottom edge of viewport.        */
        def_help_rec.view_bottom = convert_number(pcommand_params);
        if (utrange(def_help_rec.view_bottom,
                    def_help_rec.view_top, MAX_VIEW_ROWS))
            abort_error("Error: Illegal viewport dimensions.");
        break;

    case 15:
        /* Y coordinate of top edge of viewport.           */
        def_help_rec.view_top = convert_number(pcommand_params);
        if (utrange(def_help_rec.view_top, 1,
                    def_help_rec.view_bottom))
            abort_error("Error: Illegal viewport dimensions.");
        break;

    default:
        sprintf(message,
                "Error: Internal error - unknown command index %d.",
                command_index);
        abort_error(message);
    }
}

```

/\*\*

* Name	CSTR2PAS -- Convert a C string to Pascal format.
* Synopsis	length = cstr2pas(pc_string, ppascal_string,
*	max_length);
*	int length                      Length of Pascal string,
*	including length byte.
*	const char far *pc_string      C string to convert.
*	char far *ppascal_string      Buffer to place Pascal
*	string in.
*	int max_length                Maximum length of Pascal
*	string, including length.
* Description	CSTR2PAS will convert a C-style '\0' terminated string
*	into a Pascal-style string with leading length byte.
*	The maximum number of characters that will be written
*	to ppascal string will be max_length.
* Returns	int length                    The total length of the Pascal string,

```

*                               including the length byte.
**/
int estr2pas(pc_string, ppascal_string, max_length)
const char *pc_string;
char *ppascal_string;
int max_length;
{
    int pascal_length;
    int temp_index;
    max_length = (max_length > 255) ? 255 : max_length;
    pascal_length = strlen(pc_string) + 1;
    pascal_length = (max_length > pascal_length) ? pascal_length
                                                    : max_length;
    for (temp_index = pascal_length - 1; temp_index; temp_index--)
    {
        ppascal_string[temp_index] = pc_string[temp_index - 1];
    }
    ppascal_string[0] = (char) (pascal_length - 1);
    return(pascal_length);
}

```

## 第五章 用 C 进行中断服务子程序的设计

### 概述

最有用的特性之一是它对建立和使用中断服务例程(ISR)的支持。中断服务函数提供了使用硬件中断和软件中断的基本能力,包括激活和建立ISR、检测已安装的ISR、让一个程序在中止之后驻留内存、从内存摘除一个驻留程序等。

### ISR的一般用法

ISGETVEC	通过DOS功能调用返回一个中断向量的内容。
ISPUTVEC	通过DOS功能调用设置一个中断向量。
ISCALL	激活一个给定ISR程序物理地址的ISR,该ISR不必安装在一个中断向量中。一个类型为ALLREG的结构用于传递ISR使用,并返回的机器寄存器的拷贝。

### 建立ISR

ISINSTAL	将一个C函数安装成ISR,该C函数必须作为调用ISINSTAL的程序的一部分链接起来并且必须遵循下面说明的调用步骤。ISINSTAL准备一个ISR控制块,安排ISDISPAT对ISR的进入和退出施行控制。
ISPREP	以与ISINSTAL相同的方式准备一个ISR控制块但不在中断向量中安装ISR。
ISRESERV	为一个ISR内部使用的标准malloc()函数保留空间。
ISSENSE	检测一个给定的Turbo C TOOLS ISR是否是最近安装在给定中断向量中的ISR。

### 驻留程序

ISRESEXT	中止一个程序,但使之驻留在内存中。
ISREMOVE	从内存摘除一个已经中止并驻留的程序。
ISCURPRC	报告或设置DOS当前执行的“前台”进程的记录。

### 调用步骤

每个ISR在声明时带有三个参数,这些参数是指向结构的指针。在大多数情况下这些结构不被ISR修改,但它们包含的信息对于ISR可能是很有用的。

第一个参数是指向ALLREG结构的指针,该结构包含着ISR被激活时CPU所有寄存器内容的拷贝。当ISR退出时,调度程序将利用这个结构的值恢复寄存器。(硬件ISR必须在完成处理后恢复ALLREG结构或在处理过程中根本不改变寄存器,这样才能保存所有的寄存器。)

ALLREG结构在工具函数头文件butil.h中定义如下:

```
typedef struct          /* HALFREGS : 一个16位通用寄存器的两半。 */
{
    unsigned char l;    /* 低8位。 */
    unsigned char h;    /* 高8位。 */
} HALFREGS;
typedef union           /* DOUBLREG : 表达一个16位通用寄存器的 */
/* 两种方法。 */
{
    HALFREGS hl;        /* 独立的的两半。 */
    unsigned x;         /* 16位无符号值。 */
} DOUBLREG;
typedef struct          /* ALLREG : 整个CPU状态 */
```

```

{
DOUBLREG ax,bx,cx,dx; /* 通用寄存器。 */
unsigned si,di; /*索引寄存器。 */
unsigned ds,es,ss,cs; /* 段寄存器。 */
unsigned flags,bp,sp,ip; /* 其它寄存器。 */
} ALLREG;

```

第二个参数是指向ISR控制块的指针，控制块的内容由ISINSTAL或ISPREP进行初始化。调度程序使用和维护控制块中的各个项。ISR不应改变这些项，它们所包含的信息对于ISR是很有用的。

第三个参数是指向类型为ISRMSG的结构体的指针，它用于调度程序和ISR之间的通讯。常规的ISR应该忽略这个结构。

一个标准ISR可以完全忽略它的参数，这样做的唯一效果是当控制返回到调用程序时调用程序的寄存器值将完全不发生改变。事实上，硬件ISR必须使调用程序的寄存器不发生改变，否则会引起严重的问题。

## 删除

全部删除一个程序的步骤如下：

- 用ISSENSE找出驻留程序中的和每一个已安装的ISR；
- 验证已发现的ISR属于同一个程序；
- 通过在中断向量中安装不同的值使这些ISR失效；
- 用ISREMOVE摘除该程序(同时释放它占用的内存)。

## 按惯例过滤中断

用户应该有礼貌地过滤不需完全捕获的中断。下面的示例过滤掉中断8。

```

#include <bintrup.h>
void do_tasks(void);
void cdecl clock_tick(ALLREG *, ISRCTRL *, ISRMSG *);

void cdecl clock_tick(pregs, pistrblk, pmsg)
ALLREG *pregs;
ISRCTRL *pistrblk;
ISRMSG *pmsg;
{
    static int busy = 0;
    iscall(pistrblk->prev_vec, preg); /* 过滤中断 */
    utintflg(UT_INTOFF); /* 对“忙”标志进行维护期间不允许中 */
    if (!busy) /* 断，防止以嵌套方式执行“临界段”。 */
    {
        busy = 1;
        utintflg(UT_INTON);
        do_tasks();
        busy = 0;
    }
}

```

ISRCTRL结构的prev\_vec成员包含着指向一个ISR的far指针，该ISR在原来安装时使用这个中断向量。在上面的示例中，对ISCALL的调用激活以前的ISR，使传给这个ISR的寄存器值与我们的ISR接收的一组寄存器值相同。以前的ISR对这些寄存器值的修改将传递给最初的调用者。

## 从ISR中的特殊退出

DOS技术手册中说Ctrl-Break处理程序(中断类型0x23)和严重错误处理程序(中断0x24)可以按非标

准的方式退出。如果用户想使用这样的技术，下面就是使用的方法。

如果用far RETurn而不是IRET退出，需将ISRMSG结构的exit\_style成员设置为IEXIT\_RETF，如下例所示：

```
#include <bintrupt.h>
void cdecl cbreak(ALLREG *, ISRCTRL *, ISRMSG *);

void cbreak(pregs, pistrblk, pmsg)
ALLREG *pregs;
ISRCTRL *pistrblk;
ISRMSG *pmsg;
{
    extern unsigned long count;
    count++;
    pmsg->working_flags |= CF_FLAG; /* 设置进位标志 */
    pmsg->exit_style = IEXIT_RETF;
}
```

要想在一个用far RETurn返回的ISR中设置进位标志，可以象下面所示的那样修改ISRMSG结构的working\_flags成员。(示例程序CTLBRK就是这样做的。)

如果希望在用IRET返回之前废弃栈项，可以象下面语句那样对ALLREG结构的sp成员做增量操作：

```
pregs->sp += 24; /* 废弃12个堆栈层 */
```

上面所列的方法都是微妙和高级的技术。要想了解更加详细的情况，请参见isdispat.asm源代码和有关的说明。

## 扩展函数源程序

这是一份珍贵的中断子程序设计的资料，结合后面的例子，用户可用C而不一定要用汇编进行编写中断服务子程序。

## ISCALL 对软件中断调用中断服务例程进行模拟

```
#include <bintrupt.h>
int iscall( void far *pistr,
            ALLREG *preg);

pistr      指向中断服务例程的far指针。
preg       指向ALLREG结构的指针，该结构包含着机器寄存器的拷贝。(ALLREG结构在“中断服务(IS)”中说明)。
(返回)     错误代码：
            0 正常
            1 pistr是FARNIL.
```

ISCALL 激活其地址为pistr的中断服务例程，以最通用的方式返回结果。所有在\*preg中指定的值在对应的寄存器(除了CS、IP、SS和SP)中传递给中断服务例程，所有作为结果的寄存器内容在\*preg中返回(同样，除了CS、IP、SS和SP)。

中断状态(即中断是否允许)被保留。

注意，用AH寄存器中的值0x4b激活中断0x21时要留心，因为这个中断服务(装入或执行一程序的DOS功能)有重要的限制。

在许多情况下，调用ISCALL之前应该把值DEF\_FLAGS装入preg->flags中(符号DEF\_FLAGS在butil.h中定义，butil.h被包含在bintrupt.h中)，否则混乱的标志值可能导致中断被屏蔽或使处理器进入单步方式。见下面的前两个示例。

源程序(ISCALLASM):

```

        include beginasm.mac
        beginProg iscall
ERR BADVEC equ    1                ; Error code indicating invalid
                                      ; interrupt vector.

pisr    equ    dword ptr [bp + stkoff]
        if     LONGDATA
preg     equ    dword ptr [bp + stkoff + 4]
        else
preg     equ    word ptr [bp + stkoff + 4]
        endif

ALLREG struc                        ; ALLREG structure
reg_ax   dw     ?                  ; (This must match declaration in
reg_bx   dw     ?                  ; BUTIL.H).
reg_cx   dw     ?
reg_dx   dw     ?
reg_si   dw     ?
reg_di   dw     ?
reg_ds   dw     ?
reg_es   dw     ?
reg_ss   dw     ?
reg_cs   dw     ?
reg_flags dw     ?
reg_bp   dw     ?
reg_sp   dw     ?
reg_ip   dw     ?
ALLREG ends

        push    bp
        mov     bp,sp
        push    si                  ; Save registers that are important
        push    di                  ; to the C environment.
        push    ds
        push    es

                                      ; Obtain interrupt vector
        les     si,pisr
        assume es:nothing           ; Now ES:SI is address of interrupt
                                      ; handler.
        mov     ax,es               ; Quit if pisr is 0:0.
        or      ax,si
        jnz     ok_vector

bad_vector:
        mov     ax,ERR BADVEC
        jmp     short exit

ok_vector:
                                      ; Obtain contents of preg.
        if     LONGDATA
        lds     bx,preg

```

```

assume ds:nothing
else
mov    bx,preg
endif

; Now DS:BX points to *preg
; (ALLREG structure is at DS:BX).

; Begin setting up for the call:
push    ds    ; Step 1:  Save stack frame pointer
push    bp    ; so we can restore it in Step 10.

pushf     ; Step 2:  Save our own flags
           ; for restoration in Step 9.

; Step 3:  Push the stack items that
; the ISR will see:  a copy of its
; flags and its return address
; (CS:ret_addr).

mov     ax,[bx.reg_flags] ; Push the flags
push    ax                ; without clearing IF and TF.
push    cs                ; Push CS.
mov     cx,offset ret_addr
push    cx                ; Push ret_addr.

; Step 4:
; We will enter the handler by doing
; an IRET, so push the flags and the
; vector.

and     ah,0fch           ; Push the flags
push    ax                ; with IF and TF cleared.
push    es                ; Push the vector.
push    si

mov     ax,[bx.reg_ax]    ; Step 5:  Load the easy registers.
mov     cx,[bx.reg_cx]
mov     dx,[bx.reg_dx]
mov     si,[bx.reg_si]
mov     di,[bx.reg_di]
mov     es,[bx.reg_es]
assume es:nothing
mov     bp,[bx.reg_bp]

push    [bx.reg_bx]       ; Step 6:  Load DS and BX
mov     ds,[bx.reg_ds]    ; by temporarily using the stack.
assume ds:nothing
pop     bx

iret                     ; Step 7:  Load the flags and jump

```

```

; to the handler. (This IRET uses &
; removes what was pushed in Step 4.)

ret_addr:
    assume es:nothing,ds:nothing
; The interrupt handler brought us here
; by performing an IRET, thus popping
; what was pushed in Step 3.
; (However, the interrupt handler
; may choose to modify the flags
; before popping them.)

    push    bp
    push    ds
    push    bx
    pushf
; Step 8: Temporarily preserve
; returned values of BP, DS, BX,
; and flags.

    mov     bp,sp
    mov     bx,[bp+8]
    push    bx
    popf
; Step 9: Restore our own flags
; (which were pushed in Step 2).
; (They may contain special values
; required by the C environment).

    lds     bp,[bp+10]
    assume ds:nothing
; Step 10: Restore stack frame pointer
; (pushed in Step 1) so we can find preg.
; Obtain contents of preg.

    if      LONGDATA
    lds     bx,preg
    assume ds:nothing
    else
    mov     bx,preg
    endif

; Now DS:BX points to *preg
; (ALLREG structure is at DS:BX).

    mov     [bx.reg_ax],ax
    mov     [bx.reg_cx],cx
    mov     [bx.reg_dx],dx
    mov     [bx.reg_si],si
    mov     [bx.reg_di],di
    mov     [bx.reg_es],es
    pop     [bx.reg_flags]
    pop     [bx.reg_bx]
    pop     [bx.reg_ds]
    pop     [bx.reg_bp]
    add     sp,6
; Transfer values pushed in Step 8
; to the *preg structure.

    xor     ax,ax
; Discard values pushed in
; Steps 2 and 1.
; Success code.

exit:
; Now AX contains the error/success code.
; Restore registers that are important
; to the C environment.

    cld
    pop     es

```



```

pop    ds
pop    di
pop    si
pop    bp
ret
endProg iscall
end

```

## ISCURPRC 返回或设置当前执行的进程

```
#include <bintrupt.h>
```

```
unsigned iscurprc( int option,
                  unsigned newproc);
```

option      IS\_SETPROC(1)设置当前进程, IS\_RETPROC (0)报告一个进程。

newproc     如果指定了IS\_SETPROC, newproc是新的当前进程的程序段前缀的段地址。

(返回)      原进程的程序段前缀的段地址。

ISCURPRC返回(也许修改)当前执行进程的程序段前缀(PSP)的DOS记录。

通常前台程序(用户在DOS提示符下激活的程序或那个程序的最近子进程)是当前执行进程, 被打开的文件和DOS功能0x48所分配的内存块均属于当前进程。后台程序可以用ISCURPRC(中止并驻留的程序)使自己暂时成为当前程序, 以便可靠地执行它的功能和其它的DOS功能。

在DOS 3.00或更高版本中, 插入码(函数名前缀为IV)自动执行这个任务。

对于DOS版本2, 在DOS忙期间请不要使用ISCURPRC (见UTDOSRDY)。

源程序(ISCURPRC.C):

```

#include <dos.h>
#include <bintrupt.h>
unsigned iscurprc(option,newproc)
int option;
unsigned newproc;
{
    unsigned result;
    union REGS regs;
    /* DOS function 0x62 (Return PSP) is available (and documented) */
    /* for DOS versions 3.00 and later. Function 0x51 is available, */
    /* but not documented, for all versions of DOS, so prefer */
    /* function 0x62 when it is available. */
    regs.h.ah = (unsigned char) ((utdosmajor < 3) ? 0x51 : 0x62);
    intdos(&regs,&regs);
    result = regs.x.bx;
    if (option == IS_SETPROC)
    {
        regs.h.ah = 0x50; /* DOS function 0x50: Set current PSP */
        regs.x.bx = newproc; /* (an undocumented function). */
        intdos(&regs,&regs);
    }
    return result;
}

```

## ISGETVEC 返回一个中断向量

```
#include <dos.h> /* ISGETVEC需要 */
#include <bintrupt.h>
void far *isgetvec(int intype);
intype      中断类型号。
(返回)      保存在向量中的双字值。
ISGETVEC返回与指定的中断类型关联的中断向量，保存在向量中的双字值由ptr返回。
ISPUTVEC设置一个中断向量。
ISGETVEC是用宏实现的。
```

源程序(BINTRUPT.H):

该函数为宏，定义于BINTRUPT.H。请见附录C。

### isdispat 中断服务子程序调度程序。

该子程序不能从用户程序中调用。设计用来在ISR控制块的开始的进行远程调用。

Blaise的C TOOLS的中断服务子程序(ISRs)由ISINSTAL安装时为其设置的ISR控制块控制。当ISR安装后，该ISR的控制块的地址放进相应的中断向量。在每个控制块的第一项是一个这个子程序的远调用。因此当中断发生时，控制块的下一地址压进堆栈，并调用ISDISPAT。

然后该子程序执行调用用户用C编写的ISR所需的其余步骤：

- (1) 保存所有的寄存器。
- (2) 查找ISR控制块并使用其中的信息。
- (3) 确定是否为ISR的调用立一个新的堆栈。如果不建立堆栈就返回。
- (4) 设置和使用新的堆栈。
- (5) 制作ISR的参数的复本，包括含有中断发生时机器寄存器值的ALLREG结构。
- (6) 象通常的C函数一样调用ISR。
- (7) 恢复在ALLREG结构中的寄存器值。
- (8) 恢复原来的堆栈。
- (9) 放弃为ISR调用建立的堆栈，以及由不返回的插入码调用余留的堆栈。
- (10) 从ALLREG结构中恢复机器寄存器值。
- (11) 执行由ISR选择的IRET或RETF指令。

中断是否允许的中断状态保持不变，除非ISR要求改变。

返回	rcode	错误码： 如果成功为0，如果中断向量无效，则为1。
	*preg	由中断处理子程序改变后的寄存器值。

源程序(ISDISPAT.ASM):

```
include    beginasm.mac
beginMod  isdispat
if TC100
; Rely on macros defined in Turbo C's header file
; to add leading underscore (if any) on external
; names.

if LONGDATA
ExtSym@ _stklen,word,notpascal      ; Stack growth limit.
STK_GROWTH_LIMIT@ _stklen@
else
ExtSym@ _brklvl,word,notpascal      ; Stack growth limit.
STK_GROWTH_LIMIT@ _brklvl@
endif
```

```

endif ; TC100
if      MSC500
extrn   STKHQQ: word
STK_GROWTH_LIMIT equ STKHQQ
endif
beginCseg isdispat
public   _isdispat
_isdispat proc      far
ISRCTRL struc      ; ISRCTRL: ISR control block
                   ; (This must match declaration in
                   ; BINTRUPT.H.)
                   ;
icb_fcall_opcode    dw ?      ; NOP + Far call opcode
                   ; (0x9A90)
icb_isrdisp         dd ?      ; Address of ISR dispatcher
icb_iret_opcode     dw ?      ; IRET + RETF opcodes (0xcbcf)
                   ; (Offset of icb_iret_opcode
                   ; is on stack on entry to ISDISPAT.)
                   ;
icb_isrstk_r        dw ?      ; ISR stack region:   offset
icb_isrstk_s        dw ?      ;                      segment
icb_isrstksize      dw ?      ; ISR stack size
icb_isrsp           dw ?      ; ISR SP value at start of
                   ; current ISR call
                   ;
icb_isrds           dw ?      ; DS value required by ISR
icb_isrres          dw ?      ; ES value required by ISR
icb_isr             dd ?      ; Address of ISR itself
icb_isrpsp          dw ?      ; PSP of program containing ISR
                   ;
icb_prev_vec        dd ?      ; Previous value of vector
                   ;
icb_level           dw ?      ; Number of calls in progress
                   ; (0 if not in progress)
icb_limit           dw ?      ; Maximum number of nested calls
icb_signature       dw ?      ; Signature identifying this
                   ; structure
icb_sign2           dw ?      ; One's complement of
                   ; "signature"
icb_ident           db 16 dup (?) ; Identifying name
icb_control         dw ?      ; Bit fields to control
                   ; dispatcher options
icb_status          dw ?      ; Status info left by
                   ; dispatcher
icb_scratch         db 10 dup (?) ; Scratch space for use by
                   ; dispatcher & related programs
ISRCTRL ends

; Control bits in icb_control member of ISR control block.
; These must match BINTRUPT.H.

```

```

ICB_NOCHECK equ 1 ; Don't adjust stack growth limit.
; Exit style codes -- these must match BINTRUPT.H.
IEXIT_NORMAL equ 0
IEXIT_RETF equ 1
ALLREG struc ; ALLREG structure
reg_ax dw ? ; (This must match declaration in
reg_bx dw ? ; BUTIL.H).
reg_cx dw ?
reg_dx dw ?
reg_si dw ?
reg_di dw ?
reg_ds dw ?
reg_es dw ?
reg_ss dw ?
reg_cs dw ?
reg_flags dw ?
reg_bp dw ?
reg_sp dw ?
reg_ip dw ?

allreg_size db ?
ALLREG ends

; ***** BEGIN CODE HERE *****
    assume ds:nothing,es:nothing,ss:nothing
; Save initial register values so we can have some workspace.
    push bp ; Save original BP (used only here >- \ )
    mov bp,sp ;
    push bp ; (Entry value of SP) - 2
    push [bp] ; Original BP <- /
    push ax ; Original AX
    push bx ; Original BX
    push cx ; Original CX
    push dx ; Original DX
    push ds ; Original DS
    pushf ; Original flags
    push es ; Original ES
    push si ; Original SI
    add word ptr [bp-2],6 ; Convert entry SP-2 to "Caller's SP"
                        ; thus ignoring the first BP we pushed
                        ; and segment and offset pushed by
                        ; the far call in the control block.
; Set up DS:BX to address these values we just saved on caller's stack.
    mov bx,ss
    mov ds,bx
    assume ds:nothing
    mov bx,bp
                        ; Current stack contents
                        ; (high addresses to low):
                        ;

```

```

caller_flags equ word ptr [bx+10] ; Caller's flags
caller_cs     equ word ptr [bx+8]  ; Caller's CS
; "Caller's SP" points to . . .
caller_ip     equ word ptr [bx+6]  ; Caller's IP (see note below)
ret_seg       equ word ptr [bx+4]  ; Segment of ISR Control Block
; Entry SP pointed to . . .
ret_offset    equ word ptr [bx+2]  ; Offset of icb_iret opcode in
; ISR Control Block
; BX and BP point to . . .
; Original BP (won't be used again)
caller_sp     equ word ptr [bx-2]  ; "Caller's SP value"
orig_bp       equ word ptr [bx-4]  ; Original BP again
orig_ax       equ word ptr [bx-6]  ; Original AX
orig_bx       equ word ptr [bx-8]  ; Original BX
orig_cx       equ word ptr [bx-10] ; Original CX
orig_dx       equ word ptr [bx-12] ; Original DX
orig_ds       equ word ptr [bx-14] ; Original DS
orig_flags    equ word ptr [bx-16] ; Original flags
orig_es       equ word ptr [bx-18] ; Original ES
orig_si       equ word ptr [bx-20] ; Original SI

```

```

; Note: the value called "caller's SP" is the SP value
; immediately after the INT instruction that invoked
; this interrupt. Any adjustments to "caller's SP" should
; be RELATIVE (e.g., to discard some stack items just before
; returning to caller, let the ISR add a fixed value to SP in
; the ALLREG structure).

```

```

; Set up ES:SI to point to beginning of ISR control block.

```

```

    mov     es,ret_seg
    assume es:nothing
    mov     si,ret_offset
    sub     si,(icb_iret_opcode-icb_fcall_opcode)

```

```

; Make sure nesting depth isn't over limit.

```

```

    mov     ax,es:[si].icb_level
    cmp     ax,es:[si].icb_limit
    jb      level_ok
    jmp     exit ; Nested too deep: just exit.
; Check that stack size is adequate.

```

```

; Generate a new stack for the ISR: increment level & icb_isrsp.

```

```

level ok: ; Nesting depth is okay.
    inc     ax ; Increment nesting depth.
    cli
    mov     es:[si].icb_level,ax
    push    ax ; Local copy of nesting depth
; (will be popped later).
    mov     ax,es:[si].icb_isrsp; Former ISR stack pointer.
    mov     dx,ax
    add     ax,es:[si].icb_isrstksize ; Raise stack pointer
    mov     es:[si].icb_isrsp,ax ; for a new stack.

```

```

; Switch to the ISR's own stack (which we just generated).

```

```

; Interrupts must be disabled during this.

```

```

mov    ax,sp                ; Save old SP (old SS is in DS register)
if TC100 and LONGDATA
                                ; Adjust SS and SP so that SP is as small
                                ; as possible. This is necessary
                                ; for stack checking.
                                ; DX contains low address of this ISR's
                                ; stack.
add    dx,15                ; Round DX up to next multiple of 16
mov    cx,4                ; bytes. Convert to paragraph number.
shr    dx,cx
add    dx,es:[si].icb_isrstk_s ; Compute first paragraph that
                                ; begins within this ISR's stack.
                                ; This will be used for SS.
mov    cx,es:[si].icb_isrstksize ; Use stack size for SP value.
sub    cx,16                ; but allow 16 bytes for safety
                                ; because SS value was adjusted
                                ; upward (wasting up to 15 bytes).

mov    ss,dx
mov    sp,cx
mov    dx,cx                ; Stack limit is high address of
                                ; current stack. (Stack overflow
                                ; occurs if stack pushes below 0.)
else
                                ; No special SP value required.
mov    ss,es:[si].icb_isrstk_s ; Switch to ISR's own stack.
assume ss:nothing
mov    sp,es:[si].icb_isrsp
add    dx,6                ; For stack limit, use high address of
                                ; former ISR stack (plus 6 for safety).
endif
push    orig_flags          ; Restore original flags to
popff                                ; (possibly) re-enable interrupts.
; Now create an ALLREG structure on the ISR's own stack.
push    caller_ip
push    caller_sp
push    orig_bp
push    caller_flags
push    caller_cs
push    ds                  ; Caller's SS
push    orig_es
push    orig_ds
push    di
push    orig_si
push    orig_dx
push    orig_cx
push    orig_bx
push    orig_ax
mov    bp,sp                ; Now BP points to ALLREG structure.
regs    equ    [bp]
; Create two-word area for messages between dispatcher (us) and ISR.
push    orig_flags          ; "Working" flags -- depending on how

```

```

; dispatcher exits, these may persist
; on return to caller.

mov     cx, IEXIT_NORMAL
push    cx                ; Default exit style.
message equ     dword ptr [bp-4]
; Create local vector to the ISR
if      LONGPROG
push    word ptr es:[si].icb_isr+2
push    word ptr es:[si].icb_isr
vector_to_isr equ     dword ptr [bp-8]
else
push    word ptr es:[si].icb_isr
vector_to_isr equ     word ptr [bp-6]
endif
; Save items needed locally by dispatcher (us)
push    ds                ; Caller's SS
push    ax                ; SP in caller's stack just before
; stack switch
push    bx                ; Frame pointer in caller's stack.
push    es                ; Segment of ISR control block.
push    si                ; Offset of ISR control block.

; Point to ISR's data segment
mov     ds, es:[si].icb_isrds
assume ds:nothing
; Save former stack growth limit and establish new value therefor.
; (The stack growth limit variable is in the ISR's data segment.)
test    es:[si].icb_control, ICB_NOCHECK
jnz     stack_limit_adjusted
if LONGDATA
push    es
mov     ax, seg STK_GROWTH_LIMIT
mov     es, ax
mov     ax, es:STK_GROWTH_LIMIT
mov     es:STK_GROWTH_LIMIT, dx    ; New limit.
pop     es
push    ax                    ; Save old limit.

else
push    ds:STK_GROWTH_LIMIT    ; Save old limit.
mov     ds:STK_GROWTH_LIMIT, dx ; New limit.
endif

stack_limit_adjusted:

; Save control bits (in part so we will know whether to restore
; stack growth limit).

push    es:[si].icb_control
; Push ISR's arguments.

```

```

    if    LONGDATA
    push  ss                      ; Address of message area.
    lea   ax,message
    push  ax

    push  es                      ; Address of ISR control block.
    push  si

    push  ss                      ; Address of ALLREG structure.
    lea   ax,regs
    push  ax

    else

    lea   ax,message              ; Address of message area.
    push  ax

    push  si                      ; Address of ISR control block.
                                ; (For S, P, and M model programs
                                ; this must be in ISR's default
                                ; data segment.)

    lea   ax,regs                 ; Address of ALLREG structure.
    push  ax

    endif

;   Set up remaining registers required by ISR.
    cld
    mov   es,es:[si].icb_isres
    assume es:nothing

;   Actually call the ISR.
    call  vector_to_isr

;   Discard the ISR's arguments in usual C fashion.
    if    LONGDATA
    add   sp,12
    else
    add   sp,6
    endif

;   Recall former control bits into CX.
    pop   cx                      ; CX = former control bits.

;   Restore former stack growth limit.
    test  cx,ICB_NOCHECK
    jnz   stack_limit_restored
    if    LONGDATA
    mov   si,seg STK_GROWTH_LIMIT
    mov   es,si
    pop   es:STK_GROWTH_LIMIT

```



```

        else
        pop    ds:STK_GROWTH_LIMIT
        endif
stack_limit_restored:
;   Restore dispatcher's various pointers.
        pop    si
        pop    es                ; Now ES:SI point to ISR control block.
        assume es:nothing
        pop    bx
        pop    ax
        pop    ds                ; DS:AX is SS:SP in caller's stack
        assume ds:nothing        ; just before stack switch.
                                    ; DS:BX is frame pointer in caller's
                                    ; stack.
;   Discard the local vector to the ISR.
        if    LONGPROG
        add    sp,4
        else
        inc    sp
        inc    sp
        endif
;   Read the message area & react to it.
        pop    dx                ; "Exit style"
        pop    orig_flags        ; "Working flags"
;   Transfer ALLREG structure values into proper registers or into
;   holding locations.   These values may have been modified by the ISR.

        pop    orig_ax
        pop    orig_bx
        pop    orig_cx
        pop    orig_dx
        pop    orig_si
        pop    di
        pop    orig_ds
        pop    orig_es
        inc    sp                ; Discard SS value from structure.
        inc    sp
        pop    caller_cs
        pop    caller_flags
        pop    orig_bp
        pop    caller_sp
        pop    caller_ip

;   Switch back to caller's stack.
        mov    cx,ds
        cfi
        mov    ss,cx
        assume ss:nothing
        mov    sp,ax
                                    ; Now both DS and SS refer to

```

```

; caller's stack segment.
; Leave interrupts disabled.
; Discard the ISR stack created above, i.e. reset ich_level and ich_isrsp.
;
; If ich_level exceeds our local (stacked) copy of level, then
; this ISR invocation returned BEFORE a deeper-nested invocation returned.
; In this case, assume that deeper invocations will never return.
; Adjust ich_level and ich_isrsp to discard deeper stacks.
;
; Interrupts are NOT okay during this operation.

    mov     cx,es:[si].ich_level
    pop     ax                ; Local copy of nesting depth
    dec     ax
    mov     es:[si].ich_level,ax
    sub     cx,ax            ; CX = number of stacks to discard
                                ; (normally 1)
    jbe     stacks_discarded
    mov     ax,es:[si].ich_isrsize
discard_stack:
    sub     es:[si].ich_isrsp,ax    ; Decrement beginning SP once
    loop    discard_stack          ; for each stack discarded.
stacks_discarded:
                                ; Interrupts now okay.

; Consider optional exit style
    cmp     dx,EXIT_RUUF
    je      retf_exit
                                ; If exit style code not recognized.
                                ; fall through to normal exit.

; Normal exit: restore everything and do IRET.
;
; Some of the values we restore were copied from the ALLREG structure,
; so they may have been modified by the ISR.
exit:
    pop     si
    pop     es
    popff                                ; Install "original" flags.
                                ; These may have been modified via
                                ; the "WORKING_FLAGS" item in the
                                ; message area.

    pop     ds
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    pop     bp
    pop     sp                    ; Unless modified by ISR,
                                ; SP now contains the value
                                ; it had after the caller

```

```

; performed an INT.

    iret
; Special exit: restore everything and do a RETF.
;
; Some of the values we restore were copied from the ALLREG structure,
; so they may have been modified by the ISR.

retf exit:
    pop     si
    pop     es
    popf    ; Install "original" flags.
            ; These may have been modified via
            ; the "working flags" item in the
            ; message area.

    pop     ds
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    pop     bp
    pop     sp ; Unless modified by ISR,
            ; SP now contains the value
            ; it had after the caller
            ; performed an INT.

    ret

isdispat endp
endCseg isdispat
end

```

## ISINSTAL 安装一个中断服务例程(ISR)

```

#include <bintrupt.h>
int isinstal(int intype,
void cdecl(*pfunc)( ALLREG*, ISRCTRL *, ISRMSG *),
const char *pident,
ISRCTRL *pisrblk,
char *pstack,
int stksize,
int stknum);

```

**intype** 中断类型号。  
**pfunc** 中断服务例程地址。  
**pident** 标识信息16字节数组地址。  
**pisrblk**ISR 控制块(已经分配)的地址。  
**pstack** 该ISR(已经分配)使用的堆栈内存块的起始地址。  
**stksize** ISR堆栈(以字节计)的尺寸,必须至少为128字节。典型的尺寸是2048字节。  
**stknum** 该ISR的嵌套激活的最大深度(至少为1)。  
**(返回)** 错误返回代码:  
0 没有错误;  
1 **intype** 超出了0-255的范围。

ISINSTAL以指定的类型在中断向量中安装一个中断服务例程(ISR)，包括：(1)完成所有在ISR控制块中的值；(2)把ISR控制块地址保存在中断向量中。

当(2)完成时，甚至在ISINSTAL返回它的调用者以前，ISR立即生效(可以被激活)。

调用函数必须为ISR控制块分配空间，用pisrblk把该空间的地址传递过去。在ISR的使用过程中，这个空间必须是静态的。

调用函数也必须分配ISR的堆栈所要用的空间，空间地址在pstack中传递，并且必须至少为(stksize\*stknum)字节长。在ISR的使用过程中，这个空间必须是静态的。如果ISR在T、S或M模式的Turbo C程序中使用near堆栈，或在C、L、H模式的Turbo C TSR(中止并驻留)程序中使用far堆栈，则需要用ISRESERV来保留malloc()所用到的空间。

stknum是这个ISR的嵌套激活的最大深度。如果一个激活发生了并超过最大的深度，则ISR调度程序只是返回而不调用TSR，这对于IBM PC上的硬件ISR可能是有害的，因为EOI不会再次发送到Int 8259 可编程中断控制器了。

十六字符标识信息保存在ISR控制块中，以便于其它程序能够检查中断向量并检测ISR的存在。(参见示例issense.c.)

用ISPREP可以预备一个ISR控制块而并不实际上在一个中断向量中安装完整的ISR。用IVINSTAL可以安装一个插入函数。

源程序(IFINSTAL.C):

```
#include <dos.h> /* For getvect(), setvect(). */
#include <bintrupt.h>
#define NO_ERROR 0
#define BAD_INTYPE 1
typedef char CHAR16[16]; /* Aid for using utcopy */
int isinstal(intype,pfunc,pident,pisrblk,pstack,stksize,stknum)
int intype;
void (*pfunc)(ALLREG *,ISRCTRL *,ISRMSG *);
const char *pident;
ISRCTRL *pisrblk;
char *pstack;
int stksize,stknum;
{
    int ints_were_on;
    if (utrange(intype,0,255))
        return BAD_INTYPE;
    isprep(pfunc,pident,pisrblk,pstack,stksize,stknum);
    /* Now load the interrupt vector with the segment and offset */
    /* address of the ISR control block. */
    ints_were_on = utintflg(UT_INTOFF);
    pisrblk->prev_vec = isgetvec(intype);
    isputvec(intype,((char far *) pisrblk) + ICB_ENTRY_OFFSET);
    /* Setting the vector completes */
    /* the installation. */
    if (ints_were_on)
        utintflg(UT_INTON);
    return NO_ERROR;
}
```

ISPREP 预备一个ISR控制块

```
#include <bintrupt.h>
```

```
void isprep( void cdecl (*pfunc) (ALLREG *,ISRCTRL *,ISRMSG *),
            const char *pident,
            ISRCTRL *pisrblk,
            char *pstack,
            int stksize,
            int stknum);
```

pfunc      中断服务例程地址  
 pident     标识信息16字节数组地址。  
 pisrblk    ISR控制块(已分配)地址。  
 pstack    该ISR(已分配)的堆栈使用手内存块的起始。  
 stksize    ISR堆栈的尺寸(以字节计), 必须至少为64字节。典型尺寸是2048字节。  
 stknum    该ISR的嵌套激活的最大深度(必须至少为1)。

ISPREP为安装中断服务例程预备一个(ISR)控制块, 这包括通过ISR调度程序(ISDISPAT)执行ISR所必须的一切操作。

调用函数必须为ISR控制块分配空间, 用pisrblk传递空间的地址。在ISR工作过程中这个空间必须是静态的。

调用函数也必须为ISR的堆栈分配空间, 空间的地址必须用pstack传递并且必须至少为(stksize\*stknum)字节长。在ISR工作过程中, 这个空间必须是静态的。如果ISR在T、S或M模式的Turbo C程序中使用near堆栈, 或在C、L、H模式的Turbo C TSR(中止并驻留)程序中使用far堆栈, 则用ISRESERV保留malloc()所需的空间。

stknum是该ISR嵌套激活的最大深度。如果一个激活发生了并超过了最大深度, 则ISR调度程序只是返回而不调用ISR, 这对于IBM PC上的硬件ISR可能是有害的, 因为EOI不能发送到Intel8259可编程控制器上。

十六字节标识信息保存在ISR控制块上, 使得其它程序能够检查控制块, 侦测ISR的存在。(参见示例issense.c)。

ISINSTAL将预备一个ISR控制块并安装这个ISR合为一步。IVINSTAL安装一个插入函数。

源程序(ISPREP.C):

```
#include <dos.h>
#include <stdlib.h>
#include <string.h> /* For memcpy(), strcmp(). */
#include <interrupt.h>
typedef char CHAR16[16]; /* Aid for using utcopy */
static unsigned stack_limit_unsafe(void);
void isprep(pfunc,pident,pisrblk,pstack,stksize,stknum)
void (*pfunc)(ALLREG *,ISRCTRL *,ISRMSG *);
const char *pident;
ISRCTRL *pisrblk;
char *pstack;
int stksize,stknum;
{
    int i;
    extern void isdispat(void); /* ISR dispatcher */
    struct SREGS segregs;
    pisrblk->fcall_opcode = 0x9a90; /* NOP + far call opcodes */
    pisrblk->isrdisp = (void (far *)()) isdispat;
    pisrblk->iret_opcode = 0xc9cf; /* IRET + RETF opcodes */
    pisrblk->isrstk = pstack; /* Beginning of stack region */
    pisrblk->isrstksize = stksize; /* Size of stack region */
    /* Initial SP at bottom of */
}
```

```

pisrblk->isrsp    = utoff(pisrblk->isrstk);    /* stack region */
segread(&segregs);
pisrblk->isrds    = segregs.ds;
pisrblk->isres    = segregs.es;
pisrblk->isr      = (void (far *) (ALLREG *,
                                ISRCTRL *,
                                ISRMSG *)) pfunc;

pisrblk->isrsp    = utpspseg;
pisrblk->level    = 0;
pisrblk->limit    = stknum;
pisrblk->sign2    = ~(pisrblk->signature = ICB_SIGNATURE);
utcopy(pisrblk->ident,pident,CHAR16);
pisrblk->control  = stack_limit_unsafe();
pisrblk->status   = 0;
for (i = 0; i < sizeof(pisrblk->scratch); i++)
    pisrblk->scratch[i] = 0;
pisrblk->prev_vec = FARNIL;    /* Clear pointer to previous */
                                /* interrupt handler to prevent */
                                /* calling it before the ISR is */
                                /* really installed. */

return;
}
static unsigned stack_limit_unsafe()
{
    unsigned result;
#ifdef  defined( _TURBOC_ ) \
    && (defined( _SMALL_ ) || defined( _MEDIUM_ ) || defined( _TINY_ ))
    char buffer[5];    /* Check Turbo C copyright year.*/
    utpeekn(utofaru(utseg(&_psp),27),buffer,4);
    buffer[4] = '\0';
    if (0 == strcmp(buffer,"1987"))
        result = 0;    /* Safe to adjust stack limit. */
    else
        result = ICB_NOCHECK;    /* Don't adjust stack limit. */
#else
    result = 0;    /* Safe to adjust stack limit. */
#endif
    return result;
}

```

## ISPUTVEC 设置一个中断向量

```

#include <dos.h>    /* ISPUTVEC 需要.*/
#include <intrpt.h>
void isputvec(int intype,
void far *ptr);
intype    中断类型号。
ptr       准备装载到向量中的四字节值。
ISPUTVEC  把指定类型中断向量设置成*ptr所指明的地址。

```

用ISGETVEC可以取得一个中断向量的内容，用ISINSTAL则能够安装Turbo C TOOLS中断服务

例程。

ISPUTVEC是用宏实现的。

源程序(BINTRUPT.H):

该函数在头文件BINTRUPT.H中说明为宏。请见附录C。

## ISREMOVE 去除一个驻留程序

#include <bintrupt.h>

int isremove(unsigned psp\_seg);

psp\_seg 驻留程序的程序段前缀的段地址。

(返回) MMCTRL返回的错误:

0 成功。

7 内存控制块被破坏: 用户程序向内存中的某个不适当的位置做了写入操作。

9 非法的内存块: 也许psp\_seg不属于一个驻留程序。

ISREMOVE从内存中删除一个已经中止但保持驻留的程序(也许通过ISRESEXT)。

警告: 删除程序之前首先要废弃它包含的所有中断处理程序, 也许要通过ISPUTVEC来重置它们的向量。否则, 如果有一个中断被引发, 控制有可能传到一个不存在的例程中。

源程序(ISREMOVE.C):

```
#include <dos.h> /* For intdosx(), REGS, and */
/* SREGS. */
```

```
#include <bintrupt.h>
```

```
static int mm_free(unsigned); /* Internal function (see */
/* below). */
```

```
int isremove(unsigned psp_seg)
```

```
{
```

```
    unsigned memblock,nextblock;
```

```
    int result;
```

```
    MEMCTRL_t block;
```

```
    result = 9; /* In case first block address */
/* is bad, return 9 (Invalid */
/* memory block). */
```

```
    /* Begin memory block search at first memory block. */
```

```
    for (memblock = mmfirst();
```

```
        memblock != 0;
```

```
        memblock = nextblock)
```

```
    {
```

```
        if ( 9 != mmctrl(memblock,&block,&nextblock) /* If block okay */
```

```
            && block.owner_psp == psp_seg) /* and belongs to */
```

```
        { /* psp_seg. */
```

```
            result = mm_free(memblock); /* then free the block. */
```

```
            if (result != 0)
```

```
                break; /* Quit if error. */
```

```
        }
```

```
    }
```

```
    return result;
```

```
}
```

```

/**
 * Name          mm_free -- Release an allocated memory block
 * Synopsis      ercode = mm_free(seg);
 *               int ercode          Returned error code (0 if okay).
 *               unsigned seg        Segment address of memory block.
 * Description    This function frees memory blocks that have been
 *               allocated by DOS function 0x48. The segment address
 *               given must be associated with a previously allocated
 *               block.
 * Returns       ercode              DOS function return code.
 **/
static int mm_free(seg)
unsigned seg;
{
    union REGS regs;
    struct SREGS sregs;
    regs.h.ah = 0x49;
    sregs.es = seg;
    intdosx(&regs,&regs,&sregs);
    return ((regs.x.cflag) ? (regs.h.al)          /* Nonzero error code. */
           : (0));                               /* Success: return 0. */
}

```

**ISRESERV** 保留ISR所需的动态内存。

```

#include <bintrupt.h>
int isreserv(    size_t reserve,
                 size_t blksize,
                 char **ppblock);

```

**reserve** 需保留的ISR所用空间的尺寸(以字节计)。

**blksize** 通过malloc()分配的块尺寸。

**ppblock** 指针的地址, 该指针由通过malloc()分配的块的地址来填充, 或失败时用NULL填充。

(返回) 成功代码。可能值包括:

IS\_OK (0) 成功。

IS\_NO\_MEMORY (1) 无足够内存。

IS\_RANGE (2) reserve或blksize越界。

ISRESERV通过malloc()分配一块空间, 它是以这样一种方式进行的: reserve声明的空间的大小可以满足以后ISR或插入函数的分配需要。

在T、S和M内存模式的Turbo C中, 用户应该将分配之后的块作为堆栈, 用于ISR或插入函数的堆栈。在C、L和H内存模式的Turbo C中, 用户可以在任何需要的地方放置ISR堆栈或插入函数堆栈。blksize可以小到1。

size\_t在标准头文件stdlib.h中定义。

源程序(ISRESERV.C):

```

#include <stdlib.h> /* For malloc(), free(), and size_t. */
#include <bintrupt.h>
int isreserv(reserve,blksize,ppblock)
size_t reserve,blksize;
char **ppblock;
{

```



```

void *p_reserve,*ptrial,*pprev;
/* We may allocate many blocks of size "blksize", and in each one */
/* store the address of its predecessor, so we can clean up. */
/* Therefore each block must be large enough to contain a pointer; */
/* "trial_size" is the size we must use. */
size_t trial_size = max(blksize,sizeof(void *));
*ppblock = NULL; /* In case of premature error. */
if (blksize <= 0 || reserve <= 0) /* Range check. */
    return IS_RANGE;

/* Allocate reserved block. */
if (NULL == (p_reserve = malloc(reserve)))
    return IS_NO_MEMORY;
/* Search for a block of size "blksize" until we find one at a
/* higher address than the reserved block or until no more memory */
/* is available. In each block, save a pointer to predecessor. */
pprev = NULL;
while ( NULL != (ptrial = malloc(trial_size))
        && utplong(ptrial) <= utplong(p_reserve))
{
    * (void **) ptrial = pprev;
    pprev = ptrial;
}
*ppblock = ptrial; /* Return NULL if no success
/* before we ran out of memory; */
/* otherwise return successful */
/* trial block. */
while (NULL != (ptrial = pprev)) /* Free the chain of
/* temporary blocks. */
{
    pprev = * (void **) ptrial;
    free(ptrial);
}
free(p_reserve); /* Free the reserved block. */
return (*ppblock ? IS_OK : IS_NO_MEMORY);
}

```

## ISRESEXT 中止一个程序但保持驻留

```

#include <dos.h> /* ISRESEXT需要。 */
#include <bintrupt.h>
void isresext(int excode);
excode 设置的退出码(0到255)。

```

ISRESEXT关闭所有打开的文件(不使用C库函数),中止当前程序,设置退出码,程序保持驻留在内存中(是DOS的一部分)。

父程序可以利用DOS函数0x4d检查退出码或利用批命令ERRORLEVEL在command.com级上检查。如果子程序被某一个spawn...函数执行,则可以得到作为函数值的退出码。退出码必须不大于255并且不小于0。

从实际上讲,任何一个驻留内存的程序都应该尽可能小。

用ISREMOVE可以从内存摘除一个驻留程序。

ISRESEXT是用宏实现的。

源程序(BINTRUPT.C):

参见附录C, 该功能定义为宏。

## ISSENSE      检测一个已安装的中断服务例程(ISR)

#include <bintrupt.h>

ISRCTRL far \*issense(void far \*ptr,

const char \*pident);

ptr            指向可能的ISR 入口点的far指针。

pident        标识信息的16字节数组的地址。

(返回)        发现的ISR控制块的地址。若未发现, 返回FARNIL。

ISSENSE检查所给内存位置的内容, 看看是否有一个特定的Blaise ISR控制块。如果发现一个ISR控制块, ISSENSE返回该控制块的地址。

本函数只能检测最近安装的指定类型的ISR。

如果发现了ISR, 用ISPUTVEC可以把中断向量的值恢复为pctrl->prev\_vec。在检测到所有ISR并使之无效之后, 用ISREMOVE可以从内存中删除一个驻留程序。IVSENSE则能够查知Turbo C TOOLS插入码。

源程序(ISSENSE.C):

#include <dos.h>

#include <string.h>

#include <bintrupt.h>

ISRCTRL far \*issense(ptr,pident)

void far    \*ptr;

const char \*pident;

{

    unsigned int offset;

    ISRCTRL far \*ptest;

    int            i;

    if (ptr == FARNIL)

        return FARNIL;

    offset = utoff(ptr);

    /\* Make sure that the control block does not straddle a segment \*/

    /\* boundary. Notice that this computation requires that \*/

    /\* sizeof(ISRCTRL) be at least 2 (which it is). \*/

    if ( offset < ICB\_ENTRY\_OFFSET

        || offset >= (unsigned) 0xffff - sizeof(ISRCTRL) + 2)

        return FARNIL;

    /\* If this vector points to a Turbo C TOOLS, C TOOLS PLUS \*/

    /\* or LIGHT TOOLS interrupt service routine, then the \*/

    /\* actual item pointed to is at offset ICB\_ENTRY\_OFFSET in an \*/

    /\* ISRCTRL structure. \*/

    ptest = uttofar (utseg(ptr),offset - ICB\_ENTRY\_OFFSET,ISRCTRL);

    /\* Check for signature word in ISRCTRL structure. \*/

    if (ptest->signature != ICB\_SIGNATURE)

        return FARNIL;

    /\* Confirm signature by checking its one's-complement. \*/

    if (ptest->sign2 != (unsigned) ~ICB\_SIGNATURE)

        return FARNIL;

    /\* Check identifying "string". \*/

```

for (i = 0;
     i < sizeof (ptest->ident)) &&
    (ptest->ident[i] == pident[i]);
    i++;
;
if (i < sizeof (ptest->ident))
    return FARNIL;
/* Fall through: all tests passed, this is the desired ISR. */
return ptest;
}

```

## 使用中断子程序的程序设计

ISCLOCK.C用来在屏幕上显示不断更新的时间。

程序在屏幕的右上角显示时钟。运行该程序的所有支持皆由中断支持子程序(IS)提供。

ISCLOCK从每次计时中断中取得控制,但每十次显示一次时间。(稍微比半秒少一些)

ISCLICK从BIOS中以时钟为单位获得时间。

命令行格式:

islock [-r]

ISCLOCK是自删除服务程序。如果调用ISCLOCK时没有参数则安装ISCLOCK;如果调用时有“-r”参数,则从内存中删除。

ISCLOCK用来展示中断服务的功能,并提供一个中断服务代码的结构和使用的例子。

```

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <intrupt.h>
#include <bscreens.h>
#include <bvideo.h>

/* The function which will be called every timer */
/* tick. */
void intime (ALLREG *,ISRCTRL *,ISRMSG *);
#if (IS_NUM_FLOAT_VECS > 0)
/* Storage for floating point vectors. */
static void far *float_vec [IS_NUM_FLOAT_VECS];
#endif
#define STKSIZE 2000
/* Exit values. */
#define FALL_OUT 101
#define OK 0
#define NOT_FOUND 1
#define ERROR_REMOVING 3
#define ALREADY_INSTALLED 4
#define TIMER_VEC 0x08
/* Definitive signature for this version of ISLOCK.*/
char *islock_ident = "ISLOCK 03/31/89";
/* Allocation of stack space for the intervention */
/* scheduler and user function. */
char isstack [STKSIZE];
/* Control block for ISR dispatcher. */
ISRCTRL isrctrl;
/* Internal functions -- install & remove ISR. */

```

```

int install_is (void);
int remove_is (void);
void main(int, char **);

```

```

void main (argc, argv)

```

```

int  argc;

```

```

char *argv [];

```

```

{
    if (argc == 1)
        exit (install_is ());
    if ((argc == 2)
        (((argv [1][0] == '.') || (argv [1][0] == '/')) &&
         ((argv [1][1] == 'r') || (argv [1][1] == 'R'))))
        exit (remove_is ());
    printf ("usage: isclock [-r]\n");
    exit (0);
}

```

```

/**

```

INSTALL\_IS -- 安装ISCLOCK的中断矢量，终止并驻留。

ret = install\_is ();

int ret 当 安装中断服务子程序时出现问题时从ISINSTAL中返回错误码。

如果没有安装ISCLOCK则安装之。

返回 ret

```

**/

```

```

int install_is ()

```

```

{
    int i, ret;
    void far *timer_vec = isgetvec (TIMER_VEC);
    if (issense (timer_vec, isclock_ident) != FARNIL)
    {
        puts ("ISCLOCK already installed");
        return (ALREADY_INSTALLED);
    }
    #if (IS_NUM_FLOAT_VECS > 0)
        /* Save floating point vectors. */
        for (i = 0; i < IS_NUM_FLOAT_VECS; i++)
            float_vec[i] = isgetvec (IS_1ST_FLOAT_VEC + i);
    #endif
    /* Install the interrupt service routine. */
    if ((ret = isinstal (TIMER_VEC, intime, isclock_ident,
                        &isrctrl, isstack, STKSIZE, 1)) != 0)
    {
        /* Error! */
        printf ("ISINSTAL error %d.\n", ret);
        return (ret);
    }
    /* Terminate and stay resident. */
    isresext (OK);
    /* Should never get here. */
}

```

```

    return (FALL_OUT);
}

/**
    REMOVE IS -- 删除先前安装的ISCLOCK。
    ret = remove_is ();
    int ret          返回码
                     NO_ERROR(0)-
                     没有发现代码。
                     NOT_FOUND (1)-
                     ISCLOCK未找到。
                     ERROR_REMOVING (3)-
                     ISCLOCK不能 被删除(由于MALLOC指针被重写)。
    函数删除内存中活动的ISCLOCK，清除中断矢量，释放内存。
    返回    ret (如果错误为非零)。
    **/
int remove_is ()
{
    ISRCTRL far *pisrctrl;
    void far *timer_vec = isgetvec (TIMER_VEC);
    /* Check to see if ISCLOCK installed. */
    if ((pisrctrl = issense (timer_vec, isclock_ident)) == FARNIL)
    {
        puts ("ISCLOCK not found.");
        return (NOT_FOUND);
    }

    /* Restore the old interrupt vector. */
    isputvec (TIMER_VEC, pisrctrl->prev_vec);
    if (isremove (pisrctrl->isrsp))
    {
        puts ("Error removing ISCLOCK.");
        return (ERROR_REMOVING);
    }
    return (0);
}

/**
    INTIME -- 在屏幕上显示当前时间的数字。
    (只由中断服务调用程序调用)。
    intime (pallreg, pisrctrl, pisrmsg);
    ALLREG *pallreg      指向CPU寄存器的内容的结构。
    ISRCTRL *pisrctrl     指向ISR调度程序。
    ISRMSG *pisrmsg      未用。
    函数每次18.2次的调度程序中接收控制，显示当前时间的数字(每调用18次显示一次)，并退出。
    返回    无
    **/

void intime (pallreg, pisrctrl, pisrmsg)
ALLREG *pallreg;
ISRCTRL *pisrctrl;

```

```

ISRMSG *pisrmsg;
{
    static int    ticks = 0;
    long          elpsec, time;
    int           mode, act_page, columns;
    int           hours, mins, secs;
    char          time_str[10];
    #if (IS_NUM_FLOAT_VECS > 0)
        void far   *save_float [IS_NUM_FLOAT_VECS];
    #endif

        /* Get rid of compiler warning message. */
    hours = *((int *) pisrmsg);
        /* Call previous owner of the timer vector first. */
    iscall (pisrctrl->prev_vec, paddr);
        /* If we have enough ticks, get time from BIOS tick */
        /* counter (via UTGETCLK), and display it. */
    if (++ticks >= 18)
    {
        ticks = 0;
    #if (IS_NUM_FLOAT_VECS > 0)
        /* Save current floating point vectors. */
        utintflg (UT_INTOFF);
        utpeeka (uttfar (0, 4*IS_1ST_FLOAT_VEC, char), save_float,
            sizeof (save_float));
        /* Reinstall our own floating point vectors. */
        utpoken (float_vec, uttfar (0, 4*IS_1ST_FLOAT_VEC, char),
            sizeof (float_vec));
        utintflg (UT_INTON);
    #endif

        /* Get the current mode and active display page */
        /* from BIOS. */
        scmode (&mode, &columns, &act_page);
        /* Direct our display output to the current display */
        /* page. */
        scpage (act_page);
        /* Get current time of day from BIOS. */
        /* Figure out number of seconds past midnight from */
        /* (given) number of timer ticks since midnight. */
        utgetclk (&time);
        /* Figure out number of seconds past midnight from */
        /* (given) number of timer ticks since midnight. */
        elpsec = (long) (0.0549254 * (double) time);
        /* Figure out hours, minutes, seconds from seconds */
        /* past midnight. */
        hours = (int) (elpsec / 3600L);
        mins = (int) ((elpsec / 60L) % 60L);
        secs = (int) (elpsec % 60L);
        /* Display the time. */
        sprintf (time_str, "%02d:%02d:%02d", hours, mins, secs);
        vidspmsg (0, columns - strlen (time_str),

```

```

        NORMAL | INTENSITY, SC_BLACK, time_str);
#if (IS_NUM_FLOAT_VECS > 0)
        /* Restore floating point vectors. */
        utintflg (UT_INTOFF);
        utpoken (save_float, uttofar (0, 4*IS_1ST_FLOAT_VEC, char),
                sizeof (save_float));
        utintflg (UT_INTON);
#endif
    }
}

```

## CTLBRK.C 说明 control-break 处理子程序的实现和使用

程序用来说明 control-break(int 0x23)处理程序实现和使用。程序在在屏幕显示一提示,并使用户按 CTRL C 或 CTRL BREAK。当按这些键时 control-break 子程序被触发,并显示一信息。

命令行格式:

```

    ctlbrk
#include <stdio.h>
#include <bfiles.h>
#include <bintrupt.h>
#include <bcreens.h>
#include <bvideo.h>
        /* Vector 0x23 is the one to intercept. */
#define CBREAK_VEC 0x23
        /* Size of the CTRL-BREAK interrupt handler stack. */
#define STKSIZE 1500
        /* Declaration of the CTRL-BREAK handler. */
void cbreak (ALLREG *,ISRCTRL *,ISRMSG *);
int main()
{
        /* ISR control block for the handler. */
    ISRCTRL    cbrk_ctl;
        /* Allocate stack space for handler execution. */
    static char    cbrk_stack[STKSIZE];
    int          err;
    char          resp[2];
        /* String uniquely identifying this interrupt handler. */
    static char    cbrk_ident[] = "CTLBRK 03/31/89";
        /* Install Ctrl-Break handler, then prompt user to try it. */
    err = isinstal (CBREAK_VEC, cbreak, cbrk_ident,
        &cbrk_ctl, cbrk_stack, STKSIZE, 1);
    if (err != 0)
        return (err);
    printf ("Press CTRL-C or CTRL-BREAK to try interrupt handler,\n");
    fflush (stdout);
    fprompt ("or press ENTER to quit: ", resp, sizeof (resp));
        /* We do not have to re-install the previous INT 0x23 handler because DOS does so automatically on program termination. */
}

```

```

    return (0);
}
/**
    cbreak -- Ctrl break中断服务子程序。
    该函数被安装成中断0x23的处理子程序。使用BIOS或直接I/O(不使用DOS函数)显示信息，然后
    用RETF返回DOS，使程序结束。
**/
void cbreak (pregs, pisrblk, pmsg)
ALLREG *pregs;
ISRCTRL *pisrblk;
ISRMSG *pmsg;
{
    int mode, columns, act_page;
    int row, col, junk;

        /* Get rid of compiler warning messages. */
    junk = *((int *) pregs);
    junk = *((int *) pmsg);
    junk = *((int *) pisrblk);
    scmode (&mode, &columns, &act_page);
    scpage (act_page);
    scurst (&row, &col, &junk, &junk);
    vidspmsg (row, col, -1, -1, "<<Ctrl-C detected>>");
    pmsg->working_flags |= CF_FLAG;
    pmsg->exit_style     = IEXIT_RETF;
}

```

## CRITERR.C 说明严重错误处理子程序和使用

程序用来说明严重错误(into x 24)处理子程序的实现和使用。

程序在屏幕上显示一提示让用户打开A驱动器，然后按ENTER。程序设法打开“A:\GARBAGE”文件。这将产生一个严重错误(into x 24) 中断处理子程序用来报告错误，并使DOS进行失败操作的重试。

命令行格式如下：

```

    criterr
#include <dos.h>
#include <stdio.h>
#include <bfiles.h>
#include <bintrupt.h>
#define CRIT_VEC 0x24
#define STACKSIZE 3000          /* Items for the critical */
#define NUMSTACKS 2            /* error ISR. */
ISRCTRL crit_blk;
char crit_stk[STACKSIZE*NUMSTACKS];
void crit_err(ALLREG *,ISRCTRL *,ISRMSG *);
void write_str(char *);
void wait_key(void);
void main()
{
    char resp[2];
    FILE *file;

        /* Install DOS critical error handler. */
}

```



```

isinstal (CRIT_VEC, crit_err, "TCT CRITER 5.00",
          &crit_blk, crit_stk, STACKSIZE, NUMSTACKS);
          /* Display message, wait for ENTER. */
fprompt ("Open A: drive door, then press ENTER to test handler: ",
         resp, sizeof (resp));
          /* Now try to open (for reading) a file on drive A: */
file = fopen ("a:\\garbage", "r");
if (file != NULL)
    fclose (file);
          /* We do not have to re-install the previous */
          /* INT 0x24 handler because that happens */
          /* automatically on program termination. */
}
/**
crit_err——处理DOS严重错误
(只从ISR调度程序中调用。)
crit_err(pregs,pisrblk,pmsg);
ALLREG *pregs    指向发生中断时保留的寄存器项的结构体的指针
ISRCTRL *pisrblk 指向ISR控制块体的指针
ISRMSG *pmsg     指向ISR和调度程序之间传递信息的结构
函数分析DOS在机器寄存器中提供的信息，报告信息和错误发生的设备，并等待击一键，返回DOS
进入重试操作。
返回          preg->ax.hlj 值为1，要求重试。
**/
void crit_err (pregs, pisrblk, pmsg)
ALLREG *pregs;
ISRCTRL *pisrblk;
ISRMSG *pmsg;
{
    DEV_HEADER dh;
    char report[100];
    int bad_fat;
    static char *message[] =
    {
        "Attempt to write on write-protected diskette", /* 0x00 */
        "Unknown unit", /* 0x01 */
        "Drive not ready", /* 0x02 */
        "Unknown command", /* 0x03 */
        "Data error (CRC)", /* 0x04 */
        "Bad request structure length", /* 0x05 */
        "Seek error", /* 0x06 */
        "Unknown media type", /* 0x07 */
        "Sector not found", /* 0x08 */
        "Printer out of paper", /* 0x09 */
        "Write fault", /* 0x0a */
        "Read fault", /* 0x0b */
        "General failure", /* 0x0c */
        "Unknown error 13", /* 0x0d */
        "Unknown error 14", /* 0x0e */
        "Invalid disk change" /* 0x0f */
    }

```

```

};
static char *area[4] =          /* Area on disk in which error */
{                                /* may occur. */
    "DOS area (system files)",
    "file allocation table",
    "directory",
    "data area"
};

    /* Get rid of compiler warning messages. */
bad_fat = *((int *) pmsg);
bad_fat = *((int *) pbsrbk);
bad_fat = 0;
if (pregs->ax.hi.h & 0x80)
{
    /* Character device or FAT error. */
    /* Get the device header into local storage. */
    utpeekn (utfatar (pregs->bp, pregs->si, char), &dh, sizeof (dh));
    /* If it is a character device, print its name. */
    if (dh.attr & 0x8000)
        sprintf (report,
            "\015\012<<Critical error on device \%-8s\":>>",
            dh.name);
    else /* It must be a block device FAT error. */
    {
        bad_fat = 1;
        sprintf (report, "\015\012<<Bad FAT on drive %c:>>",
            pregs->ax.hi.l + 'A');
    }
}
else /* Disk error. */
    sprintf(report,
        "\015\012<<Critical error while %s %s on drive %c:>>",
        (pregs->ax.hi.h & 1) ? "writing" : "reading",
        area[(pregs->ax.hi.h & 6) >> 1],
        pregs->ax.hi.l + 'A');
write_str (report);
write_str ("\015\012");
write_str (message[utlobyte(pregs->di)]);
write_str ("\015\012");
write_str (report);
if (bad_fat)
    write_str("\015\012<<BAD FAT -- Turn the computer off!\007\015\012>>");
else
{
    write_str ("\015\012Correct condition and press any key.\015\012");
    wait_key ();
    pregs->ax.hi.l = 1; /* Retry. */
}
}
/**
write_str ——只能使用DOS 1-12号服务向DOS控制台写字串，忽略CTRL_BREAK。

```

write\_str(pstring);

char \*pstring 指向包含写信息的字符数组头的指针。通常该字符串以NUL('\0')终止。

该函数只使用DOS 1-12号服务向DOS控制台写字符串，忽略CTRL\_BREAK和CTRL\_C。

返回 无

\*/

void write\_str(pstring)

char \*pstring;

```
{
    union REGS regs;
    char ch;
    regs.x.ax = 0x0600;
    while ((ch = *pstring++) != '\0')
    {
        if (ch != (char) 0xff)
        {
            regs.x.dx = utbyword (0, ch);
            int86 (FL_DOS_INT, &regs, &regs);
        }
    }
}
```

\*/

wait\_key - 等待击键并放弃(只使用DOS 1-12号服务。

wait\_key();

该函数放弃已键入的键，等待另一击键，并放弃之。如果新键是CTRL-BREAK或CTRL\_C，程序被终止。

返回 无。函数返回类型为void。

\*/

void wait\_key()

```
{
    union REGS regs;
    regs.x.ax = 0x0c08; /* Flush keyboard & await one key. */
    int86 (FL_DOS_INT, &regs, &regs);
    /* If IBM PC extended character, read & discard the */
    /* second byte of the keystroke. */
    if (utlobyte (regs.x.ax) == 0)
    {
        regs.x.ax = 0x0600;
        regs.x.dx = 0x00ff;
        int86 (FL_DOS_INT, &regs, &regs);
    }
}
```

## 第六章 内存驻留程序设计的插入码

### 概述

插入码使建造可靠的驻留实用程序变得十分简便。它执行所有必要的调度操作,用户可以省略中断服务例程(ISR)所需的大部分复杂工作。换句话说,插入码使得 ISR 的使用更为简便。

用户利用插入码可以调度他的程序,使该程序在一个热键被按下,到达一指定的时刻或规律性地经过一个时间间隔时被激活。用户可以自由使用系统服务,包括 DOS 功能和浮点运算,在安装函数时只需声明它的特殊需要即可。仅当一个函数所需要的服务具备时插入码才会激活这个函数。

### 插入码实用函数

IVINSTAL	安装用户的插入函数。
IVDISABL	通过重新安装中断向量原值的方法使一个插入函数失效,这些中断向量为插入过滤程序所使用。
IVSENSE	检测已安装的插入码是否可摘除。
IVDETECT	检测已安装的中断码的存在,即使它的一些插入过滤程序被其它中断处理程序覆盖。
IVCTRL	返回插入控制块的地址。
IVVECS	读取或安装插入过滤程序使用的一组中断向量。

### 建立一个插入函数

#### 调度

用户设计应用程序时必须决定激活插入函数的事件。事件有两种类型:按键事件或计时器事件。

按键事件由一个 IV\_KEY 结构数组指明,每个结构指定一个热键。IV\_KEY 结构在插入码头文件 binterv.h 中定义如下:

```
typedef struct          /* IV_KEY: 指定一个热键 */
{
    char ch;             /* 字符值。 */
    char keycode;        /* 键码(伪扫描码)。 */
    int action;          /* 探测到这个键时所采取的动作(见下面的可能值) */
} IV_KEY;
```

IV\_KEY 结构的 action 成员可以是下列值之一:

```
#define IV_KY_NONE      0 /* 无热键按下。 */
#define IV_KY_SERVICE   1 /* 激活插入函数。 */
#define IV_KY_SLEEP     2 /* 使插入函数睡眠。 */
#define IV_KY_WAKE      3 /* 唤醒插入函数。 */
```

(IV\_KY\_SLEEP 动作可以防止在一个具有 IV\_KEY\_WAKE 动作的热键按下之前激活插入函数。)

定义热键时还要考虑扩展键盘的新按键。例如,如果一个热键是数字键盘上的 Ctrl-Home,这时也应该把扩展键盘的小键盘上的 Ctrl-Home 定义为一个热键,使这两个按键具有相同的作用。

计时器事件通过 IV\_TIME 结构数组指定,每个数组指明一个间隔或一天中的某个时刻。IV\_TIME 结构在 binterv.h 中定义如下:

```
typedef struct          /* IV_TIME: 指明用于插入码的一天中的 */
{                      /* 某个时刻或时间间隔。 */
    long ticks;        /* 自午夜以来或两次激活之间的计时脉冲数。 */
    int action;        /* 时间事件的类型(某时刻或每隔一段间隔)。 */
} IV_TIME;
```

IN\_TIME 结构的 action 成员可以是下列值之一:

```
#define IV_TM_INTERVAL      1    /* 激活之间的规律性间隔。 */
#define IV_TM_MOMENT        2    /* 一天中的某个时刻。 */
```

IV\_TM\_MOMENT 事件通常仅激活一次, 除非这些事件的 24 时制时间被修改(见下面的“重新调度”)。另外, 在调用 IVINSTAL 时指定 IV\_DAILY 任选项将使得瞬时事件每天在适当的时候重复发生。(UTTIM2TK 对于把 24 时制时间转换为计时脉冲计数是很有帮助的。)

## 调用步骤

声明插入函数时带有一个参数, 该参数是指向一个类型为 IV\_EVENT 结构的指针, 这个结构描述插入函数的硬件触发事件(一天中的某个时刻或热键)。一个程序可以只有一个插入函数, 所以这个函数必须处理所有引起调度的事件。

IV\_EVENT 在 binterv.h 中定义如下:

```
typedef struct /* IV_EVENT: 插入函数的触发事件。 */
{
    IV_KEY key; /* 发现的热键(如果存在)。 */
    int time action; /* 计时器事件的类型: 瞬时或间隔(如果存在)。 */
    int time_index; /* 表中计时器事件的标号。 */
    long time; /* 一天中的当前时间, 以自午夜以来的时钟脉冲
               /* 单位。 */
} IV_EVENT;
```

## 重新调度

用户在安装了插入函数之后也许想修改这个函数的按键事件或计时器事件表。要做到这一点最简单的方法就是直接修改原来的事件表。当发生按键事件时, 用户也可以修改插入控制块的 pkeytab 和 ktsize 成员。(用 IVCTRL 可以取得插入控制块的地址。)不论采用哪种方法都要注意通过 UTINTFLG 关闭中断。

计划在一天内的指定时刻(IV\_TM\_MOMENT)发生的计时器事件通常只被激活一次, 除非用户修改了这些事件的发生时间。另外, 在调用 IVINSTAL 时指定 IV\_DAILY 将使瞬时事件在每天的适当时刻重复发生。

## 摘除

将插入过滤程序所用的中断向量恢复为它们的原值可以使插入函数失效。下面是在安装了插入函数的程序内部实现这一点的方法:

```
#include <bintrupt.h>
indisabl(ivctrl());
摘除整个程序的步骤如下:
用 IVSENSE 查找驻留程序中的插入过滤程序;
用 ISSENSE 查找驻留程序中的常规 ISR;
验证发现的插入过滤程序中的常规 ISR;
通过 IVDISABL 使插入过滤程序失效;
通过在 ISR 的中断向量中设置不同值的方法使每个 ISR 失效;
用 ISREMOVE 摘除驻留程序(并释放它占用的内存)。
```

## 高级用法

### 使用扩展键盘 BIOS

(请回顾“键盘(KB)”中有关扩展键盘的说明。)

按照缺省, 如果扩展键盘 BIOS 功能可用, 插入键盘过滤程序(IVKEYDB)即可使用这些功能。无论

用户程序的其它部分是否使用扩展 BIOS，这种缺省对于在 IBM 环境下的所有应用程序都应该是很合适的。

如果希望 IVKEYBD 仅使用传统的 BIOS 服务，可在调用 IVINSTAL 之前执行下面的语句：

```
#include <binterv.h>
#include <bkeybrd.h>
b_inusex = KB_USE_NORMAL;
```

### 暂时使一个插入函数失效

用户利用将插入控制块的 no\_call 成员设置为非零值的方法可以使一个插入函数失效而不必将它从内存摘除，如下例所示：

```
#include <binterv.h>
inctrl()->no_call = 1;
```

这个方法比只是提早从插入函数中退出更为有效，因为这个方法能够防止插入调度程序在从中断 8 (计时脉冲) 返回时被调用。要想使插入码重新有效，只需将 no\_call 成员置为 0。

### 选择插入过滤程序

按照缺省，插入码自动使用五个过滤程序：IVDIS、IVKEYBD、IVTIMER、IVODS 和 IVIDLE，这五个程序控制何时激活插入函数。IVINSTAL 安装这些过滤程序，IVSENSE 检测它们，而 IVDISABL 则能够把它们摘除。Turbo C TOOLS 有两个可选的不常用的插入过滤程序：IVCOM 和 IVCOM2。

全局变量 b\_ivmask 列出了被 IVINSTAL、IVSENSE 和 IVDISABL 所处理的过滤程序。下面是 b\_ivmask 所使用的位值：

```
#define IV_F_TIMER           0x0001 /* INT 08h : 计时器脉冲。 */
#define IV_F_KEYSTROKE      0x0002 /* INT 09h : 按键。 */
#define IV_F_DISK           0x0004 /* INT 13h : BIOS 磁盘服务。 */
#define IV_F_DOS            0x0008 /* INT 21h : DOS 功能。 */
#define IV_F_IDLE           0x0010 /* INT 28h : DOS 空闲。 */
#define IV_F_2COM           0x0020 /* INT 0ch : Comm 口 1 */
#define IV_F_1COM           0x0040 /* INT 0bh : Comm 口 2 */
#define IV_STD_FILTERS
    (IV_F_TIMER | IV_F_KEYSTROKE | IV_F_DISK | IV_F_DOS | IV_F_IDLE)
#define IV_COM_FILTERS (IV_F_2COM | IV_F_1COM)
```

b\_ivmask 的缺省值是 IV\_STD\_FILTERS，它指明了五个标准过滤程序。

当使插入函数失效以摘除过滤程序时，通常应该使装载 b\_ivmask 的值与调用 IVINSTAL 时 b\_ivmask 所具有的值相同。这个值记录在插入控制块的 filter\_mask 成员中。(详见 IVDETECT 中的示例。)

### 防止异步通讯拥塞

两个可选的插入过滤过程(IVCOM1 和 IVCOM2)被设计用于防止异步通讯口上的拥塞错误。要想使用某一个或同时使用这两个过滤程序，用户必须象上述的“选择插入过滤程序”那样修改全局变量。为在两个异步口上加上保护，可在调用 IVINSTAL 之前执行下列操作：

```
#include <binterv.h>
b_ivmask |= IV_F_1COM | IV_F_2COM;
```

当使插入函数失效以摘除适当的过滤程序时，要记住使装载 b\_ivmask 的值与调用 IVINSTAL 时 b\_ivmask 所具有的值相同。

### 插入码扩展码源程序

插入码的应用见示例一节。

**IVCTRL**            报告本程序中插入控制块的地址

```
#include <binary.h>
```

```
IV_CTRL far *ivctrl(void);
```

(返回) 插入控制块的地址。

IVCTRL 返回插入控制块的地址。插入控制块是一个类型为 IV\_CTRL 的结构，用于插入过滤程序、调度程序和 IVINSTAL 之间的通信。它位于代码段中。

用户很少需要直接访问插入控制块，IVINSTAL 执行所有必要的操作来安装插入过滤程序。然而一旦插入机制出错，检查一下结构中的下列这些域也许很有帮助：

```
awake      kb_busy
cbreak     kb_ext
com1_busy  key_event
com2_busy  no_call
disk_busy  req
idle_busy  req
idle_busy  timer_busy
idle_safe  wait
filter_mask
```

源程序(IVCTRLASM):

```
include beginasm.mac
beginMod ivctrl
.....

; Define symbols and macros

; Hot key action codes:
IV_KY_NONE equ 0 ; No hot key or action.
IV_KY_WAKE equ 3 ; Invoke intervention function and
; awaken it if asleep.

BEGIN_IV_FILTER macro ; BEGIN_IV_FILTER: Macro
local begin_label ; to build pointer to
; the IV_CTRL structure
jmp short begin_label ; at the beginning of each
; filter.
dw offset begin_block
begin_label:
endm

IV_VECTORS struc ; IV_VECTORS: set of interrupt vectors
; used by intervention code.
; (Must match version in BINTERV.H exactly!)
ptimer dd 0 ; INT 08h: Timer tick.
pkeybd dd 0 ; INT 09h: Keystroke.
pdisk dd 0 ; INT 13h: BIOS disk services.
pdos dd 0 ; INT 21h: DOS functions.
pidle dd 0 ; INT 28h: DOS idle.
;
pcom1 dd 0 ; INT 0ch: First asynchronous communication
; port.
pcom2 dd 0 ; INT 0bh: Second asynchronous communication
```

```

; port.
IV_VECTORS ends
;*****
;
; Allocate and initialize the IV_CTRL structure.
beginCseg ivctrl
;Note: the following sequence of thirty variables must exactly
; match the IV_CTRL structure as defined in BINTERV.H.
begin_block equ $
signature dw 0 ; Signature identifying this structure.
sign2 dw 0 ; One's complement of "signature".
ident db 16 dup (0) ; Identifying name.
psp dw 0 ; PSP of this program.
enabled dw 0 ; Nonzero if enabled (this prevents the
; filters from being installed twice).
pgate dd 0 ; Gateway for invoking this program
; from outside.
prev_vec IV_VECTORS <> ; Previous values of interrupt vectors.
;
dos_need db 0 ; Nonzero if intervention function
; uses DOS services.
dkey_need db 0 ; Nonzero if intervention function
; uses DOS functions 1-12.
;
pkeytab dd 0 ; Table of hot keys.
ktsize dw 0 ; Size of hot key table.
;
keyfound dw 0 ; Found key, if any.
keyaction dw IV_KY_NONE ;
;
; Internal state of filters:
;
iv_wait db 0 ; Nonzero if INT 21h or INT 28h filter
; is cycling, waiting for the
; intervention function to be invoked.
req db 0 ; Nonzero if intervention function
; needs invocation.
awake db 0 ; Zero if intervention function is
; asleep.
;
; Busy flags:
timer_busy db 0 ; INT 08h filter.
kb_busy db 0 ; INT 09h filter.
com2_busy db 0 ; INT 0bh filter.
com1_busy db 0 ; INT 0ch filter.
disk_busy db 0 ; INT 13h filter.
idle_busy db 0 ; INT 28h filter.
;
idle_safe db 0 ; Nonzero if INT 28h is in progress
; and DOS functions are available

```



```

; (but not functions 1-12).
;
pschedblk      dd 0      ; Entry point into scheduler's ISR
; control block.
;
pdos_crit      dd 0      ; Address of DOS critical section flag
; (busy flag).
;
prev_16h_vec   dd 0      ; Address of previous INT 16h handler
; (if replaced).
cbreak         db 0      ; Nonzero if Ctrl-C or Ctrl-Break
; detected during this invocation of
; the intervention function.
no_call        db 0      ; Nonzero if user wants to prevent
; invocation of intervention function.
filter_mask    dw 0      ; Map of installed filters.

kb_ext         db 0      ; Nonzero if extended BIOS keyboard
; services to be used.
;
res0           db 0      ; Reserved for future use.
;
reserved       db 8 dup (0) ; Reserved for future use.
; End of IV_CTRL structure.

```

```

;*****
;
; Internal variables

```

```

timer_caller_offset dw ?      ; Return address for INT 08h.
timer_caller_seg     dw ?
kb_caller_offset     dw ?      ; Return address for INT 09h.
kb_caller_seg        dw ?
com2_caller_offset   dw ?      ; Return address for INT 0bh.
com2_caller_seg      dw ?
com1_caller_offset   dw ?      ; Return address for INT 0ch.
com1_caller_seg      dw ?
disk_caller_offset   dw ?      ; Return address for INT 13h.
disk_caller_seg      dw ?
key_caller_offset    dw ?      ; Return address for INT 16h.
key_caller_seg       dw ?
key_caller_flags     dw ?      ; Flags pushed by caller on stack.
key_entry_flags      dw ?      ; Flags on entry to IVBIOSKY.
key_func_number      db ?      ; AH value on entry to INT 16h.
key_busy             db 0      ; Busy flag for INT 16h filter.
dos_caller_offset     dw ?      ; Return address for INT 21h.
dos_caller_seg       dw ?
dos_save_flags       dw ?      ; Flags pushed by INT 21h caller.
idle_caller_offset   dw ?      ; Return address for INT 28h.
idle_caller_seg      dw ?

```

```

;*****

```

```

;
; Begin code for IVCTRL function.
beginProc    ivctrl
    assume    ds:nothing, es:nothing, ss:nothing
    mov       dx,cs                ; Load address of intervention
    mov       ax,offset begin_block ; control block into DX:AX.
    ret
endProc      ivctrl
;*****
;
; IVTIMER -- INT 8 filter
;
; This function calls the scheduler on every clock tick,
; except that it never does so in a nested fashion,
; so it is reentrant.
beginProc    ivtimer                ; (IVTIMER will perform a far return.)
    assume    ds:nothing, es:nothing, ss:nothing
    BEGIN_IV_FILTER
    pushf
    cli                                ; Disable interrupts until we can
                                      ; set the busy flag.
    cmp       timer_busy,0           ; If this filter is already active,
    jne       timer_jump_prev        ; If timer busy, jump to previous
                                      ; handler.
    cmp       no_call,0              ; If no calls wanted, jump to
    jne       timer_jump_prev        ; previous handler.
    cmp       com1_busy,0            ; If COM1 busy,
    jne       timer_jump_prev        ; jump to previous handler.
    cmp       com2_busy,0            ; If COM2 busy,
    jne       timer_jump_prev        ; jump to previous handler.
    cmp       dos_need,0             ; If DOS is not needed
    je        timer_enter
    cmp       disk_busy,0            ; or disk is idle,
    je        timer_enter            ; go call scheduler.
                                      ; Else jump to previous handler.
timer_jump_prev:
    popff
    jmp       prev_vec.ptimer        ; Jump to previous handler.
                                      ; The previous handler will return
                                      ; directly to the original caller.

timer_enter:
    mov       timer_busy,1           ; Prevent nesting.
    popff
    pop       timer_caller_offset    ; Save caller's address.
    pop       timer_caller_seg
    call      prev_vec.ptimer        ; Call previous handler, which will
                                      ; pop the flags from the top of
                                      ; the stack, send the EOI to the
                                      ; interrupt controller, and

```

```

; reenable interrupts.
pushf ; Call scheduler.
call pschedblk
push timer_caller_seg
push timer_caller_offset
mov timer_busy,0 ; No longer any need to protect
; stored copy of caller's address.
db 0cbh ; Far return.
endProc ivtimer
;*****
;
; IVKEYBD -- INT 9 filter
;
; This filter checks the most recent keystroke to see if it
; is a hot key. If so, it records the keystroke and
; removes it from the type-ahead buffer.
beginProc ivkeybd
assume ds:nothing, es:nothing, ss:nothing
BEGIN_IV_FILTER
pushf
cli ; Disable interrupts until we can
; set the busy flag.
cmp kb_busy,0 ; If this filter is already active,
je kb_enter
popff
jmp prev_vec.pkeybd ; jump to previous handler.
; The previous handler will return
; directly to the original caller.
kb_enter:
mov kb_busy,1 ; Prevent nesting.
popff
pop kb_caller_offset ; Save caller's address.
pop kb_caller_seg
call prev_vec.pkeybd ; Call previous handler, which will
; pop the flags from the top of
; the stack, send the EOI to the
; interrupt controller, and
; reenable interrupts.

pushf
push ax
push bx
push cx
push dx
push bp
push si
push di
push ds
push es
cmp keyaction,IV_KY_NONE
jne kb_exit ; Quit if any hot key has already

```

```

                                ; been detected.
                                ; Scan table of hot keys:
    mov     cx,ktsize           ;
    jcxz    kb_exit             ; Quit if table is empty.
    cmp     kb_ext,0            ; Prepare to call BIOS:
    mov     dh,10h              ; DH = 10h if extended services,
    jne     kb_chosen           ; DH = 0 if normal BIOS services.
    mov     dh,0
kb_chosen:
    mov     ah,1                ; Examine most recent keystroke.
    add     ah,dh
    int     16h
    jz      kb_exit             ; Quit if none recorded.
                                ; AX contains char & key code.
    les     di,pkeytab
    cld
kb_loop:
    scasw                                ; Check one hot key listing.
    je      kb_found
    inc     di                        ; Step over its action code.
    inc     di
    loop    kb_loop
    mov     keyfound,0              ; Most recent keystroke is not
                                ; a hot key.
    mov     keyaction,IV_KY_NONE
    jmp     short kb_exit
kb_found:
                                ; Found a hot key.
    mov     bx,es:[di]             ; Get key action for found key.
    cmp     awake,0
    jne     kb_awake              ; If intervention function is asleep
    cmp     bx,IV_KY_WAKE         ; and this is not the wakeup key,
    jnz     kb_exit              ; then quit.
kb_awake:
    mov     keyfound,ax           ; Record hot key for access by the
    mov     keyaction,bx         ; scheduler.
    mov     ah,0                 ; Drop the keystroke from the
    add     ah,dh
    int     16h                  ; type-ahead buffer.
kb_exit:
    pop     es
    pop     ds
    pop     di
    pop     si
    pop     bp
    pop     dx
    pop     cx
    pop     bx
    pop     ax
                                ; Top of stack contains flags returned
                                ; by previous INT 9 handler.

```

```

push    kb_caller_seg
push    kb_caller_offset
mov     kb_busy,0
iret
endProc  ivkeybd

```

```

;*****

```

```

;
; IVCOM2 -- INT 0bh filter (hardware events on communications port 2)
;

```

```

; The sole purpose for this filter is to maintain the com2_busy
; flag telling the scheduler when comm port 2 interrupts are in
; progress.

```

```

beginProc  ivcom2          ; (IVCOM2 will perform a far return.)
assume     ds:nothing, es:nothing, ss:nothing
BEGIN_IV_FILTER
pushf
cli                      ; Disable interrupts until we can
                        ; set the busy flag.
cmp        com2_busy,0    ; If this filter is already active,
je         com2_enter
popff
jmp        prev_vec.pcom2 ; jump to previous handler.
                        ; The previous handler will return
                        ; directly to the original caller.

```

com2 enter:

```

mov        com2_busy,1    ; Prevent nesting.
popff
pop        com2_caller_offset ; Save caller's address.
pop        com2_caller_seg

call       prev_vec.pcom2  ; Call previous handler, which will
                        ; pop the flags from the top of
                        ; the stack.

push       com2_caller_seg
push       com2_caller_offset
mov        com2_busy,0
db         0cbh           ; Far return.
endProc    ivcom2

```

```

;*****

```

```

;
; IVCOM1 -- INT 0ch filter (hardware events on communications port 1)
;

```

```

; The sole purpose for this filter is to maintain the com1_busy
; flag telling the scheduler when comm port 1 interrupts are in
; progress.

```

```

beginProc  ivcom1          ; (IVCOM1 will perform a far return.)
assume     ds:nothing, es:nothing, ss:nothing
BEGIN_IV_FILTER
pushf

```

```

        cli                                ; Disable interrupts until we can
                                           ; set the busy flag.

        cmp        com1_busy,0            ; If this filter is already active,
        je         com1_enter
        popff
        jmp        prev_vec.pcom1         ; jump to previous handler.
                                           ; The previous handler will return
                                           ; directly to the original caller.

com1_enter:
        mov        com1_busy,1            ; Prevent nesting.
        popff
        pop        com1_caller_offset     ; Save caller's address.
        pop        com1_caller_seg
        call       prev_vec.pcom1         ; Call previous handler, which will
                                           ; pop the flags from the top of
                                           ; the stack.

        push       com1_caller_seg
        push       com1_caller_offset
        mov        com1_busy,0
        db         0cbb                  ; Far return.
        endProc    ivcom1

;*****
;
; IVDISK -- INT 13h filter (BIOS disk services)
;
; The sole purpose for this filter is to maintain the disk busy
; flag telling the scheduler when BIOS disk operations are in
; progress.
        beginProc   ivdisk                ; (IVDISK will perform a far return.)
        assume      ds:nothing, es:nothing, ss:nothing
        BEGIN_IV_FILTER
        pushf
        cli                                ; Disable interrupts until we can
                                           ; set the busy flag.

        cmp        disk_busy,0            ; If this filter is already active,
        je         disk_enter
        popff
        jmp        prev_vec.pdisk         ; jump to previous handler.
                                           ; The previous handler will return
                                           ; directly to the original caller.

disk_enter:
        mov        disk_busy,1            ; Prevent nesting.
        popff
        pop        disk_caller_offset     ; Save caller's address.
        pop        disk_caller_seg
        call       prev_vec.pdisk         ; Call previous handler, which will
                                           ; pop the flags from the top of

```

```

                                ; the stack.
push    disk_caller_seg
push    disk_caller_offset
mov     disk_busy,0
db      0cbb                    ; Far return.
endProc    ivdisk
;*****
;
; IVBIOSKY -- INT 16h filter (BIOS keyboard services)
;
; This filter prevents DOS from encountering a Ctrl-C
; from the keyboard. It works by filtering all calls to
; keyboard functions 0 (wait for keystroke) and 1 (test for
; keystroke). If a Ctrl-C is encountered, the keystroke
; is discarded and the operation is repeated.
beginProc    ivbiosky
assume       ds:nothing, es:nothing, ss:nothing
pushf
cli                                ; Disable interrupts until we can
                                ; set the busy flag.
cmp         ah,2                    ; Filter only if function number
jae         key_jump_prev          ; is 0 or 1.
cmp         key_busy,0             ; If this filter is already active,
je          key_enter              ; jump to previous handler.
key_jump_prev:
popff
jmp         prev_16h_vec
                                ; The previous handler will return
                                ; directly to the original caller.
key_enter:
mov         key_busy,1              ; Prevent nesting.
mov         key_func_number,ah      ; Save function number.
popff
pushf
pop         key_entry_flags
pop         key_caller_offset      ; Save caller's address and flags
pop         key_caller_seg
pop         key_caller_flags
push        key_caller_flags      ; but keep caller's flags on
                                ; top of stack.

key_try:
call        prev_16h_vec           ; Call previous handler, which will
                                ; pop the flags from the top of
                                ; the stack.
pushf                                ; Save flags as modified by
                                ; previous handler.
cmp         key_func_number,0
jne         key_ready
                                ; This is function 0: await

```

```

                                ; keystrok:
    cmp     al,3
    jne     key_exit
    mov     cbreak,1           ; Found a Ctrl-C.
    pop     ax                 ; Discard flags from top of stack.
key_retry:
                                ; At this point, stack is empty
                                ; of things that concern us.
    push    key_entry_flags    ; Reestablish the flags that were
    popff                                       ; in effect at the beginning.
    push    key_caller_flags
    mov     ah,key_func_number
    jmp     key_try

key_ready:
                                ; This is function 1: test for
                                ; keystroke.
                                ; Check ZF to see if char was found.
    popff
    pushf
    jz      key_exit           ; Exit if no character found.
    cmp     al,3
    jnz     key_exit           ; Exit if character found is not
                                ; a Ctrl-C.
    mov     cbreak,1           ; Found a Ctrl-C.
    pop     ax                 ; Discard flags from top of stack.
    push    key_entry_flags    ; Reestablish the flags that were
    popff                                       ; in effect at the beginning.
    push    key_caller_flags
    mov     ah,0               ; Consume the Ctrl-C.
    call    prev_16h_vec
    jmp     key_retry

key_exit:
                                ; Top of stack has flags returned
                                ; by previous handler.
    push    key_caller_seg
    push    key_caller_offset
    mov     key_busy,0
    iret
endProc    ivbiosky
;*****
;
;
; IVCBREAK -- INT 1Bh handler
;
; This filter prevents DOS from encountering a Ctrl-Break.
beginProc  ivcbreak
assume     ds:nothing, es:nothing, ss:nothing
mov       cbreak,1
iret

```



```

endProc    ivcbreak
;*****
;
; IVDOS -- INT 21h filter
;
; The purpose of this filter is to delay the return from
; DOS calls if the intervention function needs DOS and is
; waiting for service. After invoking DOS, this filter may
; enter a wait loop that will soon be released by the
; scheduler (which is invoked by a clock tick).
beginProc  ivdos          ; (IVDOS will perform a far return.)
assume     ds:nothing, es:nothing, ss:nothing
BEGIN_IV_FILTER
push       bx
xchg       ax,bx          ; Save AX in BX.
lahf                          ; Save flags in AH.
cmp        dos_need,0      ; If intervention function doesn't
je         dos_jump_prev   ; need DOS, just invoke DOS directly.
cmp        timer_busy,0    ; If timer tick is busy,
jne        dos_jump_prev   ; just invoke DOS directly.
; Certain DOS functions examine their
; return address or do not return to
; their caller. We must invoke them
; directly:
;
; (Remember that we saved the function
; number in BH.)
;
cmp        bh,26h          ; Function 26h: Make PSP
je         dos_jump_prev   ; (examines caller's address)
;
cmp        bh,4Bh          ; Function 4Bh: Exec
je         dos_jump_prev   ; (can destroy registers on return)
;
cmp        bh,00h          ; Function 00h: Terminate
je         dos_jump_prev   ; (does not return)
;
cmp        bh,31h          ; Function 31h: Keep process
je         dos_jump_prev   ; (does not return)
;
cmp        bh,4ch          ; Function 4ch: Terminate
jne        dos_enter       ; (does not return)

dos_jump_prev:
sahf                          ; Restore the flags.
xchg       ax,bx
pop        bx
jmp        prev_vec.pdos     ; Jump to previous handler.
; The previous handler will return
; directly to the original caller.

```

```

dos_enter:                                ; This is a normal DOS function call.
    sahf                                  ; Restore the flags.
    xchg    ax,bx
    pop     bx
    cli                                           ; Rearrange the top items on the
    pop     dos_caller_offset ; stack to bring the flags to
    pop     dos_caller_seg   ; the top.
    pop     dos_save_flags
    push    dos_caller_seg
    push    dos_caller_offset
    push    dos_save_flags
    call    prev_vec.pdos ; On return, DOS pops the flags from
                                ; the top of the stack.

    pushf
    cli
    cmp     req,0 ; Skip the wait loop if the intervention
    je      dos_exit ; function doesn't require service
                                ; at this time.

    push    es ; Check the DOS critical section flag:
    push    di
    les     di,pdos_crit
    cmp     es:byte ptr [di],0
    pop     di
    pop     es
    jne     dos_exit ; if DOS is busy, skip the wait loop
                                ; so DOS can complete its business.

    mov     iv_wait,1
    sti

dos_wait:                                ; Wait until somebody sets iv_wait
    cmp     iv_wait,0 ; to zero.
    jne     dos_wait

dos_exit:                                ; Top of stack has flags returned by
                                ; previous INT 21h handler.

    popff
    db      0cbh ; Far return.
    endProc    ivdos

;*****
;
;
;   IDLE -- INT 28h filter
;
;   The purpose of this filter (similar to the INT 21h filter above)
;   is to delay the return from INT 28h calls if the intervention
;   function is waiting for service. After invoking the previous
;   handler, this filter may enter a wait loop that will soon be
;   released by the scheduler (which is invoked by a clock tick).
    beginProc    idle
    assume      ds:nothing, es:nothing, ss:nothing
    BEGIN_IV_FILTER
    pushf

```

```

        cli                                ; Disable interrupts until we can
                                           ; set the busy flag.
        cmp        idle_busy,0            ; If this filter is already active
        jne        idle_jump_prev
        cmp        dos_need,0            ; or if DOS is not needed
        je         idle_jump_prev
        cmp        dkey_need,0           ; or if DOS functions 1-12 are needed
        je         idle_enter
idle_jump_prev:
        popff
        jmp        prev_vec.pidle ; jump to previous handler.
                                           ; The previous handler will return
                                           ; directly to the original caller.
idle_enter:
        mov        idle_busy,1           ; Prevent nesting.
        popff
        pop        idle_caller_offset ; Save caller's address.
        pop        idle_caller_seg
        call       prev_vec.pidle ; Call previous handler. On return,
                                           ; it pops the flags from the top
                                           ; of the stack.
        mov        idle_safe,1           ; Now we're in the nice INT 28h
                                           ; environment: all DOS functions are
                                           ; allowable except functions 1-12.
        pushf
        cli
        cmp        req,0                 ; Skip the wait loop if the intervention
        je         idle_exit             ; function doesn't require service
                                           ; at this time.

        sti
        mov        iv_wait,1
idle_wait:
        cmp        iv_wait,0             ; Wait until somebody sets iv_wait
        jne        idle_wait             ; to zero.

idle_exit:
                                           ; Top of stack contains flags returned
                                           ; by previous INT 28h handler.
        push       idle_caller_seg
        push       idle_caller_offset
        mov        idle_safe,0
        mov        idle_busy,0
        iret
endProc        ividle
endCseg        ivctrl
endMod         ivctrl
end

```

**IVDETECT**      检测已安装的插入函数，即使它被部分覆盖

```
#include<binterv.h>
```

```
int ivdetect( const IV_VECTORS far *pvector,
              const char far *pident,
              IV_CTRL far **ppctrl,
              unsigned far *pfound);
```

pvector 结构的地址，该结构包含一组待检查的中断向量。

pident 标识信息的十六字节数组的地址。

ppctrl far 指针的地址，从该指针中返回发现的插入控制块的地址。如果未发现插入函数，返回 FARNIL。

pfound 变量的地址，该变量返回指示插入过滤程序的位屏蔽，这些插入过滤程序被已发现而未被其它中断处理程序覆盖的插入函数所作用。下面是可以设置的位：

符号	值	意义
IV_F_TIMER	0x001	中断 08h: 计时器脉冲。
IV_F_KEYSTROKE	0x002	中断 09h: 按键。
IV_F_DISK	0x0004	中断 13h: BIOS 磁盘。
IV_F_DOS	0x008	中断 21h: DOS 功能。
IV_F_IDLE	0x0010	中断 28h: DOS 空闲。
IV_F_2COM	0x0020	中断 0ch: Comm 1
IV_F_1COM	0x0040	中断 0bh: Comm 2

(返回) 成功代码：

IV_NO_ERROR	(0)	成功并且发现的函数可删除；
IV_NOT_FOUND	(7)	未发现；
IV_PART_COVERED	(6)	已发现，但某些 b_ivmask 指定的向量被其它 ISR 覆盖。

IVDETECT 检查一组中断向量，看看它们是否属于支持 Turbo C TOOLS 插入函数的插入过滤程序。

IVDETECT 不追踪“级联”式中断，然而，即使某些插入过滤程序被其它 ISR 遮蔽，IVDETECT 也可以定位插入控制块。\*pfound 返回时包含着一个屏蔽，指示哪一个过滤程序是“暴露的”，可以被直接摘除。

如果发现了插入函数，则仅当检测到所有的由全局变量 b\_ivmask 指明的过滤程序时，该插入函数才必然能够被删除，IV\_NO\_ERROR 作为函数值返回。(参阅“插入码(IV)”的“选择插入过滤程序”中有关 b\_ivmask 的说明。) 如果某些指定的过滤程序被其它中断处理程序覆盖，则返回 IV\_PART\_COVERED。\*pfound 指明正在使用的滤程序，该程序必定是可删除的。

如果发现了插入函数，用 IVDISABL 可以恢复中断向量的内容，利用 ISSENSE 则能够控制一个常规的 Blaise 中断服务例程(ISR)。

源程序(IVDETECT.C):

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

```
#include <string.h>
```

```
#include <binterv.h>
```

```
#include <butil.h>
```

```
/* Internal functions. */
```

```
static IV_CTRL far *check_vec(const void far *);
```

```
static int match_vec(const IV_CTRL far *,
                    const char far *);
```

```
int ivdetect(pvector,pident,ppctrl,pmask)
```

```
const IV_VECTORS far *pvector;
```

```
const char far *pident;
```

```
IV_CTRL far **ppctrl;
```

```

unsigned          far *pmask;
{
    const unsigned masks[] =          /* Array of mask values.      */
    {                                  /* These must correspond with */
                                      /* the contents of pvecs[]!   */

        IV_F_TIMER,
        IV_F_KEYSTROKE,
        IV_F_DISK,
        IV_F_DOS,
        IV_F_IDLE,
        IV_F_1COM,
        IV_F_2COM,
    };
#define NUM_VECTORS (sizeof(masks)/sizeof(masks[0]))
    const void far *pvecs[NUM_VECTORS];
    IV_CTRL    far *ptest[NUM_VECTORS];
    char        ident[sizeof((*ppctrl)->ident)];
    int         i,first_match,result;
    pvecs[0] = pvecs->ptimer;          /* pvecs[] is used to make the */
    pvecs[1] = pvecs->pkeybd;          /* main loop more convenient. */
    pvecs[2] = pvecs->pdisk;          /* Sequence of items must      */
    pvecs[3] = pvecs->pdos;           /* correspond with masks[]!    */
    pvecs[4] = pvecs->pidle;
    pvecs[5] = pvecs->pcom1;
    pvecs[6] = pvecs->pcom2;
    utpeekn(pident,ident,sizeof(ident)); /* Local copy of              */
                                      /* identification string.*/

    *pmask = 0;
    first_match = -1;
    for (i = 0; i < NUM_VECTORS; i++) /* Examine all vectors.      */
    {
        ptest[i] = check_vec(pvecs[i]); /* Fetch address of possible */
                                      /* control block.            */
        if (match_vec(ptest[i],ident)) /* Check control block.      */
        {
            if (first_match < 0)
                first_match = i;

            /* Must equal other found */
            /* blocks.                  */

            if (ptest[i] == ptest[first_match])
                *pmask |= masks[i]; /* Have found a total match. */
        }
    }
    if (first_match >= 0)
    {
        *ppctrl = ptest[first_match]; /* Found a control block.    */
                                      /* See whether enough filters */
                                      /* are exposed.              */

        if ((*pmask & b_ivmask) == b_ivmask)
            result = IV_NO_ERROR;
    }
}

```

```

        else
            result = IV_PART_COVERED;
    }
    else
    {
        *ppctrl = FARNIL;
        result = IV_NOT_FOUND;
    }
    return result;
}
/**
 *
 * Name          check_vec -- Return pointer to IV_CTRL structure (if any)
 *                  used by an intervention filter.
 *
 * Synopsis      presult = check_vec(ptr);
 *
 *                  IV_CTRL far *presult
 *
 *                  Far pointer to possible IV_CTRL
 *                  structure, or FARNIL if ptr or the
 *                  data pointed to is invalid.
 *
 * void far *ptr   Address of entry point of purported
 *                  filter.
 *
 * Description    This function examines the contents of memory pointed to
 *                  by an interrupt vector to see whether it might be an
 *                  entry point to a Blaise C TOOLS intervention filter. The
 *                  resulting value is checked so that it can be used as a
 *                  pointer to an IV_CTRL structure without fear that
 *                  segment boundaries may be crossed.
 *
 *                  By convention, intervention filters begin with a two-byte
 *                  jump instruction followed by a two-byte pointer to their
 *                  intervention control block.
 *
 *                  This function does NOT trace chains of "cascaded"
 *                  interrupt handlers.
 *
 * Returns        presult          Far pointer to possible IV_CTRL
 *                  structure, or FARNIL if ptr or the
 *                  data pointed to is invalid.
 */

```

```

static IV_CTRL far *check_vec(ptr)
const void far *ptr;
{
    unsigned int offset;          /* Scratch variable: offset */
                                   /* within segment. */
    offset = utoff(ptr);
    if (offset >= 0xffffd)        /* Make sure that the two-byte */

```

```

    return FARNIL;
    /* pointer to the control
    /* block doesn't straddle a
    /* segment boundary.
    /* Extract offset of control
    /* block.

offset = utpeekw(uttofar(utseg(ptr),offset+2));
/* Make sure that the control block does not straddle a segment
/* boundary. Notice that this computation requires that
/* sizeof(IV_CTRL) be at least 2 (which it is).
if (offset >= (unsigned) 0xffff - sizeof(IV_CTRL) + 2)
    return FARNIL;
return uttofar(utseg(ptr),offset,IV_CTRL);
}

/**
 *
 * Name      match_vec -- Check validity of potential address of
 *              intervention control block.
 * Synopsis  match = match_vec(ptr,pident);
 *              int match      Zero if not a valid intervention
 *                              control block or if identification
 *                              string fails to match,
 *                              Nonzero if valid and matching.
 *              const IV_CTRL far *ptr
 *                              Purported address of intervention
 *                              control block.
 *              const char *pident
 *                              Pointer to array of 16 bytes of
 *                              identifying information
 * Description This function examines a pointer to see whether whether
 *              it might contain the address of an intervention control
 *              block.
 * Returns    match      Zero if not a valid intervention
 *                              control block or if identification
 *                              string fails to match,
 *                              Nonzero if valid and matching.
 **/
static int match_vec(ptr,pident)
const IV_CTRL far *ptr;
const char      *pident;
{
    char scratch[sizeof(ptr->ident)];
    if ( ptr == FARNIL
        || ptr->signature != (unsigned int) IV_SIGNATURE
        || ptr->sign2      != ~ (unsigned int) IV_SIGNATURE)
        return 0;
    /* Make local copy of identifying
    /* string from control block.
    utpeekn(ptr->ident,scratch,sizeof(scratch));
    /* Return 1 if match.

```

```

    return (0 == memcmp(scratch, pident, sizeof(scratch)));
}

```

## IVDISABL 使一个插入函数失效

```
#include <blaterv.h>
```

```
int ivdisabl(IV_CTRL far *pctrl);
```

pctrl 等关闭的插入控制块的双字地址。

(返回) 错误返回代码。可能值包括:

IV\_NO\_ERROR (0) 无错误。

IV\_NOT\_ENABLED (1) 未打开插入控制块。

IV\_NOT\_INSTALLED (2) 插入过滤程序当前未安装, pctrl 不是一个合法的插入

控制块的地址。

IV\_NULL\_PTR (3) pctrl 是 FARNIL。

INDISABL 通过重新安装插入过滤程序使用的中断向量的原值使该插入函数无效。

IVDISABL 仅使列在全局变量 b\_ivmask 中的过滤程序无效。如果由 b\_ivmask 指定的过滤程序的任意一个可删除, 则不进行任何操作, IV\_NOT\_INSTALLED 作为函数值返回。

要想暂时使一个插入函数无效而不重新安装中断向量, 可参见“插入码(IV)”一章中的“暂时使一个插入函数失效”一节。

用 ISPUTVEC 可以使一个常规的中断服务例程无效, 用 ISREMOVE 则可以从内存删除一程序。

源程序(IVDISABL.C):

```

#include <dos.h>
#include <string.h>
#include <blaterv.h>
#include <butil.h>
#define OK 0 /* Error codes. */
#define IV_NOT_ENABLED 1
#define IV_NOT_INSTALLED 2
#define IV_NULL_POINTER 3
int ivdisabl(pctrl)
IV_CTRL far *pctrl;
{
    IV_VECTORS vecs;
    unsigned save_mask;
    if (FARNIL == pctrl)
        return IV_NULL_POINTER;
    if (!pctrl->enabled)
        return IV_NOT_ENABLED;
    ivvecs(IV_RETVEC, &vecs); /* Check current vectors to be */
                               /* sure that the intervention */
                               /* function is installed. */
    if (pctrl != ivsense(&vecs, pctrl->ident))
        return IV_NOT_INSTALLED;
    pctrl->enabled = 0;
    save_mask = b_ivmask; /* Save b_ivmask. */
    b_ivmask &= pctrl->filter_mask; /* Restore no more vectors */
                                   /* than were installed. */
    ivvecs(IV_SETVEC, &pctrl->prev_vec); /* Restore the vectors. */
    b_ivmask = save_mask; /* Restore b_ivmask. */
}

```



```
return IV_NO_ERROR;
```

## IVINSTAL 安装一个插入函数

```
#include <hinterv.h>
```

```
int ivinstal( void cdecl (*pfunc)(IV_EVENT *),
              const char *pident,
              char *pstack, int stksize,
              IV_KEY *pkeytable, int numkeys,
              IV_TIME *ptimetable, int numtimes,
              int option);
```

pfunc 指向插入函数的指针。

pident 指向标识信息 16 字节数组的指针。

pstack 内存块起始地址(已分配的), 该内存块将用于插入函数的堆栈。

stksize 堆栈的尺寸(以字节计), 必须至少为 256 字节。2048 字节是典型的尺寸。

pkeytable 列出热键的 IV\_KEY 结构数组的地址。

numkeys 所列热键的数目。

ptimetable 列出计时器事件的 IV\_TIME 结构数组的地址。

numtimes 所列计时器事件的数目。

option 指明插入函数需求的四位值:

符号	值	意义
IV_DOS_NEED	1	所用 DOS 功能。
IV_NO_DOS_NEED	0	不使用 DOS 功能。
IV_DKEY_NEED	2	使用了 DOS 功能 1-12。
IV_NO_DKEY_NEED	0	未使用 DOS 功能 1-12。
IV_FLOAT_NEED	4	激活时暂时重新安装浮点向量。
IV_NO_FLOAT_NEED	0	不需要重新安装浮点向量。
IV_DAILY	8	“瞬时”事件应该每日发生。

(返回) 错误返回代码。可能值包括:

IV_NO_ERROR	(0)	无错误。
IV_NO_ERROR	(0)	无错误。
IV_INSTALLED	(5)	本程序中已安装了一个插入函数。

IVINSTAL 安装一个插入函数, 安排激活它的事件。这包括(1)完成所有插入控制块中的值; (2)安排调度程序在适当的时刻激活插入函数; (3) 在适当的中断向量中放入插入过滤程序的地址。

一旦步骤(3)完成, 插入函数立即生效(可以被激活), 也就是说, 甚至在 IVINSTAL 返回它的调用程序以前即生效。

pkeytable, numkeys, ptimetable 和 numtimes 参数指明激活插入函数的事件。详见“插入码(IV)”。请不要把 IV\_TM\_NONE 作为计时器事件执行的动作。

调用函数必须为插入函数的堆栈分配空间, 这个空间的地址以 pstack 传递并且应该至少为 stksize 字节长。这个空间必须是静态的, 直至该插入函数失效。如果 ISR 在 T、S 或 M 模式的 Turbo C 程序中使用 near 堆, 或在 C、L 或 H 模式的 Turbo C TSR(中止并驻留)程序中使用 far 堆, 则用 ISRESERV 可以保留用于 Malloc()的空间。

标识信息的十六字节保存在插入控制块中以便于其它程序检查中断向量, 检测这个插入函数的存在。(参见示例 ivsense.c。)

用 IVDISABL 可以关闭一个插入函数, 用 ISINSTAL 则能够安装一个常规的中断服务例程(ISR)。

源程序(IVINSTAL.C):

```
#include <dos.h> /* For use with isgetvec(), */
/* isputvec(), REGS, and int86()*/
```

```

#include <binlrv.h>
#include <bintrup.h>
#include <bkeybrd.h>
int b ivusex = KB_USE_EXTEND; /* Whether to use extended */
/* keyboard services when */
/* testing for hot keys: */
/* KB_USE_EXTEND or */
/* KB_USE_NORMAL. */

extern void ivtimer(void); /* Intervention code filters */
extern void ivkeybd(void); /* (in ISCTRLASM) */
extern void ivcom2(void);
extern void ivcom1(void);
extern void ivdisk(void);
extern void ivdos(void);
extern void ividle(void);
extern void ivbiosky(void); /* These two filters are */
extern void ivcbreak(void); /* installed by the scheduler */
/* before invoking the */
/* intervention function and */
/* removed after it exits. */

/* Local variables used by scheduler, initialized by installer. */
static IV CTRL far *ptrf = FARNIL; /* Pointer to internal values */
/* used by filters. */
/*

static IV TIME *ptimtab; /* Address and size of user's */
static int timtabsize; /* table of timer events. */
static int timindex; /* Index of found timer event. */
static int timaction; /* Type of timer action last */
/* requested. */

static long idleticks; /* Clock ticks since last */
/* service. */

static long lasttime; /* Time of last call. */
/*

static int daily; /* Whether to reschedule moment */
/* events daily. */
/*

static void /* Pointer to user's */
(*ptask)(IV EVENT *); /* intervention function. */
/*

static unsigned char /* Pointer to interrupt 0x24 */
far *pint24h flag; /* busy flag. */

#if (IS_NUM_FLOAT_VECS > 0)
static char float_need; /* Storage for floating point */
/* vectors. */

static void far *float vec[IS_NUM_FLOAT_VECS];
#endif

/* Internal functions */

static void scheduler(ALLREG *,ISCTRL *,ISRMSG *);
static void timer req(void);

```

```

static int readytask(void);
static void unschedule(void);
static void reschedule(void);
int ivinstal(pfunc,pident,pstack,stksize,
             pkeytable,numkeys,ptimetable,numtimes,option)
void      (*pfunc)(IV_EVENT *);
const char *pident;
char      *pstack;
int       stksize;
IV_KEY    *pkeytable;
int       numkeys;
IV_TIME   *ptimetable;      /* This table must not move. */
int       numtimes;
int       option;
{
    int      ints_were_on,i;
    IV_VECTORS   vecs;
    unsigned int i24_seg,i24_offset;
    static ISRCTRL schedblk = {0}; /* ISR control block for */
                                   /* scheduler. */
    pctrl      = ivctrl(); /* Find intervention control */
                           /* structure. */
    ints_were_on = utintflg(UT_INTTOFF); /* Prevent interrupts while */
                                           /* checking enabled state. */
    if (pctrl->enabled) /* Error: this function must */
    { /* not be used more than once. */
        utintflg(ints_were_on); /* */
        return IV_INSTALLED; /* */
    } /* */
    pctrl->enabled = 1; /* Prevent second use. */
    utintflg(ints_were_on); /* Restore interrupt state. */
    /* Initialize local variables used by scheduler. */
    ptask      = pfunc; /* Address of user's */
                        /* intervention function. */
    ptimtab     = ptimetable; /* Address and size of user's */
    timtabsize  = numtimes; /* table of timer events. */
    timindex    = -1; /* Index of found timer event. */
    timaction   = IV_TM_NONE; /* Type of timer action last */
                             /* requested. */
    idleticks   = 0L; /* Clock ticks since last */
                     /* service. */

    if (option & IV_DAILY)
    { /* Moment events happen once */
      /* daily. */
      utgetclk(&lasttime); /* Start time for reschedule(). */
      unschedule(); /* Cancel today's events whose */
                   /* time has passed. */

      daily = 1;
    }
    else

```

```

        daily = 0;
#if (IS_NUM_FLOAT_VECS > 0)
        float_need = (char)(0 != (option & IV_FLOAT_NEED));
                                /* Save floating point vectors */
        for (i = 0; i < IS_NUM_FLOAT_VECS; i++)
            float_vec[i] = isgetvec(IS_1ST_FLOAT_VEC + i);
#endif

        /* Set up ISR control block through which the scheduler will be */
        /* called. */
        isprep(scheduler,"TCT6.00 INTSCHED",&schedblk,pstack,stksize,1);
        /* Initialize most items in intervention control structure. */
        pctrl->psp = utpspseg; /* PSP of this program. */
                                /*
                                /*
                                /* Save previous contents of */
                                /* Interrupt vectors. */
                                /*
                                /* Set the dos_need flag if */
                                /* either the IV_DOS_NEED or */
                                /* IV_DKEY_NEED bit is set. */

        pctrl->dos_need = (char)
                        (0 != (option & (IV_DOS_NEED | IV_DKEY_NEED)));
        pctrl->dkey_need = (char)(0 != (option & IV_DKEY_NEED));
        pctrl->pkeytab = pkeytable; /* Address and size of user's */
        pctrl->ktsize = numkeys; /* table of hot keys. */
        pctrl->key_event.ch = 0; /* No hot key found yet. */
        pctrl->key_event.keycode = 0;
        pctrl->key_event.action = IV_KY_NONE;
        if ( b_ivuser == KB_USE_NORMAL
            || kbequip() == KB_NOEXTENDED)
            pctrl->kb_ext = 0; /* Use normal keyboard services.*/
        else
            pctrl->kb_ext = 1; /* Extended keyboard services. */
        pctrl->wait = 0; /* INT 21h and INT 28h filters */
                                /* don't need to be in wait */
                                /* loop yet. */
        pctrl->req = 0; /* No service yet requested. */
        pctrl->awake = 1; /* Intervention code is not */
                                /* asleep. */
        pctrl->timer_busy = 0; /* Clear the busy flags. */
        pctrl->kb_busy = 0;
        pctrl->com2_busy = 0;
        pctrl->com1_busy = 0;
        pctrl->disk_busy = 0;
        pctrl->idle_busy = 0;
        pctrl->idle_safe = 0;

                                /* Entry point to scheduler. */
        pctrl->pschedblk = ((char far *)(&schedblk)) + ICB_ENTRY_OFFSET;
        utcrit(); /* Address of DOS critical */
        pctrl->pdos_crit = b_critad; /* section flag (busy flag). */
                                /* Address of interrupt 0x24 */

```

```

if (utdosmajor < 3) /* busy flag. */
    i24_offset = utoff(b_critad) + 1;
else if (utdosver == 0x0300)
    i24_offset = 0x0167;
else
    i24_offset = utoff(b_critad) - 1;
i24_seg = utseg(b_critad);
pint24b_flag = uttofaru(i24_seg,i24_offset);
for (i = 0; i < sizeof(pctrl->reserved); i++)
    pctrl->reserved[i] = '\0';
/* User's identifying string. */
utmovmem(pident,pctrl->ident,sizeof(pctrl->ident));
/* Identifying signature. */
pctrl->signature = (unsigned int)IV_SIGNATURE;
pctrl->sign2 = (unsigned int)~IV_SIGNATURE;
/* Install the filters. This will activate the intervention
/* function. */
pctrl->filter_mask = b_ivmask;
vecs.ptimer = ivtimer;
vecs.pkeybd = ivkeybd;
vecs.pdisk = ivdisk;
vecs.pdos = ivdos;
vecs.pidle = ividle;
vecs.pcom1 = ivcom1;
vecs.pcom2 = ivcom2;
ivvecs(IV_SETVEC,&vecs);
return IV_NO_ERROR;
)

```

/\*\*

* Name	scheduler -- Invoke intervention function if appropriate
* Description	This function checks whether the intervention function
*	is eligible to be invoked and invokes it if appropriate,
*	based on the time of day and whether DOS services are
*	necessary and available. It also maintains certain
*	fields in the intervention control block to control the
*	intervention filters.
*	This function is built as an interrupt service routine
*	(ISR) to be called from the intervention timer filter
*	via the ISR dispatcher.
*	Interrupts are enabled.
* Returns	Members of intervention control block:
*	wait Set to zero to release INT 21h and
*	INT 28h filters from wait loop.
*	awake Whether intervention code is asleep
*	or awake.
*	key_event The most recent detected hot key
*	is cleared.
*	req 1 to indicate that the intervention
*	function needs service but must wait

```

*                                     for DOS availability;
*
*                                     0 to indicate that the intervention
*                                     function does not need service.
*
*      Member of user's table of timer events:
*      ptimeable[ ].ticks
*
*      If this event is a moment event,
*      the IV RESCHED bit in the time of day
*      is set, effectively preventing
*      the event from occurring again.
*      At midnight, the bit is cleared.
*      (The user can reschedule the event
*      by changing the ticks member again.)
**/
static void scheduler(pregs,pisrblk,pmsg)
ALLREG *pregs;
ISRCTRL *pisrblk;
ISRMSG *pmsg;
{
    IV_EVENT    signal;
    void far *pint_1bh;
    unsigned    oldproc;
    int         dos_saved;
    char        *p;
#ifdef (IS_NUM_FLOAT_VECS > 0)
    void far *save_float[IS_NUM_FLOAT_VECS];
    int         float_switched;
#endif
    p = (char *) preg;          /* Get rid of pesky compiler */
    p = (char *) pisrblk;       /* warnings */
    p = (char *) pmsg;
    p++;
    pctrl->wait = 0;             /* Release INT 21h or INT 28h */
                                /* filters if they are waiting.*/
    if ((!pctrl->awake) && pctrl->key_event.action != IV KY_WAKE)
        return;
    if (daily)                   /* Reschedule yesterday's */
        reschedule();            /* moment events for today if */
                                /* this is a new day. */
    if (timaction == IV TM_NONE) /* If no timer action pending. */
        timer_req();             /* analyze time of day to see */
                                /* whether a timer action */
                                /* should be pending. */
    if ( pctrl->key_event.action != IV KY_NONE
        || timaction != IV TM_NONE)
    {
        if (readytask())
        {
            /* Now we know that we can */
            /* execute the task. */

            pctrl->req = 0;
            if (timaction == IV TM_INTERVAL)

```

```

        idleticks = 0L;
    else if (timaction == IV_TM_MOMENT)
        /* Disable the event. */
        ptimtab[timindex].ticks |= IV_RESCHED;
        /* (The user can reschedule the */
        /* event by modifying the time */
        /* table.) */

        /* Build signal to tell user */
        /* task what event(s) */
        /* triggered the task. */

    utmovmem((char far *) &pctrl->key_event,
             (char far *) &signal.key,
             sizeof(IV_KEY));
    signal.time_action = timaction;
    signal.time_index = timindex;
    utgetclk(&signal.time);
    pctrl->cbreak = 0; /* No Ctrl-Break or Ctrl-C */
                     /* encountered yet. */
    if (pctrl->dos_need) /* Take special precautions if */
    { /* DOS is used: */
        /* Prevent Ctrl-Break and */
        /* Ctrl-C. */

        pctrl->prev_16h_vec = isgetvec(0x16);
        pint_1bh = isgetvec(0x1b);
        isputvec(0x16,(void far *) ivbiosky);
        isputvec(0x1b,(void far *) ivcbreak);
        /* Set current process to point */
        /* to us. */

        if (utdosmajor >= 3)
            oldproc = iscurprc(IS_SETPROC,utpspseg);
        dos_saved = 1;
    }
    else
        dos_saved = 0;
    #if (IS_NUM_FLOAT_VECS > 0)
        if (float_need)
        {
            /* Save current floating point */
            /* vectors. */

            utintflg(UT_INTOFF);
            utpeekn(uttfar(0,4 * IS_1ST_FLOAT_VEC,char),
                   save_float,
                   sizeof(save_float));
            /* Reinstall our own floating */
            /* point vectors. */

            utpoken(float_vec,
                    uttfar(0,4 * IS_1ST_FLOAT_VEC,char),
                    sizeof(float_vec));
            utintflg(UT_INTON);
        }
    #endif

```

```

        float_switched = 1;
    }
    else
        float_switched = 0;
#endif
    (*ptask)(&signal);        /* Invoke the user task.    */
    #if (IS_NUM_FLOAT_VECS > 0)
        if (float_switched)
        {
            /* Restore float vectors.    */
            utintflg(UT_INTTOFF);
            utpoken(save_float,
                    uttofar(0,4 * IS_1ST_FLOAT_VEC,char),
                    sizeof(save_float));
            utintflg(UT_INTON);
        }
    #endif
    if (dos_saved)
    {
        /* Restore current process.    */
        if (utdosmajor >= 3)
            iscurprc(IS_SETPROC,oldproc);
        /* Restore former Ctrl-Break    */
        /* and BIOS keyboard handling. */
        isputvec(0x1b,pint_1bh);
        isputvec(0x16,pctrl->prev_16h_vec);
    }
    if (pctrl->key_event.action == IV_KY_SLEEP)
        pctrl->awake = 0;
    else if (pctrl->key_event.action == IV_KY_WAKE)
        pctrl->awake = 1;
    pctrl->key_event.ch      = 0;
    pctrl->key_event.keycode = 0;
    pctrl->key_event.action  = IV_KY_NONE;
    timaction               = IV_TM_NONE;
}
else
    pctrl->req = 1;
}
if (timaction != IV_TM_INTERVAL)
    idleticks++;
}
/**
 * Name      timer_req -- Scan table of timer events to find one
 *              that is eligible for service.
 * Synopsi   timer_req();
 * Returns   timaction      IV_TM_NONE if no timer event is
 *                          eligible;
 *                          IV_TM_MOMENT if a one-time event is
 *                          eligible;
 *                          IV_TM_INTERVAL if a periodic event is

```



```

    eligible.
    *           timindex      Index of found event in user's table of
    *                               timer events. This is meaningless
    *                               if IV_TM_NONE is returned.
    **/
static void timer req()
{
    long now;
    int i;
    utgetclk(&now);
    for (i = 0;
        i < timtabsize && timaction == IV_TM_NONE;
        i++)
    {
        if ( ( ptimtab[i].action == IV_TM_MOMENT
            && ptimtab[i].ticks <= now)
            || ( ptimtab[i].action == IV_TM_INTERVAL
            && ptimtab[i].ticks <= idleticks))
        {
            timaction = ptimtab[i].action;
            timindex = i;
        }
    }
}
/**
 * Name          readytask -- Return nonzero if intervention function
 *                  is eligible for service.
 * Synopsis      ready = readytask();
 *               int ready      1 if intervention function can be
 *                               safely executed.
 *                               0 if not.
 **/
static int readytask()
{
    if (pctrl->idle_busy) /* We're inside an INT 28h call,*/
    { /* so DOS functions 1-12 aren't */
        /* available, but other DOS */
        /* functions are. */
        if (pctrl->idle_safe && !pctrl->dkey_need)
            return 1;
        else
            return 0;
    }

    /* We're not in an INT 28h call.*/
    else if (pctrl->dos_need) /* If intervention function */
    { /* needs any DOS or disk */
        /* functions, */
        if ( pctrl->disk_busy
            || !hidswrdy()
            || *pint24h_flag)

```

```

        return 0;                /* Disk or DOS is busy so fail. */
    else
        return 1;                /* Disk & DOS are available. */
    }
    else
        return 1;
}
/**
 * Name            unschedule -- Cancel for today all "moment" events
 *                  whose time has passed.
 * Synopsis        unschedule();
 * Description      This function sets the IV_RESCHED bit in all "moment"
 *                  events if necessary, preventing them from executing
 *                  today.
 **/
static void unschedule()
{
    long now;
    int i;
    utgetclk(&now);
    for (i = 0; i < timtabsize; i++)
        if ( ptimtab[i].action == IV_TM_MOMENT
            && ptimtab[i].ticks < now)
            ptimtab[i].ticks |= IV_RESCHED;
}
/**
 * Name            reschedule -- Reschedule all "moment" events if
 *                  midnight has passed.
 * Synopsis        reschedule();
 * Description      This function clears the IV_RESCHED bit from all
 *                  "moment" events if necessary, allowing them to execute
 *                  again. It takes action only if the time of day is
 *                  observed to go backward (as it normally does at
 *                  midnight.)
 **/
static void reschedule()
{
    long now;
    int i;
    utgetclk(&now);
    if (now < lasttime)
    {
        /* Time went backward, so this */
        /* must be a new day. */
        /* Reschedule all moment events.*/

        for (i = 0; i < timtabsize; i++)
            if (ptimtab[i].action == IV_TM_MOMENT)
                ptimtab[i].ticks &= ~IV_RESCHED;
    }
    lasttime = now;                /* Save current time. */
}

```

## IVSENSE 检测一个已安装的插入函数是否是可删除的

```
#include <binterv.h>
```

```
IV_CTRL far *ivsense(const IV_VECTORS far *pvectors,  
const char far *pident);
```

pvectors 结构的地址, 该结构包含待检查的一组中断向量。

pident 标识信息十六字节数组的地址。

(返回) 查询到的插入控制块的地址。如果未查询到插入过滤程序, 返回 FARNIL。

IVSENSE 检查一组中断向量, 看它们是否属于支持 Turbo C TOOLS 插入函数的插入过滤程序。如果所有由全局变量 `b_ivmask` 声明的向量与该插入函数相匹配, 则 IVSENSE 返回插入控制块的地址。

限制: IEVSENSE 假设插入过滤程序对于相应的中断来说是最近安装的处理程序。如果由 `b_ivmask` 所指定的任意一个中断向量所指向的 ISR 不属于该过滤程序而只是将控制传递给插入函数, 则不会查询到该插入函数。用户可以部分地通过 IVDETECT 克服这个限制。IVDETECT 能够检测到一个插入控制块, 即使它的一个过滤程序面临着直接的检查(也就是说, 未被其它 ISR 覆盖)。

如果发现了插入函数, 可以用 IVDISABL 恢复中断向量的内容。用 ISSENSE 能够检测到一个常规的 Blasie 中断服务例程(ISR)。

源程序(IVSENSE.C):

```
#include <binterv.h>
```

```
#include <butil.h>
```

```
IV_CTRL far *ivsense(pvectors,pident)
```

```
const IV_VECTORS far *pvectors;
```

```
const char far *pident;
```

```
{
```

```
    unsigned mask;
```

```
    IV_CTRL far *pctrl;
```

```
    return ((IV_NO_ERROR == ivdetect(pvectors,pident,&pctrl,&mask))
```

```
        ? pctrl
```

```
        : FARNIL);
```

```
}
```

## IVVECS 设置或返回插入过滤程序所使用的中断向量

```
#include <binterv.h>
```

```
int invvecs(int option,
```

```
IV_VECTORS far *pvectors);
```

option IV\_SETVEC (1) 安装向量, IV\_RETVEC(0) 报告向量。

pvectors 包含一组中断向量的结构的地址。

(返回) 错误返回代码。可能值包括:

IV\_NO\_ERROR (0) 无错误。

IV\_BAD\_OPT (4) option 未知。

IVVECS 设置或返回插入过滤程序中使用的中断向量。参见“插入码(IV)”中有关插入过滤程序的说明。

设置向量时, IVVECS 仅设置 `b_ivmask` 所列出的向(参见(IV)”中有关插入过滤程序的说明。

设置向量时, IVECS 仅设置 `b_ivask` 所列出的向量(参见“插入码 IV)”中的“选择插入过滤程序”一节; 返回向量时, IVVECS 总是返回所有的向量而不论 `b_ivmask` 是什么内容。

源程序(IVVECS.C):

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

```

#include <bintrv.h>
#include <bintrupt.h>
unsigned b_ivmask = IV_STD_FILTERS; /* Mask indicating which */
/* intervention filters to use. */

int ivvecs(option,pvectors)
int option;
IV_VECTORS far *pvectors;
{
    int result;
    switch (option)
    {
        case IV_RETVEC:
            pvectors->ptimer = isgetvec(0x08);
            pvectors->pkeybd = isgetvec(0x09);
            pvectors->pdisk = isgetvec(0x13);
            pvectors->pdos = isgetvec(0x21);
            pvectors->pidle = isgetvec(0x28);
            pvectors->pcom1 = isgetvec(0x0c);
            pvectors->pcom2 = isgetvec(0x0b);
            result = IV_NO_ERROR;
            break;
        case IV_SETVEC:
            /* Install disk (INT 0x13) and DOS idle */
            /* (INT 0x28) filters first so their */
            /* safeguards are in place before the */
            /* other filters can invoke the */
            /* intervention function. */
            if (b_ivmask & IV_F_2COM)
                isputvec(0x0b,pvectors->pcom2);
            if (b_ivmask & IV_F_1COM)
                isputvec(0x0c,pvectors->pcom1);
            if (b_ivmask & IV_F_DISK)
                isputvec(0x13,pvectors->pdisk);
            if (b_ivmask & IV_F_IDLE)
                isputvec(0x28,pvectors->pidle);
            if (b_ivmask & IV_F_TIMER)
                isputvec(0x08,pvectors->ptimer);
            if (b_ivmask & IV_F_DOS)
                isputvec(0x21,pvectors->pdos);
            if (b_ivmask & IV_F_KEYSTROKE)
                isputvec(0x09,pvectors->pkeybd);
            result = IV_NO_ERROR;
            break;
        default: /* Unknown option. */
            result = IV_BAD_OPT;
            break;
    }
    return result;
}

```

插入码程序设计的方法请见第一章的示例程序。

## 第七章 高级键盘管理程序设计

键盘控制函数提供了对IBM个人计算机或兼容机键盘的直接控制,包括增强键盘(101或102键)。这些函数返回字符和扩展字符码,检测是否有一个按键就绪、读取或设置移位键的状态(如Shift、Scroll Lock等),还可直接将按键“插入”键盘缓冲区。

这些函数还包括一个称为键控制函数的非常有效的特性,即具有安装一个函数的能力,该函数在等待一个按键时可以被调用。这样的函数使程序在等待用户输入时能够继续工作。您还可以利用键控制函数来过滤或修改一个按键或从另一个来源提供按键,如磁盘文件或通讯口。示例程序KEYCTRL和ENTRYEDT展示了使用键控制函数的方法。

### 增强键盘

IBM增强键盘(101或102键)能够以与常规的PC和PC/AT键盘(83和84键)不同的方式工作。例如,新的IBM型号上的BIOS可以将数字小键盘上的下箭头与增强键盘的光标小键盘上的下箭头区别开来。

Turbo C TOOLS允许您指明使用扩展键盘服务(全部支持增强键盘上的新键)还是使用常规的键盘服务(不区分两个小键盘)。要作出选择,需调用KBEXTEND。其后对KBGETKEY和KBREADY的调用将使用常规功能(分别为功能0和1)或者使用与提高警惕BIOS相对应的功能(分别为功能0x10和0x11)。

### 键盘函数的功能

#### 键盘输入

KBREADY	报告下一个等待按键(如果有的话)而不把该按键从缓冲区中删除。这是一个非破坏性读取。
KBGETKEY	等待下一个按键并把该按键从缓冲区中删除,返回字符和键码。这是一个破坏性读取。
KBQUERY	从用户返回一个字符串,将每个按键回显到屏幕。它不利用DOS的标准输入或输出。(请与FLPROMKPT相比较。)

#### 处理超前键入缓冲区

KBFLUSH	废弃所有的等待按键。
KBSTUFF	迫使一个按键字符串放入超前键入缓冲区。
KBPLACE	迫使一个按放入超前键入缓冲区。
KBQUEUE	报告在缓冲区中等等的按键。

#### 处理移位键

KBSTATUS	报告“移位”键的状态: Shift、Ctrl、Alt、Num Lock、Scroll Lock和Insert。
KBSET	设置移位键的状态和键盘指示灯。

#### 使用增强键盘

KBEXTEND	选择用于KBGETKEY和KBREADY的扩展或常规的BIOS键盘服务。
KBEQUIP	检测扩展BIOS键盘服务和增强键盘的存在,报告是否选择了扩展键盘服务。

#### 使用键控制函数

KBWAIT正象KBGETKEY ;	等待下一个按键并删除之,返回字符和键码。这是一个破
--------------------	---------------------------

坏性读取。  
 KBPOLL与KBREADY相似：报告下一个等待按键(如果有的话)，可任性地删除之。  
 KBKCFUSH类似KBFLUSH：废弃所有的等待按键。

## 取得键码

KBSCANOF 返回产生指定ASCII字符的最常见的键码。

## 高级特性：键控制函数

键控制函数是一个特殊的函数，它在程序查寻、读取或清空键盘时接管控制。这是一个功能极为强大的技术，有着多种用途。通过使用一个键控制函数，用户程序在等待一个按键时可以继续工作。您可以对一个按键进行过滤或修改，还可以动态地重新定向数据的输入源而不影响下面程序的设计。

标准键盘函数KBGETKEY、KBREADY和KBFLUSH具有对应的支持键控制函数的版本：KBWAIT、KBPOLL和KBKCFUSH。

有些执行BIOS键盘输入的Turbo C TOOLS函数允许使用键控制函数。HLREAD、KBQUERY、MNLREAD、MNREAD、WNQUERY和WNREAD都使用KBWAIT或KBPOLL。它们激活其地址保存在全局变量b\_key\_ctrl中的键控制函数。如果让b\_key\_ctrl保持它的缺省值NIL，则不调用键控制函数，那些函数仍将正常工作。DEFIELD和WNFIELD允许您将一个键控制函数的地址指定为一个结构的一部分。

## 调用步骤

键控制函数在声明时带有一个参数，该参数是KB DATA结构的地址。KB DATA在bkeybrd.h头文件中声明如下：

```
typedef struct          /* KEY SEQUENCE 结构: */
{
    /* 一个键的字符和键码。 */
    unsigned char    character_code;
    unsigned char    key_code;
} KEY_SEQUENCE;

typedef struct          /* KB_DATA 结构: */
{
    int key_found;      /* 如果等于KB_KEY_FOUND, 键被按下。*/
    KEY_SEQUENCE key_seq; /* 键的字符和扫描码(如果key_found是 */
    /* KB_KEY_FOUND)。 */
    void *pfunction_data; /* 指向用户提供的信息结构的指针。 */
    int control_action;   /* 键控制函数采取的特殊动作; */
    /* KB_NO_ACTION: 不做任何操作。 */
    /* KB_FLUSH: 清空缓冲区。 */
    /* KB_REMOVE_KEY: 允许从BIOS键盘缓冲 */
    /* 区中删除挂起的按键。 */
    /* KB_NO_REMOVE_KEY: 不得从BIOS键 */
    /* 缓冲区中删除挂起的按键。 */
    int returned_actions; /* 从键控制函数返回时调用程序所采取的 */
    /* 动作。 */
    /* KB_REMOVE_KEY: 调用程序应该删除任何 */
    /* 已发现的按键。 */
    /* KB_NO_REMOVE_KEY: 不删除已发现的按键。 */
} KB_DATA;
```

## 高级键盘管理函数源程序和使用参考

## KBEQUIP

## 检测键盘环境

```
#include <bkeybrd.h>
```

```
int kbequip(void);
```

(返回) 扩展BIOS键盘功能是否可用:KB\_EXTENDED (1) 或KB\_NOEXTENDED (0)。

KBEQUIP检测扩展BIOS 键盘服务例程是否存在。如果发现扩展BIOS键盘服务例程,则存在一个增强的键盘。KBEQUIP装载下面这些全局变量:

变量装入值

b_kbnhan	KB_ENHANCED	(1)	检测到增强的键盘。
	KB_NOENHANCED	(0)	未发现。
b_kbxten	KB_EXTENDED	(1)	检测到扩展BIOS。
	KB_NOEXTENDED	(0)	未发现。

KBEXTEND 选用扩展BIOS键盘服务。

源程序(KBEQUIP.C):

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

```
#include <bkeybrd.h>
```

```
int b_kbxten = KB_NOEXTENDED; /* Presence of extended BIOS functions. */
```

```
int b_kbnhan = KB_NOENHANCED; /* Presence of enhanced keyboard. */
```

```
int b_kbusex = KB_USE_NORMAL; /* Flag controlling whether to use  
/* normal or extended keyboard functions*/
```

```
typedef struct /* KEYPAIR: Character and key */
```

```
{ /* code as stored in BIOS */
```

```
char ch; /* keyboard buffer. */
```

```
char keycode;
```

```
} KEYPAIR;
```

```
int kbequip()
```

```
{  
static int did_kbequip = 0; /* Flag indicating whether we've*/  
/* done presence test before. */
```

```
int i, ints_were_on;
```

```
struct /* Copy of BIOS keyboard buffer */
```

```
{  
unsigned int head;  
unsigned int tail;  
KEYPAIR queue[16];
```

```
} savebuf;
```

```
union REGS inregs, outregs;
```

```
if (!did_kbequip) /* Skip if we've done this before. */
```

```
{  
ints_were_on = utintflg(UT_INTOFF); /* Prevent keystrokes */  
/* while testing. */  
/* Save copy of keyboard buffer */
```

```
utmovmem(KB_BUFHEADADDR,  
(char far *) &savebuf,  
sizeof(savebuf));
```

```
/* Flush the buffer. */
```

```
utpokew(KB_BUFHEADADDR, savebuf, tail);
```

```

inregs.x.ax = 0x05ff;          /* Stuff key sequence 0x05ff */
inregs.x.cx = 0xffff;          /* using extended function 5. */
int86(KB_BIOS_INT,&inregs,&outregs);
utintflg(UT_INTOFF);
if (outregs.h.al == 0)          /* Successfully stuffed? */
    for (i = 0; b_kbxten == KB_NOEXTENDED && i < 16; i++)
    {
        inregs.h.ah = 0x10;     /* Try extended function 0x10 */
                                   /* to read the data back. */
        int86(KB_BIOS_INT,&inregs,&outregs);
        utintflg(UT_INTOFF);
        if (outregs.x.ax == 0xffff) /* Found the data, so */
            b_kbxten = KB_EXTENDED; /* must be extended. */
    }

if (b_kbxten == KB_EXTENDED
    && (0x10 & utpeekb(uttofaru(KB_DATASEG,0x0096))))
    b_kbnhan = KB_ENHANCED;
                                   /* Restore keyboard buffer. */
utmovmem((char far *) &savebuf,
        KB_BUFHEADADDR,
        sizeof(savebuf));
did_kbequip = 1;                  /* Prevent repeated tests. */
utintflg(ints_were_on);
}
return b_kbxten;
}

```

## KBEXTEND 选用扩展的或一般的BIOS键盘服务

#include <bkeybrd.h>

int kbextend(int selection);

selection KB\_USE\_EXTEND (1)或KB\_USE\_NORMAL (0)。

(返回) 如果new\_state不可用, 返回KB\_ERROR(-1);

否则返回原状态KB\_USE\_EXTEND (1) 或KB\_USE\_NORMAL (0)。

KBEXTEND检测KBGETKEY 和KBREADY, 决定选用扩展的还是传统的BIOS键盘服务。如果操作成功地完成, 则返回所作出的选择。

如果选择了KB\_USE\_EXTEND 而扩展BIOS键盘服务例程不存在, 则出现错误。

源程序(KBEXTEND.C):

#include <bkeybrd.h>

int kbextend(new\_state)

int new\_state;

```

{
    int result;
    if (new_state == b_kbsex)          /* If nothing to do, */
        result = b_kbsex;              /* just return setting. */
    else if (new_state == KB_USE_NORMAL) /* If requesting traditional */
    {                                   /* services, */
        result = b_kbsex;              /* just return former state */
    }
}

```



```

        b_kbusex = new_state;          /* and set new state.      */
    }
    else if (new_state == KB_USE_EXTEND) /* If requesting extended */
    {                                   /* services,                */
        if (kbequip() == KB_EXTENDED) /* first check for presence. */
        {                             /* Extended services are     */
            result = b_kbusex;         /* available.                */
            b_kbusex = new_state;
        }
        else
            result = KB_ERROR;          /* Error: extended services */
                                         /* are not available.        */
    }
    else
        result = KB_ERROR;              /* Unknown request.          */
    return result;                      /* Return former value or    */
                                         /* error flag.                */
}

```

## KBFLUSH      废弃所有在键盘缓冲区中等待的按键

```
#include <bkeybrd.h>
```

```
int kbflush(void);
```

(返回)      被废弃的等待按键的数量

**KBFLUSH** 用BIOS废弃并清除在BIOS超前键入缓冲区中等待的所有按键，返回值表明有多少按键被废弃。

注意：**KBFLUSH**不关闭中断，这样在**KBFLUSH**返回其调用者之前可能会出现另一个按键，这个按键不会被清洗掉。即使在调用**KBFLUSH**之前关闭中断，也存在一个短暂的间隙，其间也许会出现按键。

若使用了键控制函数，可以用**KBKCFUSH**来清空按键。

源程序(KBFLUSH.C):

```
#include <bkeybrd.h>
```

```
int kbflush()
```

```

{
    int num_found, scan;
    char ch;
    num_found = 0;          /* None found yet.          */
    while (kbready (&ch, &scan) != 0)
    {
        ++num_found;        /* Found a waiting character. */
        (void) kbgetkey (&scan); /* Read and discard it.      */
    }
    return (num_found);
}

```

## KBGETKEY      等待读入下一个按键

```
#include <bkeybrd.h>
```

int kbgetkey(int \*pkey);

pkey            按键的键码。

(返回)            按键的字符码。

KBGETKEY等待一个按键(除非已有一个按键在BIOS超前键入缓冲区中等待), 然后返回该字符的ASCII码以及代表所按物理键的键码。

特殊的头文件bkeys.h中列出了所有字符序列、字符码和键码。

用KBREADY可以对按键作非破坏性查询, KBWAIT则使用键控制函数等待一个按键。

源程序(KBGETKEY.C):

```
#include <dos.h>
#include <bkeybrd.h>
int kbgetkey (pscan)
int *pscan;
{
    union REGS regs;
        /* Set up for BIOS call to retrieve key from          */
        /* keyboard buffer, or wait for a key if none is       */
        /* present.                                             */
    regs.h.ah = (( b_kbusex != KB_USE_NORMAL
        && kbequip() == KB_EXTENDED0x10 : 0x00);
        /* Do the call.                                         */
    int86 (KB_BIOS_INT, &regs, &regs);
        /* Return appropriate values: scan and character      */
        /* codes.                                              */
    *pscan = (int) regs.h.ah;
    return ((int) regs.h.al);
}
```

## KBKCFUSH      通过键控制函数废弃所有的等待按键

#include <bkeybrd.h>

void kbkcfsh(    PKEY\_CONTROL pfunc,  
                 void \*pdata);

pfunc            键控制函数的地址。如果不存在键控制函数, 则为NIL。

pdata            传给键控制函数的补充信息的地址。

KBKCFUSH废弃并消除全部等待按键。

KBFLUSH只使用了BIOS而未使用键控制函数来清空按键。

源程序(KBKCFUSH.C):

```
#include <bkeybrd.h>
void kbkcfsh(pfunction, pfunction_data)
PKEY_CONTROL pfunction;
void *pfunction_data;
{
    KB_DATA key_control_data;
    if (pfunction == 0)
    {
        kbflush();
        return;
    }
}
```

```

key_control_data.key_seq.character_code    = 0;
key_control_data.key_seq.key_code         = 0;
key_control_data.pfunction_data           = pfunction_data;
key_control_data.control_action            = KB_FLUSH;
key_control_data.key_found                 = KB_NO_KEY_FOUND;
key_control_data.returned_action           = KB_FLUSH;
pfunction(&key_control_data);
if (key_control_data.returned_action == KB_FLUSH)
    kbflush();
return;
}

```

## KBPLACE 在键盘缓冲区中放置一个按键

```
#include <bkeybrd.h>
```

```
int kbplace(    int at_head,
                char value,
                char key);
```

at\_head      KB\_HEAD (1)    按键被放在队列的队首(下一个要读入的按键)。

KB\_TAIL (0)    按键被放在队列的队尾。

value        按键的ASCII值(如果不是ASCII码, 则为零)。

key          按键的键码(扫描键码或扩展码)。

(返回)      如果成功, 为0; 如果缓冲区满, 则为1。

如果BIOS 超前键入缓冲区中有空间的话, KBPLACE在这个缓冲区的头部或末尾放入(“插入”)一个按键。如果没有空间, 则返回错误代码1。

KBPLACE假设BIOS超前键入缓冲区位于标准位置0x40:0x1。这个函数有时会执行失常(例如, 如果安装了某个键盘增强程序)。

源程序(KBPLACE.ASM):

```

include beginasm.mac          ; Specifies the C compiler
beginProg kbplace

KB_HEAD equ 0
KB_TAIL equ 1

where equ word ptr [bp + stkoff + 0] ; Addresses of arguments
value equ byte ptr [bp + stkoff + 2]
scan equ byte ptr [bp + stkoff + 4]
buffer_head equ word ptr ds:1ah
buffer_tail equ word ptr ds:1ch
kb_buffer equ 1ch
kb_buffer_end equ 3ch

; Beginning of actual code
push bp
mov bp, sp
push ds
pushf ; Save interrupt state.
mov ax, 40h ; Segment address of BIOS data
mov ds, ax
assume ds:nothing
cli ; Disable interrupts.
cmp where, KB_HEAD

```

```

        jz      try_at_head
        cmp     where, KB_TAIL
        jz      try_at_tail
        mov     ax, 2
        jmp     short exit
try_at_head:
        mov     bx, buffer_head
        dec     bx                ; Tentatively step head back.
        dec     bx
        cmp     bx, kb_buffer
        jae     k5h
                                ; Went past beginning of buffer,
        mov     bx, kb_buffer_end-2; so wrap to end.
k5h:
                                ; Now BX has new head pointer.
        cmp     bx, buffer_tail
        je      no_room
        mov     al, value
        mov     ah, scan
        mov     [bx], ax        ; Store value & scan code
                                ; at buffer head.
        mov     buffer_head, bx ; New head pointer.
        jmp     short success
try_at_tail:
        mov     bx, buffer_tail
        mov     cx, bx
        inc     cx                ; Tentatively advance new tail.
        inc     cx
        cmp     cx, kb_buffer_end
        jb      k5
                                ; Went past end of buffer,
        mov     cx, kb_buffer    ; so wrap to beginning.
k5:
                                ; Now CX has new tail.
        cmp     cx, buffer_head
        je      no_room
        mov     al, value
        mov     ah, scan
        mov     [bx], ax        ; Store value & scan code
                                ; at buffer tail.
        mov     buffer_tail, cx ; New tail pointer.
success:
        xor     ax, ax
        jmp     short exit
no_room:
        mov     ax, 1
exit:
        popff                    ; Restore interrupt state.
        pop     ds
        pop     bp

```

```

ret
endProg kbplace
end

```

## KB POLL 通过一个键控制函数查看下一个等待按键

```
#include <bkeybrd.h>
```

```
int kbpoll( PKEY_CONTROL pfunc,
            void *pdata,
            KEY_SEQUENCE *pkey_seq,
            int option);
```

pfunc 键控制函数地址。如果不存在键控制函数，则为NIL。

pdata 传给键控制函数的补充信息的地址。

pkey\_seq 结构的地址，该结构返回按键的字符码和键码。

option KB\_NOREMOVE\_KEY(0) 允许再次读按键(即非破坏性读取);  
KB\_REMOVE\_KEY (1) 删除该按键(破坏性读取)。

(返回) 如果有一个字符可用，则为1；如果没有，则为0。

如果有一个字符可用，KB POLL返回在BIOS超前键入缓冲区中的下一个字符及键码。如果没有字符可用，则在\*pkey\_seq中返回的值无定义。

KB POLL与KB WAIT相似，但后者不返回，直至再有一个字符可用。WB WAIT总是删除这个等待字符而KB POLL可以让该字符被再次读取。

利用KB READY可以仅使用BIOS而不用键控制函数来查询键盘。

源程序(KB POLL.C):

```
#include <bkeybrd.h>
```

```
PKEY_CONTROL b_key_ctrl = 0; /* Default key control procedure*/
/* for user input functions. */
```

```
int kbpoll(pfunction, pfunction_data, pkey_sequence, option)
```

```
PKEY_CONTROL pfunction;
```

```
void *pfunction_data;
```

```
KEY_SEQUENCE *pkey_sequence;
```

```
int option;
```

```
{
```

```
KB_DATA key_control_data;
```

```
char dummy_character = '\0';
```

```
int dummy_key = 0;
```

```
int key_found;
```

```
key_found = kbready(&dummy_character, &dummy_key);
```

```
/* If there is no key control function, remove the keystroke */
```

```
/* (if there was one), and return KBREADY's data. */
```

```
if (pfunction == 0)
```

```
{
```

```
if (key_found && (option == KB_REMOVE_KEY))
```

```
dummy_character = kbgetkey(&dummy_key);
```

```
pkey_sequence->character_code = dummy_character;
```

```
pkey_sequence->key_code = (unsigned char) dummy_key;
```

```
return(key_found ? KB_KEY_FOUND : KB_NO_KEY_FOUND);
```

```
}
```

```
key_control_data.key_seq.character_code = dummy_character;
```

```
key_control_data.key_seq.key_code = (unsigned char) dummy_key;
```

```
key_control_data.pfunction_data = pfunction_data;
```

```

key_control_data.control_action      = option;
key_control_data.returned_action     = option;
key_control_data.key_found           = key_found
                                     ? KB_KEY_FOUND
                                     : KB_NO_KEY_FOUND;

pfuction(&key_control_data);
*pkey_sequence = key_control_data.key_seq;
/* Check to see if we should remove a keystroke from the buffer.*/
if ((option == KB_REMOVE_KEY)      &&
    (key_control_data.returned_action == KB_REMOVE_KEY) &&
    key_found                      &&
    (kbready(&dummy_character, &dummy_key)))
{
    kbgetkey(&dummy_key);
}
return(key_control_data.key_found);
}

```

## KBQUERY 从标准IBM控制台读邓用户的响应

```
#include <bkeybrd.h>
```

```

char kbquery( char *presp,
              int resp_size,
              int *pkey,
              int *pscrolled);

presp      指向字符缓冲区的指针, 该缓冲区返回用户的响应。
resp_size  以字节计量的输入缓冲区的尺寸(包括用于尾随的NUL('\0'字符的一个字节)。
pkey       结束输入的按键键码。
pscrolled  变量的地址, 该变量返回 屏幕滚动的行数。
(返回)     结束输入的按键ASCII(字符)值。

```

KBQUERY从用户返回一串按键, 回显这些按键, 并将所有的I/O限定在当前显示设备的活动页中。

用户可能象通常那样键入字符, 用下列方法结束输入: ENTER、Ctrl-M、Ctrl-J、功能键或其它IBM扩展键序列。输入字符被回显, 光标在整个屏幕上移动。当到达屏幕的最右列时, 光标移向下一行; 到达屏幕的右下角时, 屏幕发生滚动。当缓冲区满时, 输入的记录过程即停止。

下列按键作为输入的一部分时具有特殊的效果:

Backspace或Ctrl-H('\8')删除原先输入的字符, 将光标移到那个字符的位置。在输入开始时的退格键没有任何效果。

ENTER、Ctrl-M('\15')或Ctrl-J('\12')结束输入。

NUL( ASCII ) 被忽略并被废弃。

TAB或Ctrl-I('\9')象通常那样回显成一个或多个空格, TAB字符在缓冲区中返回。

Ctrl-G('\7')引起一声鸣叫但不作为输入的一部分返回。

特殊的非ASCII IBM 键(象功能键和一些ALT键)结束输入。

所有其它ASCII字符, 包括控制字符, 均被回显。

(控制字符回显为它们本身, 也就是说没有上箭头('^')前缀。)

一旦输入缓冲区满, 光标便停在最后一个输入字符上, 以后的按键(除了退格和结束键)引起鸣叫并被废弃。退格键象通常那样工作, 但用户可能一直退格到输入的开始处。这时函数并不返回, 直到ENTER或其它结否键被按下(即使缓冲区已满)。当输入结束时执行一个回车和换行。

结束字符不作为输入的一部分返回, 结束字符的ASCII码作为函数值返回, 它的键码(扫描码)在\*pscan中返回。

KBQUERY使用其地址保存在b\_key\_ctrl中的键控制函数。

用FLPROMPT可以从标准输入中读。

源程序(KBQUERY.C):

```
#include <bkeybrd.h>
#include <bcreens.h>
#define BEL      7      /* Bell      (CTRL-G)      */
#define BS       8      /* Backspace (CTRL-H)    */
#define TAB      9      /* Tab      (CTRL-I)     */
#define LF       10     /* Line feed (CTRL-J)    */
#define ENTER    13     /* ENTER or carriage return (CTRL-M)*/
#define CR       13
#define TAB_WIDTH 8      /* Distance between tab stops (for */
                        /* echoing TAB characters).      */
char kbquery(presp,resp_size,pscan,pnum_scrolled)
char *presp;
int  resp_size, *pscan, *pnum_scrolled;
{
    unsigned char ch;
    char *pbegin = presp;
    char *ptemp;
    int  old_page, mode, columns, act_page;
    int  is_char,junk;
    int  row, col, begin_row, begin_col;
    int  num_have = 0;
    KEY_SEQUENCE kseq;
    old_page = b_curpage; /* Save former display page. */
    scmode(&mode,&columns,&act_page);
    spage(act_page); /* Direct I/O to active page. */
    *pnum_scrolled =
    num_have = 0;
    securst(&begin_row,&begin_col,&junk,&junk);
    row = begin_row;
    col = begin_col;
    do /* Collect the response. */
    {
        /* Await a keystroke. */
        kseq = kbwait(b_key_ctrl,NIL);
        *pscan = kseq.key_code;
        ch = kseq.character_code;
        if (ch == 0xe0)
            is_char = (kseq.key_code == 0);
        else
            is_char = ch;
        if (is_char)
        {
            /* Got an ASCII character. */
            switch (ch)
            {
                case LF:
                case ENTER:
                    break; /* This is a terminating character: */
            }
        }
    }
}
```

```

/* handle it at loop exit. */
case '\0': /* Ignore NULs. */
    break;
case BEL: /* BEL: just beep. */
    settywrt((char) BEL,1);
    break;
default:
    /* If buffer full... */
    if (num_have >= resp_size - 1)
        settywrt((char) BEL, 1);
    else
    {
        do /* Echo spaces or the char, noting */
        { /* any scrolling. */
            if ((col >= columns - 1) &&
                (row >= scrrows() - 1))
                ++*pnum_scrolled;
            settywrt((ch == TAB ? (char) ' ' : ch),1);
            securst(&row,&col,&junk,&junk);
        } while (ch == TAB && (col % TAB_WIDTH));
        /* Update response buffer. */
        num_have++;
        *presp++ = ch;
    }
    break;
case BS:
    if (num_have)
    {
        /* Move cursor to last char entered.*/
        num_have--;
        /* Delete TABs specially: */
        if (*--presp == TAB)
        {
            *presp = '\0';
            row = begin_row - *pnum_scrolled;
            col = begin_col;
            ptemp = pbegin;
            /* Find new cursor location by walking */
            /* forward through the response again. */
            while ((ch = *ptemp++) != '\0')
            {
                do
                {
                    if (++col > columns - 1)
                    {
                        row++;
                        col = 0;
                    }
                } while (ch == TAB && (col % TAB_WIDTH));
            }
        }
    }

```



```

        secursel(row,col);
        /* Don't need to erase with a blank */
        /* since the TAB was already displayed */
        /* using blanks. */
        /* ("row" may be negative here if */
        /* begin_row was scrolled off the */
        /* screen, but that can only happen if */
        /* user fills the screen with input.) */
    }
    else
    {
        /* Just delete a normal */
        /* character. */
        if (col) /* Back up one column. */
            secursel(row,--col);
        else /* Back up to previous row. */
            secursel(--row,col = columns - 1);
        /* Erase previous character.*/
        secwrite((char) '\b',1);
    }
    else
    {
        /* Do nothing */ /* Can't delete past*/
        /* beginning. */
    }
    break;
}

/* Note: at this point, row and col indicate the next */
/* position on the screen. */
} while (is_char && (ch != ENTER) && (ch != LF));
*presp = '\0'; /* Terminate response string. */
settywr((char) CR,1); /* Perform a final newline. */
settywr((char) LF,1);
if (row >= scrows() - 1)
    ++*pnum_scrolled;

scpage(old_page); /* Restore current display page. */
return ch; /* Successful exit. */
}

```

## KBQUEUE 报告键盘缓冲区总容量及剩余容量

#include <bkeybrd.h>

int kbqueue(int \*ptotal);

ptotal 指向一个变量的指针，该变量存放队列的总容量。

(返回) 超前键入队列中可用空间的大小。

KBQUEUE报告BIOS超前键入缓冲区的全部容量及其中尚未填满的空间的大小。

如果要使用返回值，最好先关闭中断(通过utintlg(UT\_INTOFF))再调用KBQUEUE，防止以后的按键或其它进程改变可用空间的数量。使用完这个值后，可以通过utintlg(UT\_INTON)重新打开中断。

KBQUEUE 认为BIOS超前键入缓冲区位于标准位置0x40:0x1c。若不在这个位置，这个函数有可

能操作异常，例如安装了某个键盘增强程序。

源程序(KBQUEUE.C):

```
#include <bkeybrd.h>
#include <butil.h>
int kbqueue (ptotal)
int *ptotal;
{
    unsigned int bufhead;
    unsigned int buftail;
    int avail, wason;
    *ptotal = KB_BUFACSIZE;
    bufhead = utpeekw (KB_BUFHEADADDR);
    buftail = utpeekw (KB_BUFTAILADDR);
    wason = utintoff ();
    if (bufhead > buftail)
        avail = (bufhead - buftail);
    else
        avail = (bufhead + KB_BUFSIZE - buftail);
    if (wason)
        utinton ();
    return ((avail >> 1) - 1);
}
```

## KBREADY      检查下一个等待按键

#include <bkeybrd.h>

int kbready(char \*pch,  
                  int \*pkey);

pch            如果一个字符已准备好，返回该字符。

pkey           如果一个字符已准备好，返回其键码。

(返回)        如果缓冲区中有一个字符，则为1；否则为0。

如果有一个字符就绪，KBREADY非破坏性地返回BIOS超前键入缓冲区中的下一个字符及键值1。如果缓冲区为空，则\*pch和\*pkey的值无定义。

KBREADY与KBGETKEY相似，但后者将执行挂起，直到缓冲区中有一个可用的字符。另外，KBGETKEY 从缓冲区中删除等待字符，而KBREADY 则让字符再次被读取。

利用KB POLL及一个键控制函数可以查询键盘。

源程序(KBREADY.ASM):

```
include beginasm.mac
extProc kbequip
extSym b_kbusex,word
extSym b_kbxtcn,word
KB_USE_NORMAL equ 0            ;Values for b_kbusex.
KB_USE_EXTENDED equ 1
KB_NOEXTENDED equ 0            ;Values for b_kbxtcn.
KB_EXTENDED equ 1
beginProg kbready
if longData
    pch equ dword ptr [bp + stloff + 0] ; Address of pch
```

```

    pscan equ dword ptr [bp + stloff + 4] ; Address of pscan
else
    pch equ word ptr [bp + stloff + 0] ; Address of pch
    pscan equ word ptr [bp + stloff + 2] ; Address of pscan
endif
; Beginning of actual code
push bp
mov bp, sp
cld
pushf
mov ch,1 ; BIOS keyboard function 1: test for key
; Check whether extended services requested.
mov dx,ds ; (Save DS.)
mov ax,seg b_kbusex@
mov ds,ax
assume ds:nothing
cmp ds:b_kbusex@,KB_USE_NORMALExtended services requested?
mov ds,dx ; (Restore DS.)
je after_check ; If not, use function 1.

call kbequip@ ; Sense presence of extended services.
cmp ax,KB_EXTENDED ; If extended services absent,
jne after_check ; just use function 1.

mov ch,11h ; Use extended service 11h

```

#### after\_check:

```

mov ah,ch
int 16h

jz nochar ; Skip getting the char if there is
; not one to get.

if longData
    push es
    les bx, pch ; Set up address, get character code.
    mov byte ptr es:[bx], al
    mov al, ah
    xor ah, ah
    les bx, pscan ; Set up address, get scan code.
    mov word ptr es:[bx], ax
    pop es
else
    mov bx, pch ; Set up address, get character code.
    mov byte ptr [bx], al
    mov al, ah
    xor ah, ah
    mov bx, pscan ; Set up address, get scan code.
    mov word ptr [bx], ax
endif

```

```

        mov     ax, 1                ; Return a success code.
        jmp     short endf

nochar:
        mov     ax, 0                ; Return a failure code.

endf:
        popff
        pop     bp
        ret
        endProg kbready
        end

```

## KBSCANOF 返回一个字符的键码

```
#include <bkeybrd.h>
```

```
int kbseanof(int ch);
```

ch 所需键码的对应字符。字符的整数值必须在0到127的范围内。

(返回) 如果ch不是一个合法的ASCCC字符，返回-1；否则返回属于该字符的键码(伪扫描码)。

KBSCANOF是一个宏，它返回一个一般的ASCII字符的键码(也称为扫描码)。字符的整数值必须在0至127范围的。

请参看特殊的头文件bkeys.h。

源程序(KBSCANOF.C):

```
#include <bkeybrd.h>
```

```
#include <bkeys.h>
```

```

unsigned char b_keycod [] = {
    KBNDEF,          KB_S_C_A,
    KB_S_C_B,        KB_S_C_C,
    KB_S_C_D,        KB_S_C_E,
    KB_S_C_F,        KB_S_C_G,
    KB_S_N_BACKSPACE, KB_S_N_TAB,
    KB_S_C_J,        KB_S_C_K,
    KB_S_C_L,        KB_S_N_ENTER,
    KB_S_C_N,        KB_S_C_O,
    KB_S_C_P,        KB_S_C_Q,
    KB_S_C_R,        KB_S_C_S,
    KB_S_C_T,        KB_S_C_U,
    KB_S_C_V,        KB_S_C_W,
    KB_S_C_X,        KB_S_C_Y,
    KB_S_C_Z,        KB_S_N_ESC,
    KB_S_C_BACKSLASH, KBNDEF,
    KBNDEF,          KB_S_C_MINUS,
    KB_S_N_SPACE,    KB_S_S_1,
    KB_S_S_QUOTE,    KB_S_S_3,
    KB_S_S_4,        KB_S_C_5,
    KB_S_S_7,        KB_S_N_QUOTE,
    KB_S_S_9,        KB_S_S_0,
    KB_S_S_8,        KB_S_S_EQUALS,

```

```

KB_S_N_COMMA, KB_S_N_MINUS,
KB_S_N_PERIOD, KB_S_N_SLASH,
KB_S_N_0,      KB_S_N_1,
KB_S_N_2,      KB_S_N_3,
KB_S_N_4,      KB_S_N_5,
KB_S_N_6,      KB_S_N_7,
KB_S_N_8,      KB_S_N_9,
KB_S_S_SEMI,   KB_S_N_SEMI,
KB_S_S_COMMA,  KB_S_N_EQUALS,
KB_S_S_PERIOD, KB_S_S_SLASH,
KB_S_S_2,      KB_S_S_A,
KB_S_S_B,      KB_S_S_C,
KB_S_S_D,      KB_S_S_E,
KB_S_S_F,      KB_S_S_G,
KB_S_S_H,      KB_S_S_I,
KB_S_S_J,      KB_S_S_K,
KB_S_S_L,      KB_S_S_M,
KB_S_S_N,      KB_S_S_O,
KB_S_S_P,      KB_S_S_Q,
KB_S_S_R,      KB_S_S_S,
KB_S_S_T,      KB_S_S_U,
KB_S_S_V,      KB_S_S_W,
KB_S_S_X,      KB_S_S_Y,
KB_S_S_Z,      KB_S_N_LBRACKET,
KB_S_N_BACKSLASH, KB_S_N_RBRACKET,
KB_S_S_6,      KB_NDEF,
KB_S_N_BACKQUOTE, KB_S_N_A,
KB_S_N_B,      KB_S_N_C,
KB_S_N_D,      KB_S_N_E,
KB_S_N_F,      KB_S_N_G,
KB_S_N_H,      KB_S_N_I,
KB_S_N_J,      KB_S_N_K,
KB_S_N_L,      KB_S_N_M,
KB_S_N_N,      KB_S_N_O,
KB_S_N_P,      KB_S_N_Q,
KB_S_N_R,      KB_S_N_S,
KB_S_N_T,      KB_S_N_U,
KB_S_N_V,      KB_S_N_W,
KB_S_N_X,      KB_S_N_Y,
KB_S_N_Z,      KB_S_S_LBRACKET,
KB_S_S_BACKSLASH, KB_S_S_RBRACKET,
KB_S_S_BACKQUOTE, KB_S_C_BACKSPACE};

```

## KBSET 设置移位键的当前状态

```
#include <bkeybrd.h>
```

```
void kbset(const KEYSTATUS *pkeybd);
```

pkeybd 结构的地址，该结构包含新键盘状态。

KBSET控制移位键和其它控制键的状态。如果某一位为on，则对应标志被置位，否则该位是off。KBSET还根据状态位的情况设置键盘指示灯。

KEYSTATUS结构在键盘头文件bkeybrd.h中定义如下:

```
typedef struct kstatus /*键盘状态结构。 */
{
    unsigned right_shift :1; /*右Shift键已按下。 */
    unsigned left_shift :1; /*左Shift键已按下。 */
    unsigned ctrl_shift :1; /* Ctrl键已按下。 */
    unsigned alt_shift :1; /* Alt 键已按下。 */
    unsigned scroll_state :1; /*Scroll Lock被转置。 */
    unsigned num_state :1; /*Num Lock 被置。 */
    unsigned caps_state:1; /*Caps Lock 被转置。 */
    unsigned _ins_state :1; /* Insert 状态已激活。 */
    /* */
    unsigned filler :3; /* 用于字对齐的填充字符。 */
    unsigned hold_state :1; /*挂起键已被转置。 */
    unsigned scroll_shift : 1; /*Scroll Lock键已按下。 */
    unsigned num_shift :1; /*Num Lock 键已按下。 */
    unsigned caps_shift :1; /* Caps Lock 键已按下。 */
    unsigned ins_shift :1; /*Insert 键已按下。 */
}KEYSTATUS;
```

KBSET在内部激活KBREADY。 如果程序使用扩展BIOS键盘服务,您可以在调用KBSET之前激活kbextend(KB\_USE\_EXTEND),这就可以防止KBREADY废弃对于常规BIOS键盘服务例程未知的按键,也可以防止将扩展按键翻译成为常规的对应按键。

源程序(KBSET.C):

```
#include <bkeybrd.h>
#include <butil.h>
void kbsset (pkeybd)
const KEYSTATUS *pkeybd;
{
    char tempchar;
    int tempint;
    utpokew (KB_SHIFTADDR, utpeekw (pkeybd));
    kbready (&tempchar, &tempint);
}
```

## KBSTATUS 报告移位键的当前状态

```
#include <bkeybrd.h>
int kbstatus(KEYSTATUS *pkeybd);
pkeybd 结构的地址,该结构返回键 状态。
(返回) 表示键盘状态的整数。
```

KBSTATUS是一个宏,它在位结构中返回移位键及其它控制键的状态。如果某位是on,则对应标志被置位;否则该位是off。

返回值是BIOS的两字节键盘状态信息的拷贝,低字节是BIOS功能INT 0x16, AH=2所返回的KEYSTATUS结构在键头文件bkeybrd.h 中定义如下:

```
typedef struct kstatus /* 键盘状态结构。 */
{
    unsigned right_shift : 1; /* 右Shift 键已按下。 */
    unsigned left_shift :1; /* 左Shift 键已按下。 */
    unsigned ctrl-shift :1; /* Ctrl 键已按下。 */
}
```

```

    unsigned alt_shift :1;      /*Alt 键已按下。      */
    unsigned scroll_state :1;    /*Scroll Lock 被转置。  */
    unsigned numb_state :1;     /*Num Lock被转置      */
    unsigned caps_state :1;     /*Caps Lock 被转置     */
    unsigned ins_state :1;      /*Insert 状态已激活。   */
    unsigned filler :3;         /*用于字对齐的填充字符。*/
    unsigned hold_state :1;     /*挂起键已被转置。     */
    unsigned scroll_shift :1;    /*Scroll Lock键已按下。 */
    unsigned numb_shift :1;     /* Num Lock 键已按下。  */
    unsigned caps_shift :1;     /* Caps Lock 键已按下。 */
    unsigned ins_shift :1;      /*Insert 键已按下。    */
}KEYSTATUS;

```

源程序(BKEYBRD.H);

参见附录C中的BKEYBRD.H头文件,该功能被定义为宏。

## KBSTUFF      强行将一个字符串送入BIOS超前键入缓冲区

```
#include <bkeybrd.h>
```

```
char *kbstuff(    int at_head,
                 char *pstring);
```

at\_head      KB\_HEAD(1): 把字符串放在队列的头部。

KB\_TAIL (0): 把字符串放在队列的尾部。

pstring      字符串首字符的地址。字符串必须以NUL ('\0')结尾。

(返回)      字符串中未插入到队列的首字符的地址。若全部放入队列,则指向尾随的NUL('\0')。

KBSTUFF强行将一个按键字符串放到BIOS超前键入队列中,就好象它们是被用户键入的一样。

字符串可以安插在队列的尾部(好象是最近键入的)或安插在头部(最早键入的)。

如果要把字符串中的字符放到队列的尾部,则尽可能多的字符将被插入,指向未插入的首字符的指针作为函数值返回。

如果要把字符串放在队列的头部(下一个要读入的),则仅当它们都能放入队列时才能插入。(这样从来不会有超过15个字符的字符串能够插入到队列的头部。)

用KBQUEUE可以知道键盘缓冲区中还剩余多少空间。

对于字符串中一般的ASCII字符,KBSTUFF 插入0作为键码。

要指明字符串中的扩展键序列,可以使用字符'\377',其后尾随该键值。

利用KBPLACE可以将一个字符插入队列中。参见KBSCANOF中的第二个示例。

源程序(KBSTUFF.C);

```
#include <bkeybrd.h>
```

```
#include <butil.h>
```

```
#define EXT_FLAG ((char) '\377') /* Indicates scan code of          */
                                   /* extended key sequence.      */
```

```
char *kbstuff(where,pstring)
```

```
int    where;
```

```
char *pstring;
```

```
{
    int            length,i;
    register int    j;
    register char *ptr;
    char           *pend;
    int            ints_were_on;

```

```

int          total_size;
char         value,scan;
if (where == KB_TAIL)
{
    /* Stuff at tail of queue. */
    /* Turn interrupts off. */
    ints_were_on = utintoff();
    for (; *pstring; pstring++)
    {
        if (*pstring == EXT_FLAG)
        {
            value = '\0';
            scan = *++pstring;
        }
        else
        {
            value = *pstring;
            scan = '\0';
        }
        if (kbplace(where,value,scan)) /* Try to stuff it. */
            break; /* Quit if it didn't fit. */
    }
}
else
{
    /* Stuff at head of queue. */
    length = 0;
    for (ptr = pstring; *ptr; ptr++)
    {
        /* Measure length of string.*/
        length++;
        if (*ptr == EXT_FLAG)
            ptr++;
    }
    pend = ptr; /* Pointer to end of string.*/
    ints_were_on = utintoff(); /* Turn interrupts off. */
    if (length <= kbqueue(&total_size))
    {
        /* The string fits. */
        /* Stuff each character in */
        /* reverse order. */
        for (i = length; i; i--)
        {
            ptr = pstring;
            for (j = 1; j < i; j++)
            {
                /* Step past first (i-1) */
                /* characters in string. */
                if (*ptr == EXT_FLAG)
                    ptr++;
                ptr++;
            }
            /* Now ptr points to i-th */
            /* character in string. */
            if (*ptr == EXT_FLAG)
                kbplace(where,
                    (char) 0,
                    *(ptr + 1)); /* Scan code only. */
        }
    }
}

```



```

        else
            kbplace(where,
                    *ptr,          /* Character only. */
                    (char) 0);
    }
    pstring = pend;          /* All characters stuffed. */
}
else
{
    /* Do nothing. */          /* String too big: No */
                                /* characters stuffed. */
}
}
if (ints_were_on)
    utinton();              /* Turn interrupts back on. */
return pstring;             /* Return remainder of string. */
}

```

## KBWAIT 等待并通过键控制函数读取下一个按键

```
#include <bkeybrd.h>
```

```
KEY_SEQUENCE kbwait (PKEY_CONTROL pfunc,
                    void *pdata);
```

pfunc 键控制函数的地址。如果不存在键控制函数，则为NIL。

pdata 传给键控制函数的其它信息的地址。

(返回) 字符码和按键的键码。

KBWAIT等待一个按键然后返回该字符的ASCII码及代表按下的物理键的键码。

作为结果的字符码和键码以KEY\_SEQUENCE结构返回，该结构在头文件bkeybrd.h中定义如下：

```

typedef struct          /*KET_SEQUENCE结构: */
{
    /*一个键的字符和键码。 */
    unsigned char character_code;
    unsigned char key_code;
}KEY_SEQUENCE;

```

要了解详情可能的返回值，请参看特殊的头文件bkey.h。

利用KB POLL并通过一个键控制函数可以非破坏性地查询键盘。KBGETKEY 只使用BIOS进行等待并删除一个按键，它不使用键控制函数。

源程序(KBWAIT.C):

```
#include <bkeybrd.h>
```

```
KEY_SEQUENCE kbwait(pfunction, pfunction_data)
```

```
PKEY_CONTROL pfunction;
```

```
void *pfunction_data;
```

```

{
    KEY_SEQUENCE return_value;
    /* Loop until KB POLL reports a keystroke has been found. */
    while (kbpoll(pfunction, pfunction_data, &return_value,
                 KB_REMOVE_KEY) != KB_KEY_FOUND)
        ;
    return(return_value);
}

```

## CMKEY.C键盘宏程序示例

CMKEY是一个终止并驻留程序，用来截获某些热键，并且那些键按下时向键盘缓冲区填写扩展的字符串。把字符串填进键盘缓冲区，然后把控制传回前台进程。所有运行该键盘增强程序的中断支持皆由插入码函数提供。

CMKEY要求调度程序控制每个计时中断，以便如果只填充了字符串的一部分，可在下一时钟中断时填充其余部分，直到填充了整个字符串。

CMKEY如果正在处理一个热键时，接收到中一热键，则第二个热键被忽略。

CMKEY从插入码调度程序传递的IV\_EVENT结构中恢复按键。

命令行格式中：

cmkey [-r]

CMKEY是自删除中断服务子程序，如果调用CMKEY时没用参数且没有安装，此程序安装自己。如果执行时用参数“-r”，则从内存中删除运行的复本。

CMKEY旨在说明插入码的功能，并提供一个中等规模及复杂度的插入码程序的设计和用法。插入码的说明请见插入码一章。

```
#include <dos.h>
#include <stdio.h>
#include <binterv.h>
#include <bintrupt.h>
#include <bkeybrd.h>
#include <bkeys.h>

/* The declaration of the intervention function. */
/* It will be called on every timer tick, and also */
/* when a hot key is pressed. */

void tickhandler (IV_EVENT *);
#define TRUE 1
#define FALSE 0
#define NUL '\0'
#define STKSIZE 2000
#define OK 0
#define FALL_OUT 101
#define NOT_FOUND 1
#define ERROR_DISABLING2
#define ERROR_REMOVING3
#define ALREADY_INSTALLED 4

/* Definitive signature for this version of CMKEY. */
#define CMKEY_SIGN "CMKEY/31/89"

/* Allocation of stack space for the intervention */
/* scheduler and user function. */

char schedstk [STKSIZE];

/* This is the data structure to pass to the */
/* scheduler so that it will give control every */
/* clock tick. */

IV_TIME timetab = {1, IV_TM_INTERVAL};

/* This is the data structure to pass to the */
/* scheduler so that it will give control every */
/* time ALT 1, 2, or 3 is pressed, and will put the */
/* intervention code to sleep when ALT-minus is */
/* detected, and will wake it up again when ALT- */
/* equals is selected. */

IV_KEY keytab [] =
```

```

{
    {KB_C_A_1,    KB_S_A_1,    IV_KY_SERVICE},
    {KB_C_A_2,    KB_S_A_2,    IV_KY_SERVICE},
    {KB_C_A_3,    KB_S_A_3,    IV_KY_SERVICE},
    {KB_C_A_MINUS,KB_S_A_MINUS,IV_KY_SLEEP},
    {KB_C_A_EQUALS,KB_S_A_EQUALS,IV_KY_WA
};
#define k(c) kbscanof(c)
static char *string_table [] =
{
    "cls\r",
    "dir /w /p *.c\rdir /w /p *.asm\r",
    "<this is a test of the emergency broadcast system>"
};
static char *pstuff = NIL;
static int     awake = TRUE;
                /* Internal functions -- install and remove */
                /* intervention function. */
int install_iv (void);
int remove_iv  (void);
void main(int, char **);
void main (argc, argv)
int  argc;
char *argv [];
{
    if (argc == 1)
        exit (install_iv ());
    if ((argc == 2)
        (((argv [1][0] == '-') || (argv [1][0] == '/')) &&
         ((argv [1][1] == 'r') || (argv [1][1] == 'R'))))
        exit (remove_iv ());
    printf ("usage: cmkey [-r]\n");
    exit (0);
}
/**

```

INSTALL\_IV -- 安装CMKEY的中断矢量，终止并驻留。

语法           ret = install\_iv ();

int ret           如果遇到错即从IVINSTAL返回代码。

如果没有安装，CMKEY则安装之。

返回           ret       ALREADY\_INSTALLED (4)  
                  已经安装了CMKEY。

FALL\_OUT (101)

ISRESEXT()失败。

IV\_INSTALLED (5)

CMKEY已安装。

\*/

int install\_iv ()

```

{
    int      icode;
    IV_VECTORS vecs;

```

```

        /* Check to see if CMKEY already installed.          */
ivvecs (IV_RETVEC, &vecs);
if (ivsense (&vecs, CMKEY_SIGN) != FARNIL)
{
    puts ("CMKEY already installed.");
    return (ALREADY_INSTALLED);
}

    /* Install the interrupt service routine--i.e. tell */
    /* the scheduler about our tick and keypress          */
    /* handler routine.                                   */
if (0 !=
    (rcode =
        ivinstal (tickhandler, CMKEY_SIGN, schedstk, STKSIZE,
            keytab,
            sizeof(keytab) / sizeof(IV_KEY),
            &timetab,
            sizeof(timetab) / sizeof(IV_TIME),
            /* Neither DOS services nor DOS "key" services nor */
            /* TURBO C floating point support is needed.        */
            IV_NO_DOS_NEED | IV_NO_DKEY_NEED | IV_NO_FLOAT_NEED)))
{
    /* Error!                                                */
    printf ("TVINSTAL error %d.\n", rcode);
    return (rcode);
}

    /* Terminate and stay resident.                        */
isresext (OK);

    /* Should never get here.                               */
return (FALL_OUT);
}

```

/\*\*

REMOVE\_IV -- 删除已安装的CMKEY。

语法:

```
ret = remove_iv ();
```

int ret            返回代码。

NO\_ERROR(0)-

没有遇到错误。

NOT\_FOUND (1)-

CMKEY未找到。

ERROR\_DISABLING (2)-

CMKEY不能关闭。该错误不可见。

ERROR\_REMOVING (3)-

CMKEY不可删除(因为MALLOC指针被 重写。

函数从内存删除活动的CMKEY, 恢复中断矢量, 并释放内存。

返回            ret (如果有错返回 上述错误码。).

\*/

```
int remove_iv ()
```

```
{
```

```
    IV_VECTORS vecs;
```

```
    IV_CTRL far *pivctrl;
```

```
    /* Check to see if CMKEY installed.          */
```

```
    */
```

```

ivvecs (IV RETVEC, &vecs);
if ((pivctrl = ivsense (&vecs, CMKEY_SIGN)) == FARNIL)
{
    puts ("CMKEY not found.");
    return (NOT_FOUND);
}

if (ivdisabl (pivctrl))
{
    puts ("Error disabling CMKEY.");
    return (ERROR_DISABLING);
}

if (isremove (pivctrl->psp))
{
    puts ("Error removing CMKEY.");
    return (ERROR_REMOVING);
}

return (0);
}
/**

```

**TICKHANDLER** --处理计时和热键事件。

语法 只由插入码调度程序调用。

**tickhandler** (pevent);

**IV\_EVENT \*pevent** 指向插入事件结构的指针。结构包含检测到的热键和其它数据。

函数从每次计时中断的调用中，或每次热键按下时接受控制。两者有很大的不同：

1. 热键 如果正在处理其它热键则放弃后续热键。如果没有正在处理的热键，则尽可能多地填充热键的扩展字符串到缓冲区中，然后生成一个由计时插入码填充其余的字符的指针。
2. 计时中断 CMKEY试着填充热键扩展字符串的其余字符到缓冲区，如果一次未能填充，则调用下一次。

返回 无。

\*/

**void tickhandler** (pevent)

**IV\_EVENT \*pevent;**

```

{
    /* Do a key action, if one exists. */
    switch (pevent->key.action)
    {
        /* If there is no key action, ignore the hot key
        /* part of the handler. */
        case IV_KY_NONE:
            break;
            /* If the user typed a "service" key, start stuffing*/
            /* the appropriate string. */
        case IV_KY_SERVICE:
            /* If we are already stuffing a string, discard this*/
            /* hot key. If we are asleep, ignore it. */
            if ((pstuff == NIL) && awake)

```

```

switch (pevent->key.keycode)
{
    case KB_S_A_1:
        pstuff = string_table [0];
        break;
    case KB_S_A_2:
        pstuff = string_table [1];
        break;
    case KB_S_A_3:
        pstuff = string_table [2];
        break;
}
break;
    /* If we get a message to go to sleep, do it.          */
case IV_KY_SLEEP:
    awake = FALSE;
    utsound (800, 1);
    break;
    /* If we get a message to wake up, do it.              */
case IV_KY_WAKE:
    awake = TRUE;
    utsound (1600, 1);
    break;
}

    /* No we do the actual stuffing of the string, if      */
    /* any.                                                  */
if (pstuff != NIL)
{
    /* Use KBPLACE to stuff them in, character by         */
    /* character, until full.                               */
    for (;
        (*pstuff != NUL) &&
        (kbplace (KB_TAIL, (int) *pstuff, kbscanof (*pstuff)) == 0);
        pstuff++)
    ;
    /* If we have come to the end of the string, we        */
    /* must set pstuff to NIL so that we know we can        */
    /* accept another hot key now.                           */
    if (*pstuff == NUL)
        pstuff = NIL;
}
}

```

## 第八章 内存管理高级程序设计

内存管理函数协助Turbo C程序对DOS内存控制块进行管理, 保证驻留程序的可靠性。

### DOS内存控制块结构

我们知道, 每个内存块与一个节的边界对齐, 它的前面有一个与之关联的控制块, 该控制块为一个节(十六字节)长。内存管理头文件bmem.h定义了MEMCTRL结构, 以此作为管理控制块的手段。下面是MEMCTRL的定义:

```
typedef struct      /* MEMCTRL : DOS内存控制块结构。 */
{
    char    dummy;    /* 使无符号成员做字对齐的哑元, 使得 */
                    /* 结构内部没有浪费的空间。 */
                    /* 实际控制块的16字节长的拷贝从这儿 */
                    /* 开始: */
    char ident;        /* 标识码。 */
    unsigned owner_psp; /* 所有者进程PSP的段地址。 */
    unsigned size;      /* 以节为单位的内存块的大小。 */
    char _reserved[11]; /* 很明显, 用于DOS */
} MEMCTRL;
```

结构由一个名为-dummy额外字节开始。包含\_dummy的目的是为了防止编译器的结构边界对齐机制在结构内部产生多余的空间。控制块的十六字节内容实际上从结构的ident成员开始。

ident成员包含一个'M'或'Z'来标记这个块是合法的内存控制块。'Z'仅出现在最后一个控制块上, 即具有最高内存地址的控制块。

owner\_psp成员包含着“拥有”这个内存块的程序的段地址。例如, 当程序通过DOS功能0x48分配内存时, 已分配内存块的控制块被标记上该程序的PSP(程序段前缀)的段地址。

处于“自由”状态(未分配给任何程序)的内存块的owner\_psp被标记为零。

size包含内存块(不包括控制块)的尺寸(以节为单位)。

### 内存管理函数

MMSIZE	报告程序所占用的内存块的大小。
MMCTRL	取得与给定内存块关联的内存控制块, 计算下一个内存块的地址。
MMFIRST	报告系统中第一个内存控制块的地址。

### 内存管理函数源程序

#### MMCTRL 读取DOS内存控制块

```
#include <bmem.h>
int mmctrl( unsigned memblk,
            MEMCTRL *pctrlblk,
            unsigned *pnexthblk);
```

memblk DOS 功能0x48所分配的DOS内存控制块的段地址。

pctrlblk 结构的地址, 该结构返回与memblk关联的内存控制块的拷贝。

pnexthblk 变量的地址, 该变量返回下一个内存块的段地址。如果memblk非法或下一块不存在, 则返回零。

(返回) 返回的memblk的状态:

0: memblk是一个合法的DOS内存块但不是链中的最后一个。

9: memblk是一个非法的DOS内存块。

18: memblk是最后一个内存块。

MMCTRL读取与段地址memblk关联的控制块的拷贝并计算下一个内存块的地址。

源程序(MMCTRL.C):

```
#include <bmem.h>
#include <butil.h>
int mmctrl(memblk, pctrlblk, pnextblk)
unsigned memblk;
MEMCTRL pctrlblk;
unsigned *pnextblk;
{
    register int result;
    utpeekn(uttfarw(memblk-1,0), /* Fetch 16 bytes. */
            (char far *) &pctrlblk->ident,
            16);
    switch (pctrlblk->ident)
    {
        case 'M': /* Valid memory block, not the */
            result = 0; /* last. */
            *pnextblk = memblk + pctrlblk->size + 1;
            break;
        case 'Z': /* Last memory block in the */
            result = 18; /* chain. */
            *pnextblk = 0;
            break;
        default: /* Invalid memory block. */
            result = 9;
            *pnextblk = 0;
            break;
    }
    return result;
}
```

## MMFIRST 报告第一个内存块的地址

#include <bmem.h>

unsigned mmfirst(void);

(返回) 第一个DOS内存块的段地址(即页号)。如果出错, 返回0。

MMFIRST报告系统中第一个DOS内存块(即位于最低物理地址的内存块), 返回值是内存块第一个数据字节的段地址(见下面的示例。)

源程序(MMFIRST.C):

#include <dos.h>

#include <bmem.h>

#include <butil.h>

unsigned mmfirst()

{

union REGS regs;



```

struct SREGS sregs;
unsigned      result;
MEMCTRL      ctlblock;
unsigned      nextblock;
int           ercode;
regs.h.ah = 0x52;
intdosx(&regs,&regs,&sregs);
result = utpeekw(uttofaru(sregs.es,regs.x.bx - 2)) + 1;
ercode = mmctrl(result,&ctlblock,&nextblock);
if (ercode != 0 && ercode != 18)
    result = 0;
return result;
}

```

## MMSIZE          报告一个程序的尺寸

```

#include <bmem.h>
unsigned mmsize(void);

```

(返回)          存放一个程序的内存块的尺寸(以节为单位, 每节长度是十六字节)。如果MMCTRL报告错误, 则返回零。

MMSIZE报告程序代码和数据所占内存块的尺寸。报告的尺寸以节为单位, 每节十六字节长。程序可以“拥有”通过DOS功能0x48分配的附加内存块, MMSIZE不计入这些附加块。

源程序(MMSIZE.C):

```

#include <dos.h>                                /* For use with utpspseg.        */
#include <bmem.h>
#include <butil.h>
unsigned mmsize()
{
    MEMCTRLctlblock;
    unsigned nextblock;
    int      ercode;
    ercode = mmctrl(utpspseg,&ctlblock,&nextblock);
    if (ercode == 0 || ercode == 18)
        return ctlblock.size;
    else
        return 0;
}

```

## 第九章 选单程序设计

### 选单函数功能概述

这一章介绍了Turbo C TOOLS 6.0最为出色的特性之一：具有亮条移动和鼠标器支持的选单。

系统定义了一个按键动作的标准集，用户可以重定义、删除或暂时使任意的标准按键分配失效，也可以定义随意选择一个选项的键。

### 选单函数的种类

#### 建立、显示和释放选单

MNCREATE	分配并建立一个BMENU结构，包括用于容纳选单的Turbo CTOOLS的窗口。
MNDSPLAY	用可选的边界和标题在屏幕上显示一个选单。如果相关的话，它还将Turbo C字符窗口设置为与选单的视口相匹配。
MNVDISP	在视口中显示一个虚拟选单(带有可选的边界和标题，它们可以比选单小。)如果相关的话，它还将Turbo C字符窗口设置为与选单的视口相匹配。
MNDSTROY	从屏幕上取消一个选单，释放它的所有内部数据结构。如果选单具有活动光标，使该光标失效，Turbo C字符窗口(如果相关)被置为全屏幕。

#### 定义标准选项和按键

MNITEM	向选单中加入一个选项或删除一个已有的选项。
MNKEY	定义一个按键的效果或修改，删除一个已有的按键定义。
MNITMKEY	向选单加入一个选项及一个或多个ASCII字符，这些字符用于将亮条移到该选项上。

#### 定义Lotus形式的选项

MNLITEM	将一个选项连同说明字符串加入到选单中，在该选项被显亮的同时出现说明字符串。MNLITEM还可以修改或删除这样的选项。
MNLITKEY	与MNLITEM表现相同，但它还可以定义一个或多个ASCII字符，这些字符用于把亮条移到某选项上。

#### 使用鼠标器的准备工作

MNMSTYLE	为选单选择一个鼠标器的使用方式。
MNMOUSE	向MNREAD和MNLREAD认可的鼠标器事件表中加入、修改或删除一个选项。

#### 读取用户的反应

MNREAD	通过选单读取一个用户响应。
MNLREAD	通过Lotus\形式的选单读取用户的响应。

#### 亮条操作

MNHILITE	移动或取消亮条，可带有或不带有选项的Lotus形式的说明字符串。
----------	----------------------------------

#### 高级选单特性

## 按键动作和亮条移动

当调用MNREAD或MNLREAD时，用户接管控制。他必须按键来移动亮条或输入一个选择。您可以定义每个可能按键的确切动作。对按键动作的详细说明可以使得程序易于使用而不被例外情况所困扰。

任何按键可以执行下面的零个、任意或全部动作：

按键动作

符号	值	动作
MN_TRANSMIT	0x0010	从MNREAD或MNLREAD返回，报告所选项的行和列。
MN_BEEP	0x0020	使扬声器发声。
MN_DISABLE	0x0080	按键被忽略。
MN_ABORT	0x0100	从MNREAD或MNLREAD返回，报告所选项的行和列，返回一个错误代码。
MN_KBIGNORE	0x0200	忽略并废弃后面的按键，直至下一个认可的鼠标器操作。
MN_HIDE_BAR	0x0800	使亮条关闭，删去说明字符串。
MN_SHOW_BAR	0x1000	使亮条打开(取代MN_HIDE_BAR)。
MN_NOWRAP	0x4000	取代MN_UP、MN_DOWN、MN_RIGHT、MN_LEFT、MN_NEXT或MN_PREVIOUS操作中的回绕动作。

如果指明了MN\_DISABLE，其它可能值被忽略。

除了可能的动作之外，每个按键还使亮条以下列方式之一移动(除非声明了MN\_DISABLE)。

按键与亮条移动

符号	值	亮条移动
MN_NOMOVE	0x001	不移动。
MN_UP	0x001	同列上移。
MN_DOWN	0x002	同列下移。
MN_RIGHT	0x003	同行右移。
MN_LEFT	0x004	同行左移。
MN_NEXT	0x005	移到紧接本项安装之后安装的选项。
MN_PREVIOUS	0x006	移到恰在本项安装之前安装的选项。
MN_FIRST	0x007	移到第一个安装的选项。
MN_LAST	0x008	移到最后安装的选项。
MN_SELECT	0x009	移到指定位置上的选项。
MN_PGUP	0x00a	上移行数为视口高度。
MN_PGDN	0x00b	下移行数为视口高度。
MN_PGRIGHT	0x00c	右移列数为视口宽度。
MN_PGLEFT	0x00d	左移列数为视口宽度。

如果MN\_UP、MN\_RIGHT或MN\_LEFT亮条移动在发现另一个选项之前到达窗口的边沿，它将继续从下一行开头进行下去(即回绕)。MN\_NOWRAP动作位可以抑制这个效果。

MN\_NEXT、MN\_PREVIOUS、MN\_FIRST和MN\_LAST亮条移动依赖于选项安装的次序。当MN\_NEXT或MN\_PREVIOUS亮条移动操作到达选项内部表的尾端时即发生回绕，查找从表的另一端继续进行。MN\_NOWRAP动作位可以抑制回绕效果。如果某些选项被MNITEM删除后又被重新安装，这将影响MN\_NEXT、MN\_PREVIOUS、MN\_FIRST和MN\_LAST使用这些选项的顺序。)

## 缺省的键分配

每个选单通过内部函数MNDEFKEY设立下列缺省的按键动作和亮条移动：

缺省的选单键分配

键	ASCII值	键码	动作	亮条移动
上箭头	0	72		MN_UP
下箭头	0	80		MN_DOWN
右箭头	0	77		MN_RIGHT

左箭头	0	75		MN_LEFT
Home	0	71		MN_FIRST
End	0	79		MN_LAST
Tab	9	15		MN_NEXT
Shift-Tab	0	15		MN_PREVIOUS
PgUp	0	73		MN_PGUP
PgDn	0	81		MN_PGDN
Ctrl-右箭头	0	116		MN_PGRIGHT
Ctrl-左箭头	0	115		MN_PGLEFT
Enter	13	28	MN_TRANSMIT	MN_NOMOVE
Escape	27	1	MN_ABORT	MN_NOMOVE
增强键盘:				
上箭头	224	72		MN_UP
下箭头	224	80		MN_DOWN
右箭头	224	77		MN_RIGHT
左箭头	224	75		MN_LEFT
Home	224	71		MN_FIRST
End	224	79		MN_LAST
PgUp	224	73		MN_PGUP
PgDn	224	81		MN_PGDN
Ctrl-右箭头	224	116		MN_PGRIGHT
Ctrl-左箭头	224	115		MN_PGLEFT
Enter	13	224		MN_TRANSMIT

如果想修改缺省的键分配表，可以使全局变量**b\_mnkeytab**指向自己的表。详见**madekey.c**。  
通过**MNKEY**可以在任何时候修改或删除键分配。

## 控制内存分配

如果要建立一个中止并驻留的程序，请参见“窗口函数(WN)”一章中的相应部分“控制内存分配”。

由于选单函数利用了Turbo C TOOLS的窗口，所有有关窗口的注意事项也适用于选单。在程序中中止之前应建立自己的选单，设置所有的选项、键分配和鼠标器事件。对于每个选单请不要忘记执行下面语句

```
wnsetopt(pmenu->pwin, WN_PREV_ALLOC, 1);
```

## 选单扩展函数源程序和使用参考

### MNATR 改变菜单在项的属性。

```
#include <bmenu.h>
```

```
presult = mnatr (pmenu, pitem, mode);
```

**presult** 指向新创建的BMENU结构，如果失败，则为空。

**pitem** 要改变属性的项。

**mode** MN\_UNHIGHLIGHT把项的亮条取消，MN\_HIGHLIGHT把项加亮。

该函数改变指定菜单的给定项的属性。不改变菜单的窗口的其它项。

输出是延迟的。

\*返回 **presult** 指向新创建的BWINDOW结构，或如果失败则为NIL。  
**b\_werr** 可能为下面的值：  
(No change) 成功  
MN\_BAD\_MENU Pmenu无效

MN_BAD_ITEM	Pitem无效
WN_BAD_WIN	Pmenu->pwin无效
WN_NOT_SHOWN	内部错误
WN_BAD_DEV	内部错误
WN_NULL_PTR	内部错误

源程序(WNATRC):

```
#include <bmenu.h>
BMENU *mnatr (pmenu, pitem, mode)
BMENU *pmenu;
const BITEM *pitem;
int mode;
{
    /* Validate menu data structures. */
    mnvalidm (pmenu)
    /* Check validity of item. */
    if (pitem == NIL)
        wnreterr (WN_ILL_VALUE);
    if (pitem->signature != MN_ITEM_SIGN)
        wnreterr (MN_BAD_ITEM);
    /* Set attributes on the menu item. */
    if (mode == MN_HIGHLIGHT)
    {
        if (wnatrblk (pmenu->pwin,
            pitem->row, pitem->col,
            pitem->row,
            (pitem->col + pitem->len) - 1,
            utlonyb (pmenu->hilattr),
            uthinyb (pmenu->hilattr), WN_NO_UPDATE) == NIL)
            return (NIL);
    }
    else
        /* Un-highlight item */
        if (pitem->pcharattr)
        {
            if (NIL == wnwnsct(pmenu->pwin,
                pitem->row,pitem->col,
                pitem->row,pitem->col + pitem->len - 1,
                pitem->pcharattr,
                -1,-1,CHAR_ATTR | WN_NO_UPDATE))
                return(NIL);
        }
    else
        if (wnatrblk (pmenu->pwin,
            pitem->row, pitem->col,
            pitem->row,
            (pitem->col + pitem->len) - 1,
            utlonyb (pitem->attr),
            uthinyb (pitem->attr), WN_NO_UPDATE) == NIL)
            return (NIL);
}
```

```

    return (pmenu);
}

```

## MNCREATE 建立一个包含单信息的选单结构和窗口

```
#include <bmenu.h>
```

```
BMENU *mncreate(int height, int width,
                int textattr,
                int hilattr,
                int proattr,
                int longattr);
```

height, width

窗口数据区的行数和列数。

textattr 窗口数据区的缺省属性。

hilattr 选单亮条的属性。

proattr 受保护选项的属性。

longattr Lotus 形式的选单注脚的属性。

(返回) 建立后的BMENU结构的地址。如果失败，返回NIL。

MNCREATE是一个宏，它为BMENU结构分配空间并用缺省值填充之，这包括创建一个窗口，用具有属性textattr的空格(" ")对窗口数据区初始化，将选单的BWINDOW结构的地址保存在建立的BMENU结构的pwin成员中等待操作。

如果内存分配失败，则出现一个错误。

用MNDSTROY可以废弃BMENU结构、窗口及有关的内部数据结构。

源程序(BMENU.H):

该函数为宏，见附录C。

## MNCREAT0 分配并创建一个菜单结构。

```
pmenu = mncreat0 ( height, width,
                  textattr, hilattr, proattr, longattr,
                  signature);
```

pmenu 指向新创建的BMENU结构，如果失败则为NIL。

height 数据区中的行数。

width 数据区中的列数。

textattr 数据区中的缺省属性。

hilattr 高亮条的属性。

proattr 保护项的属性。

longattr 长描述的属性(如果有的话)。

signature 调用的函数的注脚字。

该函数为菜单结构分配空间，并用缺省值填充。其中包括对WNCREAT的调用。

如果菜单大小超过屏幕大小，或内存分配失败，则产生错误

函数由宏mncreate()调用，提供注脚值。相同的注脚值提通过宏mnvalmnu()提供给MNVALMN0。通过使用这些宏，所有的选单函数可为比较而提供注脚值。用这种方法在运行时可检测到不兼容的BMENU.H的版本。

使用MNDSTROY来放弃BMENU结构和相关的内部数据结构。

返回 presult 指向新创建的结构，如果失败则为空。

h wnerr 可能返回如下值：

(不变) 成功

WN ILL DIM 大小无效

WN\_NO\_MEMORY无足够的内存。

源程序(MNCREAT0.C):

```
#include <bmenu.h>
BMENU *mncreat0 (height, width, textattr, hilattr, proattr, longattr,
                 signature)
int      height, width, textattr, hilattr, proattr, longattr;
unsigned signature;
{
    BMENU *pmenu;
    int      err;

    /* Attempt to allocate space for BMENU structure. */
    if ((pmenu = utalloc (BMENU)) == NIL)
        wnretterr (WN_NO_MEMORY);

    /* Set pointers to NIL, stuff in given values. */
    pmenu->pitems = NIL;
    pmenu->pkeys = NIL;
    pmenu->hilattr = hilattr;
    pmenu->proattr = proattr;
    pmenu->longattr = longattr;

    /* Create this menu's window. */
    if ((pmenu->pwin = wncreate (height, width, textattr)) == NIL)
    {
        free (pmenu);
        return (NIL);
    }

    /* Create signature. */
    pmenu->signature = signature;

    /* Create default key bindings list for this menu. */
    if (mndefkey (pmenu) == NIL)
    {
        err = b_wnerr;
        mndlkeys (pmenu);
        wndstroy (pmenu->pwin);
        pmenu->signature = BMENU_DEAD;
        free (pmenu);
        wnretterr (err);
    }

    pmenu->pwin->options.cur_track = 0; /* Disable cursor tracking. */
    /* Turn the window cursor off (so our highlight bar */
    /* looks good). */
    wnsetopt (pmenu->pwin, WN_CUR_OFF, 1);
    return (pmenu);
}
```

**MNDEFKEY 增加约束菜单的缺省键。**

presult = mndefkey (pmenu);

presult 指向刚改变的BMENU结构, 如果失败则为NIL。

pmenu 指向增加约束键的BMENU的指针。

该函数增加菜单的约束键。

缺省的约束的是UP, DOWN, RIGHT和LEFT键完成标明的功能。HOME和END键完成逻辑最先和最后；TAB和BACKTAB键完成逻辑下一个和先前一个；ENTER确认选择，ESC放弃选择。

为改变缺省的键约束，可使B\_MNKEYMAP指向非B\_MNDEFKEY的其它键表。

返回	presult	指向新创建的BMENU结构，如果失败则为NIL。
	b_wuerr	取下列值：
		(不改变) 成功
		MN_BAD_MENU无效的菜单
		WN_NO_MEMORY无足够的内存。

源程序(MNDEFKEY.C):

```
#include <bmenu.h>
#include <bkeys.h>
#define FALSE 0
BMENU *mndefkey (pmenu)
BMENU *pmenu;
{
    BKEYTAB *pktab;
    /* For each entry in the key map (char code, scan */
    /* code, action) insert in menu's key list. */
    for (pktab = b_mnkeytab;
        pktab->action != MN_INVALID_ACTION;
        pktab++)
        /* Add the key to the keylist if not done. */
        if (mnkey (pmenu, -1, -1, pktab->ch, pktab->scan,
            pktab->action, MN_ADD) == NIL)
            return (NIL);
    return (pmenu);
}
```

**b\_mndefkey** 包含MNDEFKEY使用的缺省键表。

表含有缺省选单键映象表。iv MNCREAT被调用时， MNCREATE调用MNDEFKEY建立键表。

全程变量B\_MNKEYTAB指向当缺省的键表，再生时指向下面的键表。

```
BKEYTAB b_mndefkey [] =
{
    {KB_C_N_UP, KB_S_N_UP, MN_UP },
    {KB_C_N_DOWN, KB_S_N_DOWN, MN_DOWN },
    {KB_C_N_RIGHT, KB_S_N_RIGHT, MN_RIGHT },
    {KB_C_N_LEFT, KB_S_N_LEFT, MN_LEFT },
    {KB_C_N_HOME, KB_S_N_HOME, MN_FIRST },
    {KB_C_N_END, KB_S_N_END, MN_LAST },
    {KB_C_N_TAB, KB_S_N_TAB, MN_NEXT },
    {KB_C_S_TAB, KB_S_S_TAB, MN_PREVIOUS },
    {KB_C_N_ENTER, KB_S_N_ENTER, MN_TRANSMIT},
    {KB_C_N_ESC, KB_S_N_ESC, MN_ABORT },
    {KB_C_N_NEW_UP, KB_S_N_NEW_UP, MN_UP },
    {KB_C_N_NEW_DOWN, KB_S_N_NEW_DOWN, MN_DOWN },
    {KB_C_N_NEW_RIGHT, KB_S_N_NEW_RIGHT, MN_RIGHT }
}
```



```
{KB_C_N_NEW_LEFT, KB_S_N_NEW_LEFT, MN_L},
{KB_C_N_NEW_HOME, KB_S_N_NEW_HOME, MN_H},
{KB_C_N_NEW_END, KB_S_N_NEW_END, MN_E},
{KB_C_N_PADENTER, KB_S_N_PADENTER, MN_TRANSMIT},
```

```
{KB_C_N_PGUP, KB_S_N_PGUP, MN_PGUP},
{KB_C_N_PGDN, KB_S_N_PGDN, MN_PGDN},
{KB_C_N_NEW_PGUP, KB_S_N_NEW_PGUP, MN_PGUP},
{KB_C_N_NEW_PGDN, KB_S_N_NEW_PGDN, MN_PGDN},
```

```
{KB_C_C_RIGHT, KB_S_C_RIGHT, MN_PGRIGHT},
{KB_C_C_LEFT, KB_S_C_LEFT, MN_PGLEFT },
{KB_C_C_NEW_RIGHT, KB_S_C_NEW_RIGHT, MN_PGRIGHT},
{KB_C_C_NEW_LEFT, KB_S_C_NEW_LEFT, MN_PGLEFT },
```

```
{KBNDEF, KBNDEF, MN_INVALID_ACTION}
```

```
};
```

```
/**
```

```
* Name      b_mnkeytab -- Variable which points to current
*              default key table used by MNDEFKEY.
* Description This variable points to the default key map which
*              MNCREATE attaches to each newborn menu. The
*              default for B_MNKEYTAB is to point at the B_MNDEFKEY
*              table.
```

```
**/
```

```
BKEYTAB *b_mnkeytab = b_mndefkey;
```

**MNDISABL** 关闭没有选项的区域的键约束。

```
result = mndisabl (pmenu);
```

result Pointer to just-changed BMENU

```
*              structure, or NIL if failure.
```

pmenu 指向关闭某些约束键的BMENU的指针。

该函数关闭不存在或保护项的约束键，打开由MNDISABL关闭的现在已指向有效项的键。

返回 result 指向新修改的BMENU结构的指针，如果失败则为NIL。

b\_werr 可能返回如下值：

(没有改变) 成功

MN\_BAD\_MENU 无效的菜单

MN\_BAD\_KEY 无效的键表项。

MN\_BAD\_ITEM 发现无效的菜单项。

源程序(MNDISABL.C):

```
#include <bmenu.h>
```

```
BMENU *mndisabl (pmenu)
```

```
BMENU *pmenu;
```

```
{
```

```
    BKEYMAP *pkey;
```

```
    int      code;
```

```

        /* Validate menu data structures. */
mnavaildm (pmenu)
        /* Go through entire key list... */
for (pkey = pmenu->pkeys; pkey != NIL; pkey = pkey->next)
{
    /* Check key signature. */
    if (pkey->signature != MN_KEY_SIGN)
        wnretterr (MN_BAD_KEY);
    /* If the key is bound to a particular location, */
    if (MNMOVE (pkey->action) == MN_SELECT)
    {
        /* and there is no selectable item at that */
        /* location... */
        if (mnmachitm (pmenu, NIL, pkey->row, pkey->col, 0, &code) == NIL)
            /* Disable the key if no error walking the item list*/
            {
                if (code == WN_NO_ERROR)
                    pkey->action |= MN_TEMP_DISABLE;
                else
                    return (NIL);
            }
        /* Enable keys which match items now but were */
        /* temporarily disabled in the past. */
        else if (pkey->action & MN_TEMP_DISABLE)
            pkey->action &= ~(MN_TEMP_DISABLE);
    }
}
return (pmenu);
}

```

**MNDLITMS** 释放选项表中所有项的内存。

**mdlitms** (pmenu);

**pmenu** 指向要放弃的选项表的BMENU结构。

该函数完全放弃菜单项表结构。

返回 \*pmenu->pitems NIL.

源程序(MNDLITMS.C):

#include <stdlib.h>

#include <bmenu.h>

int mndlitms (pmenu)

BMENU \*pmenu;

```

{
    BITEM *pitem, *qitem;
    /* Free memory used by item list. */
    for (pitem = pmenu->pitems; pitem != NIL; pitem = qitem)
    {
        /* Check item signature, then delete it. */
        if (pitem->signature != MN_ITEM_SIGN)
            wnreturn (MN_BAD_ITEM);
    }
}

```

```

    pitem->signature = MN_DEAD_ITEM;
    qitem = pitem->next;
    if (pitem->plstring)
        free (pitem->plstring);
    if (pitem->pcharattr)
        free (pitem->pcharattr);
    free (pitem);
}
return (WN_NO_ERROR);
}

```

**MNDLKEYS** 释放键表中使用的所有的键的内存。

**mdlkeys** (pmenu);

**pmenu** 指向要放弃键表的BMENU的指针。

该函数完全放弃菜单键表结构。

返回 \*pmenu->pkeys NIL.

源程序(MNDLKEYS.C):

```

#include <stdlib.h>
#include <bmenu.h>
int mdlkeys (pmenu)
BMENU *pmenu;
{
    BKEYMAP *pkey, *qkey;
    /* Free memory used by key list. */
    for (pkey = pmenu->pkeys; pkey != NIL; pkey = qkey)
    {
        /* Check item signature, then delete it. */
        if (pkey->signature != MN_KEY_SIGN)
            wnreturn (MN_BAD_KEY);
        pkey->signature = MN_DEAD_KEY;
        qkey = pkey->next;
        free (pkey);
    }
    return (WN_NO_ERROR);
}

```

**MNDSPY** 在同尺寸视口中显示一个选单

#include <bmenu.h>

BMENU \*mndsply(BMENU \*pmenu,  
const WHERE \*pwere,  
const BORDER \*pbord);

**pmenu** 待显示的BMENU结构的地址。

**pwere** WHERE结构的地址, 该结构描述了设备、显示页和选单显示位置的坐标。

**pbord** BORDER结构的地址, 该结构指明了放在选单周围的边界和标题。

(返回) 新显示的BMENU结构的地址。如果失败, 返回NIL。

**MNDSPY**是一个宏, 它在给定的显示设备及显示页上显示一个选单并加上边界。(边界可以是“无边界”类型或包含上/下标题。)选单显示在一个与选单数据区尺寸相匹配的视口中, 选单窗口成为

当前窗口(用于I/O), 那一页上其它窗口的光标被冻结。如果该程序与NW\_???.OBJ文件相连接, 则Turbo C的正文窗口设置为与视口相匹配。

位置和边界(连同标题)分别由WHERE 和BORDER结构指明。请参见WINDSPY中有关这些结构的完整说明。

用MNVDISP可以在一个可能比选单数据区小的视口中显示选单。

如果pmenu未指向合法的选单结构或选单显示位置不适当, 则显示一个错误。

源程序(BMENU.H):

该函数为宏, 见附录C。

**MNDSTROY** 从屏幕上取消一个选单, 废弃其数据结构。

#include <bmenu.h>

int mndstroy(BMENU \*pmenu);

pmenu 待取消并废弃的BMENU结构的地址。

(返回) 错误代码。如果没有错误, 则为WN\_NO\_ERROR (0)。

MNDSTROY从显示选单的屏幕显示页上取消一个选单(如果必要的话)并完全废弃它的BMENU结构。如果该选单不是当前显示的, 这时并不出现错误。

如果选单光标是活动的(相对于该显示页上其它的窗口), 则MNDSTROY将光标关闭。如果程序另与NW\_???.OBJ文件链接, 则虽然字符窗口的属性未被改变或恢复, Turbo C字符窗口被设置为整个显示屏。

用MNDSTROY废弃窗口结构之后便不能再使用该窗口, 它的内存已被释放。

如果pmenu未指向一个合法的选单结构, 这时显示出错误。

要想从屏幕上取消一个选单而不废弃之, 可以使用下面WNREMOVE。

源程序(MNDSTROY.C):

#include <stdlib.h> /\* For free(). \*/

#include <bmenu.h>

static int mndlmice(BMNMUSE \*);

int mndstroy (pmenu)

BMENU \*pmenu;

```
{
    int code, errcode;
        /* Validate menu structure. */
    if (mvalmenu (pmenu) == NIL)
        wnreturn (MN_BAD_MENU);
        /* Validate menu's window. */
    if (wnvalwin (pmenu->pwin) == NIL)
        wnreturn (WN_BAD_WIN);
        /* If shown, undisplay window. */
    if (pmenu->pwin->where_shown.dev == SC_COLOR ||
        pmenu->pwin->where_shown.dev == SC_MONO)
        if (wnremove (pmenu->pwin) == NIL)
            return (b_wnerr);
        /* Destroy menu's window data structures. */
    if (wndstroy (pmenu->pwin))
        return (b_wnerr);
        /* Free memory used by item list. */
    code = mndlitms(pmenu);
        /* Free memory used by key list. */
}
```

```

    ercode = mndlkeys(pmenu);
    if (code == WN_NO_ERROR);
        code = ercode;
        /* Free menu mouse event list. */
    ercode = mndlmice(pmenu->pmouse);
    pmenu->pmouse = NIL;
    if (code == WN_NO_ERROR)
        code = ercode;
        /* Invalidate menu signature. */
    pmenu->signature = BMENU_DEAD;
    /* Free memory used by menu. */
    free(pmenu);
    return (code);
}
/**
 * Name      MNDLMICE -- Free linked list of mouse menu events.
 * Synopsis  ercode = mndlmice(pmouse);
 *           int ercode      Returned result code:
 *                           WN_NO_ERROR if okay;
 *                           MN_BAD_MOUSE if list corrupted.
 *           BMNMOUSE *pmouse
 *                           Address of first event to free.
 * Description This function completely discards a linked list
 *           of menu mouse event structures.
 * Returns    ercode      Returned result code:
 *                           WN_NO_ERROR if okay;
 *                           MN_BAD_MOUSE if list corrupted.
 */
static int mndlmice(pmouse)
BMNMOUSE *pmouse;
{
    BMNMOUSE *pnext_mouse;
    /* Free memory used by item list. */
    for (; pmouse != NIL; pmouse = pnext_mouse)
    {
        /* Check event signature, then delete it. */
        if (pmouse->signature != MN_MOU_SIGN)
            wuretern (MN_BAD_MOUSE);
        pmouse->signature = MN_DEAD_MOUSE;
        pnext_mouse = pmouse->pnext;
        free(pmouse);
    }
    return WN_NO_ERROR;
}

```

**MNFINDSL** 找到给出起始坐标的菜单的第一可选项。

presult = mnfindsl (pmenu, pitem, row, col, pcode);

presult 指向匹配给出的行和列的在菜单项表中的项的指针。如果失败则指向NIL。

pmenu 指向要查找项表的菜单的指针。传给MNMCHITM。

pitem           指向开始查找的表项，如果为NIL则从第一项开始查找。  
 row,            要查找的行  
 col             要查找的列  
 pcode           指向返回错误代码变量的指针。如果没有错误代码则为NIL。  
 MNFINDSL从指定点开始查找菜单的项表，以找到指定行和列的项。  
 如果没有找到，该函数在要求的列和行找任一项。  
 如果没有找到，该函数在要求的行和列找任一项。  
 如果还没有找到，函数查找任一可选项。  
 最后如果还没有找到，则该函数返回NIL。  
 返回            presult           指向找到的BITEM，如果没有找到则为NIL。

源程序(MNFINDSL.C):

```

#include <bmenu.h>
BITEM *mufindsl (pmenu, pitem, row, col, pcode)
BMENU *pmenu;
BITEM *pitem;
int row, col;
int *pcode;
{
    int code;

    /* First try to find one that matches the row and
       /* column specifications exactly.
    if (((pitem = mnmatchm (pmenu, pitem, row, col,
                           0, &code)) == NIL) &&
        (code == WN_NO_ERROR))
        /* Next try to find one that matches the column
           /* requested.
    if (((pitem = mnmatchm (pmenu, pitem, -1, col,
                           0, &code)) == NIL) &&
        (code == WN_NO_ERROR))
        /* Next try to find one that matches the row
           /* requested.
    if (((pitem = mnmatchm (pmenu, pitem, row, -1,
                           0, &code)) == NIL) &&
        (code == WN_NO_ERROR))
        /* Next try to find any selectable item at all!
    if (((pitem = mnmatchm (pmenu, pitem, -1, -1,
                           0, &code)) == NIL) &&
        (code == WN_NO_ERROR))
        return (NIL);
    if (code != WN_NO_ERROR)
    {
        if (pcode != NIL)
            *pcode = code;
        return (NIL);
    }
    else
        return (pitem);
}
  
```

**MNHILIT0** 移动或删除菜单的高亮条或项描述。

**presult** = mnhilit0(pmenu,pitem,option);

**presult** 指向新修改的BMENU结构, 如果失败指向NIL。

**pitem** 要亮显的选项。如果为NIL则没有要亮显的项。如果指定了MN\_UNHIGHLIGHT则忽略此函数的作用。

**option** 可选择位:

MN\_HIGHLIGHT

MN\_UNHIGHLIGHT (忽略行和列)

MN\_USE\_DESCRIPTION

WN\_UPDATE or WN\_NO\_UPDATE

函数亮显一菜单项或删除一亮条。可选地增加或删除项的Lotus风格的描述的字符串。

函数可处理保护的菜单项。

返回 **presult** 指向刚修改的BMENU结构, 如果失败指向NIL。

**b\_werr** 可能如下值:

(不修改)	成功
WN_BAD_OPT	未知的选择
MN_BAD_MENU	pmenu无效
WN_BAD_WIN	pmenu->pwin无效
MN_BAD_ITEM	pitem无效
WN_NOT_SHOWN	内部错误
WN_BAD_DEV	内部错误
WN_ILL_DIM	内部错误
WN_NULL_PTR	内部错误

源程序(MNHILIT0.C):

```
#include <bmenu.h>
BMENU *mnhilit0(pmenu,pitem,option)
BMENU *pmenu;
BITEM *pitem;
int option;
{
    int remove      = ( (pitem == NIL)
                        || (MN_UNHIGHLIGHT ==
                            (option & (MN_HIGHLIGHT | MN_UNHIGHLIGHT))));
    int desc        = (option & MN_USE_DESCRIPTION);
    REGION strip;   /* Region to force into viewport*/
    LOC begin_strip,end_strip; /* Points to force into viewport*/
    mnvalidm(pmenu) /* Validate menu. */
    if (pmenu->pbar_item != NIL)
    {
        /* A bar is currently shown. */
        if (pmenu->internals.desc_shown) /* If description shown */
        {
            /* and if */
            if ( remove /* no bar wanted */
                || !desc /* or no description */
                /* wanted */
                || pmenu->pbar_item != pitem) /* or bar moved */
            {
                /* then remove description. */
            }
        }
    }
}
```

```

        if (pmenu->pbar_item->llen &&
            NIL == wnscribk(pmenu->pwin,
                           pmenu->pbar_item->lrow,
                           pmenu->pbar_item->lcol,
                           pmenu->pbar_item->lrow,
                           (pmenu->pbar_item->lcol
                            + pmenu->pbar_item->llen) - 1,
                           utlonyb(pmenu->pwin->attr),
                           uthinyb(pmenu->pwin->attr),
                           WNSCR_LEFT,0,WN_NO_UPDATE))

            return NIL;
        pmenu->internals.desc_shown = 0;
    }
}

if (    remove                /* If removing bar          */
    || pmenu->pbar_item != pitem) /* or bar moved      */
{
    /* remove old highlight bar.          */
    if (NIL == mnatr(pmenu,pmenu->pbar_item,MN_UNHIGHLIGHT))
        return NIL;
    pmenu->pbar_prev = pmenu->pbar_item;
    pmenu->pbar_item = NIL;
}

if (!remove)
{
    if (pmenu->pbar_item != pitem)
    {
        /* Light new highlight bar.          */
        if (NIL == mnatr(pmenu,pitem,MN_HIGHLIGHT))
            return NIL;
        if (pmenu->pbar_item != NIL)
            pmenu->pbar_prev = pmenu->pbar_item;
        pmenu->pbar_item = pitem;
    }

    if (    desc
        && pitem->llen
        && !pmenu->internals.desc_shown)
    {
        /* Show description.          */
        if (NIL == wnwrect(pmenu->pwin,
                           pitem->lrow,pitem->lcol,
                           pitem->lrow,pitem->lrow + pitem->llen - 1,
                           pitem->plstring,
                           utlonyb(pmenu->longattr),
                           uthinyb(pmenu->longattr),
                           CHARS_ONLY | WN_NO_UPDATE))

            return NIL;
        pmenu->internals.desc_shown = 1;

        /* Force description into viewport.          */
        strip.ul.row = pitem->lrow;
        begin_strip.row = pitem->lrow;
    }
}

```



```

strip.lr.row      = pitem->lrow;
end_strip.row     = pitem->lrow;
strip.ul.col      = pitem->lcol;
begin_strip.col   = pitem->lcol;
strip.lr.col      = pitem->lcol + pitem->lien - 1;
end_strip.col     = pitem->lcol + pitem->llen - 1;
if (NIL == washoblk(pmenu->pwin,
                    &strip,&begin_strip,&end_strip,
                    WN_NO_UPDATE))

    return NIL;
}

/* Force item into viewport. */
strip.ul.row      = pitem->row;
begin_strip.row   = pitem->row;
strip.lr.row      = pitem->row;
end_strip.row     = pitem->row;
strip.ul.col      = pitem->col;
begin_strip.col   = pitem->col;
strip.lr.col      = pitem->col + pitem->len - 1;
end_strip.col     = pitem->col + pitem->len - 1;
if (NIL == wshoblk(pmenu->pwin,
                  &strip,&begin_strip,&end_strip,
                  WN_NO_UPDATE))

    return NIL;
}
if ( WN_UPDATE == (option & (WN_UPDATE | WN_NO_UPDATE))
    && !pmenu->pwin->options.delayed)
    if (NIL == wnupdate(pmenu->pwin))
        return NIL;
return pmenu; /* Success. */
}

```

## MNHILITE 移动或取消选单亮条及选项说明

#include <bmenu.h>

```
BMENU *mnhilite( BMENU *pmenu,
                 int row, int col,
                 int option);
```

pmenu 待修改的BMENU 结构的地址。

row, col 如果未指明MN\_UNHIGHLIGHT, 它们就是待修改的选项的行和列(相对于选单窗口数据区左上角(0,0))。

option 下列任选值之一:

符号	值	意义
MN_UNHIGHLIGHT	0x0000	取消亮条和说明字符串(忽略row和col)。
MN_HIGHLIGHT	0x0001	用亮条将选项显亮。
MN_USE_DESCRIPTION	0x4000	如果选项具有说明字符串, 显示之。
WN_UPDATE	0x0000	除非选单窗口被“延迟”, 否则显示挂起的输出。
WN_NO_UPDATE	0x0004	不立即显示改变的内容。

(返回) 修改后的BMENU 结构的地址。 如果失败, 返回NIL。

**MNHILITE** 显亮一个选项或取消亮条, 它可任选地加入或取消选项的Lotus形式的说明字符串。如果指定的选项是被保护的, 则NIL作为函数值返回, 但全局窗口错误值(b\_wnerr)不变。

源程序(MNHILITE.C):

```
#include <bmenu.h>
BMENU *mnhilite(pmenu,row,col,option)
BMENU *pmenu;
int row,col,option;
{
    BITEM *pitem;
    mnvalidm(pmenu) /* Validate menu. */
    switch (option & (MN_HIGHLIGHT | MN_UNHIGHLIGHT))
    {
        case MN_HIGHLIGHT:
            if (NIL == (pitem = mnchitm(pmenu,NIL,row,col,0,NIL)))
                return NIL;
            break;
        case MN_UNHIGHLIGHT:
            pitem = NIL;
            break;
        default:
            wnreterr(WN_BAD_OPT);
    }
    return mnhilit0(pmenu,pitem,option);
}
```

## MNITEM 插入、修改或删除一个选项

#include <bmenu.h>

BMENU \*mnitem(BMENU \*pmenu,

int row, int col,

int option,

const char \*pstring);

pmenu 待修改的BMENU 结构的地址。

row, col 定位选项的行和列(相对于选单窗口数据区左上角(0,0))。

option 两个有关的数据位:

符号	值	意义
MN_NOPROTECT	0	选项可以用亮条选择。
MN_PROTECT	1	选项被保护, 这条不能选择它。
MN_CHARS_ONLY	0	*pstring 仅包含字符, 以'\0'结尾。
MN_CHAR_ATTR	2	*pstring 包含(char,attr)偶对。第一个包含有'\0'字符值的偶对结束这个字符串。

pstring 待显示的正文。如果要取消这个选项, 则为空字符串("")。

(返回) 修改后的BMENU结构的地址。如果失败, 返回NIL。

**MNITEM** 向选单中加入一个新选项, 修改或删除一个已有的选项。

如果在选单中同样的位置(row,col)已有一个选项, 则**MNITEM**修改或删除这个已有的选项, 选项的正文用窗口的缺省属性(颜色)清除。如果pstring为NIL或指向空字符串(""), 则该选项连同它的选择键被废弃, 否则\*pstring中的新正文写入窗口, option的新值被记录下来。

如果在(row, col) 处没有选项, 则新选项就加入到选项表的末尾, \*pstring 的内容写入到窗口中。

如果声明了MN\_NOPROTECT, 正文使用窗口的缺省属性(颜色); 如果指明了MN\_PROTECT, 则使用选单的保护属性。

必要时选项的正文在窗口的右边沿截断。

如果row和col超出选单窗口的尺寸, 则出现一个错误。

源程序(MNITEM.C):

```
#include <string.h>
#include <bmenu.h>
#define TRUE 1
#define FALSE 0
#define NUL '\0'
static int write_item(const BMENU *,const BITEM *,const char *);
BMENU *mnitem (pmenu, row, col, option, pstring)
BMENU *pmenu;
int row, col, option;
const char *pstring;
{
    int done = FALSE;
    BITEM *pitem, *qitem;
    BKEYMAP*pkey;
    /* Validate menu data structures. */
    mvalidm (pmenu)
    /* Make sure given row and col are within bounds. */
    if (ustrange (row, 0, pmenu->pwin->img.dim.h - 1) ||
        ustrange (col, 0, pmenu->pwin->img.dim.w - 1))
        wretterr (WN_ILL_DIM);
    /* Go through all existing menu items, and make
       * qitem trail pitem through the list. */
    for (pitem = pmenu->pitems, qitem = NIL;
        (pitem != NIL) && (!done);
        qitem = pitem, pitem = pitem->next)
    {
        /* Check item signature byte. */
        if (pitem->signature != MN_ITEM_SIGN)
            wretterr (MN_BAD_ITEM);
        /* If existing row and column match, replace
           * existing item. */
        if ((pitem->row == row) && (pitem->col == col))
        {
            /* Erase the area used by the old item in the menu. */
            if (NIL == wnscribk (pmenu->pwin,
                                row, col, row, (col + pitem->len) - 1,
                                -1, -1, WNSCR_LEFT, 0, WN_UPDATE))
                return NIL;
            /* Delete existing item if replacement string is
               * NIL or the NUL-string. */
            if ((pstring == NIL) || (*pstring == NUL))
            {
                done = TRUE;
                /* Make previous item point at next item. */
            }
        }
    }
}
```

```

if (qitem == NIL)
    pmenu->pitems = pitem->next;
else
    qitem->next = pitem->next;
/* Delete item signature. */
pitem->signature = MN_DEAD_ITEM;
/* Free memory used by old item, and delete any key */
/* bindings to this row and column. */
if (pitem->plstring)
    free (pitem->plstring);
if (pitem->pcharattr)
    free (pitem->pcharattr);
free (pitem);
for (pkey = pmenu->pkeys; pkey != NIL; pkey = pkey->next)
{
/* Make sure to remove every key for the item. */
    if ((pkey->row == row) &&
        (pkey->col == col))
    {
        if (NIL == mnkey(pmenu,row,col,pkey->ch,pkey->scan,
                        0,MN_DELETE))
            return NIL;
    }
}
}
else
{
/* The replacement string is not NULL, so do a true */
/* replacement. */
done = TRUE;
/* Figure out length of new item within bounds of */
/* window it has to fit in (or be truncated by). */
if (option & MN_CHAR_ATTR)
{
/* If the string contains character/attribute pairs,*/
/* find the index of the first character equal to */
/* '\0'. Note that pitem->len is the length of the */
/* ASCII text, not the entire buffer. */
for (pitem->len = 0; pstring[pitem->len];
    pitem->len += 2)
    ;
pitem->len /= 2;
}
else
    pitem->len = strlen(pstring);
pitem->len = mntrunc (pmenu, pitem->len, col);
/* Copy new char_attr string, if present, into item.*/
if (option & MN_CHAR_ATTR)
{
    if (pitem->pcharattr != NIL)
        free(pitem->pcharattr);

```

```

        if ((pitem->pcharattr = malloc(pitem->len * 2)) == NIL)
            wnretterr (WN_NO_MEMORY);
        memmove((void *) pitem->pcharattr, (void *) pstring,
            pitem->len * 2);
    }
    /* Figure out protection value, record attribute. */
    pitem->option = option;
    pitem->attr = pmenu->pwin->attr;
    /* Write the text of the new item to the menu's */
    /* window. */
    if (write_item(pmenu, pitem, pstring))
        return NIL;
}
}

/* If there is a non-NUL string to place, and we */
/* have not already done anything with it, it must */
/* be a string to be added... so we will add it. */
/* qitem points to the last item in the list, or is */
/* NIL if there is no item list yet. */
if ((!done) && (pstring != NIL) && (*pstring != NUL))
{
    /* Attempt to allocate space for the new item. */
    if ((pitem = utalloc (BITEM)) == NIL)
        wnretterr (WN_NO_MEMORY);
    /* Figure out length of new item within bounds of */
    /* window it has to fit in (or be truncated by). */
    if (option & MN_CHAR_ATTR)
    {
        /* If the string contains character/attribute pairs,*/
        /* find the index of the first character equal to */
        /* '\0'. Note that pitem->len is the length of the */
        /* ASCII text, not the entire buffer. */
        for (pitem->len = 0; pstring[pitem->len]; pitem->len += 2)
            ;
        pitem->len /= 2;
    }
    else
        pitem->len = strlen(pstring);
    pitem->len = mntrunc (pmenu, pitem->len, col);
    /* Store values for future reference. */
    pitem->attr = pmenu->pwin->attr;
    pitem->row = row;
    pitem->col = col;
    pitem->option = option;
    pitem->pstring = NIL;
    pitem->pcharattr = NIL;
    pitem->next = NIL;
    /* Copy char attr string, if present, into item. */
    if (option & MN_CHAR_ATTR)

```

```

    {
        if ((pitem->pcharattr = malloc(pitem->len * 2)) == NIL)
            wureterr (WN_NO_MEMORY);
        memmove((void *) pitem->pcharattr, (void *) pstring,
            pitem->len * 2);
    }

    /* Create item signature. */
    pitem->signature = MN_ITEM_SIGN;
    /* If there is no item list, add this as the first
    /* item, otherwise add this item to the tail of the */
    /* item list. */
    if (pmenu->pitems == NIL)
        pmenu->pitems = pitem;
    else
        qitem->next = pitem;
        /* Write the text of the new item to the menu's
        /* window.
    if (write_item(pmenu, pitem, pstring))
        return NIL;
    done = TRUE;
}
return (pmenu);
}

/**
 * Name      WRITE_ITEM - Write the text (and attributes, if
 *                  present) of an item into its window.
 * Synopsis  error = write_item(pmenu, pitem, pstring);
 *                  int      error      Nonzero if error, zero if okay.
 *                  const BMENU *pmenu  The menu to write to.
 *                  const BITEM *pitem  Item to write to the menu.
 *                  const char  *pstring The item's text (possibly character/
 *                  attribute pairs).
 * Description This function writes an item's text into a menu.      If the
 *                  item's MN_CHAR_ATTR bit is set, pstring contains character/
 *                  attribute pairs; if the item's MN_PROTECT bit is not set,
 *                  these attributes are also written.
 * Returns    error      Nonzero if error, zero if okay.
 */

```

```

static int write_item(pmenu, pitem, pstring)
const BMENU *pmenu;
const BITEM *pitem;
const char  *pstring;
{
    return (NIL == wnwrect(pmenu->pwin,
        pitem->row, pitem->col,
        pitem->row, pitem->col + pitem->len - 1,
        pstring,
        (pitem->option & MN_PROTECT))

```

```

        ? (uthonyb (pmenu->proattr) : (-1),
        (pitem->option & MN_PROTECT)
        ? (uthinyb (pmenu->proattr) : (-1),
        WN_UPDATE |
        ((pitem->option & MN_CHAR_ATTR)
        ? CHAR_ATTR : CHARS_ONLY));
    }

```

## MNITMKEY 向选单加入一个选项，为它分配选择字符

```
#include <bmenu.h>
```

```
BMENU *mnitmkey(BMENU *pmenu,
                int row, int col,
                int option,
                const char *pstring,
                const char *pkeys,
                int action);
```

pmenu 待修改的BMENU结构的地址。

row, col 定位选项的行和列(相对于窗口选单数据区左上角(0,0))。

option 两个有关的数据位:

符号	值	意义
MN_NOPROTECT	0	选项可以用亮条选择。
MN_PROTECT	1	选项被保护，亮条不能选择它。
MN_CHARS_ONLY	0	*pstring 只包含字符，以'\0'结尾。
MNCHAR_ATTR	2	*pstring包含(char,attr)偶对。第一个包含'\0'字符值的偶对结束这个字符串。

pstring 待显示的正文。

pkeys 一个字符串，该字符串包含应该分配给这个选项的字符。

action 选择零个、任意或全部可能的按键动作。

(返回) 修改后的BMENU结构的地址。如果失败，返回NIL。

MNITMKEY向选单加入一个新选项，为它的选择分配一个或多个字符。

如果在(row,col)处已有一个选项，则结果无定义。

必要时选项的正文在窗口右边沿截断。

如果row 和col超出选单窗口的尺寸，则出现错误。

源程序(MNITMKEY.C):

```
#include <bkeybrd.h>
```

```
#include <bmenu.h>
```

```
#define NUL '\0'
```

```
BMENU *mnitmkey (pmenu, row, col, option, pstring, pkeys, action)
```

```
BMENU *pmenu;
```

```
int row, col, option;
```

```
const char *pstring, *pkeys;
```

```
int action;
```

```
{
```

```
    char c;
```

```
    int err;
```

```
        /* Add item to menu.
```

```
*/
```

```
    if (NIL == mnitem (pmenu, row, col, option, pstring))
```

```

        return (NIL);
        /* Add all of the characters pointed to by *pkeys
        /* into the menu's key binding list.
while ((c = *pkeys++) != NUL)
    if (NIL == mnkey (pmenu, row, col,
        kbcharof (c), kbscanof (c),
        /* Make sure the movement portion of the action is
        /* MN_SELECT.
            (action & ~MN_MOVE_MASK) | MN_SELECT, MN_ADD))
    {
        err = b_wnerr;
        mnitem (pmenu, row, col, option, NIL);
        wnreferr (err);
    }
return (pmenu);
}

```

## MNKEY 加入、修改或取消一个选单的键分配

```

#include <bmenu.h>
BMENU *mnkey(BMENU *pmenu,
    int row, int col,
    int ch, int key,
    int act_move,
    int howchange);

```

**pmenu** 待修改的BMENU结构的地址。  
**row,col** 如果指定了MN\_SELECT, 它们指明待显亮的选项。  
**ch, key** 按键的ASCII值和键码。  
**act\_move** 将一个按键亮条移动同零个、任意或全部可能的按键动作结合起来。  
**howchange** 下列值之一: MN\_ADD (0), MN\_CHANGE (1) 或MN\_DELETE (2)。  
**(返回)** 修改后的BMENU结构的地址。若失败, 返回NIL。

MNKEY分配选单所用按键的行为、修改按键的行为或取消一个按键分配。

请根据“选单函数(MN)”下的“按键动作和亮条移动”一节来定义act\_move值。

要修改或取消一个按键的功效必须声明确切的字符值(ch)和键码(key)以便识别这个键。如果该按键还选择一个特定的选项(MN\_SELECT), 则必须指明正确的选项位置(row, col), 否则无法找到这个按键, 出现错误。

仅当所有的按键分配都指定了MN\_SELECT, 也就是说, 如果它们都声明了一个选项, 则同一个按键(由ch和key指明)才能够具有多于一个的按键分配。如果试图把其它种类的、相互矛盾的亮条移动加入到同一个按键中, 这将导致错误。

如果MN\_SELECT是选择的亮条移动而row和col未指明一个已存在的选项, 这时出现错误。

源程序(MNKEY.C):

```

#include <bmenu.h>
#define TRUE 1
#define FALSE 0
BMENU *mnkey (pmenu, row, col, ch, scan, action, howchange)
BMENU *pmenu;
int row, col;
int ch;

```



```

int    scan, action, howchange;
{
    int    done    = FALSE;
    int    touched = FALSE;
    int    code;
    BKEYMAP *pkey, *qkey, *rkey;
        /* Validate the menu data structure. */
    mvalidm (pmenu)
    qkey = NIL;
    switch (howchange)
    {
        case MN_ADD:
        case MN_CHANGE:
            /* If movement is MN_SELECT, (row,col) must be */
            /* valid window coordinates, and must have an item. */
            if (MNMOVE (action) == MN_SELECT)
            {
                if (utrange (row, 0, pmenu->pwin->img.dim.h - 1) ||
                    utrange (col, 0, pmenu->pwin->img.dim.w - 1))
                    wnreterr (WN_ILL_DIM);
                if (NIL == mnmatchm (pmenu, NIL, row, col, 1, &code))
                {
                    if (code == WN_NO_ERROR)
                        wnreterr (MN_NO_ITEM);
                    else
                        return (NIL);
                }
            }
            pkey = pmenu->pkeys;
            /* Go check for a change of action code for old key */
            /* definition, or for a key conflict. */
            while ((pkey != NIL) && (!done))
            {
                /* Check key signature byte. */
                if (pkey->signature != MN_KEY_SIGN)
                    wnreterr (MN_BAD_KEY);

                if (((pkey->scan == scan) &&
                    (pkey->ch == ch) &&
                    (((pkey->row == row) &&
                    (pkey->col == col)) ||
                    (MNMOVE (pkey->action) != MN_SELECT)))
                    /* What we really want to do is change the action */
                    /* associated with this key. */
                    {
                        /* Check for possible key conflict. */
                        if ((MNMOVE (pkey->action) == MN_SELECT) &&
                            (MNMOVE (action) != MN_SELECT))

                            for (rkey = pmenu->pkeys;

```

```

        rkey != NIL;
        rkey = rkey->next)
    {
        if (rkey->signature != MN_KEY_SIGN)
            wnreterr (MN_BAD_KEY);
        if ((rkey != pkey) &&
            (rkey->ch == ch) &&
            (rkey->scan == scan))
            wnreterr (MN_KEY_CONFLICT);
    }
    done = TRUE;
    /* Change the key's action, but preserve */
    /* its motion. */
    pkey->action = MNMOVE(pkey->action) |
        (action & ~MN_MOVE_MASK);
}
else
    /* Check for a key conflict. A conflict is defined */
    /* as when two keys have the same character and key */
    /* codes, and are not both bound to menu items (i.e. */
    /* do not have MN_SELECT as their movement). */
    if ((pkey->ch == ch) &&
        (pkey->scan == scan) &&
        (MNMOVE (pkey->action) != MN_SELECT))
        wnreterr (MN_KEY_CONFLICT);
    qkey = pkey;
    pkey = pkey->next;
}
if (!done)
{
    /* Now we know we really want to *add* a key binding */
    /* Qkey is pointing to the last item in the keymap, */
    /* or is NIL if there is no keymap defined. */
    /* Allocate memory for new key binding. */
    if ((pkey = utalloc (BKEYMAP)) == NIL)
        wnreterr (WN_NO_MEMORY);
    pkey->action = action;
    pkey->row = row;
    pkey->col = col;
    pkey->ch = ch;
    pkey->scan = scan;
    pkey->next = NIL;
    /* Create key signature. */
    pkey->signature = MN_KEY_SIGN;
    /* Insert it at the end of the list, or as the */
    /* first list item if there is no list yet. */
    if (qkey == NIL)
        pmenu->pkeys = pkey;
    else
        qkey->next = pkey;
}

```

```

    }
    break;
case MN_DELETE:
    for (pkey = pmenu->pkeys; pkey != NIL; )
    {
        /* Check key signature. */
        if (pkey->signature != MN_KEY_SIGN)
            wnreterr (MN_BAD_KEY);
        /* Make sure to remove every definition of the key. */
        if ((pkey->ch == ch) &&
            (pkey->scan == scan))
        {
            if (((MNMOVE (pkey->action) == MN_SELECT) &&
                (pkey->row == row) &&
                (pkey->col == col)) ||
                (MNMOVE (pkey->action) != MN_SELECT))
            {
                if (MNMOVE (pkey->action) != MN_SELECT)
                    done = TRUE;
                /* Point previous key to next key. */
                if (qkey == NIL)
                    pmenu->pkeys = pkey->next;
                else
                    qkey->next = pkey->next;
                /* Delete key signature. */
                pkey->signature = MN_DEAD_KEY;
                /* Free memory used by deleted key */
                /* binding. */
                free (pkey);
                /* Set up pointers to mean something*/
                touched = TRUE;
                pkey = pmenu->pkeys;
                qkey = NIL;
            }
        }
        if (!touched)
        {
            qkey = pkey;
            pkey = pkey->next;
        }
        else
            touched = FALSE;
    }
    break;
default:
    wnreterr (WN_ILL_VALUE);
}
return (pmenu);
}

```

## MNLITEM 加入、修改或取消一个Lotus 形式的选项

```
#include <bmenu.h>
```

```
BMENU *mnlitem(BMENU *pmenu,
               int row, int col,
               int option,
               const char *pstring,
               int lrow, int lcol,
               const char *plstring);
```

pmenu 指向修改的BMENU结构的地址。

row, col 定位选项的行和列(相对于选单窗口数据区左上角(0,0))

option 两个有关的数据位

符号	值	意义
MN_NOPROTECT	0	选项可以被亮条选择。
MN_PROTECT	1	选项被保护, 亮条不能选择它。
MN_CHARS_ONLY	0	*pstring 仅包含字符, 以'\0'结尾。
MN_CHAR-ATTR	2	*pstring 包含(char, attr)偶对。第一个包含字符值'\0'的偶对结束该字符串。

pstring 待显示的正文。如果要取消这个选项, 则为NIL或空字符串("")。

lrow, lcol 待显示的\*pstring 所在的行和列。

plstring 当选项被显亮时需显示的说明字符串。

(返回) 修改后的BMENU结构的地址。如果失败, 返回NIL。

MNLITEM向lotus形式的选单中加入一个新选项或取消一个已有的选项。

如果在选单中位置(row, col)已有一个选项, 则MNLITEM修改或删除原有的选项, 选项的原正文用窗口的缺省属性(颜色)消除。如果pstring为NIL或指向空字符串(""), 则该选项被废弃, 否则将\*pstring中的新正文写入窗口, 任选的新值lrow, lcol及\*plstring被记录下来。

如果在(row, col)没有选单的选项, 则新选项加入到选项表的尾部, \*pstring的内容写入窗口中。

如果指明了MN\_NOPROTECT, 则窗口的缺省属性(颜色)用于正文; 如果指明了MN\_PROTECT, 则使用选单的保护属性。

必要时选项及说明字符串的正文在窗口右边沿截断。

如果row或col超出选单窗口尺寸, 则出现一个错误。

源程序(MNLITEM.C):

```
#include <string.h>
#include <bmenu.h>
BMENU *mnlitem (pmenu, row, col, option, pstring, lrow, lcol, plstring)
BMENU *pmenu;
int row, col, option;
const char *pstring;
int lrow, lcol;
const char *plstring;
{
    BITEM *pitem;
    int code;
    int lasterr;

    /* Add the standard text item to the menu. */
    if (NIL == mnlitem (pmenu, row, col, option, pstring))
        return (NIL);
}
```

```

        /* Get a pointer to the item so we can add special */
        /* "Lotus"-style information; if we cannot get a */
        /* pointer, the above MNITEM call was a deletion. */
if ((pitem = mnunchitm (pmenu, NIL, row, col, 1, &code)) != NIL)
{
    /* Add the information. */
    pitem->lrow = lrow;
    pitem->lcol = lcol;
    pitem->lren = mntrunc (pmenu, strlen(plstring), lcol);
    if ((pitem->plstring = malloc(pitem->lren + 1)) == NIL)
    {
        /* Error allocating memory for copy of Lotus string.*/
        lasterr = b_werrr;
        mnitem(pmenu, row, col, option, NIL);
        b_werrr = lasterr;
        return(NIL);
    }
    strncpy(pitem->plstring, plstring, pitem->lren);
    pitem->plstring[pitem->lren] = '\0';
}

if (code != WN_NO_ERROR)
    return (NIL);
else
    return (pmenu);
}

```

**MNLITKEY** 加入一个LOTUS形式的选项，为它分配选择字符。

```
#include <bmenu.h>
```

```
BMENU *mnlitkey( BMENU *pmenu,
                int row, int col,
                int option,
                const char *pstring,
                int lrow, int lcol,
                const char *plstring,
                const char *pdeys,
                int action);
```

**pmenu** 指向待修改的 BMENU 结构的指针。

**row, col** 选项位置的行和列(相对于选单窗口数据区左上角(0,0))。

**option** 两个有关的数据位:

符号	值	意义
MN_NOPROTECT	0	选项可以被亮条选择。
MN_PROTECT	1	选项被保护，亮条不能选择它。
MN_CHARS_ONLY	0	*pstring 仅包含字符，以'\0'结尾。
MN_CHAR_ATTR	2	*pstring 包含(char,attr)偶对,第一个包含'\0'作为其字符值的偶对结束该字符串。

**pstring** 待显示的正文。

**lrow, lcol** 待显示的\*plstring所在的行和列。  
**plstring** 选项被显亮时需显示的说明字符串。  
**pkeys** 一个字符串,它包含着应该分配给这个选项的字符。  
**action** 选择零个、任意或全部可能的按键动作(见“选单函数(MN)”下的“按键动作和亮条移动”一节)。  
**(返回)** 指向修改后的BMENU结构的指针。若失败,返回NIL。

MNLITKEY向Lotus 形式的选单中加入一个新选项并分配一个或多个字符来选择它。\*pstring的内容以窗口的缺省属性(如果指明了MN\_NOPROTECT)或选单的保护属性(如果指明了MN\_PROTECT)写入到窗口中。

如果在(row, col)处已有选项,则结果无定义。

必要时选项和说明字符串的正文在窗口的右边沿截断。

如果row或col超出选单窗口的尺寸,则出现一个错误。

源程序(MNLIKEY.C):

```
#include <bkeybrd.h>
#include <bmenu.h>
#define NUL '\0'
BMENU *mnlitkey (pmenu,
                  row, col, option, pstring,
                  lrow, lcol, plstring,
                  pkeys, action)
BMENU *pmenu;
int row, col, option;
const char *pstring;
int lrow, lcol;
const char *plstring;
const char *pkeys;
int action;
{
    char c;
    int err;

    /* Add the item to the menu. */
    if (NIL == mnlitem (pmenu, row, col, option, pstring,
                       lrow, lcol, plstring))
        return (NIL);

    /* Add all of the characters pointed to by *pkeys
       /* into the menu's key binding list.
    while ((c = *pkeys++) != NUL)
        if (NIL == mnkey (pmenu, row, col,
                          kbcharof (c), kbscanof (c),
                          /* Make sure the movement portion of the action is
                          /* MN SELECT.
                          (action & ~MN_MOVE_MASK) | MN_SELECT, MN_ADD))
        {
            err = bwnerr;
            mnlitem (pmenu, row, col, option, NIL);
            wnreterr (err);
        }
}
```

```

    return (pmenu);
}

```

## MNLREAD 通过Lotus形式的选单读入一个用户响应

```
#include <bmenu.h>
```

```
int mnlread( BMENU pmenu,
             int srow, int scol,
             int *prow, int *pcol,
             int *pch, int *pkey,
             int option);
```

pmenu 待修改的 BMENU 结构的地址。

srow,scol 最初需显亮的选项所在的行和列。

prow,pcol 变量的地址, 这些变量分别返回所选项的行和列。

pch, pkey 变量的地址, 这些变量分别返回最后按键的字符值和键码。

option 下列值的零个或任意个组合:

符号	值	意义
MN_PREV_BAR	0x2000	显亮原先显亮的选项。
MN_SHOW_BAR	0x1000	如果使用了 MN_PREV_BAR, 迫使亮条显示, 即使它原来不显示。
MN_HIDE_BAR	0x0800	最初不显亮选项, 取代 MN_SHOW_BAR。
MN_KEIGNORE	0x0200	忽略并废弃全部按键。
MN_UNKNOWN_BEEP	0x0010	当未分配的键按下时鸣叫。
MN_UNKNOWN_TRANSMIT	0x0020	当未分配的键按下时返回。
MN_ALL_TRANSMIT	0x0008	每次按键后返回。
MN_HOLD_BEEP	0x0080	返回时制止鸣叫, 取代MN_UNKNOWN_BEEP。
MN_KEEP_HIGHLIGHT	0x0004	使最后的选项显亮。
MN_KEEP_DESCRIPTION	0x0040	不消除最后的说明字符串。
MN_REMOVE	0x0001	事后取消选项单。
MN_DESTROY	0x0003	事后取消并废弃选单。

(返回) 错误代码。若无错误, 返回 WN\_NO\_ERROR。

MNLREAD 是一个宏, 它接受来自用户的按键, 移动选单上的亮条, 报告用户的选择及按下的最后一个键。当选项被显亮时, 其相应的说明字符串也被显亮。

除非指定了MN\_PREV\_BAR 并查寻到原来被显亮的选项, 否则 srow和scol指定最初被显亮的选项。如果在(srow, scol)处没有可选择的选项, 则MNLREAD选择一个选项, 使之显亮。

窗口被更新。如果窗口在调用 MNLREAD 之前被“延迟”, 则退出时恢复这个状态。

如果下列情况之一发生, 这时出现错误, 返回非零错误代码。

选单未显示;

选单窗口数据区的某个区域被另一个窗口覆盖;

选单中没有非保护选项;

指定了MN\_SHOW\_BAR但不能找到可选择的选项;

指定了MN\_UNKNOWN\_TRANSMIT而一个未分配键被按下;

出现具有MN\_ABORT动作的按键或鼠标器操作;

指定了MN\_REMOVE或MN\_DESTROY而不能取消这个选单;

指定了MN\_DESTROY而不能废弃这个选单。

MNREAD 读取一个选单但不显示Lotus形式的说明字符串。

源程序(BMENU.H):

参见附录C的MNLREAD的定义。

## MNREAD 从给定的开始行和列使用亮条读取菜单。

```
ret = mnread ( pmenu, srow, scol, prow, pcol,  
               pch, pscan, option);
```

**ret** 如果选择没有错误, ret为WN\_NO\_ERROR (0)。否则为MN/WN的错误码。

**pmenu** 指向要读取的菜单。

**srow** 首先加亮项的开始行。

**scol** 首先加亮项的开始列。

**prow,** 当确认键已按后, 要读取的亮条的行。

**pcol** 当确认键已按后, 要读取的亮条的列。

**pch** 指向返回终止MNREAD所按键的字符码。

**pscan** 指向返回终止MNREAD所按键的字符码。

**option** 用与下列值相与, 每项的解释见MNLREAD。

MN\_USE\_DESCRIPTION

MN\_ALL\_TRANSMIT

MN\_DESTROY

MN\_HOLD\_BEEP

MN\_KEEP\_DESCRIPTION

MN\_KEEP\_HIGHLIGHT

MN\_REMOVE

MN\_UNKNOWN\_BEEP

MN\_UNKNOWN\_TRANSMIT

MN\_PREV\_BAR

MN\_SHOW\_BAR

MN\_HIDE\_BAR

MNREAD从用户接受输入, 在指定菜单移动高条。

如果出现下述情况, 产生错误:

在菜单上没有可选项。

菜单没有显示。

WNREMOVE失败。

MNDSTROY失败。

按键未知。

**返回** **ret** 如果没有错误返回WN\_NO\_ERROR, 否则如果错误返回MN/WN错误代码。

**\*prow** 当键入接受键时高亮条所在的行和列。如果没有当前的高亮条, 则指的是最近高亮的项。

**\*pcol** 最近高亮的项。

**\*pch** 按键的字符码和扫描码, 如果是鼠标事件则为0xff。

**\*pscan**

**b\_mnmoevent**

最近的鼠标事件, 如果没有则为0L。

**b\_mnmovert, b\_mnmohoriz**

最近的鼠标事件发生的行和列。

源程序(MNREAD.C):

```
#include <bkeybrd.h>
```

```
#include <bmenu.h>
```

```
#include <bmouse.h>
```

```
/* Most recent mouse menu event.
```

```
*/
```

```
unsigned long b_mnmoevent; /* Event, or 0L if none.
```

```
*/
```



```

unsigned      b_mnmovvert;      /* Row and column where event took*/
unsigned      b_mnmohoriz;      /* place, or 0xffff if none. */

typedef struct                  /* CLEANUP ITEMSItems to */
{                               /* restore on exit. */
    /*                               */
    int        MOCATCH_was_off; /* Must remove MOCATCH. */
    unsigned delayed_flag;      /* Window was previously delayed. */
} CLEANUP_ITEMS;
typedef struct                  /* USER_RESPONSEMouse or */
{                               /* keyboard event. */
    enum {MOUSE, KEYSTROKE, INTERNAL_ERROR} mouse_or_key;
    union
    {
        struct                /* Details of mouse event: */
        {                     /*                               */
            unsigned long      event; /* Actual mouse event. */
            unsigned          vert,horiz; /* Mouse cursor location */
                                   /* (in characters). */
            const BMNMOUSE *pmouse; /* Mouse event node. */
            BITEM             *pitem; /* Menu item where mouse */
        } mouse;              /* event took place. */
        struct                /* Details of keystroke: */
        {                     /*                               */
            KEY_SEQUENCE kseq; /* Character and key codes*/
            const BKEYMAP *pkey; /* Key map entry. */
        } keystroke;
    } data;
} USER_RESPONSE;
#define TRUE 1
#define FALSE 0
/* Internal functions. */
static void clean_up(const BMENU *,const CLEANUP_ITEMS *);
static int mnmchxdt(const BKEYMAP *,const BMNMOUSE *,int);
static int in_window(const BWINDOW *,unsigned,unsigned);
static BITEM *pfind_item(const BMENU *,unsigned,unsigned);
static const BMNMOUSE *pfind_mouse_event(const BMNMOUSE *,unsigned long,int *);
static int wait_operator(const BMENU *,USER_RESPONSE *,
                        const BKEYMAP *,int);
static int interp_resp(BMENU *,const USER_RESPONSE *,int *,BITEM **,
                        int *,int *);
static int jump(BWINDOW *,int);
int mnmread (pmenu, srow, scol, prow, pcol, pch, pscan, option)
BMENU *pmenu;
int srow, scol;
int *prow, *pcol;
int *pch, *pscan;
int option;
{

```

```

int      code,          /* Returned error code from MNFINDSL */
done = FALSE,          /* Becomes true when MNREAD should */
                        /* exit at next opportunity. */
result,                /* Return code from interp_resp(). */
shouldbeep = FALSE;    /* Whether we should beep or not. */
BITEM *pitem;           /* Pointer to current menu item. */
const BKEYMAP *pkey;    /* Pointer to current key found. */
USER_RESPONSE;
CLEANUP_ITEM;

/* Validate menu data structures. */
if (mnvalmnu (pmenu) == NIL)
    wnreturn (MN_BAD_MENU);
/* Exit if menu not displayed. */
if ( ( pmenu->pwin->where shown.dev != SC_MONO
      && pmenu->pwin->where shown.dev != SC_COLOR)
    wnreturn (WN_NOT_SHOWN);
/* Exit if window is covered or frozen. */
if (pmenu->pwin->internals.frozen)
    wnreturn (WN_COVERED);
/* Disable keys which do not point either at items
   * or at (-1,-1). */
if (mndisabl (pmenu) == NIL)
    return (b_wnerr);
/* Check to make sure there is an exit route. */
if (mnchkxit(pmenu->pkeys,pmenu->pmouse,option))
    return (b_wnerr);
/* Set pitem to highlight bar location (and set
   * previous location if no bar shown). */
if (option & MN_PREV_BAR)
{
    pitem = pmenu->pbar_item; /* Use former location */
                          /* or none if no bar shown. */
    if ( (pitem == NIL)
        && (option & MN_SHOW_BAR)) /* We require a bar, */
        pitem = pmenu->pbar_prev; /* so try harder. */
}
else
{
    /* If not using previous, */
    option |= MN_SHOW_BAR; /* then we must show a bar. */
    pitem = NIL;
}
if ( (pitem == NIL)
    && (option & MN_SHOW_BAR)) /* We still need a bar. */
{
    /* Find the first selectable menu item. */
    pitem = mnfindsl(pmenu,NIL,srow,scol,&code);
    if (pitem == NIL)
    {
        if ( code == WN_NO_ERROR /* Error if none selectable */
            && !(option & MN_HIDE_BAR)) /* unless none wanted. */

```

```

        wreturn (MN_NONE_SELECT);
    else
        return (b_werr);          /* Internal error. */

    if ( (option & MN_HIDE_BAR) /* Must hide the bar and */
        && (pitem != NIL)) /* there is a bar to hide. */
    {
        pmenu->pbar_prev = pitem; /* Save previous bar site. */
        pitem = NIL; /* Show no bar. */
    }

    /* Point pkey at menu's keylist. */
    pkey = pmenu->pkeys;
    /* Save the starting row and column coordinates. */
    if (pitem)
    {
        *prow = pitem->row;
        *pcol = pitem->col;
    }

    /* Save items that may need restoration on exit. */
    dirt.MOCATCH_was_off = !b_mocatch;
    dirt.delayed_flag = pmenu->pwin->options.delayed;
    /* Miscellaneous setup. */
    pmenu->pwin->options.delayed = 0; /* Allow immediate output. */
    /* Check whether we should always transmit. */
    if (option & MN_ALL_TRANSMIT)
        done = TRUE;
    do
    {
        /* Show highlight bar and (perhaps) description. */
        if (NIL == mnhibit0(pmenu,pitem,
            (option & MN_USE_DESCRIPTION) | WN_UPDATE | MN_HIGHLIGHT))
        {
            clean_up(pmenu,&dirt);
            return b_werr;
        }
        /* Do a beep if the last keycode told us to do so.... */
        if (shouldbeep)
        {
            scttywrt ('\a', 0);
            shouldbeep = FALSE;
        }
        /* Get a keystroke or mouse event from the user. */
        if (WN_NO_ERROR != wait_operator(pmenu,&resp,pkey,option))
        {
            clean_up(pmenu,&dirt);
            return b_werr;
        }
        switch (resp.mouse_or_key)
        {

```

```

    case KEYSTROKE:
        *pch = resp.data.keystroke.kseq.character_code;
        *pscan = resp.data.keystroke.kseq.key_code;
        if (resp.data.keystroke.pkey != NIL)
            pkey = resp.data.keystroke.pkey;
        b_mnmoevent = 0L;
        b_mnmovert = 0xffff;
        b_mnmohoriz = 0xffff;
        break;
    case MOUSE:
        *pch = 0xff;
        *pscan = 0xff;
        b_mnmoevent = resp.data.mouse.event;
        b_mnmovert = resp.data.mouse.vert;
        b_mnmohoriz = resp.data.mouse.horiz;
        break;
}
/* Interpret user response in terms of menu results. */
switch (result = interp_resp(pmenu,&resp,&option,&pitem,
                             &shouldbeep,&done))
{
    case WN_NO_ERROR:
    case MN_READ_AB:
    case MN_XMIT_NOBAR:
    case MN_UNKNOWN_AB:
        break;
    default:
        clean_up(pmenu,&dirt);
        return result; /* Internal error. */
}
} while (!done);
clean_up(pmenu,&dirt);
/* Show highlight bar and (perhaps) description. */
if (NIL == mnhibit0(pmenu,pitem,
                    (option & MN_USE_DESCRIPTION) | WN_UPDATE | MN_HIGHLIGHT))
    return b_werr;
/* Check whether we should beep. */
if (shouldbeep && (!(option & MN_HOLD_BEEP)))
    scattywrt ('\a', 0);
/* Perhaps remove highlight bar and description. */
if (NIL == mnhibit0(pmenu,pitem,
                    WN_UPDATE
                    | ((option & MN_KEEP_HIGHLIGHT) ? MN_HIGHLIGHT : MN_UNHIGHLIGHT)
                    | ((option & MN_KEEP_DESCRIPTION) ? MN_USE_DESCRIPTION : 0)))
    return b_werr;
/* Return the final row and column of the highlight */
/* bar. */
if (pitem == NIL)
    pitem = pmenu->pbar_prev;
if (pitem == NIL)

```

```

{
    *pcol = -1;
    *prow = -1;
}
else
{
    *pcol = pitem->col;
    *prow = pitem->row;
}

    /* Remove menu's window from display if told to. */
if (option & MN_REMOVE)
    if (wuremove (pmenu->pwin) == NIL)
        return (b_werr);
    /* Destroy menu's data structures if told to do so. */
if ((option & MN_DESTROY) == MN_DESTROY)
    if (mndstroy (pmenu) != WN_NO_ERROR)
        return (b_werr);
return (result);
}

```

```

/**
 * Name      clean_up -- Restore miscellaneous aspects of
 *            environment.
 * Synopsis   clean_up(pmenu,pdirt);
 *            const CLEANUP ITEMS *pmenu
 *            Address of the menu structure.
 *            const CLEANUP ITEMS *pdirt
 *            Address of record structure listing
 *            items to restore on exit from
 *            MNREAD.
 * Description This function restores miscellaneous items:
 *            specifically, if MOCATCH was not installed before MNREAD
 *            was invoked, this function removes it; also restores the
 *            menu's window's "delayed" state.

```

```

 * Returns    b mocatch      Restored to value before MNREAD was
 *            called.

```

\*/

```

static void clean_up(pmenu,pdirt)
const BMENU      *pmenu;
const CLEANUP ITEMS *pdirt;
{
    if (pdirt->MOCATCH was off)
        mopreclk(MO_REMOVE);
    pmenu->pwin->options.delayed = pdirt->delayed flag;
}

```

/\*\*

```

 * Name      mnchkxit -- Check whether there is any way to
 *            exit the menu.
 * Synopsis   ret = mnchkxit(pkey,pmouse,option);
 *            int      ret      Zero if there is an exit route (i.e. no

```

```

*                                     error);
*
*                                     MN_BAD_KEY if a bad keymap
*                                     entry encountered;
*
*                                     MN_BAD_MOUSE if a bad mouse
*                                     entry encountered;
*
*                                     MN_NO_EXIT if there is no exit key.
*
* const BKEYMAP *pkey
*                                     Pointer to the start of the menu's
*                                     key list.
*
* const BMNMOUSE *pmouse
*                                     Address of first item in list of menu's
*                                     mouse actions.
*
* int option      MNREAD options including the following
*                                     bits:
*
*                                     MN_KBIGNORE
*                                     MN_ALL_TRANSMIT
*                                     MN_UNKNOWN_TRANSMIT
*
* Description     MNCHKXIT checks the option value and the menu's key and
*
*                 mouse lists to see if there is any way for the menu to
*                 be exited.
*
* Returns         ret      Zero if there is an exit route (i.e. no
*
*                 error);
*
*                 MN_BAD_KEY if a bad keymap
*                 entry encountered;
*
*                 MN_BAD_MOUSE if a bad mouse
*                 entry encountered;
*
*                 MN_NO_EXIT if there is no exit key.
*
* b_werr          Possible values:
*
*                 (No change) Success.
*
*                 MN_NO_EXIT No way to exit the
*                 menu.
*
*                 MN_BAD_KEY Bad keymap entry
*                 encountered.
*
*                 MN_BAD_MOUSE Bad mouse entry
*                 encountered.
**/
static int mnchkxit(pkey,pmouse,option)
const BKEYMAP*pkey;
const BMNMOUSE *pmouse;
int      option;
{
    if (option & (MN_ALL_TRANSMIT | MN_UNKNOWN_TRANSMIT))
        return 0;

    if (! (option & MN_KBIGNORE))
    {
        for (; pkey != NIL; pkey = pkey->next)
        {
            /* Check key signature.
            if (pkey->signature != MN_KEY_SIGN)

```

```

        wnreturn (MN_BAD_KEY);
    if (    !(pkey->action & (MN_DISABLE | MN_TEMP_DISABLE))
        && (pkey->action & (MN_TRANSMIT | MN_ABORT)))
        return (0);
    }
}
for (;    pmouse != NIL; pmouse = pmouse->pnext)
{
    /* Check mouse signature. */
    if (pmouse->signature != MN_MOU_SIGN)
        wnreturn (MN_BAD_MOUSE);
    if (    !(pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
        && (pmouse->action & (MN_TRANSMIT | MN_ABORT)))
        return (0);
}
wnreturn (MN_NO_EXIT);
}

```

```

/**
 * Name          pfind_item -- Find menu item that covers a given
 *                  screen location.
 * Synopsis      pitem = pfind_item(pmenu,row,col)
 *                  BITEM *pitem  Address of resulting menu item node,
 *                  or NIL if error or if none found.
 *                  const BMENU *pmenu
 *                  Address of menu control structure.
 *                  int row,col    Display location (relative to (0,0) at
 *                  upper left corner of display).
 * Description    This function finds a menu item (if any) that covers a
 *                  given display location.
 *                  The result is meaningless if the menu is not displayed
 *                  or (row,col) is outside the viewport.
 * Returns        *pitem          Address of resulting menu item node,
 *                  or NIL if error or if none found.
 *                  b_wnerr       Unchanged if no error or if none found;
 *                  MN_BAD_ITEM if corrupted item list.
 **/

```

```

static BITEM *pfind_item(pmenu,row,col)
const BMENU *pmenu;
unsigned      row,col;
{
    BITEM *pitem;
    int target row =      row
                    - pmenu->pwin->where_shown.corner.row
                    + pmenu->pwin->data_origin.row;
    int target col =      col
                    - pmenu->pwin->where_shown.corner.col
                    + pmenu->pwin->data_origin.col;
    for (pitem = pmenu->pitems;

```

```

        pitem != NIL;
        pitem = pitem->next)
    {
        if (pitem->signature != MN_ITEM_SIGN)
        {
            wnreferr(MN_BAD_ITEM);
        }
        if (    pitem->row == target_row
            && pitem->col <= target_col
            && target_col < pitem->col + pitem->len)
        {
            return pitem;
        }
    }
    return NIL;
}

/**
 * Name          in_window -- Report whether window data area contains
 *                  a given display location.
 * Synopsis      contains = in_window(pwin,row,col);
 *                  int contains      Nonzero if window contains the
 *                                  given location,
 *                                  zero if not.
 *                  const BWINDOW *pwin
 *                                  Address of window control structure.
 *                  int row,col      Display location (relative to (0,0) at
 *                                  upper left corner of display).
 * Description    This function tells whether a given display location
 *                  lies within a window's data area.  The window must be
 *                  displayed.
 * Returns        contains      Nonzero if window contains the
 *                                  given location,
 *                                  zero if not.
 */
static int in_window(pwin,row,col)
const BWINDOW *pwin;
unsigned      row,col;
{
    /* Macros to return edge of data area (plus 1). */
#define BOT_ROW_P1 (pwin->where_shown.corner.row + wnview_h(pwin))
#define RT_COL_P1 (pwin->where_shown.corner.col + wnview_w(pwin))
    return (    pwin->where_shown.corner.row <= row
        && row < BOT_ROW_P1
        && pwin->where_shown.corner.col <= col
        && col < RT_COL_P1);
#undef BOT_ROW_P1
#undef RT_COL_P1
}

```



```

/**
 * Name      pfind_mouse_event -- Find mouse event in list.
 * Synopsis  pfound = pfind_mouse_event(pmouse,event,perror)
 *           const BMNMOUSE *pfound
 *           Returned address of found mouse event
 *           record.
 *           const BMNMOUSE *pmouse
 *           Address of first mouse node to check.
 *           unsigned long event
 *           Actual mouse event to search for.
 *           int *perror      Address into which to return success
 *                           code.
 * Description This function searches for a matching mouse event
 * in a list of mouse events, taking into account
 * what aspects should be ignored.
 * Returns    pfound      Address of found mouse event record,
 *                       or NIL if error or if none found.
 *           *perror      WN_NO_ERROR or MN_BAD_MOUSE
 */
static const BMNMOUSE *pfind_mouse_event(pmouse,event,perror)
const BMNMOUSE *pmouse;
unsigned long   event;
int             *perror;
{
    for ( ;
        pmouse != NIL;
        pmouse = pmouse->pnext)
    {
        if (pmouse->signature != MN_MOU_SIGN)
        {
            wretterr(*perror = MN_BAD_MOUSE);
        }
        if ( 0 == (pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
            && (pmouse->ignore | event) == (pmouse->ignore | pmouse->event))
        {
            *perror = WN_NO_ERROR;
            return pmouse;
        }
    }
    *perror = WN_NO_ERROR;
    return NIL;
}

/**
 * Name      wait_operator -- Wait for mouse or keyboard command
 *           from operator.
 * Synopsis  ercode = wait_operator(pmenu,prsp,pkey,option);
 *           int ercode      Error code:
 *                           WN_NO_ERROR if okay;
 *                           other if internal error.
 *           const BMENU *pmenu

```

```

*                               Address of menu structure.
*
*   USER_RESPONSE *presp
*
*                               Address of structure containing the
*                               returned user response.
*
*   const BKEYMAP *pkey
*
*                               Address of keymap entry for most
*                               recent keystroke.
*
*   int option      Option flag.   Relevant bits:
*                               MN_KBIGNORE
*                               MN_KBDISCARD (both have same effect).
*
* Description      This function polls the mouse and keyboard for a response
*                  from the user.
*                  Mouse events are reported only if they are listed
*                  in the menu's list of mouse events.
*
* Returns          ercode          Error code:
*
*                  WN_NO_ERROR if okay;
*                  other if internal error.
*
*                  b_wnerr         Unchanged if no error;
*                  same as "ercode" if error detected.
*
*                  *presp          User response.
**/
static int wait_operator(pmenu,presp,pkey,option)
const BMENU      *pmenu;
USER_RESPONSE *presp;
const BKEYMAP    *pkey;
int              option;
{
    unsigned long event;
    BITEM          *pitem;
    unsigned       vert,hORIZ;
    int            ercode;
    const BMNMOUSE *pmouse;
    for (;;)      /* Infinite loop:   First poll mouse, then      */
    {              /* keyboard.   Exit when anything happens.    */
                  /* Poll mouse history separately for each      */
                  /* listed mouse event.                        */
        for (pmouse = pmenu->pmouse;
             pmouse != NIL;
             pmouse = pmouse->pnext)
        {
            if (pmouse->signature != MN_MOU_SIGN)
            {
                wnreturn(MN_BAD_MOUSE);
            }
            if ( 0 == (pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
                && MO_OK == mocheck(pmouse->event,pmouse->ignore,
                                     MO_CLEAR,&event,&vert,&horiz)
                && event)
            {
                /* Analyze location of mouse event:   whether on a */

```

```

/* menu item and whether outside the menu. */
vert >>= 3; /* Convert from pixels to characters. */
horiz >>= 3;
if (in_window(pmenu->pwin,vert,horiz))
{
    if (NIL == (pitem = pfind_item(pmenu,vert,horiz)))
        event = (event | MN_OFF_ITEMS)
                & ~(unsigned long) (MN_ITEMS | MN_OUT_MENU);
    else
        event = (event | MN_ITEMS)
                & ~(unsigned long) (MN_OFF_ITEMS | MN_OUT_MENU);
}
else
    event = (event | MN_OUT_MENU)
            & ~(unsigned long) (MN_OFF_ITEMS | MN_ITEMS);
/* Mouse event is now completely characterized, so */
/* scan list of menu events for this menu. Ignore */
/* event if not found in list. */
if (NIL != (presp->data.mouse.pmouse
            = pfind_mouse_event(pmenu->pmouse,event,&rcode)))
{
    /* Found mouse event. */
    presp->mouse_or_key = MOUSE;
    presp->data.mouse.event = event;
    presp->data.mouse.vert = vert;
    presp->data.mouse.horiz = horiz;
    presp->data.mouse.pitem = pitem;
    return rcode;
}
}
/* No relevant mouse event, so poll keyboard. */
if (kbpoll(b_key_ctrl,pmenu,
           &presp->data.keystroke.kseq,
           KB_REMOVE_KEY))
if (0 == (option & (MN_KBIGNORE | MN_KBDISCARD)))
{
    presp->mouse_or_key = KEYSTROKE;
    presp->data.keystroke.pkey =
        namchkey(pmenu,pkey,
                presp->data.keystroke.kseq.character_code,
                presp->data.keystroke.kseq.key_code,
                &rcode);
    return rcode;
}
}
}

```

/\*\*

\* Name           interp\_resp -- Interpret user response and decide  
\*   actions to take.

\* Synopsis       rcode = interp\_resp(pmenu,presp,poption,ppitem,

```

*                                     pshouldbeep,pdone);
*
* int ercode      Error code:
*
*      WN_NO_ERROR if okay;
*      MN_READ_AB if user requested abort;
*      MN_XMIT_NOBAR if user requested transmit while
*      no bar shown;
*      MN_UNKNOWN_AB if user pressed an unknown key
*      the MN_UNKNOWN_TRANSMIT option in *poption
*      was set;
*      other if internal error.
*
* BMENU *pmenu Address of menu structure.
*
* const USER_RESPONSE *presp
*      Address of structure containing the
*      actual user response.
*
* int *poption   Address of option flag.
*
* BITEM **ppitem Address of pointer to highlighted menu
*      item (*pitem == NIL if none).
*
* int *pshouldbeep
*      Address of flag indicating whether
*      a beep was requested.
*
* int *pdone     Address of flag indicating whether
*      calling function should return without
*      further user input.
*
* Description    This function interprets the user's response (in *presp)
*
* and determines what action to take.
*
* Returns       ercode      Error code:
*
*      WN_NO_ERROR if okay;
*      MN_READ_AB if user requested abort;
*      MN_XMIT_NOBAR if user requested transmit while
*      no bar shown;
*      MN_UNKNOWN_AB if user pressed an unknown key
*      the MN_UNKNOWN_TRANSMIT option in *poption
*      was set;
*      other if internal error.
*
* h_werr        Unchanged if no error;
*      same as "ercode" if error detected.
*
* *poption      The MN_KB_DISCARD bit may be set or
*      cleared.
*
* *ppitem       Menu item to highlight (NIL if no
*      highlight bar).
*
* *pshouldbeep  TRUE if beep requested;
*      unchanged if no beep requested.
*
* *pdone        TRUE if caller should return;
*      unchanged if further input okay.
*
**/
*
*      /* Macro to do a for loop which makes the variable */
*      /* qitem take on all of the pointer values from */
*      /* pitem to the end of the item list in the menu's */
*      /* item list.  ritem is initialized to be pitem. */
*
#define FORALL(pitem)

```

```

        for (qitem = pmenu->pitems, ritem = (pitem); \
            qitem != NIL; \
            qitem = qitem->next) \
        { \
            if (qitem->signature != MN_ITEM_SIGN) \
                wreturn (MN_BAD_ITEM); \
        } \
#define FOREND() } \
/* Macro which maximizes a "rangefunc". This is */ \
/* used by MN_UP, DOWN, RIGHT, and LEFT to determine*/ \
/* the correct place to move the highlight bar. */ \
/* Arguments: */ \
/* matchrc -- Either "row" or "col"; the */ \
/* component of the item's */ \
/* position which is supposed to */ \
/* match that of the other item. */ \
/* diffrc -- Either "row" or "col"; the */ \
/* opposite of "matchrc". */ \
/* size -- Either height (if matchrc is */ \
/* "col") or width (if matchrc is */ \
/* "row") of menu's window data */ \
/* area. */ \
/* rangefunc- Either ULRANGE or DRRANGE. */ \
#define IFMATCH(matchrc, diffrc, size, rangefunc, pitem) \
    if ((!(qitem->option & MN_PROTECT)) && \
        (qitem->matchrc == (pitem)->matchrc) && \
        (rangefunc ((pitem), qitem, diffrc, size) > \
         rangefunc ((pitem), ritem, diffrc, size))) \
#define DRRANGE(pitem, qitem, rowcol, size) \
    ((qitem->rowcol <= (pitem)->rowcol) \
     ? ((pitem)->rowcol - qitem->rowcol) & wrap_mask \
     : ((size - qitem->rowcol) + (pitem)->rowcol)) \
#define ULRANGE(pitem, qitem, rowcol, size) \
    ((qitem->rowcol >= (pitem)->rowcol) \
     ? (qitem->rowcol - (pitem)->rowcol) & wrap_mask \
     : ((size - (pitem)->rowcol) + qitem->rowcol)) \
/* BEYOND: Macro that maximizes a "beyondfunc": returns */ \
/* nonzero value if *qitem is a better candidate than *ritem. */ \
/* Use is similar to IFMATCH. */ \
/* size - height or width of data area. */ \
/* floor - preferred amount of motion (negative for upward */ \
/* or leftward. */ \
#define BEYOND(matchrc,diffrc,size,beyondfunc,pitem,floor) \
    ( (!(qitem->option & MN_PROTECT)) \
      && (qitem->matchrc == (pitem)->matchrc) \
      && (beyondfunc((pitem),qitem,diffrc,size,floor) > \
         beyondfunc((pitem),ritem,diffrc,size,floor))) \
#define DRBEYOND(pitem,qitem,rowcol,size,floor) \
    ( (qitem->rowcol >= (pitem)->rowcol + (floor)) \
      ? size - qitem->rowcol \
      : qitem->rowcol - size)

```

```

#define ULBEYOND(pitem,qitem,rowcol,size.ceiling) \
    ( (qitem->rowcol <= (pitem->rowcol + (ceiling)) \
      ? qitem->rowcol \
      : -1 - qitem->rowcol) \
      /* Height of menu's data area. */
#define H (pmenu->pwin->img.dim.h) \
      /* Width of menu's data area. */
#define W (pmenu->pwin->img.dim.w)
static int interp_resp(pmenu,presp,poption,ppitem,
                      pshouldbeep,pdone)
BMENU          *pmenu;
const USER_RESPONSE *presp;
int             *poption;
BITEM          **ppitem;
int             *pshouldbeep,*pdone;
{
    int    code;
    BITEM *pitem,*qitem,*ritem;
    int    result = WN_NO_ERROR;
    int    action;
    int    disp;
    int    wrap_mask;
    pitem = *ppitem;
        /* Do operations that are different for mouse and */
        /* keystroke events:    in particular, get action code*/
    switch (presp->mouse_or_key)
    {
        case KEYSTROKE:
            if (presp->data.keystroke.pkey != NIL)
            {
                action = presp->data.keystroke.pkey->action;
                if (MNMOVE(action) == MN_SELECT)
                {
                    const BKEYMAP *pkey;
                    /* Move highlight bar to specified row/column of a */
                    /* menu item. */
                    pkey = presp->data.keystroke.pkey;
                    if ((qitem =
                        mnmatchm (pmenu, pitem, pkey->row,
                                pkey->col, 0, &code)) != NIL)
                        pitem = qitem;
                    else if (code != WN_NO_ERROR)
                    {
                        *pdone = TRUE;
                        return (b_wnerr);
                    }
                }
            }
        else /* Unknown (or disabled) key pressed. */

```

```

    action = 0;

    /* Check whether we should beep when unknown key */
    /* pressed. */
    if (*poption & MN_UNKNOWN_BEEP)
        *pshouldbeep = TRUE;
    /* Check whether we should transmit when unknown */
    /* key pressed. */
    if (*poption & MN_UNKNOWN_TRANSMIT)
    {
        result = wnterror(MN_UNKNOWN_AB);
        *pdone = TRUE;
    }
}
break;
case MOUSE:
    action = presp->data.mouse.pmouse->action;
    if (MNMOVE(action) == MN_SELECT)
        pitem = presp->data.mouse.pitem;
    break;
}
switch (MNMOVE(action))
{
    case MN_UP:
    case MN_DOWN:
    case MN_RIGHT:
    case MN_LEFT:
    case MN_NEXT:
    case MN_PREVIOUS:
    case MN_PGDN:
    case MN_PGUP:
    case MN_PGRIGHT:
    case MN_PGLEFT:
        /* Handle relative movements specially: if bar */
        /* not shown, use previously highlighted item. */
        /* This has the effect of showing the bar unless */
        /* MN_HIDE_BAR is also specified in the action code.*/
        if (pitem == NIL)
            pitem = pmenu->pbar_prev;
        if (pitem != NIL)
        {
            /* wrap_mask is used by the DRRANGE and ULRANGE */
            /* macros for the MN_UP, MN_DOWN, MN_RIGHT, and */
            /* MN_LEFT motions. When MN_NOWRAP is specified, */
            /* then wrap_mask is 0, thus causing DRRANGE and */
            /* ULRANGE to return 0 for coordinates in the */
            /* opposite direction. */
            wrap_mask = ((action & MN_NOWRAP) ? 0 : ~0);
            switch (MNMOVE(action))

```

```

{
/* Move highlight bar up.  Wrap to bottom if at top.*/
case MN_UP:
    FORALL(pitem)
        IFMATCH (col, row, H, ULRANGE, pitem)
            ritem = qitem;
    FOREND ()
    pitem = ritem;
    break;
case MN_DOWN:
/* Move highlight bar down.  Wrap to top if at */
/* bottom. */
    FORALL(pitem)
        IFMATCH (col, row, H, DRRANGE, pitem)
            ritem = qitem;
    FOREND ()
    pitem = ritem;
    break;
case MN_RIGHT:
/* Move highlight bar right.  Wrap to left if at */
/* rightmost item. */
    FORALL(pitem)
        IFMATCH (row, col, W, DRRANGE, pitem)
            ritem = qitem;
    FOREND ()
    pitem = ritem;
    break;
case MN_LEFT:
/* Move highlight bar left.  Wrap to right if at */
/* leftmost item. */
    FORALL(pitem)
        IFMATCH (row, col, W, ULRANGE, pitem)
            ritem = qitem;
    FOREND ()
    pitem = ritem;
    break;
case MN_NEXT:
/* Move highlight bar to "logical-next" menu */
/* position.  This is the next item in the order */
/* that items were entered by MNITEM and/or */
/* MNITEMKEYWrap to beginning if at end unless */
/* MN_NOWRAP specified. */
    if (action & MN_NOWRAP)
    {
        for (qitem = pitem->next;
            qitem != NIL;
            qitem = qitem->next)
        {
            if (!(qitem->option & MN_PROTECT))
            {

```



```

        pitem = qitem;
        break;
    }
}
else
{
    /* MNMCHITM automatically wraps. */
    pitem = mnmchitm (pmenu, (*ppitem)->next,
                    -1, -1, 0, &code);
    if (code != WN_NO_ERROR)
    {
        *pdone = TRUE;
        return (b wnerr);
    }
}
break;

case MN_PREVIOUS:
/* Move highlight bar to "logical-previous" menu */
/* position. This is the previous item in the */
/* order that items were entered by MNITEM and/or */
/* MNITMKEYWrap to end if at beginning. */
if (pitem == mnmchitm (pmenu, pmenu->pitems,
                    -1, -1, 0, &code))
{
    /* pitem is first in list. */
    if (action & MN_NOWRAP)
        ritem = pitem;
    else
    {
        /* Find last in list. */
        FORALL(pitem)
            if (! (qitem->option & MN_PROTECT))
                ritem = qitem;
        FOREND ()
    }
}
else
{
    if (code != WN_NO_ERROR)
    {
        *pdone = TRUE;
        return (b wnerr);
    }
    for (ritem = pitem, qitem = pmenu->pitems;
        qitem != pitem;
        qitem = qitem->next)
        if ( ! (qitem->option & MN_PROTECT))
            ritem = qitem;
}

pitem = ritem;
break;

```

```

case MN_PGDN:
    disp = jump(pmenu->pwin,MN_PGDN);
    FORALL(pitem)
        if (BEYOND(col,row,H,DRBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;
case MN_PGUP:
    disp = jump(pmenu->pwin,MN_PGUP);
    FORALL(pitem)
        if (BEYOND(col,row,H,ULBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;
case MN_PGRIGHT:
    disp = jump(pmenu->pwin,MN_PGRIGHT);
    FORALL(pitem)
        if (BEYOND(row,col,W,DRBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;
case MN_PGLEFT:
    disp = jump(pmenu->pwin,MN_PGLEFT);
    FORALL(pitem)
        if (BEYOND(row,col,W,ULBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;
    }
}
break;
case MN_FIRST:
    /* Move highlight bar to first item entered by */
    /* MNITEM and/or MNITMKEY. */
    pitem = mnmcchitm (pmenu, pmenu->pitems, -1, -1, 0, &code);
    if (code != WN_NO_ERROR)
    {
        *pdone = TRUE;
        return (b_werr);
    }
    break;
case MN_LAST:
    /* Move highlight bar to last item entered by */
    /* MNITEM and/or MNITMKEY. */
    FORALL(pitem)
        if ( !(qitem->option & MN_PROTECT))

```

```

        ritem = qitem;
    FOREND ()
    pitem = ritem;
    break;
}
if (action & MN_SHOW_BAR)
{
    if (pitem == NIL)          /* If no menu item lit,      */
        pitem = pmenu->pbar_prev; /* find an item to highlight*/
}
else if (action & MN_HIDE_BAR)
{
    if (pitem != NIL)          /* If an item is lit,      */
    {
        pmenu->pbar_prev = pitem; /* save location          */
        pitem = NIL;             /* and unlight the bar.    */
    }
}
if (action & MN_BEEP)
    *pshouldbeep = TRUE;      /* Forward beep request.   */
if (action & MN_KBIGNORE)
    *poption |= MN_KBDISCARD; /* Ignore keystrokes.      */
else
    *poption &= ~MN_KBDISCARD; /* Begin watching keys.    */
if (action & MN_ABORT)
{
    /* User requested abort. */
    result = werror(MN_READ_AB);
    *pdone = TRUE;
}
else if (action & MN_TRANSMIT)
{
    /* User requested transmit. */
    if (pitem == NIL)
        result = werror(MN_XMIT_NOBAR); /* Error if transmit */
                                          /* while no item lit.*/
    *pdone = TRUE;
}
}
*ppitem = pitem;
return result;
}

```

/\*\*

\* Name            jump -- Adjust window origin for MN\_PGUP, MN\_PGDN,  
                  MN\_PGRIGHT, or MN\_PGLEFT motion.

\* Synopsis        displacement = jump(pwin,dir);

\*                int displacement   Preferred amount of motion by  
                  highlight bar (negative for  
                  upward or toward left edge).

\*                BWINDOW \*pwin   Address of menu's window.

```

*          int dir          MN_PGUP, MN_PGDN, MN_PGRIGHT, or
*                           MN_PGLEFT.
*
* Description    This function moves the window's origin and
*                computes the distance the highlight bar should move.
*
* Returns       displacement Preferred amount of motion by
*                highlight bar (negative for
*                upward or toward left edge).
*                b_werr      Window error if any.
*
**/
static int jump(pwin,dir)
BWINDOW *pwin;
int      dir;
{
    int new_origin_row,old_origin_row,new_origin_col,old_origin_col;
    int displacement;
    new_origin_row = old_origin_row = pwin->data.origin.row;
    new_origin_col = old_origin_col = pwin->data.origin.col;
    switch (dir)
    {
        case MN_PGDN:
            new_origin_row = utmin(old_origin_row + wnview_h(pwin) - 1,
                                   wndata_h(pwin) - wnview_h(pwin));
            if (new_origin_row == old_origin_row)
                displacement = wnview_h(pwin) - 1;
            else
                displacement = new_origin_row - old_origin_row;
            break;
        case MN_PGUP:
            new_origin_row = utmax(old_origin_row - (wnview_h(pwin) - 1),
                                   0);
            if (new_origin_row == old_origin_row)
                displacement = 1 - wnview_h(pwin);
            else
                displacement = new_origin_row - old_origin_row;
            break;
        case MN_PGRIGHT:
            new_origin_col = utmin(old_origin_col + wnview_w(pwin) - 1,
                                   wndata_w(pwin) - wnview_w(pwin));
            if (new_origin_col == old_origin_col)
                displacement = wnview_w(pwin) - 1;
            else
                displacement = new_origin_col - old_origin_col;
            break;
        case MN_PGLEFT:
            new_origin_col = utmax(old_origin_col - (wnview_w(pwin) - 1),
                                   0);
            if (new_origin_col == old_origin_col)

```

```

        displacement = 1 - wnview_w(pwin);
    else
        displacement = new_origin_col - old_origin_col;
    break;
}
wnorigin(pwin,new_origin_row,new_origin_col,WN_NO_UPDATE);
return displacement;
}

```

## MNMCHITM 匹配—说明条件的菜单项。

presult = mnmmchitm (pmenu, pitem, row, col, iprot, pcode);

presult 指向菜单项中匹配于该行和列的项，若匹配失败则指向NIL。

pmenu 指向要查找的项表的菜单。如果到达项表尾和需带整字换行，则需要该函数。

pitem 指向开始查找的项表中项，如果从第一项开始查找则指向NIL。

row, 指定的行

col 指定的列

如果上述两参数其中一个或两个为-1，则忽略指定的条件。

iprot 如果iprot是1，则也搜寻保护项。

pcode 指向返回错误代码的变量的指针。如果没有代码返回，则为指向NIL。

MNMCHITM从给定点在菜单的项表中查找匹配说明的一个项。

函数只匹配非保护项，如果想保护项也匹配，则使iprot为1。

如果row或col(或两者)为-1，则采用通配说明。例如：

```
presult = mnmmchitm (pmenu, pitem, -1, col);
```

将匹配(从pitem开始的)col列的首项。与row无关。

下面的语句匹配下一非保护项：

```
presult = mnmmchitm (pmenu, pitem->next, -1, -1, 0);
```

返回 presult 指向 找到的BITEM,如果未找到则指向NIL。

源程序(MNMCHITM.C):

```
#include <bmenu.h>
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
BITEM *mnmmchitm (pmenu, pitem, row, col, iprot, pcode)
```

```
BMENU *pmenu;
```

```
BITEM *pitem;
```

```
int row, col;
```

```
int iprot;
```

```
int *pcode;
```

```

{
    int gotnil = FALSE;
    if (pcode != NIL)
        *pcode = WN_NO_ERROR;
    for (;;)
    {
        /* If we already found the tail of the list, and we */
        /* just found it again, we know that we will not */
        /* find what we are looking for. */
        if ((pitem == NIL) && gotnil)
            return (NIL);
    }
}

```

```

        /* Wrap around to head of list if at tail.      Record */
        /* that we got to tail once so we know when we are */
        /* done if we don't find what we are looking for. */
else if (pitem == NIL)
{
    if ((pitem = pmenu->pitems) == NIL)
        return (NIL);
    gotail = TRUE;
}

    /* Check item signature. */
if ((pitem != NIL) && (pitem->signature != MN_ITEM_SIGN))
{
    if (pcode != NIL)
        *pcode = MN_BAD_ITEM;
    wnretterr (MN_BAD_ITEM);
}

    /* If *pitem matches the given specification, we */
    /* are done, so return the item we found. */
if (((!(pitem->option & MN_PROTECT)) || !prot)
    && (((row == -1) && (col == -1)) ||
        ((row == -1) && (pitem->col == col)) ||
        ((col == -1) && (pitem->row == row)) ||
        ((pitem->row == row) && (pitem->col == col))))
    return (pitem);

    /* Didn't find it yet--go on to next. */
pitem = pitem->next;
}
}

```

### MNMCHKEY 匹配—说明的键。

result = mnmcchkey (pmenu, pkey, ch, scan, perror);

result 指向菜单键表中匹配于ch和scan的键，若匹配失败则指向NIL

pmenu 指向要查寻键表中的菜单。如果到达键表尾并需带整字换行则该调用是必须的。

pkey 指向表中开始查寻的键，或如果从第一键开始查寻，则指向NIL。

ch, 键表中用于查寻的字符码

scan 键表中用于查寻的扫描码。

pcode 指向返回错误代码变量的指针。如果不必返回错误代码则指向NIL。

MNMCHKE在菜单的键表中从给定点查找匹配说明的键码项。

该函数只匹配MN\_DISABLE位没有设置的键。

它跳过和pkey同列、行和行为域的键。举例如下：

```

+-----+
| Quit   | <-- bound to "Qq"
| Create | <-- bound to "Cc"
| Close  | <-- bound to "Cc"
+-----+

```

当第一次按大写C或小写c时，高亮条移到“Create”，后续的C使高亮条在“Close”和“Create”之间切换。

返回

presult

指向找到的BKEYMAP, 如找到则为NIL。

源程序(MNMCHKET.C):

```
#include <bkeybrd.h>
#include <bmenu.h>
#define TRUE 1
#define FALSE 0
const BKEYMAP *mnmmchkey (pmenu, pkey, ch, scan, pcode)
const BMENU *pmenu;
const BKEYMAP *pkey;
int ch, scan;
int *pcode;
{
    BKEYMAP *pstart = (pkey == NIL) ? (pmenu->pkeys) : (pkey);
    BKEYMAP *qkey = pstart;
    BKEYMAP *rkey = NIL;
    BKEYMAP *skey = NIL;
    int done = FALSE;
    int first = TRUE;
    if (pcode != NIL)
        *pcode = WN_NO_ERROR;
        /* Exit immediately if there is no key list. */
    if (pstart == NIL)
        return (NIL);
        /* Check each key list entry to see if it matches
        /* the given specification.
    for (;
        !done;
        qkey = qkey->next)
    {
        if (qkey == NIL)
            qkey = pmenu->pkeys;
        if ((qkey == pstart) && !first)
            done = TRUE;
            /* Check key signature.
        if (qkey->signature != MN_KEY_SIGN)
        {
            if (pcode != NIL)
                *pcode = MN_BAD_KEY;
                wnreterr (MN_BAD_KEY);
        }
            /* If *qkey is not disabled,
        if ((!(qkey->action & MN_DISABLE)) &&
            (!(qkey->action & MN_TEMP_DISABLE)) &&
            /* and the character and scan codes match the ones
            /* we are looking for, save a pointer to it.
            ((qkey->ch == ch) && (qkey->scan == scan)))
            skey = qkey;
            /* If *qkey is not disabled,
        if ((!(qkey->action & MN_DISABLE)) &&
```

```

        ( !(qkey->action & MN_TEMP_DISABLE))    &&
        /* and the character and scan codes match the ones */
        /* we are looking for, */
        ((qkey->ch == ch) && (qkey->scan == scan)) &&
        /* and at least one of the following is true: */
        /* 1. There was no previous key; */
        ((pkey == NIL) ||
        /* 2. We went all the way around the list and the */
        /* key we were given at first is the only one */
        /* which matches the specification; */
        ((qkey == pkey) && !first) ||
        /* 3. At least one component of the key we are */
        /* looking at is different from that of the key */
        /* we were given to start with. */
        (pkey->row != qkey->row) ||
        (pkey->col != qkey->col) ||
        (pkey->action != qkey->action)))
    {
        rkey = qkey;
        done = TRUE;
    }
    first = FALSE;
}
return ((rkey == NIL) ? skey : rkey);
}

```

## MNMOUSE 加入、修改或删除一个选单认可的鼠标器事件

#include <bmouse.h> /\*用于鼠标器符号。\*/

#include <bmenu.h>

BMENU \*mnmouse, BMENU \*pmenu,  
         unsigned long event,  
         unsigned long ignore,  
         int action,  
         int option);

pmenu 待修改的 BMENU 结构的地址。

event 指明待处理事件的位屏蔽。下面是有关的位:

符号	值	意义
MO_LEFT	0x0001	左鼠标器按钮。
MO_RIGHT	0x0002	右鼠标器按钮。
MO_MIDDLE	0x0004	中鼠标器按钮。
MO_PRESS	0x0008	按钮按下。
MO_RELEASE	0x0010	按钮释放。
MO_CLICK	0x0040	按钮快速按下并释放。
MO_DCLICK	0x0030	按钮两次快速按下并释放。
MO_HOLD	0x0800	按钮短暂地处于指定位置。
MO_RSHIFT	0x1000	右shift键按下。
MO_LSHIFT	0x2000	左shift键按下。
MO_CSHIFT	0x4000	Ctrl键按下。



MO_ASHIFT	0x0000	Alt键按下。
MN_ITEMS	0x0200	鼠标位于非保护性选项上。
MN_OFF_ITEMS	0x0400	鼠标在视口中但不在任何非保护性选项上。
MN_OUT_MENU	0x0100	鼠标在视口数据区之外。

ignore 指示需忽略的事件位的位屏蔽。下面是可设置的位值:

MO_LEFT	MO_RSHIFT	MN_ITEMS
MO_RIGHT	MO_LSHIFT	MN_OFF_ITEMS
MO_MIDDLE	MO_CSHIFT	MN_OUT_MENU
MO_ASHIFT		

action 对指定的鼠标器事件作出反应过程中所采取的动作(见“选单函数(MN)”的“按键动作和亮条移动”一节)。

option 应该如何修改被认可的事件表中的选项: MN\_ADD (0)、MN\_CHANGE(1) 或MN\_DELETE(2)。

(返回) 修改后的BMENU结构的地址。如果失败, 返回NIL。

在MNREAD或MNLREAD期间, 当由event和ignore指定的鼠标器事件发生时, MNMOUSE记录一个要采取的动作。

event中一些位组合具有匹配几个实际鼠标器事件的效用。例如,

(MO\_LEFT / MO\_RIGHT / MO\_PRESS / MO\_RELEASE)

探测两个按钮的按下或释放。

当下列情况之一发生时出现错误:

已指定MN\_ADD而对于这个事件/忽略的组合已经记录了一个动作;

已指定MN\_CHANGE或MN\_DELETE而对于这个事件/忽略的组合未记录任何动作;

鼠标器动作表已毁坏;

内存不够用。

源程序(MNMOUSE.C):

```
#include <stdlib.h>
```

```
#include <bmenu.h>
```

```
/* Internal function. */
```

```
static BMNMOUSE *cdecl mnmchmou(BMNMOUSE *,unsigned long,
                                unsigned long,int *);
```

```
BMENU *mnmouse(pmenu,event,ignore,action,option)
```

```
BMENU *pmenu;
```

```
unsigned long event,ignore;
```

```
int action,option;
```

```
{
```

```
    BMNMOUSE *pmouse;
```

```
    int error;
```

```
    mnavlidm(pmenu) /* Validate the menu data structure. */
```

```
    switch (option)
```

```
    {
```

```
        case MN_ADD:
```

```
            if (pmenu->pmouse == NIL)
```

```
            {
```

```
                if (NIL == (pmouse = utalloc(BMNMOUSE)))
```

```
                {
```

```
                    wnretterr(WN_NO_MEMORY);
```

```
                }
```

```
                pmouse->event = event;
```

```

        pmouse->ignore      = ignore;
        pmouse->action      = action;
        pmouse->signature   = MN_MOUSE_SIGN;
        pmenu->pmouse      = pmouse;
    }
    else
    {
        if (NIL != mnmchmou(pmenu->pmouse,
                            event,ignore,&error))
        {
            wnretterr(MN_MOUSE_CONFLICT);
        }
        else if (error != WN_NO_ERROR)
        {
            return NIL;          /* Error was recorded by      */
        }                      /* MNMCHMOU.            */

        if (NIL == (pmouse = utalloc(BMNM_MOUSE)))
        {
            wnretterr(WN_NO_MEMORY);

            pmouse->event      = event;
            pmouse->ignore     = ignore;
            pmouse->action     = action;
            pmouse->signature  = MN_MOUSE_SIGN;

            pmouse->pnext      = pmenu->pmouse;
            pmenu->pmouse->pprev = pmouse;
            pmenu->pmouse      = pmouse;
        }
        break;

    case MN_CHANGE:
        if (NIL == (pmouse = mnmchmou(pmenu->pmouse,
                                       event,ignore,&error)))
        {
            if (error == WN_NO_ERROR)
                wnerror(MN_NO_MOUSE_EVENT);
            return NIL;
        }

        pmouse->action = action;
        break;

    case MN_DELETE:
        if (NIL == (pmouse = mnmchmou(pmenu->pmouse,
                                       event,ignore,&error)))
        {
            if (error == WN_NO_ERROR)

```

```

        wnerror(MN_NO_MOUSE_EVENT);
        return NIL;
    }

    if (NIL != pmouse->pprev)
        pmouse->pprev->pnext = pmouse->pnext;
    if (NIL != pmouse->pnext)
        pmouse->pnext->pprev = pmouse->pprev;
    if (pmenu->pmouse == pmouse)
        pmenu->pmouse = pmouse->pnext;
    pmouse->signature = MN_DEAD_MOUSE;
    free(pmouse);
    break;

```

```

default:
    wnreterr(WN_ILL_VALUE);
}

```

```

return pmenu;
}

```

/\*\*

\*

\* Name            mnmchmou -- Search list of mouse events.

\*

\* Synopsis        presult = mnmchmou(pmouse,event,ignore,perror);

\*

\*                BMNMOUSE \*presult

\*

                 Address of the found mouse event  
                 record, or NIL for failure.

\*

\*                BMNMOUSE \*pmouse

\*

                 Address of initial mouse event  
                 record from which to begin search.

\*

\*                unsigned long event

\*

                 Bit mask designating event to handle.

\*

\*                unsigned long ignore

\*

                 Bit mask indicating event bits to  
                 ignore.

\*

\*                int \*perror        Address of variable in which to

\*

                 return resulting error.

\*

\* Description    This function searches a linked list of mouse event

\*

                 records for an entry matching a particular

\*

                 specification.    Both "event" and "ignore" must be

\*

                 matched exactly.

\*

\* Returns        presult            Pointer to mouse event record found, or

\*

                 NIL if not found or if list corrupted.

\*

\*                \*perror            WN\_NO\_ERROR if list is healthy,

\*

                 WN\_BAD\_MOUSE if event list corrupted,

\*

\*                b\_wnerr            WN\_BADMOUSE if event list corrupted.

```

*
**/

#include <bmenu.h>

static BMNMOUSE *mnmchmou(pmouse,event,ignore,perror)
BMNMOUSE *pmouse;
unsigned long event,ignore;
int *perror;
{
    for ( ;
        pmouse != NIL;
        pmouse = pmouse->pnext)
    {
        if (pmouse->signature != MN_MOU_SIGN)
        {
            wnreterr(MN_BAD_MOUSE);
        }
        if (pmouse->event == event && pmouse->ignore == ignore)
        {
            *perror = WN_NO_ERROR;
            return pmouse;
        }
    }
    *perror = WN_NO_ERROR;
    return NIL;
}

```

## MNMSTYLE 设立一个标准选单鼠标器格式

```

#include<bmouse.h> /* 用于按钮符号 */
#include <bmenu.h>
BMENU *mnmstyle(BMENU *pmenu,
                int style
                unsigned button);
pmenu 待修改的BMENU结构的地址。
style 下列标准形式之一：
      符号      值      意义
MN_MOU_CLICK  1  点按方式(一次点按选择，两次点按传送，界外点按取消)。
MN_MOU_DRAG   2  拖动方式(拖动选择，界外拖动取消亮条，释放传送)。
MN_MOU_ALT_DRAG 另一种拖动方式(拖动选择，释放传送)。
button 鼠标器按钮：MO_LEFT(1)、MO_RIGHT(2)或MO_MIDDLE(4)。
(返回) 修改后的BMENU结构的地址。若失败，返回NIL。

```

MNMSTYLE通过建立一个认可的鼠标器事件和对这些事件的反应的内部表来设立一个选单鼠标器的使用格式。

如果button未知或指定的按钮不存在，则不产生任何效果。

源程序(MNMSTYLE.C):

```

#include <bmenu.h>

```

```

#include <bmouse.h>
typedef struct
{
    unsigned long event,ignore;
    int          action;
} BMNMOUSE_EVENT;

static const BMNMOUSE_EVENT click[] =
{
    {
        MO_CLICK | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT
    },
    {
        MO_DCLICK | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT | MN_TRANSMIT
    },
    {
        MO_CLICK | MN_OUT_MENU,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_ABORT
    }
};

```

```

static const BMNMOUSE_EVENT drag[] =
{
    {
        MO_HOLD | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT | MN_KBIGNORE
    },
    {
        MO_HOLD | MN_OUT_MENU,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_HIDE_BAR | MN_KBIGNORE
    },
    {
        MO_RELEASE | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT | MN_TRANSMIT | MN_KBIGNORE
    },
    {
        MO_RELEASE | MN_OFF_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_TRANSMIT | MN_KBIGNORE
    },
    {
        MO_RELEASE | MN_OUT_MENU,

```

```

        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_ABORT | MN_KBIGNORE
    }
};

static const BMNMOUSE_EVENT alt_drag[] =
{
    {
        MO_HOLD | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT
    },
    {
        MO_RELEASE | MN_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_SELECT | MN_TRANSMIT
    },
    {
        MO_RELEASE | MN_OFF_ITEMS,
        MO_LSHIFT | MO_RSHIFT | MO_CSHIFT | MO_ASHIFT,
        MN_TRANSMIT
    }
};

BMENU *mnmmstyle(pmenu,style,button)
BMENU *pmenu;
int style;
unsigned button;
{
    const BMNMOUSE_EVENT *ptable;
    int num_events,i;
    mnvalidm(pmenu) /* Validate menu structure. */
    switch (button) /* Validate button specification*/
    {
        case MO_LEFT:
        case MO_RIGHT:
        case MO_MIDDLE:
            break;
        default:
            wnreterr(WN_BAD_OPT);
    }
    /* Choose a list of mouse events to implement the specified style.*/
    switch (style)
    {
        case MN_MOU_CLICK:
            ptable = click;
            num_events = sizeof(click) / sizeof(click[0]);
            break;
        case MN_MOU_DRAG:
            ptable = drag;

```

```

        num_events = sizeof(drag) / sizeof(drag[0]);
        break;
case MN_MOU_ALT_DRAG:
    ptable = alt_drag;
    num_events = sizeof(alt_drag) / sizeof(alt_drag[0]);
    break;
default:
    wnreterr(WN_BAD_OPT);
}
/* Install all of the events from the specified list. */
for (i = 0; i < num_events; i++)
{
    if (NIL == mnmouse(pmenu,ptable[i].event | button,
                      ptable[i].ignore,
                      ptable[i].action,
                      MN_ADD))

        return NIL;
}

return pmenu; /* Success. */
}

```

## MNREAD 读取来自选单的用户响应

#include<bmenu.h>

```

int mnread( BMENU pmenu,
            int srow, int scol,
            int *prow,int *pcol,
            int *pch,int *pkey,
            int option);

```

pmenu 待修改的BMENU结构的地址。  
srow scol 最初显亮的选项所在的行和列。  
prow,pcol 变量的地址，这些变量返回被选中选项的行和列。  
pch,pkey 变量的地址，这些变量返回最后按键的字符值和键码。  
option 下列值的零个或任意个组合：

符号	值	意义
MN_USE_DESCRIPTION	0x4000	显示被显亮选项的Lotus形式的说明(如果有的话)
MN_PREV_BAR	0x2000	显亮原先显亮的选项。
MN_SHOW_BAR	0x1000	如果使用了MN_PREV_BAR，迫使亮条显示，即使它原来未显示。
MN_HIDE_BAR	0x0800	最初不显亮任何选项，取代MN_SHOW_BAR。
MN_KEIGNORE	0x0200	忽略并废弃所有按键。
MN_UNKNOWN_BEEP	0x0010	当未分配键按下时鸣叫。
MN_UNKNOWN_TRANSMIT	0x0020	当未分配键按下时返回。
MN_ALL_TRANSMIT	0x0008	每次按键后返回。
MN_HOLD_BEEP	0x0080	返回时制止鸣叫，取代MN_UNKNOWN_BEEP。
MN_KEEP_HIGHLIGHT	0x0004	使最后的选项显亮。
MN_KEEP_DESCRIPTION	0x0040	不消除最后的说明字符串。
MN_REMOVE	0x0001	事后取消选单。

**MN\_DESTROY**                      0x0003      事后取消并废弃选单。

(返回)                      错误代码。若无错误，返回**WN\_NO\_ERROR(0)**。

**MNREAD**接受来自用户的按键，移动选单上的亮条，报告用户的选择及最后按下的键。

除非指定了**MN\_PREV\_BAR**，否则**srow**和**scol**指定最初显亮的选项。如果在(**srow,scol**)处没有可选择的选项，则**MNREAD**选择第一个选项，使之显亮。

窗口被更新。如果在调用**MNREAD**之前窗口被“延迟”，则退出时恢复这个状态。

通过指明**MN\_USE\_DESCRIPTION**可以对Lotus形式的选单使用**MNREAD**。

如果下列情况之一发生，则出现一个错误，返回一个非零值：

- 选单未显示；
- 选单窗口数据区某一部分被另一个窗口覆盖；
- 选单中无非保护选项；
- 指明了**MN\_SHOW\_BAR**而未发现可供选择的选项；
- 指明了**MN\_UNKUNOWN\_TRANSMIT**而一个未分配的键被按下；
- 发生了具有**MN\_ABORT**动作的按键或鼠标器操作；
- 指明了**MN\_REMOVE**或**MN\_DESTROY**而不能取消该选单；
- 指明了**MN\_DESTROY**而不能废弃该选单。

源程序(MNREAD.C):

```
#include <bkeybrd.h>
#include <bmenu.h>
#include <bmouse.h>

/* Most recent mouse menu event. */
unsigned long b_mnmoevent; /* Event, or OL if none. */
unsigned b_mnmovert; /* Row and column where event took */
unsigned b_mnmohoriz; /* place, or 0xffff if none. */
typedef struct /* CLEANUP_ITEMS Items to */
{ /* restore on exit. */
    int MOCATCH_was_off; /* Must remove MOCATCH. */
    unsigned delayed_flag; /* Window was previously delayed. */
} CLEANUP_ITEMS;
typedef struct /* USER_RESPONSE Mouse or */
{ /* keyboard event. */
    enum {MOUSE, KEYSTROKE, INTERNAL_ERROR} mouse_or_key;
    union
    {
        struct /* Details of mouse event: */
        {
            unsigned long event; /* Actual mouse event. */
            unsigned vert,horiz; /* Mouse cursor location */
            /* (in characters). */
            const BMNMOUSE *pmouse; /* Mouse event node. */
            BITEM *pitem; /* Menu item where mouse */
        } mouse; /* event took place. */
        struct /* Details of keystroke: */
        {
            KEY_SEQUENCE kseq; /* Character and key codes */
            const BKEYMAP *pkey; /* Key map entry. */
        } keystroke;
    } data;
}
```



```

} USER_RESPONSE;
#define TRUE 1
#define FALSE 0

/* Internal functions. */
static void clean_up(const BMENU *,const CLEANUP_ITEMS *);
static int mnchxkit(const BKEYMAP *,const BMNMOUSE *,int);
static int in_window(const BWINDOW *,unsigned,unsigned);
static BITEM *pfind_item(const BMENU *,unsigned,unsigned);
static const BMNMOUSE *pfind_mouse_event(const BMNMOUSE *,unsigned long,int *);
static int wait_operator(const BMENU *,USER_RESPONSE *,
                        const BKEYMAP *,int);
static int interp_resp(BMENU *,const USER_RESPONSE *,int *,BITEM **,
                      int *,int *);
static int jump(BWINDOW *,int);

int mnread (pmenu, srow, scol, prow, pcol, pch, pscan, option)
BMENU *pmenu;
int srow, scol;
int *prow, *pcol;
int *pch, *pscan;
int option;
{
    int code, /* Returned error code from MNFINDSL */
    done = FALSE, /* Becomes true when MNREAD should */
    /* exit at next opportunity. */
    result, /* Return code from interp_resp(). */
    shouldbeep = FALSE; /* Whether we should beep or not. */
    BITEM *pitem; /* Pointer to current menu item. */
    const BKEYMAP *pkey; /* Pointer to current key found. */
    USER_RESPONSE *resp;
    CLEANUP_ITEMS *cirt;

    /* Validate menu data structures. */
    if (mnvalmenu (pmenu) == NIL)
        wnreturn (MN_BAD_MENU);
    /* Exit if menu not displayed. */
    if (pmenu->pwin->where_shown.dev != SC_MONO
        && pmenu->pwin->where_shown.dev != SC_COLOR)
        wnreturn (WN_NOT_SHOWN);
    /* Exit if window is covered or frozen. */
    if (pmenu->pwin->internals.frozen)
        wnreturn (WN_COVERED);
    /* Disable keys which do not point either at items
    /* or at (-1,-1). */
    if (mndisabl (pmenu) == NIL)
        return (b_wnerr);
    /* Check to make sure there is an exit route. */
    if (mnchxkit(pmenu->pkeys,pmenu->pmouse,option))
        return (b_wnerr);
    /* Set pitem to highlight bar location (and set
    /* previous location if no bar shown). */

```

```

if (option & MN_PREV_BAR)
{
    pitem = pmenu->pbar_item;      /* Use former location      */
                                   /* or none if no bar shown. */

    if ( (pitem == NIL)
        && (option & MN_SHOW_BAR)) /* We require a bar,      */
        pitem = pmenu->pbar_prev; /* so try harder.        */
}
else
{
    /* If not using previous,      */
    option |= MN_SHOW_BAR;          /* then we must show a bar. */
    pitem = NIL;
}

if ( (pitem == NIL)
    && (option & MN_SHOW_BAR)) /* We still need a bar.      */
{
    /* Find the first selectable menu item.      */
    pitem = mnfindsl(pmenu,NIL,srow,scol,&code);
    if (pitem == NIL)
    {
        if ( code == WN_NO_ERROR) /* Error if none selectable */
            && !(option & MN_HIDE_BAR) /* unless none wanted. */
            wnreturn (MN_NONE_SELECT);
        else
            return (b_wnerr);      /* Internal error.          */
    }
}

if ( (option & MN_HIDE_BAR)
    && (pitem != NIL)) /* Must hide the bar and    */
/* there is a bar to hide. */
{
    pmenu->pbar_prev = pitem;      /* Save previous bar site.  */
    pitem = NIL;                  /* Show no bar.            */
}

/* Point pkey at menu's keylist.      */
pkey = pmenu->pkeys;
/* Save the starting row and column coordinates. */

if (pitem)
{
    *prow = pitem->row;
    *pcol = pitem->col;
}

/* Save items that may need restoration on exit.      */
dirt.MOCATCH was_off = !b_mocatch;
dirt.delayed_flag = pmenu->pwin->options.delayed;
/* Miscellaneous setup.      */
pmenu->pwin->options.delayed = 0; /* Allow immediate output. */
/* Check whether we should always transmit.      */

if (option & MN_ALL_TRANSMIT)
    done = TRUE;

```

```

do
{
    /* Show highlight bar and (perhaps) description.          */
    if (NIL == mnhighlight0(pmenu,pitem,
        (option & MN_USE_DESCRIPTION) | WN_UPDATE | MN_HIGHLIGHT))
    {
        clean_up(pmenu,&dirt);
        return b_wnerr;
    }
    /* Do a beep if the last keycode told us to do so....      */
    if (shouldbeep)
    {
        scattywrt ('\a', 0);
        shouldbeep = FALSE;
    }
    /* Get a keystroke or mouse event from the user.           */
    if (WN_NO_ERROR != wait_operator(pmenu,&resp,pkey,option))
    {
        clean_up(pmenu,&dirt);
        return b_wnerr;
    }
    switch (resp.mouse_or_key)
    {
        case KEYSTROKE:
            *pch = resp.data.keystroke.kseq.character_code;
            *pscan = resp.data.keystroke.kseq.key_code;
            if (resp.data.keystroke.pkey != NIL)
                pkey = resp.data.keystroke.pkey;
            b_mnmoevent = 0L;
            b_mnmovvert = 0xffff;
            b_mnmohoriz = 0xffff;
            break;
        case MOUSE:
            *pch = 0xff;
            *pscan = 0xff;
            b_mnmoevent = resp.data.mouse.event;
            b_mnmovvert = resp.data.mouse.vert;
            b_mnmohoriz = resp.data.mouse.horiz;
            break;
    }
    /* Interpret user response in terms of menu results.        */
    switch (result = interp_resp(pmenu,&resp,&option,&pitem,
        &shouldbeep,&done))
    {
        case WN_NO_ERROR:
        case MN_READ_AB:
        case MN_XMIT_NOBAR:
        case MN_UNKNOWN_AB:
            break;
        default:

```

```

        clean_up(pmenu,&dirt);
        return result;                                /* Internal error. */
    }
} while (!done);
clean_up(pmenu,&dirt);
    /* Show highlight bar and (perhaps) description. */
if (NIL == mnhighlight(pmenu,pitem,
    (option & MN_USE_DESCRIPTION) | WN_UPDATE | MN_HIGHLIGHT))
    return b_werr;
    /* Check whether we should beep. */
if (shouldbeep && ( !(option & MN_HOLD_BEEP)))
    scattywrt ('\a', 0);
    /* Perhaps remove highlight bar and description. */
if (NIL == mnhighlight(pmenu,pitem,
    WN_UPDATE
    | ((option & MN_KEEP_HIGHLIGHT) MN_HIGHLIGHT : MN_UNHIGHLIGHT)
    | ((option & MN_KEEP_DESCRIPTION) ? MN_USE_DESCRIPTION : 0)))
    return b_werr;
    /* Return the final row and column of the highlight */
    /* bar. */
if (pitem == NIL)
    pitem = pmenu->pbar_prev;
if (pitem == NIL)
{
    *pcol = -1;
    *prow = -1;
}
else
{
    *pcol = pitem->col;
    *prow = pitem->row;
}

    /* Remove menu's window from display if told to. */
if (option & MN_REMOVE)
    if (wnremove (pmenu->pwin) == NIL)
        return (b_werr);
    /* Destroy menu's data structures if told to do so. */
if ((option & MN_DESTROY) == MN_DESTROY)
    if (mndstroy (pmenu) != WN_NO_ERROR)
        return (b_werr);
return (result);
}

```

/\*\*

```

* Name      clean_up -- Restore miscellaneous aspects of
*            environment.
* Synopsis  clean_up(pmenu,pdirt);
*            const CLEANUP_ITEMS *pmenu
*            Address of the menu structure.

```

```

*      const CLEANUP_ITEMS *pdirt
*
*      Address of record structure listing
*      items to restore on exit from
*      MNREAD.
* Description    This function restores miscellaneous items:
*                specifically, if MOCATCH was not installed before MNREAD
*                was invoked, this function removes it; also restores the
*                menu's window's "delayed" state.
* Returns       b_mocatch    Restored to value before MNREAD was
*                          called.
**/

```

```

static void clean_up(pmenu,pdirt)
const BMENU      *pmenu;
const CLEANUP_ITEMS *pdirt;
{
    if (pdirt->MOCATCH_was_off)
        mopreclk(MO_REMOVE);
    pmenu->pwin->options.delayed = pdirt->delayed_flag;
}

```

```

/**
* Name          mnchkxitt -- Check whether there is any way to
*                exit the menu.
* Synopsis      ret = mnchkxitt(pkey,pmouse,option);
*                int      ret      Zero if there is an exit route (i.e. no
*                                error);
*                                MN_BAD_KEY if a bad keymap
*                                entry encountered;
*                                MN_BAD_MOUSE if a bad mouse
*                                entry encountered;
*                                MN_NO_EXIT if there is no exit key.
*                const BKEYMAP *pkey
*                                Pointer to the start of the menu's
*                                key list.
*                const BMNMOUSE *pmouse
*                                Address of first item in list of menu's
*                                mouse actions.
*                int option      MNREAD options including the following
*                                bits:
*                                MN_KBIGNORE
*                                MN_ALL_TRANSMIT
*                                MN_UNKNOWN_TRANSMIT
* Description    MNCHKXITT checks the option value and the menu's key and
*                mouse lists to see if there is any way for the menu to
*                be exited.
* Returns       ret            Zero if there is an exit route (i.e. no
*                                error);
*                                MN_BAD_KEY if a bad keymap
*                                entry encountered;

```

```

*
*           MN_BAD_MOUSE if a bad mouse
*           entry encountered;
*
*           MN_NO_EXIT if there is no exit key.
*
*           b_werr      Possible values:
*
*                       (No change)  Success.
*
*                       MN_NO_EXIT  No way to exit the
*                                   menu.
*
*                       MN_BAD_KEY  Bad keymap entry
*                                   encountered.
*
*                       MN_BAD_MOUSE Bad mouse entry
*                                   encountered.
*
**/

```

```

static int mnchkxit(pkey,pmouse,option)
const BKEYMAP*pkey;
const BMNMOUSE *pmouse;
int          option;
{
    if (option & (MN_ALL_TRANSMIT | MN_UNKNOWN_TRANSMIT))
        return 0;
    if (! (option & MN_KBIGIGNORE))
    {
        for (; pkey != NIL; pkey = pkey->next)
        {
            /* Check key signature. */
            if (pkey->signature != MN_KEY_SIGN)
                wreturn (MN_BAD_KEY);
            if ( !(pkey->action & (MN_DISABLE | MN_TEMP_DISABLE))
                && (pkey->action & (MN_TRANSMIT | MN_ABORT)))
                return (0);
        }
    }
    for (; pmouse != NIL; pmouse = pmouse->pnext)
    {
        /* Check mouse signature. */
        if (pmouse->signature != MN_MOU_SIGN)
            wreturn (MN_BAD_MOUSE);
        if ( !(pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
            && (pmouse->action & (MN_TRANSMIT | MN_ABORT)))
            return (0);
    }
    wreturn (MN_NO_EXIT);
}

```

```

/**
*
* Name          pfind_item -- Find menu item that covers a given
*                  screen location.
*
* Synopsis      pitem = pfind_item(pmenu,row,col)

```

```

*
*      BITEM *pitem    Address of resulting menu item node,
*                      or NIL if error or if none found.
*
*      const BMENU *pmenu
*                      Address of menu control structure.
*
*      int row,col     Display location (relative to (0,0) at
*                      upper left corner of display).
*
*
* Description      This function finds a menu item (if any) that covers a
*                  given display location.
*
*
*                  The result is meaningless if the menu is not displayed
*                  or (row,col) is outside the viewport.
*
*
* Returns          *pitem      Address of resulting menu item node,
*                              or NIL if error or if none found.
*
*                  b_wnerr     Unchanged if no error or if none found;
*                              MN_BAD_ITEM if corrupted item list.
*
**/

```

```

static BITEM *pfind_item(pmenu,row,col)
const BMENU *pmenu;
unsigned      row,col;
{
    BITEM *pitem;

    int target_row =      row
                        - pmenu->pwin->where_shown.corner.row
                        + pmenu->pwin->data_origin.row;
    int target_col =      col
                        - pmenu->pwin->where_shown.corner.col
                        + pmenu->pwin->data_origin.col;

    for (pitem = pmenu->pitems;
         pitem != NIL;
         pitem = pitem->next)
    {
        if (pitem->signature != MN_ITEM_SIGN)
        {
            wnreterr(MN_BAD_ITEM);
        }
        if ( pitem->row == target_row
            && pitem->col <= target_col
            && target_col < pitem->col + pitem->len)
        {
            return pitem;
        }
    }
    return NIL;
}

```

```

}

/**
 * Name          in_window -- Report whether window data area contains
 *                a given display location.
 * Synopsis      contains = in_window(pwin,row,col);
 *                int contains      Nonzero if window contains the
 *                                given location,
 *                                zero if not.
 *                const BWINDOW *pwin
 *                                Address of window control structure.
 *                int row,col      Display location (relative to (0,0) at
 *                                upper left corner of display).
 *
 * Description    This function tells whether a given display location
 *                lies within a window's data area.      The window must be
 *                displayed.
 * Returns        contains      Nonzero if window contains the
 *                                given location,
 *                                zero if not.
 */
static int in_window(pwin,row,col)
const BWINDOW *pwin;
unsigned      row,col;
{
    /* Macros to return edge of data area (plus 1). */
#define BOT_ROW_P1 (pwin->where_shown.corner.row + wnview_h(pwin))
#define RT_COL_P1 (pwin->where_shown.corner.col + wnview_w(pwin))
    return (    pwin->where_shown.corner.row <= row
                && row < BOT_ROW_P1
                && pwin->where_shown.corner.col <= col
                && col < RT_COL_P1);
#undef BOT_ROW_P1
#undef RT_COL_P1
}

/**
 * Name          pfind_mouse_event -- Find mouse event in list.
 * Synopsis      pfound = pfind_mouse_event(pmouse,event,perror)
 *                const BMNMOUSE *pfound
 *                                Returned address of found mouse event
 *                                record.
 *                const BMNMOUSE *pmouse
 *                                Address of first mouse node to check.
 *                unsigned long event
 *                                Actual mouse event to search for.
 *                int *perror      Address into which to return success
 *                                code.
 * Description    This function searches for a matching mouse event
 *                in a list of mouse events, taking into account

```



```

*          what aspects should be ignored.
* Returns   pfound      Address of found mouse event record,
*           or NIL if error or if none found.
*          *perror      WN_NO_ERROR or MN_BAD_MOUSE.
*
**/

```

```

static const BMNMOUSE *pfind_mouse_event(pmouse,event,perror)
const BMNMOUSE *pmouse;
unsigned long   event;
int             *perror;
{
    for ( ;
        pmouse != NIL;
        pmouse = pmouse->pnext)
    {
        if (pmouse->signature != MN_MOU_SIGN)
        {
            wnreterr(*perror = MN_BAD_MOUSE);
        }
        if ( 0 == (pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
            && (pmouse->ignore | event) == (pmouse->ignore | pmouse->event))
        {
            *perror = WN_NO_ERROR;
            return pmouse;
        }
    }
    *perror = WN_NO_ERROR;
    return NIL;
}

```

```

/**
* Name      wait_operator -- Wait for mouse or keyboard command
*           from operator.
* Synopsis  ercode = wait_operator(pmenu,presp,pkey,option);
*           int ercode      Error code:
*           WN_NO_ERROR if okay;
*           other if internal error.
*           const BMENU *pmenu
*           Address of menu structure.
*           USER_RESPONSE *presp
*           Address of structure containing the
*           returned user response.
*           const BKEYMAP *pkey
*           Address of keymap entry for most
*           recent keystroke.
*           int option      Option flag. Relevant bits:
*           MN_KBIGNORE
*           MN_KBDISCARD (both have same effect).
*

```

```

* Description      This function polls the mouse and keyboard for a response
*                  from the user.
*
*
*                  Mouse events are reported only if they are listed
*                  in the menu's list of mouse events.
*
* Returns          ercode          Error code:
*                  WN_NO_ERROR if okay;
*                  other if internal error.
*                  b_wnerr         Unchanged if no error;
*                  same as "ercode" if error detected.
*                  *presp          User response.
*
**/
static int wait_operator(pmenu,presp,pkey,option)
const BMENU      *pmenu;
USER_RESPONSE *presp;
const BKEYMAP    *pkey;
int              option;
{
    unsigned long event;
    BITEM        *pitem;
    unsigned      vert,horiz;
    int           ercode;
    const BMNMOUSE *pmouse;
    for (;;)      /* Infinite loop:      First poll mouse, then          */
    {              /* keyboard.  Exit when anything happens.            */
                  /* Poll mouse history separately for each              */
                  /* listed mouse event.                                */
        for (pmouse = pmenu->pmouse;
             pmouse != NIL;
             pmouse = pmouse->pnext)
        {
            if (pmouse->signature != MN_MOU_SIGN)
            {
                wnreturn(MN_BAD_MOUSE);
            }
            if ( 0 == (pmouse->action & (MN_DISABLE | MN_TEMP_DISABLE))
                && MO_OK == mocheck(pmouse->event,pmouse->ignore,
                                     MO_CLEAR,&event,&vert,&horiz)
                && event)
            {
                /* Analyze location of mouse event:      whether on a      */
                /* menu item and whether outside the menu. */
                vert >>= 3; /* Convert from pixels to characters. */
                horiz >>= 3;
                if (in_window(pmenu->pwin,vert,horiz))
                {
                    if (NIL == (pitem = pfind_item(pmenu,vert,horiz)))
                        event = (event | MN_OFF_ITEMS)

```

```

        & ~(unsigned long) (MN_ITEMS | MN_OUT_MENU);
    else
        event = (event | MN_ITEMS)
        & ~(unsigned long) (MN_OFF_ITEMS | MN_OUT_MENU);
    }
    else
        event = (event | MN_OUT_MENU)
        & ~(unsigned long) (MN_OFF_ITEMS | MN_ITEMS);

    /* Mouse event is now completely characterized, so */
    /* scan list of menu events for this menu.      Ignore */
    /* event if not found in list.                  */
    if (NIL != (presp->data.mouse.pmouse
                = pfind_mouse_event(pmenu->pmouse,event,&rcode)))
    {
        /* Found mouse event. */
        presp->mouse_or_key      = MOUSE;
        presp->data.mouse.event  = event;
        presp->data.mouse.vert   = vert;
        presp->data.mouse.horiz  = horiz;
        presp->data.mouse.pitem  = pitem;
        return rcode;
    }
}

/* No relevant mouse event, so poll keyboard. */
if (kbpoll(b_key_ctrl,pmenu,
           &presp->data.keystroke.kseq,
           KB_REMOVE_KEY))
if (0 == (option & (MN_KBIGNORE | MN_KBDISCARD)))
{
    presp->mouse_or_key      = KEYSTROKE;
    presp->data.keystroke.pkey =
        mmchkey(pmenu,pkey,
                presp->data.keystroke.kseq.character_code,
                presp->data.keystroke.kseq.key_code,
                &rcode);
    return rcode;
}
}

/**
 * Name      interp_resp -- Interpret user response and decide
 *           actions to take.
 * Synopsis  rcode = interp_resp(pmenu,presp,poption,ppitem,
 *                               pshouldbeep,pdone);
 *           int rcode      Error code:
 *                               WN_NO_ERROR if okay;
 *                               MN_READ_AB if user requested abort;

```

```

*           MN_XMIT_NOBAR if user requested transmit while
*           no bar shown;
*           MN_UNKNOWN_AB if user pressed an unknown key
*           the MN_UNKNOWN_TRANSMIT option in *poption
*           was set;
*           other if internal error.
*
* BMENU *pmenu Address of menu structure.
*
* const USER_RESPONSE *presp
*           Address of structure containing the
*           actual user response.
*
* int *poption Address of option flag.
*
* BITEM **ppitem Address of pointer to highlighted menu
*           item (*pitem == NIL if none).
*
* int *pshouldbeep
*           Address of flag indicating whether
*           a beep was requested.
*
* int *pdone Address of flag indicating whether
*           calling function should return without
*           further user input.
*
* Description This function interprets the user's response (in *presp)
*           and determines what action to take.
*
* Returns      ercode      Error code:
*
*           WN_NO_ERROR if okay;
*           MN_READ_AB if user requested abort;
*           MN_XMIT_NOBAR if user requested transmit while
*           no bar shown;
*           MN_UNKNOWN_AB if user pressed an unknown key
*           the MN_UNKNOWN_TRANSMIT option in *poption
*           was set;
*           other if internal error.
*
*
* b_wncrr      Unchanged if no error;
*           same as "ercode" if error detected.
*
* *poption      The MN_KB_DISCARD bit may be set or
*           cleared.
*
* *ppitem      Menu item to highlight (NIL if no
*           highlight bar).
*
* *pshouldbeep TRUE if beep requested;
*           unchanged if no beep requested.
*
* *pdone      TRUE if caller should return;
*           unchanged if further input okay.
*
**/

/* Macro to do a for loop which makes the variable */
/* qitem take on all of the pointer values from */
/* pitem to the end of the item list in the menu's */
/* item list.      ritem is initialized to be pitem. */

```

```

#define FORALL(pitem)                                \
    for (qitem = pmenu->pitems, ritem = (pitem);    \
         qitem != NIL;                              \
         qitem = qitem->next)                        \
    {                                                \
        if (qitem->signature != MN_ITEM_SIGN)        \
            wnreturn (MN_BAD_ITEM);                 \
    }

#define FOREND() }

/* Macro which maximizes a "rangefunc". This is */
/* used by MN_UP, DOWN, RIGHT, and LEFT to determine */
/* the correct place to move the highlight bar. */
/* Arguments: */
/*     matchrc -- Either "row" or "col"; the */
/*                component of the item's */
/*                position which is supposed to */
/*                match that of the other item. */
/*     diffrc -- Either "row" or "col"; the */
/*                opposite of "matchrc". */
/*     size -- Either height (if matchrc is */
/*              "col") or width (if matchrc is */
/*              "row") of menu's window data */
/*              area. */
/*     rangefunc- Either ULRANGE or DRRANGE. */

#define IFMATCH(matchrc, diffrc, size, rangefunc, pitem) \
    if ((!(qitem->option & MN_PROTECT)) &&              \
        (qitem->matchrc == (pitem)->matchrc) &&         \
        (rangefunc ((pitem), qitem, diffrc, size) > \
         rangefunc ((pitem), ritem, diffrc, size)))

#define DRRANGE(pitem, qitem, rowcol, size) \
    ((qitem->rowcol <= (pitem)->rowcol)      \
     ? ((pitem)->rowcol - qitem->rowcol) & wrap_mask \
     : ((size - qitem->rowcol) + (pitem)->rowcol))

#define ULRANGE(pitem, qitem, rowcol, size) \
    ((qitem->rowcol >= (pitem)->rowcol)      \
     ? (qitem->rowcol - (pitem)->rowcol) & wrap_mask \
     : ((size - (pitem)->rowcol) + qitem->rowcol))

/* BEYOND: Macro that maximizes a "beyondfunc": returns */
/* nonzero value if *qitem is a better candidate than *ritem. */
/* Use is similar to IFMATCH. */
/*     size - height or width of data area. */
/*     floor - preferred amount of motion (negative for upward */
/*            or leftward. */

#define BEYOND(matchrc,diffrc,size,beyondfunc,pitem,floor) \
    ( (!(qitem->option & MN_PROTECT))                      \
      && (qitem->matchrc == (pitem)->matchrc)                \
      && (beyondfunc((pitem),qitem,diffrc,size,floor) > \
          beyondfunc((pitem),ritem,diffrc,size,floor)))

```

```

#define DRBEYOND(pitem,qitem,rowcol,size,floor) \
    ( (qitem->rowcol >= (pitem)->rowcol + (floor)) \
      ? size - qitem->rowcol \
      : qitem->rowcol - size)

#define ULBEYOND(pitem,qitem,rowcol,size,ceiling) \
    ( (qitem->rowcol <= (pitem)->rowcol + (ceiling)) \
      ? qitem->rowcol \
      : -1 - qitem->rowcol)

/* Height of menu's data area. */
#define H (pmenu->pwin->img.dim.h)

/* Width of menu's data area. */
#define W (pmenu->pwin->img.dim.w)

static int interp_resp(pmenu,presp,poption,ppitem,
                      pshouldbeep,pdone)
BMENU          *pmenu;
const USER_RESPONSE *presp;
int             *poption;
BITEM          **ppitem;
int             *pshouldbeep,*pdone;
{
    int code;
    BITEM *pitem,*qitem,*ritem;
    int result = WN_NO_ERROR;
    int action;
    int dlsp;
    int wrap_mask;
    pitem = *ppitem;

    /* Do operations that are different for mouse and */
    /* keystroke events: in particular, get action code*/
    switch (presp->mouse_or_key)
    {
        case KEYSTROKE:
            if (presp->data.keystroke.pkey != NIL)
            {
                action = presp->data.keystroke.pkey->action;
                if (MNMOVE(action) == MN_SELECT)
                {
                    const BKEYMAP *pkey;
                    /* Move highlight bar to specified row/column of a */
                    /* menu item. */
                    pkey = presp->data.keystroke.pkey;
                    if ((qitem =
                        mnmatchm (pmenu, pitem, pkey->row,
                                pkey->col, 0, &code)) != NIL)
                        pitem = qitem;
                }
            }

```

```

        else if (code != WN_NO_ERROR)
        {
            *pdone = TRUE;
            return (b_wnerr);
        }
    }
}
else /* Unknown (or disabled) key pressed. */
{
    action = 0;

    /* Check whether we should beep when unknown key */
    /* pressed. */
    if (*poption & MN_UNKNOWN_BEEP)
        *pshouldbeep = TRUE;
    /* Check whether we should transmit when unknown */
    /* key pressed. */
    if (*poption & MN_UNKNOWN_TRANSMIT)
    {
        result = wnerror(MN_UNKNOWN_AB);
        *pdone = TRUE;
    }
}
break;

case MOUSE:
    action = presp->data.mouse.pmouse->action;
    if (MNMOVE(action) == MN_SELECT)
        pitem = presp->data.mouse.pitem;
    break;
}

switch (MNMOVE(action))
{
    case MN_UP:
    case MN_DOWN:
    case MN_RIGHT:
    case MN_LEFT:
    case MN_NEXT:
    case MN_PREVIOUS:
    case MN_PGDN:
    case MN_PGUP:
    case MN_PGRIGHT:
    case MN_PGLEFT:
        /* Handle relative movements specially: if bar */
        /* not shown, use previously highlighted item. */
        /* This has the effect of showing the bar unless */
        /* MN_HIDE_BAR is also specified in the action code.*/
        if (pitem == NIL)
            pitem = pmenu->pbar_prev;

```

```

if (pitem != NIL)
{
    /* wrap_mask is used by the DRRANGE and ULRANGE */
    /* macros for the MN_UP, MN_DOWN, MN_RIGHT, and */
    /* MN_LEFT motions. When MN_NOWRAP is specified, */
    /* then wrap_mask is 0, thus causing DRRANGE and */
    /* ULRANGE to return 0 for coordinates in the */
    /* opposite direction. */
    wrap_mask = ((action & MN_NOWRAP) ? 0 : ~0);

    switch (MNMOVE(action))
    {
        /* Move highlight bar up.  Wrap to bottom if at top.*/
        case MN_UP:
            FORALL(pitem)
                IFMATCH (col, row, H, ULRANGE, pitem)
                    ritem = qitem;
            FOREND ()
            pitem = ritem;
            break;
        case MN_DOWN:
            /* Move highlight bar down.  Wrap to top if at */
            /* bottom. */
            FORALL(pitem)
                IFMATCH (col, row, H, DRRANGE, pitem)
                    ritem = qitem;
            FOREND ()
            pitem = ritem;
            break;
        case MN_RIGHT:
            /* Move highlight bar right.  Wrap to left if at */
            /* rightmost item. */
            FORALL(pitem)
                IFMATCH (row, col, W, DRRANGE, pitem)
                    ritem = qitem;
            FOREND ()

            pitem = ritem;
            break;

        case MN_LEFT:
            /* Move highlight bar left.  Wrap to right if at */
            /* leftmost item. */
            FORALL(pitem)
                IFMATCH (row, col, W, ULRANGE, pitem)
                    ritem = qitem;
            FOREND ()

            pitem = ritem;
            break;
    }
}

```



```

    case MN_NEXT:
/* Move highlight bar to "logical-next" menu      */
/* position. This is the next item in the order    */
/* that items were entered by MNITEM and/or        */
/* MNITEMKEYWrap to beginning if at end unless     */
/* MN_NOWRAP specified.                            */
    if (action & MN_NOWRAP)
    {
        for (qitem = pitem->next;
             qitem != NIL;
             qitem = qitem->next)
        {
            if (!(qitem->option & MN_PROTECT))
            {
                pitem = qitem;
                break;
            }
        }
    }
    else
    {
        /* MNMCHITM automatically wraps. */
        pitem = mnmmchitm (pmenu, (*ppitem)->next,
                           -1, -1, 0, &code);
        if (code != WN_NO_ERROR)
        {
            *pdone = TRUE;
            return (b_wnerr);
        }
    }
    break;

```

```

    case MN_PREVIOUS:
/* Move highlight bar to "logical-previous" menu    */
/* position. This is the previous item in the      */
/* order that items were entered by MNITEM and/or  */
/* MNITEMKEYWrap to end if at beginning.          */
    if (pitem == mnmmchitm (pmenu, pmenu->pitems,
                           -1, -1, 0, &code))
    {
        /* pitem is first in list. */
        if (action & MN_NOWRAP)
            ritem = pitem;
        else
        {
            /* Find last in list. */
            FORALL(pitem)
                if (!(qitem->option & MN_PROTECT))
                    ritem = qitem;
            FOREND ()
        }
    }
}

```

```

else
{
    if (code != WN_NO_ERROR)
    {
        *pdone = TRUE;
        return (b_werr);
    }

    for (ritem = pitem, qitem = pmenu->pitem;
        qitem != pitem;
        qitem = qitem->next)

        if ( !(qitem->option & MN_PROTECT))
            ritem = qitem;
    }
    pitem = ritem;
    break;

case MN_PGDN:
    disp = jump(pmenu->pwin,MN_PGDN);
    FORALL(pitem)
        if (BEYOND(col,row,H,DRBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;

case MN_PGUP:
    disp = jump(pmenu->pwin,MN_PGUP);
    FORALL(pitem)
        if (BEYOND(col,row,H,ULBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;

case MN_PGRIGHT:
    disp = jump(pmenu->pwin,MN_PGRIGHT);
    FORALL(pitem)
        if (BEYOND(row,col,W,DRBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()
    pitem = ritem;
    break;

case MN_PGLEFT:
    disp = jump(pmenu->pwin,MN_PGLEFT);
    FORALL(pitem)
        if (BEYOND(row,col,W,ULBEYOND,pitem,disp))
            ritem = qitem;
    FOREND()

```

```

        pitem = ritem;
        break;
    }
}
break;

case MN_FIRST:
    /* Move highlight bar to first item entered by */
    /* MNITEM and/or MNITMKEY. */
    pitem = mnmcHitm (pmenu, pmenu->pitema, -1, -1, 0, &code);
    if (code != WN_NO_ERROR)
    {
        *pdone = TRUE;
        return (b_werr);
    }
    break;

case MN_LAST:
    /* Move highlight bar to last item entered by */
    /* MNITEM and/or MNITMKEY. */
    FORALL(pitem)
        if ( ! (qitem->option & MN_PROTECT))
            ritem = qitem;
    FOREND ()

    pitem = ritem;
    break;
}
if (action & MN_SHOW_BAR)
{
    if (pitem == NIL) /* If no menu item lit, */
        pitem = pmenu->pbar_prev; /* find an item to highlight */
}
else if (action & MN_HIDE_BAR)
{
    if (pitem != NIL) /* If an item is lit, */
    {
        pmenu->pbar_prev = pitem; /* save location */
        pitem = NIL; /* and unlight the bar. */
    }
}

if (action & MN_BEEP)
    *pshouldbeep = TRUE; /* Forward beep request. */

if (action & MN_KBIGNORE)
    *poption |= MN_KBDISCARD; /* Ignore keystrokes. */
else
    *poption &= ~MN_KBDISCARD; /* Begin watching keys. */

```

```

if (action & MN_ABORT)
{
    /* User requested abort. */
    result = wncerror(MN_READ_AB);
    *pdone = TRUE;
}
else if (action & MN_TRANSMIT)
{
    /* User requested transmit. */
    if (pitem == NIL)
        result = wncerror(MN_XMIT_NOBAR); /* Error if transmit */
        /* while no item lit.*/
    *pdone = TRUE;
}

*ppitem = pitem;
return result;
}

/**
 * Name      jump - Adjust window origin for MN_PGUP, MN_PGDN,
 *             MN_PGRIGHT, or MN_PGLEFT motion.
 * Synopsis  displacement = jump(pwin,dir);
 *             int displacement Preferred amount of motion by
 *             highlight bar (negative for
 *             upward or toward left edge).
 *             BWINDOW *pwin Address of menu's window.
 *             int dir      MN_PGUP, MN_PGDN, MN_PGRIGHT, or
 *             MN_PGLEFT.
 *
 * Description This function moves the window's origin and
 *             computes the distance the highlight bar should move.
 *
 * Returns     displacement Preferred amount of motion by
 *             highlight bar (negative for
 *             upward or toward left edge).
 *             b_wncerr      Window error if any.
 */
static int jump(pwin,dir)
BWINDOW *pwin;
int dir;
{
    int new_origin_row,old_origin_row,new_origin_col,old_origin_col;
    int displacement;
    new_origin_row = old_origin_row = pwin->data_origin.row;
    new_origin_col = old_origin_col = pwin->data_origin.col;
    switch (dir)
    {
        case MN_PGDN:
            new_origin_row = utmin(old_origin_row + wnview_h(pwin) - 1,
                                   wndata_h(pwin) - wnview_h(pwin));

```

```

        if (new_origin_row == old_origin_row)
            displacement = wvview_h(pwin) - 1;
        else
            displacement = new_origin_row - old_origin_row;
        break;

case MN_PGUP:
    new_origin_row = utmax(old_origin_row - (wvview_h(pwin) - 1),
                           0);
    if (new_origin_row == old_origin_row)
        displacement = 1 - wvview_h(pwin);
    else
        displacement = new_origin_row - old_origin_row;
    break;

case MN_PGRIGHT:
    new_origin_col = utmin(old_origin_col + wvview_w(pwin) - 1,
                           wdata_w(pwin) - wvview_w(pwin));
    if (new_origin_col == old_origin_col)
        displacement = wvview_w(pwin) - 1;
    else
        displacement = new_origin_col - old_origin_col;
    break;

case MN_PGLEFT:
    new_origin_col = utmax(old_origin_col - (wvview_w(pwin) - 1),
                           0);
    if (new_origin_col == old_origin_col)
        displacement = 1 - wvview_w(pwin);
    else
        displacement = new_origin_col - old_origin_col;
    break;
}

wnorigin(pwin,new_origin_row,new_origin_col,WN_NO_UPDATE);
return displacement;
}

```

## MNVALMNO 检查BMENU结构的有效性。

presult = mnvalmno(pmenu,signature);

presult           指向有效的BMENU结构，如果失败则指向NIL。

pmenu            指向要检查的BMENU结构的指针。

signature        调用函数要求的标签字。

函数由mnvalnu()宏调用，后者用来提供标签值。相同的标签值提供给MNCREATO。通过这些宏，所有的菜单函数都提供比较的标签字。用这种方法在运行时可检测到BEMNU.H的不兼容版本。

返回            s    presult           指向有效的BMENU结构，如果失败则指向NIL。

源程序(MNVALMNO.C)

```
#include <bmenu.h>
```

```

BMENU *mnvalidm0(pmenu,signature)
BMENU *pmenu;
unsigned signature;
{
    if (pmenu == NIL || signature != pmenu->signature)
        return (NIL);

    return pmenu;
}

```

## MNVDISP 在视口中显示一个虚拟选单

```

#include <bmenu.h>
#include <bmenu.h>
BMENU *mnvdisp(BMENU *pmenu,
               const WHERE *pwhere,
               int view_ht,int view_wid,
               int org_row,int org_col,
               const BORDER *pbord);
pmenu      待显示的BMENU结构的地址。
pwhere WHERE结构的地址, 该结构指明设备、显示页和选单显示位置的坐标。
view ht,view wid 视口中行和列的数量。
org_row,org_col 将出现在视口左上角的选单数据区中的行和列(相对于(0,0))。
pbord BORDER结构的地址, 该结构指明放置在视口四周的边界和标题。
(返回) 显示后BMENU结构的地址。若失败, 返回NIL。

```

MNVDISP在视口中显示一个虚拟选单并加一个边界。(border可以是“无边界”类型或包含上/下标题。)视口不必与选单数据区的尺寸相匹配。选单窗口变成当前窗口(用于I/O), 那一页上的其它窗口的光标被冻结。如果程序与NW\_???.OBJ相链接, 则Turbo C字符窗口被设置成与视口相匹配。

欲指明视口的位置和它的边界, 需要按WINDSPY中说明的那样建立\*pwhere和\*pbord结构。(除了MNDSPY使用与选单尺寸相同的视口这一点之外, MNDSPY与MNVDISP相同。)

如果视口的数据区与屏幕边沿相邻接(也就是说没有地方用于边界), 则不画出边界。

org\_row和/或org\_col将被调整使行视口中显示的区域不超出选单的数据区。

如果pmenu未指向一个合法的选单结构, 视口超出选单数据区或超出屏幕的尺寸, 这时出现一个错误; 如果选单的显示位置不合理, 例如请求了一个未知设备, 这时出现错误。

源程序(MNVDISP.C):

```

#include <bmenu.h>
BMENU*mnvdisp(pmenu,pwhere,view h,view w,origin_row,origin_col,pborder)
BMENU      *pmenu;
const WHERE *pwhere;
int         view h,view_w,origin_row,origin_col;
const BORDER *pborder;
{
    /* Validate menu data structure. */
    mnvalidm (pmenu);

    /* Display the menu. */
}

```

```

return ((wrvdisp (pmenu->pwin, pwhere, view_h, view_w,
                origin_row, origin_col, pborder) == NIL)
        ? NIL
        : pmenu);
)

```

## 示例程序

MENU.C 用来显示几个键盘激活类型的菜单。

```

/**
 * This program displays a series of menus on the screen, allowing
 * the user to choose which of several types of menus will be
 * displayed.
 * Five menus are used. The main menu allows the user to select
 * one of four other menus or to terminate the program. The
 * main menu is an example of a vertical menu style; the other
 * four exhibit horizontal, grid, Lotus-style, and virtual menus,
 * respectively.
 * The command line format is as follows:
 * menu [/c | /d | /a]
 * The user may specify either /c, /d, or /a, corresponding to the
 * type of mouse style they wish to use; click, drag, or alternate
 * drag, respectively. If no switch is specified, the default is
 * click.
 * The two purposes of MENU are to show off the menuing
 * capabilities of Turbo C TOOLS, and to provide a working example
 * of the proper method of construction and use of the menu
 * functions.
 */
#include <ctype.h>
#include <stdio.h>
#include <bkeybrd.h>
#include <bkeys.h>
#include <bmenu.h>
#include <buttl.h>
#include <bmouse.h>

/* Text color is intense white on blue. */
#define MYTEXTATR (utnybbyt (SC_BLUE, NORMAL | INTENSITY))
/* "Lotus" description color is intense magenta on */
/* blue. */
#define MYLNGATTR (utnybbyt (SC_BLUE, SC_MAGENTA | INTENSITY))
/* Color of highlight bar is black text on a white */
/* background. */
#define MYHILATR (utnybbyt (NORMAL, SC_BLACK))
/* Color of "protected" items is green on blue. */
#define MYPROATR (utnybbyt (SC_BLUE, SC_GREEN))

```

```

        /* Color of menu borders is cyan on black. */
#define MYBORDATR (utnybbyt (SC_BLACK, SC_CYAN))
        /* Color of menu titles is the same as menu borders.*/
#define MYTTATR MYBORDATR
#define NUL '\0'
#define TRUE 1
#define FALSE 0
        /* Terminate-on-error macro. */
#define exitBad()
{
    mobide(MO_HIDE);
    fprintf (stderr,
        "menu: died with b_werr = %d in line %d",
        b_werr, __LINE__);
    exit (b_werr);
}

        /* Declare function prototypes. */
void main (int, char **);
void horizontal (int, int, int, int, int *, int *);
void lotus (int, int, int, int, int *, int *);
void grid (int, int, int, int, int *, int *);
void virtual (int, int, int, int, int *, int *);
void main (argc, argv)
int argc;
char **argv;
{
    BMENU *pmenu;
    BORDER border;
    WHERE where;
    int ch, scan;
    int row, col;
    int rrow, rcol;
    int showrow, showcol;
    int adapter, mode, cols, apage;
    int coff, crow, ccol, chigh, clow;
    int done = FALSE;
    int mouse_style = MN_MOU_CLICK;
    int bad_command_line = 0;
        /* First, check to see if the user specified a
        /* mouse style.
if (argc == 2)
{
        /* Make sure the switch character is valid.
if ((argv[1][0] == '/') || (argv[1][0] == '-'))
{

```



```

        /* Make sure the option character is valid. */
switch(toupper(argv[1][1]))
{
case 'C':
    mouse_style = MN_MOU_CLICK;
    break;
case 'D':
    mouse_style = MN_MOU_DRAG;
    break;
case 'A':
    mouse_style = MN_MOU_ALT_DRAG;
    break;
default:
    bad_command_line = 1;
    break;
}
}
else
    bad_command_line = 1;
}
else
    if (argc > 2)
        bad_command_line = 1;
if (bad_command_line)
{
    fprintf(stderr, "Usage: menu [/c | /d | /a].\n");
    exit(1);
}

    /* Now we make certain that we are in 80 column */
    /* text video mode. */
adapter = scmode (&mode, &cols, &apage);
switch (mode)
{
case 2:
case 3:
case 7:
    break;
default:
    fprintf (stderr,
        "menu: This demonstration works only in 80 column\n");
    fprintf (stderr,
        "    text modes (modes 2, 3, and 7)\n");
    exit (1);
}

    /* Save cursor position and style to restore later. */

```

```

coff = scurst (&crow, &ccol, &chigh, &clow);
        /* Create the menu data structure. */
if ((pmenu = mncreate (8, 14,
        MYTEXTATR, MYHILATR,
        MYPROATR, MYLNGATTR)) == NIL)
    exitBad ()
        /* Set up items on the menu, and define */
        /* corresponding keys (upper and lower case of the */
        /* first letters of the items). */
if (mnitmkey (pmenu, 0, 1, 0, "Horizontal", "Hh", MN_NOMOVE) == NIL)
    exitBad ()
if (mnitmkey (pmenu, 1, 1, 0, "Lotus-Style", "Ll", MN_NOMOVE) == NIL)
    exitBad ()
if (mnitmkey (pmenu, 2, 1, 0, "Grid", "Gg", MN_NOMOVE) == NIL)
    exitBad ()
if (mnitmkey (pmenu, 3, 1, 0, "Virtual", "Vv", MN_NOMOVE) == NIL)
    exitBad ()
if (mnitmkey (pmenu, 5, 1, 0, "Quit", "QqXx", MN_NOMOVE | MN_BEEP)
    == NIL)
    exitBad ()
        /* Define the following keys as selection & */
        /* transmission keys: ALT-H, ALT-L, ALT-G, ALT-V, */
        /* ALT-Q and X. */
if (mnkey (pmenu, 0, 1, KB_C_A_H, KB_S_A_H,
    MN_SELECT | MN_TRANSMIT, MN_ADD) == NIL)
    exitBad ()
if (mnkey (pmenu, 1, 1, KB_C_A_L, KB_S_A_L,
    MN_SELECT | MN_TRANSMIT, MN_ADD) == NIL)
    exitBad ()
if (mnkey (pmenu, 2, 1, KB_C_A_G, KB_S_A_G,
    MN_SELECT | MN_TRANSMIT, MN_ADD) == NIL)
    exitBad ()
if (mnkey (pmenu, 3, 1, KB_C_A_V, KB_S_A_V,
    MN_SELECT | MN_TRANSMIT, MN_ADD) == NIL)
    exitBad ()
if (mnkey (pmenu, 5, 1, KB_C_A_Q, KB_S_A_Q,
    MN_SELECT | MN_TRANSMIT | MN_BEEP, MN_ADD) == NIL)
    exitBad ()
if (mnkey (pmenu, 5, 1, KB_C_A_X, KB_S_A_X,
    MN_SELECT | MN_TRANSMIT | MN_BEEP, MN_ADD) == NIL)
    exitBad ()
        /* Disable ESC key. */
if (mnkey (pmenu, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
    MN_ABORT, MN_DELETE) == NIL)
    exitBad ()

```

```

        /* Now that we have put all of the menu items on */
        /* the menu in item color, set the native window */
        /* color so that additional text will appear in the */
        /* menu's window in "long-item" (description) color.*/
wnsetopt (pmenu->pwin, WN_ATTR, MYLNGATTR);
        /* Figure out where to display the menu. */
where.dev      = (adapter == 0) ? SC_MONO : SC_COLOR;
where.page     = 0;
where.corner.row = 1;
where.corner.col = 1;
        /* Make a border with a top centered title. */
border.type    = BBRD_SSSS | BBRD_TCT;
border.attr    = MYBORDATTR;
border.ch      = NUL;
border.pttitle = "Menu Styles";
border.ttattr  = MYTITATTR;
        /* Display the menu on the screen. */
if (mndisplay (pmenu, &where, &border) == NIL)
    exitBad ()
        /* Set up starting row and column for highlight bar.*/
row = 0;
col = 1;
        /* If the mouse is present, enable its cursor. */
if (MO_OK == mhide(MO_SHOW))
    if (NULL == mnmstyle(pmenu, mouse_style, MO_LEFT))
        exitBad ()
            /* Leave this menu on the screen until they select */
            /* the "Quit" entry. */
do
{
    /* Read a response from the menu. */
    if (mnread (pmenu, row, col, &row, &col, &ch, &scan,
        MN_KEEP_HIGHLIGHT))
        exitBad ()
            /* Set up location to show sub-menu. */
showrow = where.corner.row + row + 2;
showcol = where.corner.col + col + 0;
        /* Go do what was requested. */
switch (row)
{
    case 0:
        horizontal (where.dev, showrow, showcol, mouse_style,
            &row, &col);
        break;
    case 1:

```

```

        lotus (where.dev, showrow, showcol, mouse_style,
               &rrow, &rcol);
        break;
    case 2:
        grid (where.dev, showrow, showcol, mouse_style,
              &rrow, &rcol);
        break;
    case 3:
        virtual (where.dev, showrow, showcol, mouse_style,
                 &rrow, &rcol);
        break;
    case 5:
        done = TRUE;
}

/* At this point, rrow and rcol contain the row and */
/* column of the item selected from the submenu.    */
if (!done)
{
    /* Tell the user what was selected from the submenu.*/
    if (b_pcurwin != pmenu->pwin)
        wnselect (pmenu->pwin);
    wnsrblk (b_pcurwin, 6, 0, 6, 13, -1, -1, 0, 0, 0);
    wncurmov (6, 1);
    wnprintf ("xmit (%d %d)", rrow, rcol);
}
} while (!done);
mndstroy (pmenu);
mohide(MO_HIDE);
/* Restore cursor position and style.                */
securset (crow, ccol);
scpgcur (coff, chigh, clow, CUR_NO_ADJUST);
}
/**
 *
 * Name          HORIZONTAL -- Display and allow user selection from
 *                  a simple horizontal menu.
 *
 * Synopsis      horizontal (dev, row, col, prrow, prcol);
 *
 *               int dev          Device on which to display the
 *                               menu.  Either SC_COLOR or SC_MONO.
 *               int row, col      Row and column where the upper-
 *                               left corner of the menu's data
 *                               area should appear.
 *               int mouse_style   The mouse style to use.

```

```

*          int *prrow,      Pointers to variables in which to
*          *prcol          return the row and column selected
*                          from the menu.
*
* Description    This function constructs a simple horizontal menu,
*                and waits for user input. It then returns the row
*                and column of the selection to its caller.
*
* Returns       *prrow, *prcol  Row and column (relative to menu) of
*                                user selection.
**/

```

```

void horizontal (dev, row, col, mouse_style, prrow, prcol)
int dev;
int row, col;
int mouse_style;
int *prrow, *prcol;
{
    BMENU *pmenu;
    BORDER border;
    WHERE where;
    int ch, scan;

    /* Figure out where to display the menu. */
    where.dev = dev;
    where.page = 0;
    where.corner.row = row;
    where.corner.col = col;

    /* Create the menu data structure. */
    if ((pmenu = mncreate (1, 28,
                          MYTEXTATR, MYHILATR,
                          MYPROATR, MYLNGATTR)) == NIL)
        exitBad ()

    /* Set up items on the menu, and add keys to the key*/
    /* binding list (upper and lower case of the first */
    /* letters of the items. */
    if (mnitmkey (pmenu, 0, 1, 0, "My dog", "MmDd", MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 0, 9, 0, "has", "Hh", MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 0, 14, 0, "itchy", "I", MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 0, 21, 0, "fleas.", "Ff", MN_NOMOVE) == NIL)
        exitBad ()
}

```

```

        /* Disable ESC key. */
if (mnkey (pmenu, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
        MN_ABORT, MN_DELETE) == NIL)
    exitBad ()

        /* Make a border with a bottom centered title. */
border.type      = BBRD_SSSS | BBRD_BCT;
border.attr      = MYBORDATR;
border.ch        = NUL;
border.pbtitle   = "Horizontal Menu";
border.btattr    = MYTITATR;

        /* Display the menu on the screen. */
if (mndisplay (pmenu, &where, &border) == NIL)
    exitBad ()

        /* Make menu aware of the mouse. */
if (NULL == mnmstyle(pmenu, mouse_style, MO_LEFT))
    exitBad ()

        /* Read a response from the menu. */
if (mread (pmenu, 0, 1, prrow, prcol, &ch, &scan, MN_DESTROY))
    exitBad ()
}
/**
 * Name          LOTUS -- Display and allow user selection from a
 *                "Lotus"-style menu.
 * Synopsis      lotus (dev, row, col, prrow, prcol);
 *                int dev          Device on which to display the
 *                                menu. Either SC_COLOR or SC_MONO.
 *                int row, col     Row and column where the upper-
 *                                left corner of the menu's data
 *                                area should appear.
 *                int mouse_style  The mouse style to use.
 *                int *prrow,      Pointers to variables in which to
 *                *prcol           return the row and column selected
 *                                from the menu.
 * Description    This function constructs a "Lotus" menu, and
 *                waits for user input. It then returns the row
 *                and column of the selection to its caller.
 * Returns        *prrow, *prcol   Row and column (relative to menu) of
 *                                user selection.
 */
void lotus (dev, row, col, mouse_style, prrow, prcol)

```

```

int dev;
int row, col;
int mouse_style;
int *prrow, *prcol;
{
    BMENU *pmenu;
    BORDER border;
    WHERE where;
    int ch, scan;

    /* Figure out where to display the menu. */
    where.dev = dev;
    where.page = 0;
    where.corner.row = row;
    where.corner.col = col;

    /* Create the menu data structure. */
    if ((pmenu = mncreate (2, 70,
                          MYTEXTATR, MYHILATR,
                          MYPROATR, MYLNGATTR)) == NIL)
        exitBad ()
        /* Set up items on the menu, and add keys to the key*/
        /* binding list (upper and lower case of the first */
        /* letters of the items. */
        if (mnlitkey (pmenu, 0, 0, 0, "Worksheet", 1, 0,
                    "Global, Insert, Delete, Column-Width, Erase, Titles, Window, Status",
                    "Ww", MN_NOMOVE | MN_TRANSMIT) == NIL)
            exitBad ()
        if (mnlitkey (pmenu, 0, 11, 0, "Range", 1, 0,
                    "Format, Label-Prefix, Erase, Name, Justify, Protect, Unprotect, Input",
                    "Rr", MN_NOMOVE | MN_TRANSMIT) == NIL)
            exitBad ()
        if (mnlitkey (pmenu, 0, 18, 0, "Copy", 1, 0,
                    "Copy a cell or range of cells",
                    "Cc", MN_NOMOVE | MN_TRANSMIT) == NIL)
            exitBad ()
        if (mnlitkey (pmenu, 0, 24, 0, "Move", 1, 0,
                    "Move a cell or range of cells",
                    "Mm", MN_NOMOVE | MN_TRANSMIT) == NIL)
            exitBad ()
        if (mnlitkey (pmenu, 0, 30, 0, "File", 1, 0,
                    "Retrieve, Save, Combine, Xtract, Erase, List, Import, Directory",
                    "Ff", MN_NOMOVE | MN_TRANSMIT) == NIL)
            exitBad ()
        if (mnlitkey (pmenu, 0, 36, 0, "Print", 1, 0,
                    "Output a range to the printer or a print file",
                    "Pp", MN_NOMOVE | MN_TRANSMIT) == NIL)

```

```

        exitBad ()
    if (mnlitkey (pmenu, 0, 43, 0, "Graph", 1, 0,
        "Create a graph",
        "Gg", MN_NOMOVE | MN_TRANSMIT) == NIL)
        exitBad ()
    if (mnlitkey (pmenu, 0, 50, 0, "Data", 1, 0,
        "Fill, Table, Sort, Query, Distribution",
        "Dd", MN_NOMOVE | MN_TRANSMIT) == NIL)
        exitBad ()
    if (mnlitkey (pmenu, 0, 56, 0, "Quit", 1, 0,
        "End 3-2-1 Session (Have you saved your play ?)",
        "Qq", MN_NOMOVE | MN_TRANSMIT) == NIL)
        exitBad ()
        /* Disable ESC key. */
    if (mkey (pmenu, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
        MN_ABORT, MN_DELETE) == NIL)
        exitBad ()
        /* Make a border with a top centered title. */
    border.type = BBRD_SDDD | BBRD_TCT;
    border.attr = MYBORDATR;
    border.ch = NUL;
    border.pttitle = "Lotus Menu (with top-centered title)";
    border.ttattr = MYTITATR;
        /* Display the menu on the screen. */
    if (mndsplay (pmenu, &where, &border) == NIL)
        exitBad ()
        /* Make menu aware of the mouse. */
    if (NULL == mpmstyle(pmenu, mouse_style, MO_LEFT))
        exitBad ()
        /* Read a response from the menu. */
    if (mnlread (pmenu, 0, 0, prrow, prcol, &ch, &scan, MN_DESTROY))
        exitBad ()
}
/**
 * Name          GRID -- Display and allow user selection from a
 *                grid menu.
 * Synopsis      grid (dev, row, col, prrow, prcol);
 *                int dev          Device on which to display the
 *                                menu. Either SC_COLOR or SC_MONO.
 *                int row, col     Row and column where the upper-
 *                                left corner of the menu's data
 *                                area should appear.
 *                int mouse_style  The mouse style to use.
 *                int *prrow,      Pointers to variables in which to
 *                *prcol           return the row and column selected

```



```

*                                     from the menu.
* Description   This function constructs a grid menu, and
*               waits for user input. It then returns the
*               row and column of the selection to its caller.
* Returns      *prrow, *prcol   Row and column (relative to menu) of
*                               user selection.
**/
void grid (dev, row, col, mouse_style, prrow, prcol)
int dev;
int row, col;
int mouse_style;
int *prrow, *prcol;
{
    BMENU *pmenu;
    BORDER border;
    WHERE where;
    int ch, scan, x, y;
    char s[3];

    /* Figure out where to display the menu. */
    where.dev = dev;
    where.page = 0;
    where.corner.row = row;
    where.corner.col = col;

    /* Create the menu data structure. */
    if ((pmenu = mncreate (6, 40,
                          MYTEXTATR, MYHILATR,
                          MYPROATR, MYLNGATR)) == NIL)
        exitBad ()

    /* Set up items on the menu. We will make a grid
    /* of numbers from 0 to 47, in rows across the
    /* menu. */
    for (y = 0; y < 6; y++)
        for (x = 0; x < 8; x++)
            /* Put a the ASCII characters for a number (y*8+x)
            /* at location (y, x*5) in the menu. */
            if (mnitem (pmenu, y, (x * 5), 0,
                      itoa (((y * 8) + x), s, 10)) == NIL)
                exitBad ()

    /* Disable ESC key. */
    if (mnkey (pmenu, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
              MN_ABORT, MN_DELETE) == NIL)
        exitBad ()

```

```

        /* Make a border with a bottom left title.          */
border.type      = BBRD_SSSS | BBRD_BLT;
border.attr      = MYBORDATR;
border.ch        = NUL;
border.pbtitle   = "Grid Menu (bottom left title)";
border.btattr    = MYTITATR;

        /* Display the menu on the screen.                  */
if (mndisplay (pmenu, &where, &border) == NIL)
    exitBad ()

        /* Make menu aware of the mouse.                   */
if (NULL == mnmstyle(pmenu, mouse_style, MO_LEFT))
    exitBad ()

        /* Read a response from the menu.                  */
if (mnread (pmenu, 0, 0, prrow, prcol, &ch, &scan, MN_DESTROY))
    exitBad ()
}

/**
 * Name          VIRTUAL -- Display and allow user selection from
 *                a virtual menu.
 * Synopsis      virtual (dev, row, col, prrow, prcol);
 *
 *                int dev          Device on which to display the
 *                                menu. Either SC_COLOR or SC_MONO.
 *                int row, col     Row and column where the upper-
 *                                left corner of the menu's data
 *                                area should appear.
 *                int mouse_style  The mouse style to use.
 *                int *prrow,      Pointers to variables in which to
 *                *prcol           return the row and column selected
 *                                from the menu.
 * Description    This function constructs a virtual menu, and waits
 *                for user input. It then returns the row and column of
 *                the selection to its caller.
 * Returns       *prrow, *prcol    Row and column (relative to menu) of
 *                                user selection.
 */
void virtual (dev, row, col, mouse_style, prrow, prcol)
int dev;
int row, col;
int mouse_style;
int *prrow, *prcol;

```

```

{
    BMENU *pmenu;
    BORDER border;
    WHERE where;
    int ch, scan;

        /* Figure out where to display the menu. */
    where.dev = dev;
    where.page = 0;
    where.corner.row = row;
    where.corner.col = col;

        /* Create the menu data structure. */
    if ((pmenu = mcreate (7, 39,
                        MYTEXTATR, MYHILATR,
                        MYPROATR, MYLNGATTR)) == NIL)

        exitBad ()
            /* Set up items on the menu, and add keys to the key*/
            /* binding list (upper and lower case of the first */
            /* letters of the items). */
            /* First, items on the menu at a Chinese restaraunt.*/
    if (mnitmkey (pmenu, 0, 0, 0, "Mandarin Chicken", "Mm",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 1, 0, 0, "Broccoli Beef", "Bb",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 2, 0, 0, "Sweet & Sour Pork", "Ss",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 3, 0, 0, "Won Ton Soup", "Ww",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 4, 0, 0, "Pork Fried Rice", "Pp",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 5, 0, 0, "Potstickers", "Pp",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 6, 0, 0, "Fortune Cookies", "Ff",
                MN_NOMOVE) == NIL)

        exitBad ()
        /* Now, items on the menu at an Italian restaraunt. */
    if (mnitmkey (pmenu, 0, 19, 0, "Lasagna", "Ll",
                MN_NOMOVE) == NIL)

        exitBad ()
    if (mnitmkey (pmenu, 1, 19, 0, "Tortellini Pesto", "Tt",

```

```

        MN_NOMOVE) == NIL)
    exitBad ()
    if (mnitmkey (pmenu, 2, 19, 0, "Ravioli w/Meat Sauce", "Rr",
        MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 3, 19, 0, "Fettucini Al Fredo", "Ff",
        MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 4, 19, 0, "Veal Parmesan", "Vv",
        MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 5, 19, 0, "Linguini", "Ll",
        MN_NOMOVE) == NIL)
        exitBad ()
    if (mnitmkey (pmenu, 6, 19, 0, "Calamari", "Cc",
        MN_NOMOVE) == NIL)
        exitBad ()
        /* Disable ESC key. */
    if (mnkey (pmenu, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
        MN_ABORT, MN_DELETE) == NIL)
        exitBad ()
        /* Make a border with a bottom centered title. */
    border.type = BBRD_SSSS | BBRD_BCT;
    border.attr = MYBORDATR;
    border.ch = NUL;
    border.pbtitle = "Virtual";
    border.btattr = MYTITATR;

        /* Display the menu in a 4 X 30 viewport. */
    if (mnvdisp (pmenu, &where, 4, 30, 0, 0, &border) == NIL)
        exitBad ()

        /* Make menu aware of the mouse. */
    if (NULL == mnmstyle(pmenu, mouse_style, MO_LEFT))
        exitBad ()

        /* Read a response from the menu. */
    if (mread (pmenu, 0, 0, prrow, prcol, &ch, &scan, MN_DESTROY))
        exitBad ()
}

```

#### 源程序(PULLMENU.C):

PULLMENU.C是下拉菜单系统示例程序。

```

/**
 * This program displays a menu bar across the top of the screen (the
 * "root" menu) which contains several items, each of which has an
 * associated vertical "pulldown" menu associated with it.
 * The command line format is as follows:
 *      pullmenu
 * The purpose of PULLMENU is to provide a working example of the
 * proper method of construction and use of a pulldown menu system.
 * Version      6.00 (C)Copyright Blaise Computing Inc. 1989
 **/
#include <ctype.h>
#include <stdio.h>

#include <bkeybrd.h>
#include <bkeys.h>
#include <bmenu.h>
#include <butil.h>
#include <bcreens.h>
#include <bvideo.h>
#include <bmouse.h>

/* Text color is intense white on blue. */
#define MYTEXTATR (utnybbyt (SC_BLUE, SC_WHITE | INTENSITY))

/* Color of highlight bar is black text on a white */
/* background. */
#define MYHILATR(utnybbyt (SC_WHITE, SC_BLACK))

/* Color of "protected" items is green on blue. */
#define MYPROATR(utnybbyt (SC_BLUE, SC_GREEN))

/* Color of menu borders is cyan on black. */
#define MYBORDATR (utnybbyt (SC_BLACK, SC_CYAN))

/* Color of menu titles is the same as menu borders.*/
#define MYTITATR MYBORDATR

#define NUL    '\0'
#define TRUE   1
#define FALSE  0

/* Constants returned by pulldown menu functions. */
#define MOVE_LEFT  -1
#define MOVE_RIGHT 1
#define STAY_PUT   0

/* Terminate-on-error macro. */
#define exitBad() \
{ \
    mohide(MO_HIDE); \
}

```

```

        fprintf (stderr,
                "pullmenu: died with b_wnerr = %d in line %d",
                b_wnerr, __LINE__);
        exit (b_wnerr);
    }

typedef struct          /* PULL_ITEM: Describes one item in a */
{                      /* pulldown menu. */
    int    row;         /* The row on which the item is located.*/
    char *pitem_text;   /* The item's text. */
    char *pselection_keys; /* The keys which will select the item. */
} PULL_ITEM;

typedef struct          /* ROOT_ITEM: Describes one item in the */
{                      /* root menu. */
    int    column;      /* The item's column. */
    char *pitem_text;   /* The item's text. */
    char *pselection_keys; /* The keys which will select the item. */
    int    display_pulldown; /* Whether to display the item's
                          /* pulldown automatically.
} ROOT_ITEM;

        /* Constants describing the locations of the menu */
        /* items in the root menu. */
#define ITEM_ONE_COL 1
#define ITEM_TWO_COL 12
#define ITEM_THREE_COL2
#define ITEM_FOUR_COL27

        /* Item descriptors for the root menu. */
ROOT_ITEM root_items[] =
{
    {ITEM_ONE_COL, " Language ", "Ll", 0},
    {ITEM_TWO_COL, " Machine ", "Mm", 0},
    {ITEM_THREE_COL, " OS ", "Oo", 0},
    {ITEM_FOUR_COL, " Quit ", "Qq", 0}
};

int num_root_items = sizeof(root_items) / sizeof(root_items[0]);

        /* Item descriptors for the "Language" menu. */
PULL_ITEM language_items[] =
{
    {1, " C++ ", "Cc"},
    {2, " C ", "Cc"},
    {3, " Pascal ", "Pp"},
    {4, " FORTRAN 77 ", "Ff"},
    {5, " COBOL ", "Cc"},
    {6, " Logo ", "Ll"},

```

```

    {7, " Pilot      ", "Pp"}
};
int num_language_items = sizeof(language_items) /
    sizeof(language_items[0]);

/* Item descriptors for the "Machine" menu. */
PULL_ITEM machine_items[] =
{
    {1, " Cray 2      ", "Cc"},
    {2, " VAX 11/780  ", "Vv"},
    {3, " IBM PS/2 80  ", "Ii"},
    {4, " Apple //GS   ", "Aa"},
    {5, " Sinclair ZX80 ", "Ss"}
};
int num_machine_items = sizeof(machine_items) /
    sizeof(machine_items[0]);

/* Item descriptors for the "OS" menu. */
PULL_ITEM os_items[] =
{
    {1, " VMS          ", "Vv"},
    {2, " UNIX 4.2 BSD ", "Uu"},
    {3, " OS/2         ", "Oo"},
    {4, " DOS 2.0      ", "Dd"},
    {5, " CP/M 80      ", "Cc"}
};
int num_os_items = sizeof(os_items) / sizeof(os_items[0]);

/* Declare internal subroutines. */
BMENU *create_menu(int, int, PULL_ITEM *, int);
int read_pulldown(BMENU *, int, int, PULL_ITEM *, int, int);

void main ()
{
    int      mode, columns, active_page;
    BMENU    *proot;
    BMENU    *planguage_menu;
    BMENU    *pmachine_menu;
    BMENU    *pos_menu;
    BORDER    border;
    WHERE     where;
    int      ch, scan;
    int      row, col;
    int      showrow, showcol;
    int      done = FALSE;
    ADAP_STATEadapter_state;

```

```

PAGE_STATEpage_state;
int         current_item;
int         ret_value;
int         read_root = 1;
int         error_code;

        /* First we make certain that we are in 80 column      */
        /* text video mode.                                    */

scgetvid(&adapter_state);

switch (adapter_state.mode)
{
case 2:
case 3:
case 7:
        break;

default:
        fprintf (stderr, "Pullmenu: This demonstration works only
                        "in 80 column\n");
        fprintf (stderr, "                        text modes (modes 2, 3, and 7)\n");
        exit (1);
}

        /* Save the image and cursor of the current page to */
        /* restore later.                                    */

scsavepg(&page_state);

scpclr();

        /* Create the menu data structures.                  */

proot = mncreate(1, 80, MYTEXTATR, MYHILATR, MYPROATR, 0);
if (proot == NULL)
        exitBad()

planguage_menu = create_menu(9, 12, language_items,
                             num_language_items);
if (planguage_menu == NULL)
        exitBad ()

pmachine_menu = create_menu(7, 15, machine_items,
                             num_machine_items);
if (pmachine_menu == NULL)
        exitBad ()

pos_menu = create_menu(7, 14, os_items, num_os_items);
if (pos_menu == NULL)
        exitBad ()

        /* Highlight the first item in each of the pulldown */

```



```

        /* menus. */
        mnhilite(planguage_menu, language_items[0].row, 0, MN_HIGHLIGHT);
        mnhilite(pmachine_menu, machine_items[0].row, 0, MN_HIGHLIGHT);
        mnhilite(pos_menu, os_items[0].row, 0, MN_HIGHLIGHT);

        /* Disable ESC key. */
        if (mnkey (proot, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
            MN_ABORT, MN_DELETE) == NULL)
            exitBad ()

        /* Set up items on the menu, and define
        /* corresponding keys (upper and lower case of the
        /* first letters of the items). */
        for (current_item = 0; current_item < num_root_items;
            current_item++)
        {
            mninitkey(proot, 0, root_items[current_item].column, 0,
                root_items[current_item].pitem_text,
                root_items[current_item].pselection_keys, MN_TRANSMIT) ==
                NULL)
            {
                exitBad ()
            }
        }

        /* Figure out where to display the menu. */
        where.dev = scmode(&mode, &columns, &active_page);
        where.page = active_page;
        where.corner.row = 0;
        where.corner.col = 0;

        /* Specify no border. */
        border.type = BBRD_NO_BORDER;

        /* Display the menu on the screen. */
        if (mndisplay (proot, &where, &border) == NULL)
            exitBad ()

        /* Display a status bar at the bottom of the screen.*/
        viatrect(scrows() - 1, 0, scrows() - 1, 79, SC_WHITE | INTENSITY,
            SC_RED);
        vldspmsg(scrows() - 1, 0, -1, -1, "Current selection:");

        /* If the mouse is present, enable its cursor. */
        if (MO_OK == mohide(MO_SHOW))
            if (NULL == mamstyle(proot, MN_MOU_CLICK, MO_LEFT))
                exitBad ()

        /* Set up starting item for highlight bar. */
        current_item = 0;

```

```

do
{
    /* Leave this menu on the screen until they select */
    /* the "Quit" entry */
    row = 0;
    col = root_items[current_item].column;

    /* If read_root is non-zero, then we need to read a */
    /* user response from the root menu; otherwise */
    /* read a response from the pulldown associated */
    /* with the currently highlighted item. */
    if (read_root)
    {
        error_code = muread (proot, row, col, &row, &col,
                             &ch, &scan,
                             MN_KEEP_HIGHLIGHT | MN_ALL_TRANSMIT);
        if (error_code == MN_READ_AB)
        {
            /* If the user aborted the menu, ignore it. */
            b_wmerr = WN_NO_ERROR;
        }
        else
            if (error_code)
                exitBad ()

        /* Find the index of the selected item. */
        for (current_item = 0; current_item < num_root_items;
             current_item++)
        {
            if (root_items[current_item].column == col)
                break;
        }
    }

    /* Set up location to show sub-menu. */
    showrow = where.corner.row + 2;
    showcol = where.corner.col + col;

    ret_value = STAY PUT;

    /* If the user transmitted an item, display its */
    /* pulldown; if it was the "Quit" item, quit. */
    if (read_root &&
        ((ch == KB_C_N_ENTER) && (scan == KB_S_N_ENTER)) ||
        ((ch == 0xff) && (scan == 0xff) &&
         ((b_mnmoevent & MO_DCLICK) == MO_DCLICK)))
    {
        if (current_item == num_root_items - 1)

```

```

        done = TRUE;
    else
        root_items[current_item].display_pulldown = 1;
}

/* If the pulldown menu associated with this item
/* should be displayed, then display it and read a
/* user response from it.
if (root_items[current_item].display_pulldown)
{
    /* Go do what was requested.
    switch (col)
    {
    case ITEM_ONE_COL:
        ret_value = read_pulldown(planguage_menu,
                                showrow, showcol,
                                language_items,
                                num_language_items, 20);

        break;

    case ITEM_TWO_COL:
        ret_value = read_pulldown(pmachine_menu,
                                showrow, showcol,
                                machine_items,
                                num_machine_items, 32);

        break;

    case ITEM_THREE_COL:
        ret_value = read_pulldown(pos_menu,
                                showrow, showcol,
                                os_items,
                                num_os_items, 47);

        break;
    }

    /* Assume that we're going to have to read a user
    /* response from the main menu.
    read_root = 1;

    /* If the key was a left or right arrow key, we
    /* move the highlight bar the appropriate direction.*/
    if ((ret_value == MOVE_LEFT) || (ret_value == MOVE_RIGHT),
    {
        /* First, make sure this item's pulldown will be
        /* displayed next time it is highlighted.
        root_items[current_item].display_pulldown = 1;

        /* Now unhighlight the current item and determine
        /* which item should be highlighted next.
        mnhilite(proot, 0, root_items[current_item].column,
                MN_UNHIGHLIGHT);

```

```

        current_item = (current_item + ret_value +
                        num_root_items) % num_root_items;

        /* Check to see if we need to read the root menu
        /* next time around.
        if (root_items[current_item].display_pulldown)
        {
            read_root = 0;
            mnhilite(proot, 0, root_items[current_item].column,
                    MN_HIGHLIGHT);
        }
    }
    else
        root_items[current_item].display_pulldown = 0;
}

} while (!done);

mmdstroy(proot);

        /* Turn mouse cursor off.
mohide(MO_HIDE);

        /* Restore cursor position and style.
scsetvid(&adapter_state);
screstpg(&page_state);
}

/**
 *
 * Name          READ_PULLDOWN -- Display and allow user selection from
 *                  a pulldown menu.
 *
 * Synopsis      ret_value = read_pulldown(pmenu, row, col,
 *                  pitems, num_items, msg_col);
 *
 * int ret_value      The return value for the parent menu
 *                  to act on.
 * BMENU *pmenu      The pulldown menu to read from.
 * int row, col      Row and column where the upper
 *                  left corner of the menu's data
 *                  area should appear.
 * PULL_ITEM *pitems An array of item descriptors for
 *                  the menu.
 * int num_items      The number of items in *pitems.
 * int msg_col      The column at which to display a
 *                  message indicating which item was
 *                  selected.

```

```

*
*
* Description      This function displays pmenu and waits for user input.
*                  If the user presses a left or right arrow key, it will
*                  return MOVE_LEFT or MOVE_RIGHT to its caller, as
*                  appropriate.  An abort key or mouse event will cause
*                  b_wmerr to be cleared, and the return code will be
*                  STAY_PUT. A transmission key or mouse event causes
*                  a return value of STAY_PUT.
*
* Returns          ret_value    MOVE_LEFT, MOVE_RIGHT, or STAY_PUT.
*
**/

```

```

int read_pulldown(pmenu, row, col, pitems, num_items, msg_col)
BMENU *pmenu;
int row;
int col;
PULL_ITEM *pitems;
int num_items;
int msg_col;
{
    int mode, columns, active_page;
    BORDER border;
    WHERE where;
    int ch, scan;
    int error_code;

    /* Figure out where to display the menu. */
    where.dev = scmode(&mode, &columns, &active_page);
    where.page = 0;
    where.corner.row = row;
    where.corner.col = col;

    /* Make a border with no title. */
    border.type = BBRD_SSSS;
    border.attr = MYBORDATR;
    border.ch = NUL;

    /* Display the menu on the screen. */
    if (mndisplay(pmenu, &where, &border) == NULL)
        exitBad ()

    /* Read a response from the menu. */
    error_code = mnread(pmenu, 1, 0, &row, &col, &ch, &scan,
        MN_KEEP_HIGHLIGHT | MN_REMOVE | MN_PREV_BAR);

    if (error_code == MN_READ_AB)
    {

```

```

        b_wmerr = WN_NO_ERROR;
        return(STAY_PUT);
    }
    if (error_code)
        exitBad ()

        /* Tell the caller whether a left or right arrow */
        /* key was hit. */
        if ((ch == KB_C_N_LEFT) && (scan == KB_S_N_LEFT))
            return(MOVE_LEFT);

        if ((ch == KB_C_N_RIGHT) && (scan == KB_S_N_RIGHT))
            return(MOVE_RIGHT);

        /* Display a message indicating which item was */
        /* selected. */
        if ((row - 1) < num_items)
            vidspmsg(scrows() - 1, msg_col, -1, -1,
                pitems[row - 1].pitem_text);

        /* Tell the caller not to move left or right, and */
        /* not to call us immediately the next time our */
        /* root menu item is selected. */
        return(STAY_PUT);
    }

```

```

/**
 *
 * Name          CREATE_MENU -- Create a vertical menu given its
 *                  dimensions and an array of its items.
 *
 * Synopsis      menu = create_menu(height, width, pitems, num_items);
 *
 *              BMENU *pmenu    Pointer to the newly created menu.
 *              int height,      The height and width of the menu to
 *              width            create.
 *              PULL_ITEMS *pitems Array of pulldown menu item
 *                  descriptors.
 *              int num_items    Number of items in pitems.
 *
 * Description    This function constructs a menu given its dimensions
 *                  and an array of item descriptors for its items.      It
 *                  designates the left and right arrow keys to be
 *                  transmission keys, and disables the ESC key.      It also
 *                  sets the window up to use the "click" style of mouse
 *                  events.
 *
 * Returns       pmenu    Pointer to the newly created menu, or NULL
 *                  if an error occurs.

```

```

*
**/

BMENU *create menu(height, width, pitems, num_items)
int      height;
int      width;
PULL_ITEM *pitems;
int      num_items;
{
    BMENU *pcreated;
    int current_item;

    pcreated = mncreate(height, width, MYTEXTATR, MYHILATR,
                        MYPROATR, 0);
    if (pcreated == NULL)
        return(NULL);

        /* Change the left and right arrow keys.          */
    if (mnkey(pcreated, 0, 0, KB_C_N_LEFT, KB_S_N_LEFT,
              MN_TRANSMIT, MN_CHANGE) == NULL)
        exitBad ()
    if (mnkey(pcreated, 0, 0, KB_C_N_RIGHT, KB_S_N_RIGHT,
              MN_TRANSMIT, MN_CHANGE) == NULL)
        exitBad ()
        /* Disable ESC key.                                */
    if (mnkey (pcreated, 0, 0, KB_C_N_ESC, KB_S_N_ESC,
              MN_ABORT, MN_DELETE) == NULL)
        exitBad ()
        /* Install mouse events.                            */
    if (NULL == mnmstyle(pcreated, MN_MOU_CLICK, MO_LEFT))
        exitBad ()

        /* Now install all the specified items.              */
    for (current_item = 0; current_item < num_items; current_item++)
    {
        if (mnitmkey(pcreated, pitems[current_item].row, 0, 0,
                    pitems[current_item].pitem_text,
                    pitems[current_item].pselection_keys, MN_TRANSMIT) ==
            NULL)
        {
            exitBad ()
        }
    }
    return(pcreated);
}

```

## 第十章 鼠标器编程

鼠标器支持函数为与标准Microsoft兼容的鼠标器驱动程序模块提供了一个接口, 这些函数提供了检测鼠标器存在、控制其光标、感知其按钮状态的直接手段。除了这些基本函数外, 您还可以探测一次点按及两次点按, 调整作为检测基准的时间间隔。您甚至还可以安装自己的用C写的函数, 使之在出现某种类型的鼠标器事件时被调用。与插入码相结合, 您还可以在对鼠标器事件作出反应时访问DOS功能。

### 鼠标器事件的种类

#### 访问鼠标器状态

MOEQUIP	报告鼠标器的存在以及鼠标器按钮的数量。
MOSTAT	报告鼠标器的当前位置, 它的按钮是处于释放还是按下状态。
MOBUTTON	报告指定的鼠标器按钮的按下或释放的次数, 它还报告最后指定的按下或释放事件发生时鼠标器的位置。
MOCHECK	报告任何类型的鼠标器事件的发生, 这些事件包括点按、两次点按、按钮的按下或释放以及鼠标器的移动。事件也可以包括对移位键的操作。

#### 控制鼠标器位置

MOCURMOV	移动鼠标。
MORANGE	在某个方向上限定鼠标的移动(垂直或水平)。
MOGETMOV	报告自上次查询MOGETMOV以来物理鼠标器的移动。

#### 控制鼠标的外观

MOHIDE	操作驱动程序的内部光标标志来隐藏或显示光标。
MOAVOID	定义屏幕上的一个区域, 当鼠标处于显示状态而鼠标器进入该区域时, 这个函数隐藏光标, 将内部光标标志置为-1。
MOSOFT	将字符方式的鼠标设置为软件光标, 它以改变屏幕上字符或属性的方法显示自己。
MOHARD	将字符方式鼠标与显示适配器上的闪烁硬件光标联系起来(但不是BIOS所记录的光标位置)。
MOGRAPH	定义图形方式下鼠标的外观。

#### 对鼠标器硬件中断的反应

MOHANDLR	在某个鼠标器操作出现时安装或摘除一个例程, 以便接管控制。这些操作包括: 鼠标器移动、鼠标器按钮的按下或释放。
MOPRECLK	安装一个内部例程(MOCATCH)来维护MOCHECK检查的历史记录。

#### 其它鼠标器操作

MOSPEED	设置鼠标器灵敏度, 即鼠标的移动对物理鼠标器移动的响应速度。
MOJUMP	在鼠标器快速移动时使鼠标移动得比正常速度更快。
MOLITPEN	使鼠标器对光笔的模拟有效或无效。
MORESET	对鼠标器驱动程序初始化。
MOGATE	是所有由Microsoft鼠标器驱动程序提供的服务例程的总入口。



## 处理鼠标器中断

鼠标器驱动程序在对鼠标器运动以及鼠标器按钮的按下和释放作出反应时可以调用一个用户函数。Turbo C TOOLS提供了一个在C中建立和安装这种鼠标器中断处理函数的简便方法。

鼠标器中断处理程序的设计与一般的中断服务例程非常相似。在阅读下面的内容之前可以浏览一下“中断服务例程(ISR)”一章。

## 调用屏蔽：相关事件组

必须决定激活处理函数的鼠标器事件的类型，这些事件由一个十六位值中的位组来表示，该十六位值称为调用屏蔽。用户一次可以仅安装一个鼠标器中断处理程序，所以应该安排好调用屏蔽，使之包括全部有关的事件。调用屏蔽位定义如下：

### 鼠标器硬件中断调用屏蔽

符号	值	意义
MO_MOVE	0x0001	鼠标器移动。
MO_L_PRESS	0x0002	左按钮按下。
MO_L_RELEASE	0x0004	左按钮释放。
MO_R_PRESS	0x0008	右按钮按下。
MO_R_RELEASE	0x0010	右按钮释放。
MO_M_PRESS	0x0020	中按钮按下。
MO_M_RELEASE	0x0040	中按钮释放。

## 调用步骤

一个鼠标器中断处理程序在声明时带有一个参数，该参数是一个ALLREG结构的地址。(ALLREG在butil.h中定义，“中断服务支持(IS)”一章对它作了说明。)下面这个简单的处理程序展示了一个恰当的调用步骤：

```
#include <bmouse.h>
extern unsigned long count = 0L;
void cdecl myhandler(const ALLREG *);
void cdecl myhandler(preg)
const ALLREG *preg;
{
    count++;
}
```

这个处理程序只做一件事：对全局变量count加1。

所有ALLREG结构包含从鼠标器驱动程序传来的机器寄存器的一个拷贝，这些寄存器的值定义如下：

寄存器	值
AX	指明事件发生的条件屏蔽，它的位设置与调用屏蔽相匹配。
BX	指明已按下按钮的位组： MO_LEFT (1)、MO_RIGHT (2) 和/或MO_MIDDLE (4)。
CX	以像素为单位的水平光标坐标。
DX	以像素为单位的垂直光标坐标。
DI	以米基为单位的水平鼠标器移动。
SI	以米基为单位的垂直鼠标器移动。

其它寄存器中的值由驱动程序产生，但Microsoft没有定义它们的值。

## 鼠标器编程函数源程序函数使用参考

mocheck, mohide, moprecik 提供鼠标器支持的子程序。  
ercode = mocheck(events,ignore,option,pfound,pvert,phoriz);  
ercode = mohide(option);

```

ercode = mopreclk(option);
ercode      返回MO_NOLINK
u_row      忽略
u_col      忽略
l_row      忽略
l_col      忽略
events     忽略
ignore     忽略
option     忽略
pfound     忽略
*pvert,*phoriz 忽略
option     忽略

```

这些函数除了用来返回MO\_NOLINK错误代码外不做任何事情。设计它们是用来防止在没有直接使用鼠标的程序中包括鼠标支持子程序。

； 本模块是独立的。在BMOUSE.H头文件中，该函数被声明为“cdecl far”，强迫所有的调用程序以正确的方法调用。

可以任何方式重汇编该文件，并可以任何内存模式连接生成的.OBJ文件。建议使用小模式的COMPILER.MAC，在所有内存模式中使用该结果.OBJ文件。

返回 ercode 返回MO\_NOLINK。

```

include beginasm.mac
MO_NOLINK equ -3 ; Must match BMOUSE.H.
beginMod bnomouse
; MOCHECK
beginCseg mocheck
beginProc mocheck ; Will exit with far RETURN.
mov ax,MO_NOLINK ; Error code.
db 0cbh ; Far return.
endProc mocheck
endCseg mocheck
; MOHIDE
beginCseg mohide
beginProc mohide ; Will exit with far RETURN.
mov ax,MO_NOLINK ; Error code.
db 0cbh ; Far return.
endProc mohide
endCseg mohide
; MOPRECLK
beginCseg mopreclk
beginProc mopreclk ; Will exit with far RETURN.
mov ax,MO_NOLINK ; Error code.
db 0cbh ; Far return.
endProc mopreclk
endCseg mopreclk
endMod bnomouse
end

```

**MOAVOID** 在指定区域中隐藏鼠标。

```
#include<bmouse.h>
```

```
int moavoid(unsigned u_row, unsigned u_col,
            unsigned l_row, unsigned l_col);
u_row, u_col
```

区域的左上角(以像素为单位, 相对于屏幕左上角(0,0))。

```
l_row, l_col
```

区域的右下角(以像素为单位, 相对于屏幕左上角(0,0))

(返回) 错误代码:

```
MO_OK      (0)    成功;
MO_ABSENT (2)    鼠标器驱动程序未安装;
MO_BAD_OPT (1)    任选项不被承认;
MO_RANGE  (2)    u_row超过l_row或u_col超过l_col。
```

MOAVOID定义屏幕上的一个区域, 当鼠标处于显示状态而进入该区域时, 该函数使鼠标隐藏, 将内部光标标志置为-1。从下次起鼠标进入该区域时保持隐藏, 直至mohide (MO\_SHOW)被激活足够的次数。在激活MOHIDE之前该区域一直被指定为隐藏区域。

如果鼠标已被隐藏, 则MOAVOID根本不起作用, 不会影响内部光标标志。

通常使用这个函数的方法是调用MOAVOID, 然后执行一个屏幕操作, 再调用 mohide (MO\_SHOW)恢复鼠标。这个函数比调用mohide(MO\_HIDE)略快一些, 它不会使鼠标不必要地闪灭。

### 源程序(MOAVOID.C):

```
#include <bmouse.h>
int far moavoid(u_row, u_col, l_row, l_col)
unsigned u_row, u_col, l_row, l_col;
{
    int result;
    DOSREG regs;
    if (u_row > l_row || u_col > l_col)
        result = MO_RANGE;
    else
    {
        regs.ax = 16;
        regs.cx = u_col;
        regs.dx = u_row;
        regs.si = l_col;
        regs.di = l_row;
        result = mogate(&regs, &regs);
    }
    return result;
}
```

### MOBUTTON 报告鼠标器按钮的按下/释放历史

```
#include <bmouse.h>
int mobutton( int event,
              unsigned *pbuttons,
              unsigned *pcount,
              unsigned *pvert, unsigned *phoriz);
event        待报告的按钮事件, 由两个二位域中的值指定: MO_LEFT(1)、MO_RIGHT (2) MO_MIDDLE
(3)及MO_PRESS (8)、MO_RELEASE(16)
```

pbuttons 返回按钮当前状态的变量的地址。变量中的某一位被置位表示相应按钮现在已被按下：  
MO\_LEFT (1)、MO\_RIGHT (2)和/或MO\_MIDDLE(4)。  
(如果不存在第三个按钮，MO\_MIDDLE位总是被清空。)

pcount 变量的地址，该变量返回上次查询以来指定按钮的指定事件发生的次数。

pvert, phoriz

变量的地址，该变量返回指定按钮最后一次发生指定事件时鼠标器的位置(以像素计，  
相对于显示幕左上角(0,0)。

(返回) 错误代码：

MO\_OK (0) 成功；  
MO\_ABSENT (-2) 鼠标器驱动程序未安装；  
MO\_BAD\_OPT (1) 事件的位组合未被承认。

MOBUTTON 报告指定鼠标器按钮的按下或释放次数，还报告上次指定事件发生时鼠标器的位置，  
然后清除指定事件的内部计数器。

报告的事件数目反映了自上次查询以来事件发生的次数。如果另一个程序询问同一项内容，则本程  
序报告的数量将比正确值小，因为那个程序的询问会清除内部计数器。

如果询问MO\_MIDDLE而鼠标器只有两个按钮，这时不报告错误。

### 源程序(MOBUTTON.C):

```
#include <mouse.h>
int mobutton(event,pbuttons,pcount,pvert,phoriz)
int event;
unsigned *nbuttons,*pcount,*pvert,*phoriz;
{
    int result;
    DOSREG regs;
    if (moequip() <= 0) /* Check mouse presence. */
        return MO_ABSENT;
    /* Check mouse function number. */
    switch (event & (MO_PRESS | MO_RELEASE))
    {
        case MO_PRESS: regs.ax = 5; break;
        case MO_RELEASE: regs.ax = 6; break;
        default: return MO_BAD_OPT;
    }
    switch (event & MO_ALLBUTTONS)
    {
        case MO_LEFT: regs.bx = 0; break;
        case MO_RIGHT: regs.bx = 1; break;
        case MO_MIDDLE: regs.bx = 2; break;
        default: return MO_BAD_OPT;
    }
    if (regs.bx > b mouse)
        return MO_BAD_OPT; /* Requested button missing. */
    result = inogate(&regs,&regs);
    if (result == MO_OK)
    {
        *pbuttons = regs.ax;
        *pcount = regs.bx;
```

```

        *phoriz    = regs.cx;
        *pvert     = regs.dx;
    }
    return result;
}

```

## **mocatch**      捕获鼠标器按钮的按下与释放。

不能直接从C中调用。

入口的寄存器值:

AX 条件屏蔽位:

```

0001h  鼠标移动
0002h  MO_L_PRESS  左按钮按下
0004h  MO_L_RELEASE 左按钮释放
0008h  MO_R_PRESS  右按钮按下
0010h  MO_R_RELEASE 右按钮释放
0020h  MO_M_PRESS  中按钮按下
0040h  MO_M_RELEASE 中按钮释放

```

BX 按钮状态位:

```

0001h  MO_LEFT
0002h  MO_RIGHT
0004h  MO_MIDDLE

```

CX 水平光标的象素位置。

DX 垂直光标的象素位置。

DI 水平鼠标的米基位置。

SI 垂直鼠标的米基位置。

出口的寄存器值:

所有的寄存器都不改变。

函数从某些鼠标事件中接受控制。它保存鼠标按钮按下与释放的位置和次数以及那时事件的移位键状态的历史。

如果安装了用户中断处理子程序(MOHANDLR.C), 函数把控制传给MOHANDLR中的调度程序。并且只当鼠标事件匹配用户函数的调用掩码时才传递控制。

返回                    b\_mohist            更新最近 鼠标事件的数组。

```

#include beginasm.mac
beginMod mocatch
MO_EVENT struc
; MO_EVENT: One mouse button
; event. This must match
; definition in BMOUSE.H.
;
MO_EV_EVENT    dw 0
; MO_PRESS or MO_RELEASE.
MO_EV_TIME     dd 0fffffffh
; BIOS clock ticks since midnight.
MO_EV_C_VERT   dw 0
; Cursor location in pixels
MO_EV_C_HORIZ  dw 0
MO_EV_VERT     dw 0
; Mouse location in mickeys.
MO_EV_HORIZ    dw 0
MO_EV_KSTAT    dw 0
MO_EV_RELEVANT dw 0
; Newness bits:
; MO_CLICK, MO_DCLICK, MO_PRESS,
; and MO_RELEASE

```

```

MO_EVENT ends                                ; indicating whether this event
                                              ; has been read & cleared.

MO_LEFT      equ 0001h                      ; Event symbols.
MO_RIGHT     equ 0002h                      ; Must match BMOUSE.H!
MO_MIDDLE    equ 0004h
MO_PRESS     equ 0008h
MO_RELEASE   equ 0010h
MO_CLICK     equ 0040h
MO_DCLICK    equ 0080h
MO_HIST_LIMIT equ 5                        ; Number of events recorded.
                                              ; Must match BMOUSE.H!

MO_L_PRESS   equ 0002h                      ; Condition mask bits in AX.
MO_L_RELEASE equ 0004h
MO_R_PRESS   equ 0008h
MO_R_RELEASE equ 0010h
MO_M_PRESS   equ 0020h
MO_M_RELEASE equ 0040h

REG_INPUT struc                             ; Saved copy of register
    REG_AX    dw ?                         ; values, pushed on entry.
    REG_BX    dw ?
    REG_CX    dw ?
    REG_DX    dw ?
    REG_DI    dw ?
    REG_SI    dw ?
REG_INPUT ends
    ; External variables defined elsewhere.
    beginDSeg
    extSym b_modispat,dword , Address of dispatcher that
                                ; calls user interrupt handler.
    extSym b_mohanmask,word  ; Mask value (if any) for user
                                ; mouse interrupt handler.
    ; Define & initialize global variables.
    pubSym b_mohist,<MO_EVENT (3 * MO_HIST_LIMIT) dup (<>)>
    endDSeg
    beginCSeg mocache
temp_vector dd ?                        ; Temporary vector in code
                                              ; segment

;*****
;
; Internal routine DO_ONE_BUTTON:
;
; Entry registers:
;
;     DX:AX  32-bit BIOS time of day (clock ticks).
;     DI      Copy of BIOS keyboard shift status word.
;     DS:BX   Address of first MO_EVENT structure for this button.
;     ES      Duplicate copy of DS.

```

```

;      SS:BP   Address of REG_INPUT structure containing register
;              values on entry to MOCATCH.
;      CX      Mask to capture button state (MO_LEFT, MO_RIGHT, or
;              MO_MIDDLE)
;      DF      (Direction Flag) = 1 (downward)
;
;      Exit registers:
;
;      CX,SI          Undefined
;      AX,BX,DX,DS,ES,DF,SS,BP,DI   Unchanged'

do_one_button    proc    near
                    assume  ds:nothing,es:nothing
;      Copy button's history backward one event to make room for new event.
        push    di
        push    cx
        mov     si,bx
        add     si,((MO_HIST_LIMIT - 1) * type b_mohist@) - 2
        mov     di,si
        add     di,type b_mohist@
        mov     cx,((MO_HIST_LIMIT - 1) * type b_mohist@) / 2
        rep     movsw
        pop     cx
        pop     di

;      Record event type:  press or release.
        test    [bp].REG_BX,cx
        jnz     button_down
        mov     [bx].MO_EV_EVENT, MO_RELEASE
        mov     [bx].MO_EV_RELEVANT,MO_RELEASE or MO_CLICK or MO_DCLICK
        jmp     short state_done

button_down:
        mov     [bx].MO_EV_EVENT, MO_PRESS
        mov     [bx].MO_EV_RELEVANT,MO_PRESS or MO_CLICK or MO_DCLICK

state_done:
;      Record other details of this event.
        mov     [bx].MO_EV_KSTAT,di          ; Keyboard status.
        mov     word ptr [bx].MO_EV_TIME,ax  ; Time of day.
        mov     word ptr [bx].MO_EV_TIME+2,dx
        mov     cx,[bp].REG_CX              ; Mouse cursor location.
        mov     [bx].MO_EV_C_HORIZ,cx
        mov     cx,[bp].REG_DX
        mov     [bx].MO_EV_C_VERT,cx
        mov     cx,[bp].REG_DI              ; Mouse location.
        mov     [bx].MO_EV_HORIZ,cx
        mov     cx,[bp].REG_SI
        mov     [bx].MO_EV_VERT,cx
        ret
do_one_button    endp

```

\*\*\*\*\*

```

;
; Actual entry point for MOCATCH.
beginProc mocatch          ; Will exit via far RETurn.
    push    bp
    mov     bp,sp
    push    ds
    push    es
    pushf
    push    si              ; Save entry register values.
    push    di
    push    dx
    push    cx
    push    bx
    push    ax

; Load registers with values that apply to all buttons.
    mov     bp,sp          ; SS:BP -> set of input registers.
    xor     ax,ax          ; Fetch BIOS time of day.
    mov     ds,ax
    assume  ds:nothing
    les     ax,ds:[046ch]
    assume  es:nothing
    mov     dx,es          ; DX:AX = time of day.
    mov     di,ds:[0417h]  ; Shift key status word.
    mov     bx,seg b_mohist@
    mov     ds,bx          ; DS -> default data segment.
    assume  ds:nothing
    mov     es,bx
    assume  es:nothing
    lea     bx,b_mohist@

                                ; DS:BX -> history for this button
                                ; ES == DS

    std

; Record left button event if it was pressed or released.
    test    [bp].REG_AX,MO_L_PRESS or MO_L_RELEASE
    jz      did_left
    mov     cx,MO_LEFT
    call    do_one_button
did_left:
    add     bx,(MO_HIST_LIMIT * type b_mohist@)

; Record right button event if it was pressed or released.
    test    [bp].REG_AX,MO_R_PRESS or MO_R_RELEASE
    jz      did_right
    mov     cx,MO_RIGHT
    call    do_one_button
did_right:

; Record middle button event if it was pressed or released.
    test    [bp].REG_AX,MO_M_PRESS or MO_M_RELEASE
    jz      did_middle
    add     bx,(MO_HIST_LIMIT * type b_mohist@)
    mov     cx,MO_MIDDLE

```



```

        call    do_one_button
did_middle:
;   All done.
;   Check whether to pass control to user's interrupt handler also.
                                ; DS -> default data segment.
        mov     ax,ds:b_mohanmask@ ; User event bits.
        test    [bp].REG_AX,ax      ; Actual event bits.
        jz      straight_exit       ; Zero implies user handler isn't
                                ; interested in this event.
;   Jump to user's interrupt handler.
        les     ax,ds:b_modispat@   ; Set up copy of vector in
        assume  es:nothing          ; code segment.
        mov     word ptr temp_vector ,ax
        mov     word ptr temp_vector+2,es
        pop     ax                  ; Restore registers.
        pop     bx
        pop     cx
        pop     dx
        pop     di
        pop     si
        popff
        pop     es
        assume  es:nothing
        pop     ds
        assume  ds:nothing
        pop     bp
        jmp     temp_vector         ; Jump to user's handler.
;   Return directly to mouse driver.
straight_exit:
        pop     ax                  ; Restore registers and exit.
        pop     bx
        pop     cx
        pop     dx
        pop     di
        pop     si
        popff
        pop     es
        assume  es:nothing
        pop     ds
        assume  ds:nothing
        pop     bp
        db      0c0h                ; Far return.
        endProc mocatch
        endCseg mocatch
        endMod mocatch
        end

```

## MOCHECK 检查最近发生的鼠标器事件

```
#include <bmouse.h>
```

```

int far decl mocheck( unsigned long events,
                     unsigned long ignore,
                     int option,
                     unsigned long *pfound,
                     unsigned *pvert, unsigned *phoriz);

```

events 指明待检查事件的位屏蔽。下面是有关的位值:

符号	值	意义
MO_LEFT	0x0001	左鼠标器按钮。
MO_RIGHT	0x0002	右鼠标器按钮。
MO_MIDDLE	0x0004	中鼠标器按钮。
MO_PRESS	0x0008	按钮按下。
MO_RELEASE	0x0010	按钮释放。
MO_CLICK	0x0040	按钮快速地按下和释放。
MO_HOLD	0x0800	按钮Q时处于指定位置。
MO_RSHIFT	0x1000	右shift键按下。
MO_LSHIFT	0x2000	左shift键按下。
MO_CSHIFT	0x4000	Ctrl键按下。
MO_ASHIFT	0x8000	Alt键按下。

ignore 指示需忽略的事件位的位屏蔽, 它可以被设置为下列值之一:

MO_LEFT	MO_RSHIFT
MO_RIGHT	MO_LSHIFT
MO_MIDDLE	MO_CSHIFT
MO_ASHIFT	

option 附加的任选位: MO\_CLEAR(0x0010)消除事件(即破坏性读);  
MO\_NOCLEAR(0x0020)允许再次探测到事件的发生。

pfound 探测到的事件

pvert, phoriz

变量的地址, 该变量返回最近记录到的事件完成的鼠标的位置(以象素计)。

(返回) 错误代码:

MO_OK	(0)	成功;
MO_NOLINK	(-3)	b_nomouse.obj 与该程序链接;
MO_ABSENT	(-2)	鼠标器驱动程序未安装;
MO_BAD_OPT	(1)	任选项未被承认;
MO_BAD_COMBO	(3)	events列出一个不可能事件。

MOCHECK报告任何种类的鼠标器事件的发生以及事件发生时鼠标器的位置。这个函数可以任选地清除事件, 使之不再被检测检测到。

MOCHECK自动调用MOPRECLK来安装MOCATCH。(MOCATCH是一个内部函数, 它探测由MOCHECK监视的事件。)从内存摘除用户程序之前必须激活mopreclk (MO\_REMOVE)。

用户通过设置events中不同的位组合可以指定多重事件, 比如在一个或多个按钮上的点按与/或两次点按。

利用ignore值可以指明事件位的通配符。例如, 设置ignore为MO\_CSHIFT使行无论Ctrl键是否按下都可以探测到指定的事件。

用全局变量b\_clicklimit和b\_dclicklimit可以分别调整点按及两次点按的最大时间间隔, 两个变量均以BIOS时钟为单位测算时间。如果不改变它们, 则b\_clicklimit的缺省值是4个时钟单位, b\_dclicklimit是9个时钟单位。见下面的示例。

如果指定了MO\_CLEAR, 则鼠标器按钮事件的历史将被标记下来, 这样系统将不再报告鼠标器的按下和释放事件。

当指明MO\_CLGEAR时, 探测一个组合事件(点按或两次点按)具有清除基本事件(按下、释放和一次点按)的效果。例如, 如果一个按钮被点按而您探测到并清除了这个点按, 您就再也不能将那个点按的按下和释放探测为单独的事件。

当指明MO\_CLEAR时, 探测一个基本事件(按下、释放或点按)将检测使用这个单独事件的组合事件成为可能。例如, 如果用户点按了一个按钮。您就可以探测并清除这个按钮, 以后也可以探测到这个点按动作。

鼠标器位置以相对于屏幕左上角(0,0)的象素来计算。

如果指定了MO\_MIDDLE而鼠标器上没有第三个按钮, 则不报告错误。

## 源程序(MOCHECK.C):

```
#include <bkeybrd.h>
#include <bmouse.h>
long b_clicklimit    = 4L;    /* Time limits for clicks and double    */
long b_dclicklimit = 9L;    /* clicks (measured in clock ticks).    */
#define TICKS_PER_DAY 0x1800b0L
int far mocheck(events, ignore, option, pfound, pvert, phoriz)
unsigned long  events, ignore;
int           option;
unsigned long *pfound;
unsigned      *pvert, *phoriz;
{
    int          button, index, ints_were_on;
    int          i;
    long         interval;
    MO_EVENT     hist[MO_HIST_LIMIT];
    unsigned char check_shift[MO_HIST_LIMIT];
    unsigned     buttons, vert, horiz;
    unsigned char shift_events, shift_ignore;
    int          clearing, match;
    KEYSTATUS    kstat;
    unsigned char shift;
    const int     button_mask[] = {MO_LEFT, MO_RIGHT, MO_MIDDLE};
    int           ercode;
    /* Install MOCATCH if necessary.    */
    if (b_mocatch == 0)
        if (MO_OK != (ercode = mopreclk(MO_INSTALL)))
            return ercode;
    /* Validate arguments.    */
    switch (option)
    {
        case MO_CLEAR: clearing = 1; break;
        case MO_NOCLEAR: clearing = 0; break;
        default: return MO_BAD_OPT; /* Unknown option.    */
    }
    if (events & (MO_DCLICK | MO_CLICK | MO_RELEASE | MO_PRESS))
        if (0 == (events & MO_ALLBUTTONS))
            return MO_BAD_COMBO;
    /* General setup    */
    ignore = (~ignore) | MO_DCLICK | MO_CLICK | MO_RELEASE | MO_PRESS | MO_HOLD;
    events &= ignore;
    shift_events = (unsigned char) uthinyb(uthibyte((unsigned int) events));
```

```

shift_ignore = (unsigned char) uthinyb(uthbyte((unsigned int) ignore));
/* For each button in turn, check all specified events. */
for (button = 0; button <= 2; button++)
{
    if (events & button_mask[button])
    {
        /* Disable interrupts so history */
        /* stands still. */
        ints_were_on = utintflg(UT_INTTOFF);
        /* Make a local copy of button history */
        /* for faster access. */
        memcpy(hist,&b_mohist[button][0],sizeof(hist));
        if (!clearing)
            utintflg(ints_were_on);
        /* Select relevant shift key bits from */
        /* history. */
        for (i = 0; i < MO_HIST_LIMIT; i++)
            check_shift[i] = shift_ignore & * (int *) &hist[i].kstat;
/* Check for double clicks. */
        if (events & MO_DCLICK)
        {
            /* Double clicks require four */
            /* consecutive events:  press, release, */
            /* press, release.  If most recent */
            /* event is press, skip over it. */
            if (hist[0].event == MO_PRESS)
                index = 1; /* Button is down. */
            else
                index = 0; /* Button is up. */
            /* Check that recent events are */
            /* relevant & that they match shift keys*/
            match = 1;
            for (i = index; i <= index + 3; i++)
            {
                if ( shift_events != check_shift[i]
                    || !(hist[i].relevant & MO_DCLICK))
                {
                    match = 0;
                    break;
                }
            }
            if (match)
            {
                /* Event qualifies; check duration. */
                interval = hist[index].time
                    - hist[index+3].time;
                if (interval < 0)
                    interval += TICKS_PER_DAY; /* Midnight wrap. */
                if (interval <= b_dclicklimit)
                {
                    if (clearing)

```

```

        { /* Clear all clicks, presses, and releases. */
            for (i = index; i <= index + 3; i++)
                b_mohist[button][i].relevant = 0;
            utintflg(ints_were_on);
        }
        *pfound = MO_DCLICK | button_mask[button]
            | (utlonyb(* (int *) &hist[index].kstat) << 12);
        *pvert = hist[index].c_vert;
        *phoriz = hist[index].c_horiz;
        return MO_OK;
    }
}

/* Check for clicks. */
if (events & MO_CLICK)
{
    /* Clicks require two consecutive */
    /* events: press, release. If most */
    /* recent event is press, skip over it. */
    if (hist[0].event == MO_PRESS)
        index = 1; /* Button is down. */
    else
        index = 0; /* Button is up. */
    /* Check that recent events are */
    /* relevant & that they match shift keys*/
    match = 1;
    for (i = index; i <= index + 1; i++)
    {
        if ( shift_events != check_shift[i]
            || !(hist[i].relevant & MO_CLICK))
        {
            match = 0;
            break;
        }
    }
    if (match)
    {
        /* Event qualifies; check duration. */
        interval = hist[index].time
            - hist[index+1].time;
        if (interval < 0)
            interval += TICKS_PER_DAY; /* Midnight wrap. */
        if (interval <= b_clicklimit)
        {
            if (clearing)
            {
                /* Clear single clicks, presses, and releases. */
                for (i = index; i <= index + 1; i++)
                    b_mohist[button][i].relevant &=
                        ~(MO_CLICK | MO_PRESS | MO_RELEASE);
            }
        }
    }
}

```

```

        utintflg(ints_were_on);
    }
    *pfound = MO_CLICK | button_mask[button]
        | (utlonyb(* (int *) &hist[index].kstat) << 12);
    *pvert = hist[index].c_vert;
    *phoriz = hist[index].c_horiz;
    return MO_OK;
}

}

}

/* Check for releases */
if (events & MO_RELEASE)
{
    /* Check last two events. */
    for (index = 0; index < 2; index++)
    {
        if ( (hist[index].relevant & MO_RELEASE)
            && check_shift[index] == shift_events)
        {
            if (clearing)
            {
                /* Clear this event. */
                b_mohist[button][index].relevant &= ~MO_RELEASE;
                utintflg(ints_were_on);
            }
            *pfound = MO_RELEASE | button_mask[button]
                | (utlonyb(* (int *) &hist[index].kstat) << 12);
            *pvert = hist[index].c_vert;
            *phoriz = hist[index].c_horiz;
            return MO_OK;
        }
    }
}

/* Check for presses */
if (events & MO_PRESS)
{
    /* Check last two events. */
    for (index = 0; index < 2; index++)
    {
        if ( (hist[index].relevant & MO_PRESS)
            && check_shift[index] == shift_events)
        {
            if (clearing)
            {
                /* Clear this event. */
                b_mohist[button][index].relevant &= ~MO_PRESS;
                utintflg(ints_were_on);
            }
            *pfound = MO_PRESS | button_mask[button]
                | (utlonyb(* (int *) &hist[index].kstat) << 12);
            *pvert = hist[index].c_vert;
            *phoriz = hist[index].c_horiz;

```

```

        return MO_OK;
    }
}
}
}
}
if (clearing)
    utintflg(ints_were_on);

/* Check for held button state. */

*pfound = 0L;
if (events & MO_HOLD)
{
    mostat(&buttons,&vert,&horiz);
    kbstatus(&kstat);
    shift = utlonyb(* (int *) &kstat);
    if ( (shift & shift_ignore) == shift_events
        && (buttons & ignore) == (events & MO_ALLBUTTONS))
    {
        *pfound = MO_HOLD | buttons | (shift << 12);
        *pvert = vert;
        *phoriz = horiz;
    }
}
return MO_OK;
}

```

## MOCURMOV 移动鼠标

```

#include <bmouse.h>
int mocurmov( unsigned vert,
              unsigned horiz);
vert, horiz  相对于显示左上角(0,0)的以像素计的鼠标器新位置。
(返回)      错误代码:
            MO_OK      (0)   成功;
            MO_ABSENT  (-2)  鼠标器驱动程序未安装。

```

MOCURMOV 移动鼠标。

新位置受限于MORANGE 所设置的当前鼠标范围。 如果所要求的位置超出了范围，则最终位置将在允许的范围内尽可能接近要求的位置。

(必要时)坐标取整为最接近当前显示方式所允许的值。

### 源程序(MOCURMOV.C):

```

#include <bmouse.h>

int mocurmov(vert,horiz)
unsigned vert,horiz;
{

```

```

DOSREG regs;
regs.ax = 4;
regs.cx = horiz;
regs.dx = vert;
return mogate(&regs,&regs);
}

```

## MOEQUIP 检查鼠标器驱动程序的存在

```
#include <bmouse.h>
```

```
int moequip(void);
```

(返回) 如果驱动程序已安装，返回按钮的数量；如果未安装，返回值不大于零。

MOEQUIP感知与Microsoft兼容的鼠标器驱动程序的存在。如果发现了驱动程序，可以肯定地得出鼠标器也已安装的结论，因为鼠标器驱动程序在启动时寻找鼠标器。

MOEQUIP也把它的返回值保存在全局变量b\_mouse中。

该函数第一次被调用时将驱动程序初始化。

用MORESET可以显式地初始化驱动程序。

### 源程序(MOEQUIP.C):

```

#include <dos.h> /* For int86(), REGS. */
#include <bintrpt.h>
#include <bmouse.h>
int b_mouse = MO_UNKNOWN; /* Mouse presence: MO_UNKNOWN, */
/* 0 if absent, or number of buttons. */

int moequip()
{
    const unsigned char far *pvector;
    union REGS inregs,outregs;
    if (b_mouse == MO_UNKNOWN)
    {
        /* Check interrupt 0x33: */
        /* absent if NIL or IRET. */
        pvector = isgetvec(0x33);
        if (pvector == FARNIL || *pvector == 0xf)
            b_mouse = MO_ABSENT;
        else
        {
            /* Test for driver presence by */
            /* initializing the driver. */
            inregs.x.ax = 0;
            inregs.x.bx = 0;
            int86(0x33,&inregs,&outregs);

            if (outregs.x.ax == 0)
                b_mouse = 0; /* AX unchanged if no driver. */
            else
                /* Number of buttons. */
                b_mouse = ((outregs.x.bx == 0xffff) ? 2 : outregs.x.bx);
        }
    }
}

```



```

    return b_mouse;
}

```

## MOGATE 鼠标器驱动程序的入口

```

#include <bmouse.h>
int mogate( const DOSREG *pinregs,
            DOSREG *poutregs);
pinregs    结构的地址, 该结构包含传给 鼠标器驱动程序的寄存器值。
poutregs   结构的地址, 该结构包含从 鼠标器驱动程序返回的寄存器值。
(返回)     成功代码:
            MO_OK      (0)    成功;
            MO_ABSENT  (-2)   驱动程序未安装。
MOGATE     是通向所有鼠标器驱动程序服务例程的总入口。
DOSREG     在butil.h头文件中定义如下:
struct dreg
{
    unsigned ax,bx,cx,dx,si,di,ds,es;
};
#define DOSREG struct dreg
    *pinregs 与*poutregs可以是同一个结构。

```

### 源程序(MOGATE.C):

```

#include <dos.h> /* For int86x(), REGS, SREGS. */
#include <bmouse.h>
int mogate(pinregs,poutregs)
const DOSREG *pinregs;
DOSREG *poutregs;
{
    int result;
    union REGS inregs,outregs;
    struct SREGS sregs;
    if (b_mouse == MO_UNKNOWN)
        moequip();
    if (b_mouse <= 0)
        result = MO_ABSENT;
    else
    {
        inregs.x.ax = pinregs->ax;
        inregs.x.bx = pinregs->bx;
        inregs.x.cx = pinregs->cx;
        inregs.x.dx = pinregs->dx;
        inregs.x.si = pinregs->si;
        inregs.x.di = pinregs->di;
        sregs.ds = pinregs->ds;
        sregs.es = pinregs->es;
        int86x(0x33,&inregs,&outregs,&sregs);
    }
}

```

```

        poutregs->ax = outregs.x.ax;
        poutregs->bx = outregs.x.bx;
        poutregs->cx = outregs.x.cx;
        poutregs->dx = outregs.x.dx;
        poutregs->si = outregs.x.si;
        poutregs->di = outregs.x.di;
        poutregs->ds = sregs.ds;
        poutregs->es = sregs.es;
        result = MO_OK;
    }
    return result;
}

```

## MOGETMOV 报告自上次查询以来物理鼠标器的移动

```
#include <bmouse.h>
```

```
int mogetmov( int *pvert,
              int *phoriz);
```

pvert 变量的地址，该变量返回以米基为单位的鼠标器垂直运动。正值表示向下移动。

phoriz 变量的地址，该变量返回以米基为单位的鼠标器水平运动。正值表示向右移动。

(返回) 错误代码:

MO\_OK (0) 成功;

MO\_ABSENT (-2) 鼠标器驱动程序未安装。

MOGETMOV报告自上次调用该函数以来鼠标器的物理运动，单位是米基。米基是物理鼠标器移动的单位。(对于Microsoft鼠标器，米基为1/200英寸。)

### 源程序(MOGETMOV.C):

```
#include <bmouse.h>
```

```
int mogetmov(pvert,phoriz)
```

```
int *pvert,*phoriz;
```

```
{
```

```
    int    result;
```

```
    DOSREG regs;
```

```
    regs.ax = 11;
```

```
    result = mogate(&regs,&regs);
```

```
    if (result == MO_OK)
```

```
    {
```

```
        *phoriz = regs.cx;
```

```
        *pvert = regs.dx;
```

```
    }
```

```
    return result;
```

```
}
```

## MOGRAPH 设置鼠标器图形方式光标

```
#include <bmouse.h>
```

```
int mograph( const unsigned *pmasks,
             int hot_row,int hot_col);
```

pmasks      32 字数组的地址。该数组描述光标的点及颜色模式。

hot\_row,hot\_col

光标热点的行列位置，以相对于象素模式左上角(0,0)的象素为单位。在屏幕方式4和方式5下，hot\_col应该为9数。

(返回)      成功代码：

MO\_OK          (0)      成功；

MO\_ABSENT      (-2)      鼠标器驱动程序未安装。

MOGRAPH定义图形方式下鼠标的形状、颜色和中心(“热点”)。

pmasks 所指向的前十六字是“屏幕屏蔽”，指明已有的屏幕数据的哪些位应被忽略；后十六字是光标屏蔽”，指明在作用了屏幕屏蔽后需反转的屏幕数据位。参见MOSOFT中有关屏幕和光标屏蔽位操作的更详细说明。

## 源程序(MOGRAPH.C):

```
#include <bmouse.h>
```

```
int mograph(pmasks,hot_row,hot_col) /
```

```
const unsigned *pmasks;
```

```
int                hot_row,hot_col;
```

```
{
    DOSREG regs;
    regs.ax = 9;
    regs.bx = hot_col;
    regs.cx = hot_row;
    regs.dx = utoff(pmasks);
    regs.es = utseg(pmasks);
    return mogate(&regs,&regs);
}
```

## MOHANDLR      安装或摘除中断处理程序

```
#include <bmouse.h>
```

```
int mohandlr( PMOHANDLR pfunc,
```

```
              unsigned call_mask,
```

```
              char *pstack,
```

```
              int stksize,
```

```
              int option);
```

pfunc      待安装的鼠标器中断处理程序的地址。

call\_mask      位屏蔽。指明调用中断处理程序的事件的类型。

符号

值

意义

MO_MOVE	0x0001	鼠标器移动。
MO_L_PRESS	0x0002	左按钮按下。
MO_L_RELEASE	0x0004	左按钮释放。
MO_R_PRESS	0x0008	右按钮按下。
MO_R_RELEASE	0x0010	右按钮释放。
MO_M_PRESS	0x0020	中按钮按下。
MO_M_RELEASE	0x0040	中按钮释放。

pstack 用于中断处理程序堆栈的内存块的起始地址。  
 stksize 中断处理程序堆栈的尺寸(以字节计), 应该至少为512, 典型值是2048。  
 option MO\_INSTALL (0) 或MO\_REMOVE (1)。 如果指定了MO\_REMOVE, 则忽略所有其它参量。  
 (返回) 错误代码:

MO_OK	(0)	成功;
MO_ABSENT	(-2)	鼠标器驱动程序未安装;
MO_BAD_OPT	(1)	option未被承认;
MO_ALREADY	(4)	鼠标器驱动程序已安装而指定了MO_INSTALL;
MO_RANGE	(2)	下端超过上端;
MO_NOT_INSTALLED	(5)	鼠标器驱动程序未安装而指定了MO_REMOVE。

MOHANDLR安装一个用户函数(pfunc), 当某个鼠标器事件上发生时, 该函数接管控制。鼠标器事件包括鼠标器的运动、鼠标器按钮的按下或释放。

在同一时间内只能有一个鼠标器处理程序可以是活动的。要改变处理程序或改变调用屏蔽, 首先应指定MO\_REMOVE删除原来的处理程序, 然后再用新的调用屏蔽安装新的处理程序。

### 源程序(MOHANDLR.C):

```
#include <bmouse.h>

/* Local static variables. */
static ISRCTRL dispat_block = {0}; /* ISR control block. */
static PMOHANDLER p_user_handler = 0; /* Pointer to user function. */

/* Internal functions. */

static void cdecl mo_dispatcher(ALLREG *,ISRCTRL *,ISRMSG *);
int mohandlr(pfunc,call_mask,pstack,stksize,option)
PMOHANDLER pfunc;
unsigned call_mask;
char *pstack;
int stksize,option;
{
    int result;
    unsigned new_mask;
    if (moequip() <= 0) /* Test mouse presence. */
        return MO_ABSENT;
    switch (option)
    {
        case MO_INSTALL:
            if (p_user_handler)
                result = MO_ALREADY;
            else
            {
                /* Set up dispatcher. */
                p_user_handler = pfunc;
                isprep(mo_dispatcher,"MOUSEDISPATCHER",&dispat_block,
                    pstack,stksize,1);
                /* Install interrupt handler
                 * if necessary.
                */
            }
        }
    }
}
```

```

it (b_mocatch)
{
    /* MOCATCH installed; need not */
    /* install new handler, but */
    /* perhaps augment call mask. */
    new_mask = (call_mask | b_momask);
    if (new_mask == b_momask)
        result = MO_OK; /* No change in call mask. */
    else
        /* Install new call mask. */
        result = moinst(b_mocatch,new_mask);
}
else
    /* Install dispatcher directly. */
    result = moinst((void (far *) ()) &dispat_block,
        call_mask);
if (result == MO_OK)
{
    /* Set up pointer & call mask */
    /* so MOCATCH can call */
    /* dispatcher. */
    b_modispat = &dispat_block;
    b_mohanmask = call_mask;
}
else
    p_user_handler = 0;
}
break;
case MO_REMOVE:
    if (!p_user_handler)
        result = MO_NOT_INSTALLED;
    else
    {
        b_mohanmask = 0;
        b_modispat = NIL;
        p_user_handler = 0;
        result = moinst(b_mocatch,b_momask);
    }

    break;

default:
    result = MO_BAD_OPT;
    break;
}

return result;
}

/**

```

```

*
* Name          mo_dispatcher -- Invoke user mouse interrupt handler.
*
* Description    This function calls the user mouse interrupt handler.
*
*               This function is built as an interrupt service routine
*               (ISR) to be called from the mouse driver via the ISR
*               dispatcher.  If MOCATCH is installed, the mouse driver
*               calls MOCATCH directly, then MOCATCH calls this function
*               (via the ISR dispatcher) if the event is among those
*               specified in the call mask.
*
**/

```

```

static void cdecl mo_dispatcher(pregs,pisrctrl,pmsg)
ALLREG *pregs;
ISRCTRL *pisrctrl;
ISRMSG *pmsg;
{
    ALLREG regs;          /* Declare local copy of registers      */
                          /* so user handler can't affect *pregs.*/
    const char *p;

    p = (const char *) pisrctrl; /* Suppress compiler warning */
    p++;                      /* that pisrctrl isn't used. */

    if (p_user_handler)
    {
        regs = *pregs;
        p_user_handler(&regs); /* Call user handler.      */
    }

    pmsg->exit_style = IEXIT_RETF;
}

```

## MOHARD 设置鼠标器硬件字符方式光标

```

#include <bmouse.h>
int mohard(int high,
           int low);

```

high, low 高低扫描行值。

(返回) 代码：如果成功，为MO\_OK(0)；如果鼠标器驱动程序未安装，则为MO\_ABSENT。

MOHARD将鼠标器字符方式光标同显示适配器的闪烁硬件光标联接起来(但不与BIOS已记录的光标位置相联系)，使软件光标失效。

MOSOFT可以用软件产生字符方式光标。

### 源程序(MOHARD.C):

```

#include <bmouse.h>

```

```

int mohard(high,low)
int high,low;
{
    DOSREG regs;

    regs.ax = 10;
    regs.bx = 1;
    regs.cx = high;
    regs.dx = low;
    return mogate(&regs,&regs);
}

```

## MOHIDE 隐藏或显示鼠标

```

#include <bmouse.h>
int far cdecl mohide(int option);
option      待执行的动作:
            MO_HIDE (1)   使内部光标标志减量(于是将光标隐藏);
            MO_SHOW (0)   使内部光标标志增量(也许使光标显示);
(返回)      错误代码:
            MO_OK      (0)   成功;
            MO_NOLINK (-3)   b_nomouse.obj与本程序链接;
            MO_ABSENT (-2)   鼠标器驱动程序未安装;
            MO_BAD_OPT (1)   option 未被承认。

```

MOHIDE操作鼠标器驱动程序的内部光标标志来显示或隐藏光标。

驱动程序通过内部光标标志来决定鼠标的显示或隐藏,这个标志是一个计数器,其范围是从-32,768到0。当标志值为0时光标显示;当值小于0时光标隐藏。对驱动程序初始化或重置显示方式时,标志的初始值为-1。

### 源程序(MOHIDE.C):

```

#include <bmouse.h>

int far mohide(option)
int option;
{
    DOSREG regs;

    switch (option)
    {
        case MO_SHOW: regs.ax = 1;          break;
        case MO_HIDE: regs.ax = 2;          break;
        default:       return MO_BAD_OPT;
    }

    return mogate(&regs,&regs);
}

```

**moinst** 无条件安装鼠标器中断处理子程序。

```

ercode = moinst(pfunc,call_mask);
ercode      错误返回码
            MO_OK 成功
            MO_ABSENT 未发现鼠标。
            void (far *pfunc)()
                安装函数
            unsigned call_mask
                标明调用处理子程序事件种类的位掩码
            MO_MOVE      鼠标移动
            MO_L_PRESS   左按钮按下
            MO_L_RELEASE 左按钮释放
            MO_R_PRESS   右按钮按下
            MO_R_RELEASE 右按钮释放
            MO_M_PRESS   中按钮按下
            MO_M_RELEASE 中按钮释放
        该函数安装鼠标器中断处理程序
    返回      ercode      错误返回代码
            MO_OK 成功
            MO_ABSENT 未发现鼠标
            b_mouse      鼠标按钮数(如果没有鼠标驱动程序)
    
```

**源程序(MOINST.C):**

```

#include <bmouse.h>

int moinst(pfunc,call_mask)
void (far *pfunc)();
unsigned call_mask;
{
    DOSREG regs;

    regs.ax = 12;
    regs.cx = call_mask;
    regs.dx = utoff(pfunc);
    regs.es = ulseg(pfunc);
    return mogate(&regs,&regs);
}
    
```

**MOJUMP** 设置鼠标器加速阈值

```

#include <bmouse.h>
int mojump(unsigned speed);
speed      鼠标运动自动加速的阈值速度(以每秒米基数为单位)
(返回)     错误代码:
            MO_OK      (0)    成功;
            MO_ABSENT  (-2)    鼠标器驱动程序未安装。
    
```



MOJUMP 允许鼠标器快移时鼠标移动得比正常速度更快。通常，鼠标根据MOSPEED设置的速度移动。然而，当鼠标器移动得比指定速度更快时，鼠标的速度也加快，使得鼠标更容易移过较大距离。

米基是物理鼠标器运动的单位。对于Microsoft mouse,一个米基为0.005英寸。

鼠标器重置时缺省跳速阈值是每秒64个米基，也就是说，如果鼠标器移动的速度超过每秒64个米基，鼠标移动得比通常更快。如果鼠标器移动得较慢，鼠标移动得也较慢，这就有利于控制微小移动。

加速的程序取决于鼠标器驱动程序。对于Microsoft mouse,速度加倍。

用户可能通过把阈值设置为一个很大的值(如32,767)的办法取消跳速效果。

### 源程序(MOJUMP.C):

```
#include <bmouse.h>
int mojump(speed)
unsigned speed;
{
    DOSREG regs;
    regs.ax = 19;
    regs.dx = speed;
    return mogate(&regs,&regs);
}
```

### MOLITPEN 使鼠标器光笔模拟有效或失效

```
#include <bmouse.h>
int molitpen(int option);
option      待执行的动作:
            MO_PEN      (1)   模拟光笔(于是隐藏鼠标);
            MO_NO_PEN   (0)   不模拟光笔。
(返回)      错误代码:
            MO_OK       (0)   成功;
            MO_ABSENT   (-2)  鼠标器驱动程序未安装;
            MO_BAD_OPT  (1)   option 未被承认。
```

MOLITPEN 使鼠标器光笔模拟有效或失效。

当光笔模拟有效时，按下鼠标器按钮模拟“笔落”状态，鼠标位置被记录下来，直至下一次按下按钮；当所有鼠标器按钮释放时，则模拟(“光笔出屏幕”状态。

鼠标器重置时(通过MORESET)自动进入光笔模拟状态。如果在鼠标器之外还想使用光笔，必须通过(MO\_NO\_PEN)使模拟失效。

### 源程序(MOLITPEN.C):

```
#include <bmouse.h>

int molitpen(option)
int option;
{
    DOSREG regs;

    switch (option)
    {
```

```

        case MO_PEN:      regs.ax = 13;          break;
        case MO_NO_PEN:   regs.ax = 14;          break;
        default:          return MO_BAD_OPT;
    }

    return mogate(&regs,&regs);
}

```

## MOPRECLK 安装或删除MOCHECK所有的内部例程。

```
#include <bmouse.h>
```

```
int far cdecl mopreclk(int option);
```

option MO\_INSTALL(0) 或 MO\_REMOVE(1)。

(返回) 错误代码:

MO_OK	(0)	成功;
MO_NOLINK	(-3)	_nomouse.obj 与本程序链接;
MO_ABSENT	(-2)	鼠标器驱动程序未安装;
MO_BAD_OPT	(1)	option未被承认;
MO_RANGE	(2)	低值超过高值;
MO_ALREADY	(4)	驱动程序已安装而又指定了MO_INSTALL;
MO_NOT_INSTALLED	(5)	驱动程序未安装而又指定了MO_RESERVE。

MOPRECLK 安装或删除MOCATCH。(MOCATCH 监视 鼠标器按钮的按下和释放, 维护MOCHECK所依赖的按钮事件历史b\_mohist。)

警告: 从内存摘除一个程序之前需删除MOCATCH。

## 源程序(MOPRECLK.C)示例:

```
#include <bmouse.h>
```

```
int far mopreclk(option)
```

```
int option;
```

```
{
```

```
    int result;
```

```
    switch (option)
```

```
    {
```

```
        case MO_INSTALL:
```

```
            if (b_mocatch)
```

```
                result = MO_ALREADY;
```

```
            else
```

```
            {
```

```
                /* Use combined call mask that */
```

```
                /* encompasses MOCATCH and user */
```

```
                /* interrupt handler, if any. */
```

```
                result = moinst(mocatch,MOCATCH_MASK | b_mohanmask);
```

```
                if (result == MO_OK)
```

```
                {
```

```
                    b_mocatch = mocatch;
```

```
                    b_momask = MOCATCH_MASK;
```

```
                }
```

```

    }
    break;

case MO_REMOVE:
    if (b_mocatch)
    {
        /* Reinstall user handler (if any). */
        result = moinst((void (far *) (void)) b_modispat,
                        b_mohanmask);
        if (result == MO_OK)
        {
            b_mocatch = 0;
            b_momask = 0;
        }
    }
    else
        result = MO_NOT_INSTALLED;
    break;
default:
    result = MO_BAD_OPT;
    break;
}
return result;
}

```

## MORANGE 设置鼠标器范围界限

#include <bmouse.h>

int morange( int option,

unsigned low, unsigned high);

option 限制方向: MO\_VERT (3) 或 MO\_HORIZ (0).

low,high 低象素界限和高象素界限(低值从显示的左上边沿开始).

(返回) 错误代码:

MO\_OK (0) 成功;

MO\_ABSENT (-2) 鼠标器驱动程序未安装;

MO\_BAD\_OPT (1) option未被承认

MO\_RANGE (2) 低值超过高值。

MORANGE在某个方向上(水平或垂直)设置鼠标器运动的界限。

如果鼠标欲游离到界限之外的某个位置,它立即被移到界限内最靠近该位置的地方。

## 源程序(MORANGE.C):

#include <bmouse.h>

int morange(option,low,high)

int option;

unsigned low,high;

{

DOSREG regs;

```

switch (option)
{
    case MO_HORIZ:regs.ax = 7;          break;
    case MO_VERT: regs.ax = 8;          break;
    default:      return MO_BAD_OPT;
}

if (low > high)
    return MO_RANGE;

regs.cx = low;
regs.dx = high;
return mogate(&regs,&regs);
}

```

## MORESET 重置鼠标器驱动程序

```
#include <bmouse.h>
```

```
int moreset(void);
```

(返回) 如果已安装驱动程序，返回按钮的数量；如果鼠标器驱动程序未安装，返回0。

MORESET是一个宏，它将一个与Microsoft兼容的鼠标器驱动程序初始化成初始条件，包括：

使鼠标不可见；

将鼠标移到显示屏幕的中央；

将鼠标的移动限制在最初的25行80列屏幕中；

禁止任何由鼠标器驱动程序自动调用的用户例程；

使光笔模拟有效；

将灵敏度值设置为它的缺省值。

MORESET也把它返回值保存在全局变量b\_mouse中。

## 源程序(BMOUSE.H):

参见附录C的宏定义。

## MOSOFT 设置鼠标器软件字符方式光标

```
#include <bmouse.h>
```

```
int mosoft( unsigned screen_mask,
```

```
            unsigned cursor_mask);
```

screen\_mask 屏幕屏蔽值：0位表示忽略显示内存中已有的字符和属性。

cursor\_mask 光标屏蔽值：1位表示反显的字符和属性。

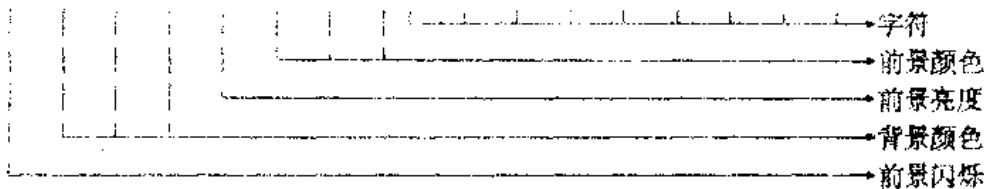
(返回) 成功代码；如果成功，为MO\_OK (0)；如果鼠标器驱动程序未安装，则为MO\_ABSENT (-2)。

MOSOFT 将鼠标器字符方式光标设置为软件光标，它通过在屏幕上改变字符和/或属性的方法显示自己，不影响一般的闪烁硬件光标。

字符方式鼠标不影响闪烁硬件光标，而MOHARD使字符方式鼠标影响硬件光标。

两个屏蔽字定义如下：

16	15	14	13	12	10	9	8	7	7	5	5	4	3	2	1	0
Fb	B	B	B	Fi	F	F	F	C	C	C	C	C	C	C	C	C



这些位与显示内存中字符和属性字节的正常安排相同。

用户需按下列方法使用屏幕和光标屏蔽值来建立软件光标的外观：

从显示内存中光标所在位置取得的已有字符和属性值是一个16位值，该值通过AND位操作与屏幕屏蔽相结合，也就是说，屏幕屏蔽中的1位保留从内存来的相应位，而屏幕屏蔽中的0位消除从内存来的相应位。结果值通过XOR位操作与光标屏蔽相结合，也就是说，光标位中的1位使得相应数据位变反，而光标屏蔽中的0位保留相应的数据位。16位结果值以字符—属性对的形式写入显示内存中。

### 源程序(MOSOFT.C):

```
#include <bmouse.h>
int mosoft(screen_mask,cursor_mask)
unsigned screen_mask,cursor_mask;
{
    DOSREG regs;
    regs.ax = 10;
    regs.bx = 0;
    regs.cx = screen_mask;
    regs.dx = cursor_mask;
    return mogate(&regs,&regs);
}
```

### MOSPEED 设置鼠标器敏度

```
#include <bmouse.h>
int mospeed( unsigned vert,
              unsigned horiz);
vert, horiz  新的垂直和水平灵敏度(每8个象素所需的米基数)，范围是1到32,767。
(返回)       错误代码：
              MO_OK      (0)    成功；
              MO_ABSENT  (-2)   鼠标器驱动程序未安装。
```

MOSPEED设置鼠标器的灵敏度，即鼠标移动速度对物理鼠标器运动的反应快慢。

灵敏度以光标移动一个象素所对应的米基数来表达。(米基是物理鼠标器运动的单位。对于Microsoft mouse，一米基为0.005英寸。)低值对应较快的光标移动，而高值提供了对细微运动的更有效的控制。

鼠标器重置时的缺省灵敏度在垂直方向上是每8个象素16米基，水平方向上每8个象素8米基。

### 源程序(MOSPEED.C):

```
#include <bmouse.h>
int mospeed(vert,horiz)
unsigned vert,horiz;
{
    DOSREG regs;
```

```

regs.ax = 15;
regs.cx = horiz;
regs.dx = vert;
return mogate(&regs,&regs);
}

```

## MOSTAT 报告鼠标器位置和按钮状态

```

#include <hmouse.h>
int mostat( unsigned *pbuttons,
            unsigned *pvert, unsigned *phoriz);
pbuttons    返回按钮状态的变量的地址。这些位中的任意一个置位表示相应按钮已按下：MO_LEFT
(1)、MO_RIGHT (2) 和/或MO_MIDDLE (4)。
            (如果不存在第三个按钮，则MO_MIDDLE 位总被清除。)
pvert, phoriz 变量的地址，该变量返回以象素为单位的鼠标器位置(相对于显示屏幕的左上角(0,0))。
(返回)      成功代码：
            MO_OK (0): 成功；
            MO_ABSENT (-2): 鼠标器驱动程序未安装。
MOSTAT 报告鼠标器按钮的状态和鼠标的当前位置。

```

### 源程序(MOSTAT.C):

```

#include <hmouse.h>

int mostat(pbuttons,pvert,phoriz)
unsigned *pbuttons,*pvert,*phoriz;
{
    int result;
    DOSREG regs;

    regs.ax = 3;
    result = mogate(&regs,&regs);

    if (result == MO_OK)
    {
        *pbuttons = regs.bx;
        *phoriz = regs.cx;
        *pvert = regs.dx;
    }

    return result;
}

```

**movars** 一些MO函数使用的全程变量：b\_mocatch, b\_momask, b\_modispat, b\_mohanimask。

该模块定义被MOHANDLR、和MOCATCH使用的四个全程变量。这些变量允许MOCATCH与用户中断处理子程序自由并存。

如果MOCATCH通过MOPRECLK安装，则b\_mocatch含有MOCATCH的地址并且b\_momash包含其调用屏蔽(名为MOCATCH\_MASK)。如果不安装MOCATCH，则两个变量包含0值。

此处在本独立模块中提供的这些变量允许MOPRECLK和MOHANDLR被用户而不必连接其它必要的模块。

### 源程序(MOVARS.C):

```
#include <bmouse.h>

void (far *b_mocatch)(void) = 0; /* Pointer to MOCATCH if */
/* installed, or 0 if not. */
/* */
unsigned b_momask = 0; /* Current MOCATCH event mask */
/* (0 or MOCATCH_MASK). */
/* */
ISRCTRL far *b_modispat = FARNIL; /* Address of dispatcher that */
/* invokes user interrupt */
/* handler, or 0 if no handler */
/* installed. */
/* */
unsigned b_mohanmask = 0; /* Call mask for user mouse */
/* interrupt handler (0 if */
/* none). */
```

### MOUSEHAN.C 鼠标中断处理子程序演示程序

```
/**
 * This program shows the same menu as the MNEXAMPL sample program,
 * with the addition of a simple mouse interrupt handler, mouse_handler().
 * Every time the mouse moves, the handler receives control and records
 * the current mouse position. The intervention function display()
 * continuously displays the mouse's position and the number of calls
 * to the mouse interrupt handler.
 */
#include <stdio.h>
#include <stdlib.h>
#include <binterv.h>
#include <bintrupt.h>
#include <bmenu.h>
#include <bmouse.h>
#include <bvideo.h>
int cdecl main(void);
void cleanup(int);
void mouse_handler(const ALLREG *);
void display(TV_EVENT *);
#define MOUSE_STACKSIZE1000 /* Size of mouse handler stack. */
#define INTERV_STACKSIZE1000 /* Size of intervention */
/* function stack. */
unsigned long num_calls = 0; /* Global variables maintained */
int mouse_row, mouse_col; /* by mouse_handler(), */
/* displayed by display(). */

int main()
```

```

{
    BMENU*pmenu;
    BORDERbord;
    WHERE location;
    int mode,columns,act_page;
    int cursor_was_off,cursor_row,cursor_col,high,low;
    int ercode,row,col,ch,keycode;
    static char mouse_stack[MOUSE_STACKSIZE];
    static char interv_stack[INTERV_STACKSIZE];
        /* Display mouse position on every */
        /* clock tick. */
    static IV_TIME interval = {1,IV_TM_INTERVAL};
    /* Create the menu and window structures in memory. */
    pmenu = mncreate(3,15, /* Dimensions of window data area. */
        SC_CYAN, /* Attributes of normal menu items. */
        REVERSE, /* Attributes of highlight bar. */
        NORMAL, /* Attributes of protected items. */
        NORMAL); /* Attributes of item descriptions. */
    if (pmenu == NIL)
        return b_werr; /* Quit if failure. */
        /* (b_werr records the most */
        /* recent menu or window error.) */
    /* Build the menu items and assign keys to select them. */
    if (NIL == mnitmkey(pmenu,0,0,MN_NOPROTECT,"Yes","yY",MN_SELECT))
        return b_werr;
    if (NIL == mnitmkey(pmenu,1,0,MN_NOPROTECT,"No","nN",MN_SELECT))
        return b_werr;
    if (NIL == mnitmkey(pmenu,2,0,MN_NOPROTECT,"Maybe","mM",MN_SELECT))
        return b_werr;
    /* Choose style of border. */
    bord.type = BBRD_DDDD; /* Box drawn with double lines. */
    bord.attr = SC_MAGENTA; /* Border will be magenta on black. */
    /* Choose where to display the menu: the active page on the */
    /* current display device. */
    location.dev = scmode(&mode,&columns,&act_page);
    location.page = act_page;
    location.corner.row = 10;
    location.corner.col = 5;
    /* If mouse is installed, turn mouse cursor on, select a mouse */
    /* usage style, and install mouse interrupt handler and the */
    /* display function. */
    if (MO_OK == mohide(MO_SHOW))
    {
        mnmstyle(pmenu,MN_MOU_CLICK,MO_LEFT);
        mohandlr(mouse_handler,MO_MOVE,
            mouse_stack,sizeof(mouse_stack),MO_INSTALL);
        ivinstal(display,"MOUSEHANDisplay",
            interv_stack,sizeof(interv_stack),
            NIL,0,
            &interval.1

```



```

        IV_NO_DOS_NEED | IV_NO_DKEY_NEED | IV_NO_FLOAT_NEED);
    }
    else
        printf("No mouse driver is installed.\n");
    /* Save former cursor position and size. */
    cursor_was_off = securst(&cursor_row,&cursor_col,&high,&low);
    /* Actually display the menu */
    if (NIL == mndisplay(pmenu,&location,&bord))
        cleanup(b_werr); /* Quit if failure. */
    /* Await the user's selection. Start the highlight bar at the
    /* item located at (0,0). Beep if unknown keystrokes are
    /* pressed. Discard the menu when done. */
    ercode = mnread(pmenu,0,0,&row,&col,&ch,&keycode,
        MN_UNKNOWN_BEEP | MN_DESTROY);
    /* Turn the mouse cursor back off. */
    mohide(MO_HIDE);
    /* Restore the cursor. */
    securset(cursor_row,cursor_col);
    sepgeur(cursor_was_off,high,low,CUR_NO_ADJUST);
    /* Report the user's selection: error code, final key, and
    /* location of item selected. */
    printf("Error code from MNREAD: %d\n",ercode);
    printf("Keystroke: ch = %d, keycode = %d\n",ch,keycode);
    printf("Item location: row = %d, column = %d\n",row,col);
    cleanup(0); /* Success. */
    return 255; /* We shouldn't fall through to
    /* this statement. */
}
/**
 * Name cleanup -- Remove interrupt handlers and terminate.
 * Synopsis cleanup(errno);
 * int errno Error code to be returned as ERRORLEVEL.
 * Description This function removes the intervention code and the
 * mouse interrupt handler and terminates the program.
 */
void cleanup(errno)
int errno;
{
    ivdisabl(ivctrl()); /* Disable intervention function (no
    /* damage if it wasn't installed). */
    /* Note: resetting the mouse driver may also reinstall the
    /* mouse's handler for asynch interrupts, making it awkward to
    /* disable intervention code. Therefore for some programs it's
    /* better to disable intervention code first. (Actually, this
    /* program doesn't need this precaution, because by default
    /* intevention code doesn't filter asynch interrupts.) */
    moreset(); /* Turn mouse cursor off and disable
    /* mouse interrupt handler. */
    exit(errno);
}

```

```

/**
 * Name          mouse_handler -- Mouse interrupt handler.
 * Synopsis      (Not to be called directly from C code:      to be called
 *               by mouse interrupt dispatcher only.)
 * Description    This function is called whenever the mouse moves.      It
 *               records the number of calls and the current mouse
 *               position (in terms of character rows and columns) in
 *               global variables.
 */
void mouse_handler(pregs)
const ALLREG *pregs;
{
    ++num_calls;
    mouse_row = preg->dx.x >> 3;
    mouse_col = preg->cx.x >> 3;
}

/**
 * Name          display -- Intervention function to report mouse position.
 * Synopsis      (Not to be called directly from C code:      to be called
 *               by intervention scheduler only.)
 * Description    This function is called on every clock tick.      It
 *               displays the current location of the mouse and the
 *               number of times the mouse interrupt handler has been
 *               called.
 */
void display(pevent)
IV_EVENT *pevent;
{
    char message[81];
    sprintf(message, "Calls to mouse interrupt handler: %8u", num_calls);
    vidspmsg(4, 35, SC_WHITE | INTENSITY, SC_BLACK, message);
    sprintf(message, "Row = %2d, Column = %2d", mouse_row, mouse_col);
    vidspmsg(5, 35, SC_WHITE | INTENSITY, SC_BLACK, message);
    pevent++;          /* Suppress compiler warning.      */
}

```

## 第十一章 打印机编程

打印机函数提供了一个与IBM BIOS打印机函数的接口, 也提供了与标准PC\_DOS假脱机打印程序print.com的接口。

### 打印机函数的种类

#### BIOS打印机接口

PRINT	对打印口初始化。
PRCHAR	打印一个字符。
PRSTATUS	报告打印口的状态。

#### 与PRINT程序的接口

PRINSTLD	反映PRINT驻留程序是否已安装。(需要DOS 3.00或更高版本。)
PRGETQ	查看假脱机队列的状态及队列中的一个文件名(由序列号指定), 它需要DOS 3.00或更高版本。
PRSPool	向假脱机队列尾加入一个文件名。它需要DOS 3.00或更高版本。
PRCANCEL	从假脱机队列中取消一个或多个文件。它需要DOS 3.00或更高版本。
PRERROR	将来自于其它PRINT接口函数的错误码转换为一条错误信息。

### 打印机控制函数源程序

#### PRCANCEL 删除假脱机打印队列中一个或全部文件

```
#include <bprint.h>
```

```
int precancel(char *pfile);
```

pfile 待删除的文件名, 以NUL("\0")结尾。如果pfile为NIL或空字符串(""), 则取消所有的文件。

(返回) 错误代码。可能的返回值包括:

PR\_OK (0) 无错误。

PR\_INSTAL (1) 未发现假脱机打印系统(未安装假脱机打印系统或DOS版本不大于3.00)。

PR\_EXIST (2) 未发现文件。

PR\_BUSY (9) 无法同假脱机打印系统通讯, 再试一次。

PRCANCEL从DOS假脱机打印队列中取消一个或全部文件名。

如果 pfile指向一个非空字符串, 则被指名的文件从打印队列中删除, 这与在命令行上键入PRINT/c file相同。

不允许使用DOS文件名扩展字符('\*'和'?')。

如果pfile为NIL或pfile所指向的字符串长度为零, 则从队列中删除全部文件。这与在DOS命令行上键入

PRINT/a

等效。

这个函数在DOS版本2上总是无效的, 它返回PR\_INSTAL(1), 表示未发现假脱机打印系统。

源程序(PRCANCEL.C) :

```

#include <dos.h>
#include <bprint.h>
#include <butil.h>

#define NUL '\0'

int precancel (pfile)
char *pfile;
{
    struct SREGS sregs;
    union REGS regs;

    /* If print spooler not installed, exit. */
    if (! printstld ())
        return (PR_INSTAL);

    /* If path is NIL or of length 0, set up to cancel
       up to cancel all files in queue. */
    if ((pfile == NIL) || (pfile[0] == NUL))
        regs.x.ax = 0x0103;

    else
    {
        /* When we put 0x0102 in AX, we are telling the
           request processor that we mean device 1 (print
           spooler), request 2 (cancel file). */
        regs.x.ax = 0x0102;

        /* Point to the pathname of the file(s) to cancel
           from the queue. */
        sregs.ds = utseg ((char far *) pfile);
        regs.x.dx = utoff ((char far *) pfile);
    }

    /* Do the request. */
    int86x (PR_DOS_INT, &regs, &sregs);

    /* If there was an error, return it. */
    if (regs.x.cflag)
        return (regs.x.ax);
    return (PR_OK);
}

```

## PRCHAR 通过BIOS向打印机发送一个字符

```

#include <bprint.h>
int prechar( int port,
             char byte);

```

port      打印机口编号(0 = lpt1, 1 = lpt2, 等等)。

byte      待打印的字符。

(返回)    打印机的最后状态(见下面的位值)。

PRCHAR是一个宏，它试图在指定的打印机上打印一个字符。

返回的打印机状态字指示打印是否成功及其它状态信息，它由下列位值组成：

符号	值	意义
PR_S_TIMEOUT	0x01	打印机时间到：字符未被打印。
PR_S_IOERR	0x08	打印机I/O 错误。
PR_S_ONLINE	0x10	打印机已联机。
PR_S_NOPAPER	0x20	打印机纸用完。
PR_S_ACK	0x40	打印机应答。
PR_S_NOTBUSY	0x80	打印机忙。

打印一个字符之前可以用PRSTATUS 测试打印机的状态。

源程序(BPRINT.H):

定义于头文件BPRINT.H中，见附录C。

**PRERROR** 返回解释错误代码的字符串，这些错误代码来自PR(打印机)函数

```
#include <print.h>
```

```
const char *prerror (int icode);
```

**icode** 由一个PR(打印机)函数返回的错误代码。

(返回) 错误信息字符串的地址。

**PRERROR** 返回一个与假脱机打印错误代码相对应的错误信息的地址。这些错误代码值由Turbo C TOOLS 的假脱机打印接口函数返回。

函数的返回值是一个静态字符串的地址，请不要修改这个字符串。

源程序(PRERROR.C):

```
#include <bprint.h>
```

```
#define PR_NUM_MAP6 /* Number of elements in the error */  
/* mapping array. */
```

```
#define PR_NUM_ERRS 13 /* Number of actual error messages. */
```

```
static const int err_map[] =  
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -1, -1, 10, -1, -1, 11, 12};
```

```
static const char *errors [] =
```

```
{  
    "no error",  
    "spooler not installed",  
    "file not found",  
    "path not found",  
    "out of handles",  
    "access denied",  
    "queue empty",  
    "out of range",  
    "queue full",  
    "spooler busy",  
    "path too long",  
    "invalid drive",  
    "unknown error"  
};
```

```
const char *prerror (err)
```

```

int err;
{
    if ((err < 0) || (err > PR_NUM_MAP) || (err_map [err] == -1))
        err = PR_NUM_MAP;

    return (errors [err_map [err]]);
}

```

## PRGETQ 报告假脱机打印队列中的一个文件名

```
#include <bprint.h>
```

```

rcode = prgetq( int n,
                int *psize,
                char *pfile);

```

n 队列中所需项的序列号。0代表当前被打印的文件，1代表下一个，等等。

psize 变量的地址，该变量返回队列中文件的数目，这个数目包括当前被打印的文件。

pfile 返回队列中第n个文件名的缓冲区的地址，这个缓冲区必须至少为65字节长，包括用于结尾NUL('\0')的一个字节。

(返回) 错误代码。可能值包括：

PR\_OK (0) 无错误。

PR\_INSTAL (1) 未发现假脱机打印系统(未安装假脱机打印系统或DOS版本小于3.00)。

PR\_EMPTY (6) 假脱机打印队列为空。

PR\_RANGE (7) 超出范围。

PR\_BUSY (9) 无法同假脱机系统通讯，再试一次。

PRGETQ 读取假脱机打印队列中的一个文件名，返回队列中文件的数目。

这个函数在DOS版本2上不能工作。在DOS版本2上它总是返回PR\_INSTAL (1)，表示未发现假脱机打印系统。

在出现错误的情况下，\*psize返回零，\*pfile返回空字符串("")。

源程序(PRGETQ.C):

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

```
#include <bprint.h>
```

```
#include <butil.h>
```

```

#define PR_QPATH_SIZE 64 /* the size of a path name in the */
                        /* printer queue. */

```

```
#define NUL '\0'
```

```
int prgetq (n, psize, pfile)
```

```
int n;
```

```
int *psize;
```

```
char *pfile;
```

```

{
    char far *pqueue;
    int err;
    struct SREGS sregs;
    union REGS regs;
    err = PR_OK;
    pfile[0] = NUL;
    *psize = 0;
    if (! prinstld ())

```

```

    return (PR_INSTAL);
/* We get the address of the spooler queue, which freezes the */
/* queue until we thaw it. This is so that data in the queue */
/* does not move around while we are reading it. */
regs.x.ax    = 0x0104; /* device 1 (spool), request 4 (queue */
                    /* address) */

int86x (PR_DOS_INT, &regs, &regs, &sregs);

if (regs.x.cflag)
    err = regs.x.ax;

else
{
    /* Get a char * to the queue items. */
    pqueue = uttofar (sregs.ds, regs.x.si, char);

    /* Normalize to inhibit wrap-around. */
    pqueue = utnorm (pqueue, char);

    /* Count the number of queue items. */
    for (; (*pqueue != NUL); (*psize)++, pqueue += PR_QPATH_SIZE)
        ;

    if (*psize == 0)
        err = PR_EMPTY;

    else if ((*psize < (n + 1)) || (n < 0))
        err = PR_RANGE;

    /* When we get to here, pqueue is pointing past the last */
    /* queue item. We have to back it up to point at the item */
    /* requested */
    else
    {
        for (; (n < *psize); pqueue -= PR_QPATH_SIZE, n++)
            ;

        /* Copy the string. Can't use strepy because pqueue and */
        /* path are in different memory models. */
        for (; (*pfile = *pqueue) != NUL; pfile++, pqueue++)
            ;

        *pfile = NUL;
    }
}

regs.x.ax    = 0x0105; /* device 1 (spool), request 5 (thaw */
                    /* the queue). */

int86 (PR_DOS_INT, &regs, &regs);
if (err != 0)
{
    *psize    = 0;

```

```

        pfile[0] = NUL;
    }
    return (err);
}

```

## PRINT 通过BIOS 初始化一个打印口

```
#include <bprint.h>
```

```
int print(int port);
```

port 打印口编号(0=lpt1, 1=lpt2,等等)。

(返回) 打印机的最终状态(见下面的位值)。

PRINT是一个宏,它初始化指定的打印口。返回的状态字由下面的位值组成:

符号	值	意义
PR_STIMEOUT	0x01	打印机时间到。
PR_S_IOERR	0x08	打印机I/O错误。
PR_S_ONLINE	0x10	打印机已联机。
PR_S_NOPAPER	0x20	打印机已用完。
PR_S_ACK	0x40	打印机应答。
PR_S_NOTBUSY	0x80	打印机忙。

源程序(BPRINT.H):

参见附录C的宏定义。

## PRINSTLD 检查驻留式假脱机打印系统PRINT 是否已安装

```
#include <bprint.h>
```

```
int prinstld(void);
```

(返回) 如果发现PRINT,返回值非零;如果未安装PRINT或DOS版本号小于3.00,返回零。

PRINSTLD 检查DOS的假脱机打印系统PRINT是否已安装。

如果DO版本比3.00旧,检查结果总是零,表示未发现PRINT (虽然实际上它有可能驻留在内存中)。

源程序(PRINSTALD.C):

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

```
#include <bprint.h>
```

```
#include <util.h>
```

```
#define FALSE 0
```

```
int prinstld ()
```

```
{
```

```
    union REGS regs;
```

```
        /* If DOS version is older than 3.00, we know that */
```

```
        /* there is no spooler we can talk to.
```

```
    if (utdosmajor < 3)
```

```
        return (FALSE);
```

```
    else
```

```
    {        /* When we put 0x0100 in AX, we are telling the */
```

```
        /* request processor that we mean device 1 (print */
```

```
        /* spooler), request 0 (installed??). */
```



```

regs.x.ax      = 0x0100;
                /* Do the request. */
int86 (PR_DOS_INT, &regs, &regs);
return ((regs.x.ax & 0xff) == 0xff);
}
}

```

## PRSPPOOL 将一个文件提交给假脱机打印系统

#include <print.h>

int prspool(const char \*pfile);

pfile 待提交打印的文件名, 以NUL ('\0') 结尾。

(返回) 错误代码。可能值包括:

PR_OK	(0)	无错误。
PR_INSTAL	(1)	未发现假脱机打印系统(未安装假脱机打印系统或DOS版本比3.00旧)。
PR_EXIST	(2)	未发现文件。
PR_PATH	(3)	未找到路径。
PR_HANDLES	(4)	打开的文件太多(没有多余的文件柄)。
PR_ACCESS	(5)	文件访问违例。(不能将卷标、目录等提交给假脱机打印系统)。
PR_FULL	(8)	队列已满, 再试一次。
PR_BUSY	(9)	无法与假脱打印系统通话, 再试一次。
PR_NAMELEN	(12)	全路径名超过64个字符。(文件名是否以NUL ('\0')结尾?)

PRSPPOOL 将指定的文件提交给DOS假脱机打印系统打印。

不允许使用DOS文件名扩展字符('\*' 和 '?')。

这个函数在DOS版本2下不能工作。在DOS版本2下, 它总是返回PR\_INSTAL (1), 表示未发现假脱机打印系统。

源程序(PRSPPOOL.C):

#include <dos.h>

#include <bprint.h>

#include <butil.h>

int prspool (pfile)

const char \*pfile;

```

{
    union REGS regs;
    struct SREGS sregs;

    /* This structure is a "request packet," of the type*/
    /* needed to communicate with the spooler. (but only*/
    /* for spool-file requests) */

    struct {
        /* The "filler" makes for unconditional structure */
        /* alignment. */
        unsigned char filler;
        /* "Level" of a packet. Undocumented, but Microsoft*/
        /* says it should be 0. */
        unsigned char level;
        /* Pointer to the name of the file to spool. */
        const char far *pfile;
    };
}

```

```

} packet;
/* If spooler is not installed, exit. */
if (! prinstld ()
    return (PR_INSTAL);
/* Set up spooler request packet. */
packet.pfile = pfile;
packet.level = 0;

/* Point to packet with registers. */
sregs.ds = utseg (&packet.level);
regs.x.dx = utoff (&packet.level);
/* When we put 0x0101 in AX, we are telling the
/* request processor that we mean device 1 (print
/* spooler), request 1 (submit file). */
regs.x.ax = 0x0101;
/* Do the request. */
int86x (PR_DOS_INT, &regs, &regs, &sregs);
/* If there was an error, return it. */
if (regs.x.cflag)
    return (regs.x.ax);
return (PR_OK);
}

```

## PRSTATUS 通过BIOS 报告打印机的状态

#include <bprint.h>

int prstatus (int port);

port 打印机口编号(0=lpt1, 1=lpt2, 等等)。

(返回) 打印机状态(见下面的位值)。

PRSTATUS 是一个宏, 它报告打印机的状态。返回的状态字由下列位值组成:

符号	值	意义
PR_S_TIMEOUT	0x01	打印机时间到。
PR_S_IOERR	0x08	打印机I/O错误。
PR_S_ONLINE	0x10	打印机已联机。
PR_S_NOPAPER	0x20	打印机纸用完。
PS_S_ACK	0x40	打印机应答。
PR_S_NOTBUSY	0x80	打印机不忙。

源程序(BPRINT.H):

参见附录C的宏定义。

## 第十二章 屏幕(视频)编程

Turbo C TOOLS 的屏幕操作函数提供了一个调用 BIOS 视频服务的高级接口,因而也提供了一个与 IBM 显示硬件的接口。用户可以选取许多 I/O 函数,将它们用于标准的显示方式或显示页。当需要最快的速度时,用户还可直接访问显示内存。

### 屏幕操作函数的种类

#### 读取屏幕方式信息

SCEQUIP	识别所安装的各种不同的显示适配器以及安装时设置的可选参数及开关。
SCMODE	报告当前的显示设备、当前方式、列数和活动的(即当前可见的)显示页。SCROWS 报告当前显示设备和方式所支持的正文行的数目。
SCPAGES	报告当前显示设备在当前方式下所支持的显示页的数目。
SCGETVID	报告当前显示设备的全部状态:方式、当前页、活动页、行、列及光标。

#### 选择显示设备和方式

SCNEWDEV	选择一个显示方式,基于所需方式重置合适的设备(于是清除屏幕)。它可以选择 25-、30-、43-或 50-行方式。
SCCHGDEV	切换至给定的显示适配器。(请注意下面提到的重要限制。)
SCSETVID	将显示适配器恢复成原先由 SCGETVID 所记录的状态,包括方式、显示页和光标。

#### 在显示页之间切换

SCAPAGE	显示(激活)当前显示设备上的所需页。它不改变光标的尺寸。
SCPAGE	选择给定的当前设备上的一个显示页,将该页作为 Turbo C TOOLS 屏幕 I/O 函数使用的页。它不显示(激活)任何显示页。

#### 控制/读取光标形状和位置

SCCURS	报告当前显示页上的光标位置和尺寸。
SCCURSET	移动当前显示页上的光标。
SCPGCUR	在当前显示页是活动(即可见)的情况下设置光标的尺寸。

#### 清除和滚动

SCPCLR	清除当前显示页。
SCCLRMSG	清除当前显示页上的一条信息。
VISROLL	使当前显示页上的一个矩形区域上滚一页或下滚一页。
VIHORIZ	使当前显示页上的一个矩形区域左移或右移。

#### 常规的屏幕写入

SCATTRIB	从当前显示页上的光标位置开始写入一个字符和属性 9 对的拷贝。
SCWRITE	在当前显示页上的光标位置写入一个字符的多个拷贝,原有的属性不变。
SCTTYWRT	向当前显示页(如果是活动的)写入一个字符并移动光标。回车、换行、响铃和退格以电传(TTY)方式处理。必要时屏幕发生滚动。
VIDSPMSG	用给定属性在当前显示页上的指定位置显示一条消息。

SCCLRMSG      清除当前显示页上的一条消息。

## 写入一个矩形区域

VIWRRECT      用一个缓冲区的内容填充当前显示页上的一个矩形区域, 该函数的任选项可以对属性进行控制。

VIWRSECT      类似于 VIWRRECT, 但 VIWRSECT 可以显示更大的矩形缓冲区中的一段。

VIATRECT      改变一个矩表区域的属性, 其中的字符不变。

SCTTYWIN      以 TTY 方式(正如 SCTTYWRT)写一个字符, 但所有输出及滚动均限定在一个矩形区域内。

SCWRAP      以 TTY 方式写入一个字符串(正象 SCTTYWIN), 但它可以“整字换行”, 这样能够避免在区域的行之间把一个字折断。

SCBOX      利用特殊的字符方式下的画框字符用单线或双线画一个方框。

SCRESTPG      利用 SCSAVEPG 保存的一个压缩拷贝恢复一个显示页。

## 屏幕读取

SCREAD      报告当前显示页光标位置的字符及其属性。

VIRDRECT      读取显示在一个矩形区域中的字符, 可带有或不带有属性。

VIRDSECT      类似 VIRDRECT, 但 VIRDSECT 可以将数据读入更大矩形区域的一段中。

SCSAVEPG      以压缩格式保存当前显示页的内容。

## 调色板支持

SCPALETT      定义由 EGA、VGA 或 MCGA 显示的整个颜色调色板。

SCPAL1      定义由 EGA、VGA 或 MCGA 显示的一个颜色。

SCMODE4      选择用于显示方式 4 (320X200 四色图形)的色板及背景颜色。

SCBORDER      设置边界颜色。

SCBLINK      选择在 Enhanced Graphics Adapter 下属性位 7 的作用: 前景闪烁背景亮度。

## 对直接视频访问的支持

VIPTTR      返回一个 far 指针, 该指针指向当前显示页上的一个给定位置(在显示内存中)。

## 保存和恢复整个显示状态

让应用程序在结束之后不干扰显示环境, 这是一种良好的行为, 有时甚至是必须的。这就需要保存显示状态, 在程序退出前小心地恢复状态。

Turbo C TOOLS 提供了一些函数来完成这些任务。SCGETVID 和 SCSETVID 读取和恢复显示方式及光标尺寸而 SCSAVEPG 和 SCRESTPG 记录并恢复所显示的正文数据。

下面是函数的列表, 这些函数保存和恢复显示环境的特定参数。

保存和恢复显示状态的函数

	读取	恢复
当前显示设备	SCMODE	SCNEWDEV
方式	SCMODE	SCNEWDEV
正文行数	SCROWS	SCNEWDEV
活动显示页	SCMODE	SCAPAGE
光标位置	SCCURST	SCCURSET
光标尺寸	SCCURST	SCPGCUR
屏幕上的正文	SCREAD、 VIRDRECT SCSAVEPG	SCATTRIB、 SCWRITE、 VIWRRECT 或 SCRESTPG

屏幕上的属性	SCREAD, VIRDIRECT 或 SCSAVEPG	SCATTRIB, VIWRRECT, VIATRECT 或 SCRESTPG
--------	------------------------------------	--

由于 BIOS 未记录某些显示状态, 所以有时不可能获取这些参数, 例如色板。

## 强制快速屏幕访问

直接显示(VI)函数检测 Color/Graphics Adapter 的存在, 计算操作的时间, 防止视频干扰(“下雪”)。然而在某些情况下用户有时希望避免出现延迟。例如, 有些显示适配器可以模拟 CGA 而不会出现雪花, 它们并不需要特殊的计时工作。有时雪花不是一个问题。

要想使 VI 函数工作得尽可能快而不考虑干扰问题, 可使用下面的语句:

```
#include <bvideo.h>
```

```
b_vifast = 1;
```

下面的语句将 VI 函数重置为一般的方式, 避免 CGA 上的干扰。

```
b_vifast = 0;
```

## 屏幕(视频)控制函数源程序

### SCAPAGE 显示(激活)一个显示页

```
#include <bcreens.h>
```

```
int scapage(int page);
```

page 待显示(激活)的页号。

(返回) 实际显示的显示页。

SCAPAGE 设置活动的显示页, 即显示该页。它不改变光标尺寸, 也不打开或关闭光标。

SCAPAGE 检查当前的显示方式。如果 page 超出范围, 则该值被强置为合适的值: 负页号置成 0, 过大的正页号置为可用的最大页号, 实际显示的页号作为函数值返回。

用 SCPAGES 可以查出当前设备在当前方式下能够支持多少页。SCPAGE 指示 Turbo C TOOLS 将屏幕输出送到一个给定页, 不论它是活动的(显示的)还是不活动的。

源程序(SCAPAGE.C):

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

```
#include <bcreens.h>
```

```
int scapage(page)
```

```
int page;
```

```
{
    int max_page;
    union REGS inregs,outregs;
    max_page = scpages() - 1; /* Maximum page value */
    utbound(page,0,max_page)
    if (max_page) /* Switch active page only if */
    { /* this device supports more */
        inregs.h.ah = 5; /* than one. */
        inregs.h.al = (unsigned char) page;
        int86(16,&inregs,&outregs);
    }
    return(page);
}
```

## SCATTRIB

### 用指定的显示属性显示一个字符的拷贝

```
#include <bscreens.h>
int scattrib( int fore,
              int back,
              char ch,
              unsigned cnt);
```

fore        前景显示属性。

back       背景显示属性。

ch         待显示的字符。

cnt        待显示的 ch 拷贝的数量。

(返回)     返回值总为 0。

SCATTRIB 从当前光标位置开始显示字符 ch 的 cnt 份拷贝,光标保持不动。拷贝在当前页(b\_curpage)上以属性(fore,back)显示,字符以一行接一行的方式写入。

如果写入时超出屏幕的末端,或(在图形方式下)写入时超出当前的正文行,这时有可能出现未预料到的结果。SCATTRIB 不滚动屏幕。

源程序(SCATTRIB.C):

```
#include <dos.h>
#include <bscreens.h>
int scattrib(fore,back,ch,cnt)
int        fore;
int        back;
char       ch;
unsigned cnt;
{
    union REGS inregs,outregs;
    if (cnt)
    {
        inregs.h.ah = 9;
        inregs.h.al = ch;
        inregs.h.bh = (unsigned char) b_curpage;
        inregs.h.bl = (unsigned char) utnybbyt(back,fore);
        inregs.x.cx = cnt;
        int86(16,&inregs,&outregs);
    }
    return(0);
}
```

## SCBLINK

### 选择前景闪烁或背景亮度

```
#include <bscreens.h>
```

```
int scblink(int option);
```

option      SC\_BLINK (1) 使属性位 7 控制前景闪烁, SC\_INTENSITY(0)  
             使属性位 7 控制背景亮度。

(返回)      返回的错误代码。可能值包括:

SC\_NO\_ERROR        (0)    成功。

SC\_BAD\_OPT         (1)    当前显示适配器不支持这个操作。

SCBLINK 控制在 EGA、VGA 和 MCGA 字符方式下属性位 7 的作用。

如果指定了 SC\_BLINK, 属性位 7 控制字符的闪烁。这是建立当显示适配器重置的缺省条件, 此时背景可用 8 种颜色, 由属性位 4-6 指定。

如果指定了 SC\_INTENSITY, 属性位 7 用于增加背景颜色的这度, 使得背景可以象前景一样选择 16 种颜色。

源程序(SCBLINK.C):

```
#include <dos.h>
#include <bcreens.h>
#define BLINK_BIT      0x20
#define OK              0          /* Error codes.          */
#define INVALID_REQUEST 1
int scblink(option)
int option;
{
    int      result;
    int      mode,act_page,columns;
    union REGS inregs,outregs;
    scequip();
    scmode(&mode,&columns,&act_page);
    if ( b_device == b_ega
        || b_device == b_vga
        || b_device == b_mega
        || b_pmodel == IBM_JR)
    {
        inregs.x.ax = 0x1003;      /* Set intensify/blinking bit. */
        inregs.h.bl = (unsigned char) option;
        int86(0x10,&inregs,&outregs);
        result = OK;
    }
    else
        result = INVALID_REQUEST;
    return result;
}
```

## SCBORDER 设置当前显示屏幕的边界颜色

#include <bcreens.h>

int scborder(unsigned color);

color 选择的颜色。

(返回) 返回的错误代码。可能值包括:

SC\_NO\_ERROR (0) 成功。

SC\_BAD\_OPT (1) 当前显示适配器和方式不支持这个操作。

SCBORDER 设置当前显示设备的边界(“屏幕外”)的颜色。

在方式 7(单色字符)或 Professional Graphics Controller 的 High\_Function

Graphics 方式下或 Enhanced Color Display 的 350 扫描线方式下不能使用非黑颜色。如果这些条件中的某一个成立, 该函数返回 SC\_BAD\_OPT, 不做任何操作。

适配器的重置将边界颜色置为 0(黑色)。

在 EGA、VGA 和 MCGA 下, 边界颜色与在色板中选择的其它颜色无关(见 SCPALETTE 和 SCPALI)。

源程序(SCBORDER.C):

```

#include <dos.h>
#include <bscreens.h>
int scborder(color)
unsigned color;
{
    int result;
    int mode,act_page,columns;
    union REGS inregs,outregs;
    seequip(); /* Sense equipment and switch */
                /* settings. */
    scmode(&mode,&columns,&act_page); /* Retrieve current device */
                /* (b_device). */
    if (mode == 7) /* Rule out monochrome text mode*/
        result = SC_BAD_OPT;
    else if ( b_device == b_ega
        || b_device == b_vga
        || b_pmodel == IBM_JR)
    {
        /* Rule out Enhanced Color */
        /* Display 350-line modes. */
        if ( (mode <= 3 || mode == 15 || mode == 16)
            && b_device == b_ega
            && (b_sw_ega == 3 || b_sw_ega == 9))
            result = SC_BAD_OPT;
        else
        {
            /* Use BIOS function. */
            inregs.x.ax = 0x1001;
            inregs.h.bh = (unsigned char) color;
            int86(SC_BIOS_INT,&inregs,&outregs);
            result = SC_NO_ERROR;
        }
    }
    else
    {
        /* Use the CGA BIOS function. */
        inregs.h.ah = 0x0b;
        inregs.h.bh = 0;
        inregs.h.bn = (unsigned char) color;
        int86(SC_BIOS_INT,&inregs,&outregs);
        result = SC_NO_ERROR;
    }
    return result;
}

```

## SCBOX 用图形字符在屏幕上画一个方框

```

#include <bscreens.h>
int scbox( int u_row, int u_col,
           int l_row, int l_col,
           int boxtype,
           char boxchar,
           int attrib);

```



u\_row, u\_col 框的上行左列。  
 l\_row, l\_col 框的下行右列。  
 boxtype 方框字符的类型: 如果使用 boxchar, 为-1; 如果使用 IBM 方框字符(见下面), 则为 0 到 15。  
 boxchar 如果 boxtype 为-1, 则 boxchar 是用于绘制方框的字符。  
 attrib 使用的属性。低字节的高四位描述前景属性, 低字节的低四位描述背景属性。  
 (返回) 错误代码: 如果方框尺寸非法, 返回 1; 如果没有错误, 返回 0。

SCBOX 在当前显示页上画一个方框。

如果某个角超出了屏幕、方框的高度或宽度小于两个字符或者左上角(u\_row, u\_col)在右下角(l\_row, l\_col)的下方或右方, 则方框的尺寸非法。如果出现这种情况, 1 作为函数值返回。

如果 boxtype 是-1, 则使用 boxchar 指定的字符画方框; 如果 boxtype 在范围 0 到 15 内, 则使用 IBM PC 字符集的方框字符绘制该方框。根据下表, boxtype 的实际值指明方框的边线是单线是双线。

SCBOX 的方框类型码

方框	类型	下	右	上	左
0	(0000)	单	单	单	单
1	(0001)	单	单	单	单
2	(0010)	单	单	双	单
3	(0011)	单	单	双	双
4	(0100)	单	双	单	单
5	(0101)	单	双	单	双
6	(0110)	单	双	双	单
7	(0111)	单	双	双	双
8	(1000)	双	单	单	单
9	(1001)	双	单	单	双
10	(1010)	双	单	双	单
11	(1011)	双	单	双	双
12	(1100)	双	双	单	单
13	(1101)	双	双	单	双
14	(1110)	双	双	双	单
15	(1111)	双	双	双	双

如果选择了上表中的一个方框类型, 函数自动选择角字符与边线匹配。

源程序(SCBOX.C):

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

/* Macro to write a single character at a given point on the */
/* screen */
#define write_one(row,col,ch,fore,back) \
{ \
    scurset(row,col); \
    scattrib(fore,back,ch,1); \
}

/* Macro to write more than one character at a given point on the */
/* screen */
#define write_some(row,col,ch,fore,back,number) \
{ \
    if (number) \
    { \

```

```

        sccurset(row,col);
        scattrib(fore,back,ch,number);
    }
}

int sbox(u_row,u_col,l_row,l_col,boxtype,boxchar,attrib)
int u_row,u_col,l_row,l_col,boxtype;
char boxchar;
int attrib;
{
    int mode,act_page,columns; /* Returned by SCMODE */
    int save_row,save_col,high,low;
    int row,width,fore,back;
    char topleft,topright,botleft,botright,
        top,bottom,left,right; /* Characters to write */
    /* Tables of box characters */
    static char upleft[] /* Top left corners */
        = {'\332','\326','\325','\311'},
        upright[] /* Top right corners */
        = {'\277','\270','\267','\273'},
        lowleft[] /* Bottom left corners */
        = {'\300','\323','\324','\310'},
        lowright[] /* Bottom right corners */
        = {'\331','\275','\276','\274'};
    static char horiz[] /* @horizontal lines */
        = {'\304','\315'},
        vert[] /* Vertical lines */
        = {'\263','\272'};
    /* Validate the box position and dimensions */
    semode(&mode,&columns,&act_page);
    if (
        0 > u_row
        || u_row >= l_row
        || l_row >= scrows()
        || 0 > u_col
        || u_col >= l_col
        || l_col >= columns
    )
        return 1;
    width = l_col - u_col - 1; /* Distance between the sides */
    if ((mode > 3) && (mode != 7)) /* Force attribute for graphics */
        attrib = min(3,max(0,attrib));
    fore = utlonyb(attrib);
    back = uthinyb(attrib);
    /* Look up the box characters from the tables */
    if (boxtype < 0)
        topleft = topright = botleft = botright
            = left = right = top = bottom = boxchar;
    else
    {
        topleft = upleft [ boxtype & 3];
        topright = upright [ (boxtype >> 1) & 3];
    }
}

```

```

    botleft = lowleft [((boxtype >> 2) & 2) | (boxtype & 1)];
    botright = lowright[ (boxtype >> 2) & 3];

    top      = horiz[(boxtype >> 1) & 1];
    bottom   = horiz[(boxtype >> 3) & 1];
    left     = vert [ boxtype      & 1];
    right    = vert [(boxtype >> 2) & 1];
}

/* Draw the box, starting from upper left corner, working      */
/* horizontally.                                              */

/* Save cursor location */
scurst(&save_row,&save_col,&high,&low);
write_one (u_row,u_col,      topleft, fore,back);
write_some(u_row,(u_col + 1),top,    fore,back,width);
write_one (u_row,l_col,      topright,fore,back);
for (row = u_row + 1; row < l_row; row++)
{
    write_one(row,u_col,left, fore,back);    /* Left side */
    write_one(row,l_col,right,fore,back);    /* Right side */
}
write_one (l_row,u_col,      botleft, fore,back);
write_some(l_row,(u_col + 1),bottom, fore,back,width);
write_one (l_row,l_col,      botright,fore,back);
scurset(save_row,save_col);    /* Restore cursor location */
return 0;
}

```

## SCCHGDEV 切换至彩色或单色显示

```
#include <bcreens.h>
int scchgdev(int device);
device      待选择的设备(SC_MONO (0) 或 SC_COLOR (1))。
(返回)      返回的错误代码；如果设备不存在必须用 SCNEWDEV 重量，返回 1；如果没有错误，
返回 0。
```

SCCHGDEV 不经过重置就能够为以后的 I/O 选择显示适配器，这个过程包括测试所需设备是否存在、设置 BIOS 装备字、恢复 BIOS 显示状态变量、恢复全局变量 b\_device 和 b\_curpage，从而指明保存在新设备上的当前项。

程序启动时的当前设备可以在任何时候用 SCCHGDEV 或 SCNEWDEV 来选择。然而，其它设备(如果存在)在被 SCCHGDEV 选择之前必须至少要重置一次，否则 SCCHGDEV 报告一个错误。

如果所需设备不是当前设备，则 SCCHGDEV 用保存在一个内部表中的值恢复 BIOS 的内部显示状态变量。

源程序(SCCHGDEC.C):

```
#include <dos.h>
#include <bcreens.h>
extern int b_vidpy(int,int);    /* Internal (see SCNEWDEV.C). */
#define FROM_BIOS      0        /* Direction flags for copying */
#define TO_BIOS        1        /* BIOS video state variables. */

```

```

#define VIDEO_FIELD 0x0030          /* The bits to be modified in */
                                   /* the BIOS equipment word. */

#define MONO_FLAG 0x0030
#define COLOR_80 0x0020
#define COLOR_40 0x0010

                                   /* Address of BIOS equipment word. */
#define EQUIP_WORD_LOC (utfar(0x0040,0x0010,unsigned int))

int scchgdev(device)
int device;
{
    int old_mode,old_columns,old_act_page; /* Previous state */
    int old_dev;
    union REGS regs;
#define our_word (regs.x.ax) /* Our copy of equipment word. */
    int old_flag,new_flag; /* Old and new values of video */
                                   /* device bit field. */

    if (device != SC_MONO && device != SC_COLOR)
        return 1; /* Unknown device. */

    /* Note current state of BIOS. */
    old_dev = scmode(&old_mode,&old_columns,&old_act_page);
    if (device == old_dev)
        return 0; /* Nothing to do. */

    /* Record current BIOS variables in case we ever need to change */
    /* back to the old device. */
    if (b_videpy(FROM_BIOS,old_dev))
        return 1;

    if (!b_adap_state[device].known) /* Abort if we've never */
        return 1; /* recorded the BIOS state */
                                   /* variables for new device. */

    scequip(); /* Sense which hardware is */
                                   /* present. */

    switch (device)
    {
        case SC_COLOR:
            if ( b_cga != SC_COLOR && b_pcmodel != IBM_JR
                && b_ega != SC_COLOR && b_pgc != SC_COLOR
                && b_vga == SC_ABSENT && b_mcga == SC_ABSENT)
                return 1; /* Color/Graphics Adapter */
                                   /* functions are not available. */

            new_flag = COLOR_80;
            break;

        case SC_MONO:
            if ( b_mdpa != SC_MONO
                && b_ega != SC_MONO
                && b_vga == SC_ABSENT)
                return 1; /* Monochrome Adapter functions */
                                   /* are not available. */

            new_flag = MONO_FLAG;
            break;
    }
}

```

```

}

b_device = device;
/* If we are switching VGA from color to monochrome or vice */
/* versa, note the change in our global variables. */
if ( device != b_mdpa && device != b_cga && device != b_cga
    && device != b_pgc)
{
    if (b_vga == old_dev)
        b_vga = device;
}
/* The BIOS keeps track of the color vs. monochrome distinction */
/* by two bits in the equipment word, so we must poke those bits */
/* to signal the change in device. */
int86(17,&regs,&regs); /* Retrieve equipment word. */

old_flag = (our_word & VIDEO_FIELD);
if (new_flag != old_flag /* There is a change. */
    && (old_flag != COLOR_40 || new_flag != COLOR_80))
    /* (Skip the change if 40-col */
    /* mode is set and we're */
    /* setting color.) */
{
    /* Install modified equipment */
    /* word. */
    utpokeb(EQUIP_WORD_LOC,
        (unsigned char) ((our_word & ~VIDEO_FIELD) | new_flag));
}
/* Restore BIOS variables that were in effect when we last quit */
/* this device. */
b_vidcpy(TO_BIOS,b_device);
scpage(b_adap_state[device].curpage); /* Restore current page. */
return 0;
}

```

## SCCLRMSG 清除屏幕上的消息

```

#include <bcreens.h>
int scclrmsg( int row,
              int col,
              int len);
row, col    待清除的位置(0,0 是屏幕的左上角)。
len         待清除的消息的长度。
(返回)      返回值总是 0。

```

这个函数在当前显示页上的指定位置清除一个消息而不改变屏幕的显示属性，光标位置保持不变。

在图形方式下，被清除区域不能超出当前行而发生折转；在字符方式下，被清除区域可以折转超出当前行(但不能超出屏幕的末端)。SCCLRMSG 不滚动屏幕。

源程序(SCCLRMSG.C):

```

#include <bcreens.h>

```

```

#define BLANK ' '
int sccrmsg(row,col,len)
int row,col,len;
{
    int c_row,c_col,high,low;

    sccurst(&c_row,&c_col,&high,&low);/* Current position.          */
    sccurset(row,col);
    sewrite((char) BLANK,len);      /* Write len blanks.      */
    sccurset(c_row,c_col);          /* Return the original position.*/
    return(0);
}

```

## SCCURSET 移动当前显示页上的光标

```

#include <bscreens.h>
int sccurset( int row,
              int col);
row          行位置(0=屏幕顶行)
col          列位置(0 = 最左列)
(返回)       结果位置。行在高八位，列在低八位。如果所需位置超出范围，结果位置可能与所需位置不相同。

```

SCCURSET 将光标移到由 row 和 col 指定的当前显示页的一个位置。

为使光标移动时可见，当前页必须是活动(即当前显示)。如果当前页是不活动，则当该页呈活动状态时，新光标位置将会出。

如果所需位置超出范围，则位置将位于屏幕边缘上，尽可能与所需位置接近。

源程序(SCCURSET.C):

```

#include <dos.h>
#include <bscreens.h>
int sccurset(row,col)
int row,col;
{
    union REGS inregs,outregs;      /* Registers for BIOS call */
    int mode,columns,act_page;
    scmode(&mode,&columns,&act_page);
    utbound(row,0,scrows() - 1)     /* Force reasonable values */
    utbound(col,0,columns-1)
    inregs.h.ah = 0x02;              /* Set up call to the BIOS */
    inregs.h.bh = (unsigned char) b_curpage;
    inregs.h.dh = (unsigned char) row;
    inregs.h.dl = (unsigned char) col;
    int86(16,&inregs,&outregs);
    return (int) inregs.x.dx;
}

```

## SCCURST 返回当前显示页上的光标位置和尺寸

```

#include <bscreens.h>
int sccurst( int *prow,

```

```

        int *pcol,
        int *phigh,
        int *plow);
prow    指向返回的行值(0=屏幕顶行)。
pcol    指向返回的列值(0=最左列)。
phigh   指向返回的高扫描行。
plow    指向返回的低扫描行。
(返回)  如果光标关闭, 为 1; 若打开, 为 0。

```

SCCURST 返回当前显示设备和显示页上的光标位置及光标尺寸, 报告光标是打开还是关闭的。

这个函数询问 ROM BIOS 有关光标状态的信息。有些 IBM 程序改变了光标尺寸但不通知 BIOS, 所以 BIOS 报告的光标尺寸也许不正确。SCCURST 也面临着这种不准确性。

Turbo C Tools 和 BIOS 都不记录非活动页的光标尺寸和打开/关闭状态, 也就是说, 活动页的光标尺寸也适用于非活动页。

源程序(SCCURST.C):

```

#include <dos.h>
#include <bscreens.h>
int securst(prow,pcol,phigh,plow)
int *prow,*pcol,*phigh,*plow;
{
    union REGS inregs,outregs;      /* Registers for BIOS call */
    inregs.h.ah = 0x03;              /* Set up the call to the BIOS */
    inregs.h.bh = (unsigned char) b_curpage;
    int86(SC_BIOS_INT,&inregs,&outregs);
    *prow = outregs.h.dh;
    *pcol = outregs.h.dl;
    *phigh = utlonyb(outregs.h.ch);  /* Use actual values. */
    *plow = utlonyb(outregs.h.cl);

    return ((outregs.h.ch & 0x60) != 0);
}

```

## SCEQUIP 检测显示硬件环境

```
#include <bscreens.h>
```

```
char scequip();
```

(返回) IBM 型号代码, 正象 UTMODEL 返回的那样。

SCEQUIP 取得有关安装的显示适配器及其可选设置的一般信息。下面的全局变量将被置位:

变量	头文件	说明
b_know_hw	bscreens.h	设置成 1, 表示 SCEQUIP 已经运行。
b_mdpa	bscreens.h	存在单色适配器(MDPA):SC_ABSENT (-2)或 SC_MONO(0) (如果是 Hercules 图形适配器, 则为 SC)MONO。)
b_cga	bscreens.h	存在 Color/Graphics Adapter: SC_ABSENT (-2) 或 SC_COLOR (1)。
b_ega	bscreens.h	Enhanced Graphics Adapter (EGA) 状态:SC_ABSENT (-2), SC_MONO (0) 或 SC_COLOR (1)。
b_vga	bscreens.h	Video Graphics Array (VGA)状态:SC_ABSENT (-2), SC_MONO (0) 或 SC_COLOR 的(1)。
b_mega	bscreens.h	Multi-Color Graphics Array (MCGA)状态:SC_ABSENT (-2), SC_MONO (0) 或 SC_COLOR (1)。
b_herc	bscreens.h	存在 Hercules 单色图形适配器:SC_ABSENT (-2) 或 SC_MONO (0)

b\_pgc            bscreens.h      Professional Graphics Controller 状态:SC\_ABSENT (-2), SC\_COLOR (1) 或 SC\_HIFUNC (2).

b\_mem\_ega       bscreens.h      EGA 上安装的内存, 以 1024 字节为单位: 64, 128 或 256.

b\_sw\_ega        bscreens.h      EGA 开关设置. 低位被置位表示 SW1 断开, 等.

b\_pemodel       butil.h          IBM 型号代码.

b\_submod        butil.h          IBM 子型号代码. 如果不存在, 则为 0.

b\_biosre        butil.h          IBM BIOS 修订版号. 如果不存在, 则为 0.

SCMODE、SCROWS 和 SCPAGES 可以取得在软件控制下能够修改的显示信息(如显示方式或 屏幕尺寸)。

源程序(SCEQUIP.C):

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

int b_know_hw = 0;                    /* Flag stating whether we have yet    */
                                     /* run SCEQUIP: 0 if no, 1 if yes */

int b_mdpa = SC_DONT_KNOW; /* Monochrome Display & Printer Adapter    */
                             /* SC_DONT_KNOW, SC_ABSENT, or SC_MONO    */
                             /* (SC_MONO if Hercules graphics adapter.) */
                             /*                                            */

int b_ega = SC_DONT_KNOW; /* Color/Graphics Monitor Adapter       */
                             /* SC_DONT_KNOW, SC_ABSENT, or SC_COLOR    */
                             /*                                            */

int b_ega = SC_DONT_KNOW; /* Enhanced Graphics Adapter: SC_DONT_KNOW, */
                             /* SC_ABSENT, SC_MONO or SC_COLOR        */
                             /*                                            */

int b_mega = SC_DONT_KNOW; /* Multicolor Graphics Array: SC_DONT_KNOW, */
                             /* SC_ABSENT, or SC_COLOR                */
                             /*                                            */

int b_vga = SC_DONT_KNOW; /* Video Graphics Array: SC_DONT_KNOW,    */
                             /* SC_ABSENT, SC_MONO or SC_COLOR        */
                             /*                                            */

int b_herc = SC_DONT_KNOW; /* Hercules monochrome graphics adapter    */
                             /* SC_DONT_KNOW, SC_ABSENT, or SC_MONO    */
                             /*                                            */

int b_pgc = SC_DONT_KNOW; /* Professional Graphics Controller:      */
                             /* SC_DONT_KNOW, SC_ABSENT, SC_COLOR, or    */
                             /* SC_HIFUNC                                */

int b_mem_ega = 0;                    /* Amount of memory installed on EGA in */
                                     /* 1024-byte units: 64, 128, 192, 256 */

unsigned b_sw_ega = 0xffff; /* Switch settings on EGA                */
                                     /* Low-order bit set means SW1 off, etc. */
                                     /* (Bit value 1 implies that            */
                                     /* corresponding switch is off.)        */

static volatile int waiter; /* Variable to assign to to wait for    */
                                     /* CGA and PGC to stabilize when using */
                                     /* fast processors.                    */

#define PGC_SEG 0xc600 /* Segment of memory-mapped PGC ports. */
                                     /* PGC port for presence test.        */

#define PGC PRES_BYTE (utfar(PGC_SEG,0x03db,unsigned char))
```



```

#define PGC_INDEX 0x03d4      /* Offset of PGC/CGA index port      */
#define PGC_IND_BYTE (utfar(PGC_SEG,PGC_INDEX,unsigned char))
#define WAIT
{
    waiter = 0;
    waiter += 1;
}

static void note_adapter(unsigned char,int); /* Internal function */
static int adap_present(int); /* Internal function */
static int pgc_present(void); /* Internal function */
static int pgc_emulating(void); /* Internal function */
static int here(void); /* Internal function */
/* Internal variables containing interim knowledge of adapter */
/* state and modes. */

#define _DONT_KNOW (-1)
#define _ABSENT 0
#define _ACTIVE 1
#define _ALTERNATE 2
#define _PRESENT 3
static int pgc_state = _DONT_KNOW;
static int vga_state;

char seequip()
{
    union REGS inregs,outregs; /* Registers for BIOS calls */
    unsigned color_or_mono,mem;
    int pgc_emul_tested;
    int mode,columns,act_page;
    if (b_know_hw)
    {
        scmode(&mode,&columns,&act_page); /* Check VGA state. */
        return b_pmodel; /* No need to do this work again */
    }

    if (utmodel() == IBM_JR) /* Fetch & save model code. */
    { /* This is a PCjr */
        b_mdpa = SC_ABSENT;
        b_cga = SC_ABSENT;
        b_eqa = SC_ABSENT;
        b_pgc = SC_ABSENT;
        b_vga = SC_ABSENT;
        b_mega = SC_ABSENT;
        b_herc = SC_ABSENT;
    }
    else if (b_pmodel == IBM_CV)
    { /* IBM PC Convertible acts like */
        b_mdpa = SC_MONO; /* both MDPA & CGA. */
        b_cga = SC_COLOR;
    }
}

```

```

b_ega = SC_ABSENT;
b_pgc = SC_ABSENT;
b_vga = SC_ABSENT;
b_mega = SC_ABSENT;
b_herc = SC_ABSENT;
}
else
{
/* Neither PCjr nor Convertible */

/* First test for VGA or MCGA: attempt BIOS video function 0x1a. */
inregs.x.ax = 0x1a00;
int86(16,&inregs,&outregs);
if (outregs.h.al == 0x1a)
{
b_mdpa = SC_ABSENT; /* Flag these devices as absent; */
b_cga = SC_ABSENT; /* note_adapter will change the */
b_ega = SC_ABSENT; /* value for devices actually found. */
b_mega = SC_ABSENT;
b_vga = SC_ABSENT;
b_pgc = SC_ABSENT;
b_mega = SC_ABSENT;
vga_state = _ABSENT;

/* Record the active adapter. */
note_adapter(outregs.h.bl, _ACTIVE);

/* Record the other adapter. */
note_adapter(outregs.h.bh, _ALTERNATE);
if (vga_state == _ABSENT)
b_vga = SC_ABSENT;
else
{
/* Use current mode to learn */
/* whether VGA is monochrome or */
/* color. */

inregs.h.ah = 0x0f;
int86(16,&inregs,&outregs);
if (outregs.h.al == 7 || outregs.h.al == 15)
/* Active adapter is SC_MONO. */
b_vga = (vga_state == _ACTIVE ? SC_MONO : SC_COLOR);
else
/* Active device is SC_COLOR. */
b_vga = (vga_state == _ACTIVE ? SC_COLOR : SC_MONO);
}
}
else
{
b_mega =
b_vga = SC_ABSENT;
}
}

/* Next test for the presence of an EGA: attempt BIOS video */
/* function 18, subfunction BL=0x10 ("Return EGA Information"). */

```

```

/* If any information returned is invalid, then the EGA must be */
/* absent. */
    if (b_ega != SC_ABSENT)
    {
        inregs.h.ah = 18;
        inregs.x.bx = 0xff10;
        inregs.x.cx = 0x000f;
        int86(16,&inregs,&outregs);
        color_or_mono = outregs.h.bh;
        mem           = outregs.h.bl;
        b_sw_ega      = outregs.h.cl;
        if (b_sw_ega >= 12 || color_or_mono > 1 || mem > 3)
            b_ega      = SC_ABSENT; /* Invalid value(s) found */
        else
        {
            /* EGA is present */
            b_ega      = (color_or_mono ? SC_MONO : SC_COLOR);
            b_mem_ega   = 64 * (mem + 1);
        }
    }

/* If the EGA, VGA, or MCGA is emulating the MDPA or the CGA, */
/* then the emulated board cannot itself be present, but the */
/* other might be. */
    if (b_mdpa == SC_DONT_KNOW)
        if (b_ega==SC_MONO || b_vga==SC_MONO || b_mcg==SC_MONO
            || !adap_present(0)) /* Sense monochrome adapter */
            b_mdpa = SC_ABSENT;
        else
            b_mdpa = SC_MONO;

/* Further, if we sense a color adapter, then exactly one of the */
/* following conditions must be true: EGA, VGA, or MCGA is in */
/* color mode; PGC is emulating the CGA; or a genuine CGA is */
/* present. */

    pgc_emul_tested = 0;
    if (b_cga == SC_DONT_KNOW)
    {
        if (b_ega==SC_COLOR || b_vga==SC_COLOR || b_mcg==SC_COLOR)
            b_cga = SC_ABSENT;
        else if (adap_present(1)) /* Sense color adapter */
        {
            if (pgc_emulating() == SC_COLOR)
            {
                b_pgc = SC_COLOR;
                b_cga = SC_ABSENT;
            }
            else
                b_cga = SC_COLOR;
            pgc_emul_tested = 1;
        }
    }

```

```

        else
            b_cga = SC_ABSENT;
    }
    /* If the PGC is present but not emulating the CGA, then it must */
    /* be in its high-function graphics mode. */
    if (b_pgc == SC_DONT_KNOW)
    {
        if (pgc_present())
            b_pgc = (pgc_emul_tested ? SC_HIFUNC : pgc_emulating());
        else
            b_pgc = SC_ABSENT;
    }
}
if (b_mdpa == SC_MONO && here()) /* Hercules adapter may be */
    b_here = SC_MONO; /* emulating monochrome. */
else
    b_here = SC_ABSENT;
b_know_hw = 1;
return b_pcmode;
}

```

```

/**
 * Name          note_adapter — Record installed adapters based on BIOS
 *                display combination code from video
 *                function 0x1a
 *
 * Synopsis      note_adapter(code,active_flag);
 *
 *                unsigned char code  Display code from BIOS video
 *                                   function 0x1a.
 *                int active_flag    _ACTIVE or _ALTERNATE indicating
 *                                   whether we are dealing with the
 *                                   active or alternate adapter.
 *
 * Description    This function extracts adapter presence information from
 *                a display combination code from BIOS function 0x1a.
 *
 * Result        b_mdpa, b_cga, b_ega, vga_state, b_mega, or pgc_state
 *                (Depending on which adapter is
 *                indicated by the code).
 */

```

```

**/
static void note_adapter(code,active_flag)
unsigned char code;
int          active_flag;
{
    switch (code)
    {
        case 0x01: b_mdpa    = SC_MONO;    break;
        case 0x02: b_cga     = SC_COLOR;   break;
    }
}

```

```

    case 0x04: b_ega      = SC_COLOR;    break;
    case 0x05: b_ega      = SC_MONO;     break;
    case 0x06: pgc_state = _PRESENT;     break;
    case 0x07:
    case 0x08: vga_state = active_flag; break;
    case 0x0b:
    case 0x0c: b_mega      = SC_COLOR;    break;
}
}
/**
 * Name          adap_present -- Test for Monochrome or Color/Graphics
 *               Adapter or emulator
 * Synopsis      present = adap_present(adapter);
 *               int present      1 if present or being emulated,
 *                               0 if not.
 *               int adapter      0 if testing for Monochrome Adapter,
 *                               1 if Color/Graphics Adapter
 * Description    This function tests for the presence of the IBM
 *               Monochrome Display and Printer Adapter or the IBM
 *               Color/Graphics Adapter or other device (such as the EGA
 *               or PGC) that may be emulating them.
 *               We do an assignment between the UTINP and UTOUTP calls
 *               so that the device has time to properly stabilize
 *               when using slow devices with fast processors.
 * Method        Write a bogus value into the cursor position register of
 *               the CRT controller chip, then read it back again. If
 *               the same value is found in the register, then the CRTC
 *               must be present; if the value is different, then it must
 *               be absent.
 * Result        present          1 if present or being emulated,
 *                               0 if not.
 */
static int adap_present(adapter)
int adapter;
{
    int      crtc_there;
    char      save_cursor;
    unsigned int port;      /* Address of the CRT controller chip */
    port = (adapter ? 0x03d4 : 0x03b4);
    utoutp(port++,0xf);     /* Set cursor register */
    WAIT
                           /* Save previous contents */
    save_cursor = (unsigned char) utinp(port);
    utoutp(port,0x5a);     /* Set to column 90 */
    WAIT
    crtc_there = (utinp(port) == 0x5a); /* Read the value back again */
    WAIT
    utoutp(port,save_cursor); /* Restore previous contents */
    return crtc_there;
}

```

```

/**
 * Name          pgc_present -- Test for IBM Professional Graphics
 *                  Controller
 * Synopsis      present = pgc_present();
 *                  int present      1 if present, 0 if not.
 * Description   This function tests for the presence of the IBM
 *                  Professional Graphics Controller.
 * Method        Write a bogus value into the supposed PGC, then read it
 *                  back again. If the same value is found, then the PGC
 *                  must be present; if the value is different, then it must
 *                  be absent.
 * Result        present      1 if present, 0 if not.
 **/
static int pgc_present()
{
    int result;
    char save,dummy;
    switch (pgc_state)
    {
        case _ABSENT:
            result = 0;
            break;
        case _PRESENT:
            result = 1;
            break;
        case _DONT_KNOW:
            save = utpeekb(PGC_PRES_BYTE); /* Save previous contents */
            WAIT
            utpokeb(PGC_PRES_BYTE,0x5a); /* Store dummy value */
            WAIT
            dummy = utpeekb(PGC_PRES_BYTE); /* Read it back again */
            WAIT
            utpokeb(PGC_PRES_BYTE,save); /* Restore previous contents */
            result = (dummy == 0x5a);
            break;
    }
    return result;
}

```

```

/**
 * Name          pgc_emulating -- Test for whether IBM Professional
 *                  Graphics Controller is emulating the IBM
 *                  Color/Graphics Adapter
 * Synopsis      emulating = pgc_emulating();
 *                  int emulating      SC_COLOR if PGC emulating the CGA,
 *                  SC_HIFUNC if not.
 * Description   This function tests for whether the IBM Professional
 *                  Graphics Controller is emulating the IBM Color/Graphics
 *                  Adapter.
 * Method        Write a bogus value into the PGC through a memory

```

```

*          address, then read it back again through an input port.
*          If the same value is found, then the PGC must be present
*          and in CGA emulation mode; if the value is different,
*          then (if the PGC is indeed present) it must be in
*          high-function graphics mode.
* Result      present          SC_COLOR if PGC emulating the CGA,
*                  SC_HIFUNC if not.
**/

```

```

static int pgc_emulating()
{
    char save,dummy;

    save = utpeekb(PGC_IND_BYTE);      /* Save previous contents */
    WAIT
    utpokeb(PGC_IND_BYTE,0x28);        /* Store dummy value */
    WAIT
    dummy = (char) utinp(PGC_INDEX);    /* Read it back again */
    WAIT
    utpokeb(PGC_IND_BYTE,save);        /* Restore previous contents */

    return (dummy == 0x28) ? SC_COLOR : SC_HIFUNC;
}

```

```

/**
*
* Name          here - Sense presence of Hercules monochrome graphics
*                  adapter.
*
* Synopsis      present = here();
*                  int present      Zero if Hercules adapter absent,
*                                  nonzero if present.
* Description    This function reports whether the Hercules monochrome
*                  graphics adapter is present.
*
* Result        present          Zero if Hercules adapter absent,
*                                  nonzero if present.
**/

```

```

static int here()
{
    unsigned char save;
    unsigned int i;
    int          result;

    result = 0;
    save = (unsigned char) (utinp(0x3ba) & 0x80);
    for (i = 0; i < (unsigned int) 32768L; i++)
        if (save != (unsigned char) (utinp(0x3ba) & 0x80))
        {
            result = 1;

```

```

        break;
    }
    return result;
}

```

## SCGETVID 记录整个显示状态

```
#include <bcreens.h>
```

```
void scgetvid(ADAP_STATE *pstate);
```

pstate 结构的地址, 该结构返回适配器的状态。

SCGETVID 记录当前显示适配器的全部状态(但不包括显示数据), 信息在 ADAP\_STATE 结构中返回。该结构定义如下:

```

typedef struct          /*CUR_TYPE 结构 */
{
    int high,low;       /*光标的高低扫描行。*/
} CUR_TYPE;
typedef struct          /*ADAP_STATE 结构;*/
{
    int mode;           /*当前显示方式。*/
    int act_page;       /*活动(显示页)。*/
    int cur_page;       /*当前页。*/
    int rows,columns;   /* 屏幕尺寸。*/
    int curs_off;       /*光标打开或关闭。*/
    CUR_TYPE curs_size; /*光标的高低扫描行。*/
} ADAP_STATE;

```

SCSETVID 可以用保存在 ADAP\_STATE 结构中的信息恢复显示适配器的状态, SCSAVEPG 和 SCRESTPG 可以保存和恢复显示的正文数据。

源程序(SCGETVID.C):

```

#include <bcreens.h>
void scgetvid(pstate)
ADAP_STATE *pstate;
{
    int row, column;
    scmode(&(pstate->mode), &(pstate->columns), &(pstate->act_page));
    pstate->cur_page = b_curpage;
    pstate->rows = scrows();
    pstate->curs_off = scurst(&row, &column, &(pstate->curs_size.high),
                             &(pstate->curs_size.low));
}

```

## SCMODE 返回屏幕的显示方式

```
#include <bcreens.h>
```

```
int scmode( int *pmode,
```

```
           int *pcolumns,
```

```
           int *papage);
```

pmode 变量的指针, 该变量返回屏幕的方式。

pcolumns 变量的指针, 该变量返回屏幕的列数。



papage 变量的指针, 该变量返回活动页号。

(返回) 返回代码表示当前的显示方式:

Color/Graphics Adapter (或 EGA, VGA, MCGA, PGC 模拟 CGA) 返回 SC\_COLOR(1)。

Monochrome Adapter (或 EGA, VGA, MCGA 模拟 Monochrome Adapter) 返回 SC\_MONO

(0)。

SCMODE 返回当前的显示方式设置、屏幕行数和活动显示页。在显示方式表中列出了可能的方式值。函数值指示当前适配器被设置成单色适配器还是彩色适配器。

这个函数还根据使用的设备将全局变量 b\_device(在 bscreen.h 中声明)设置成 SC\_COLOR(1)或 SC\_MONO(0)。

源程序(SCMODE.C):

```
#include <dos.h>
#include <bSCREENS.h>
int b_device = SC_DONT_KNOW;      /* Current adapter: SC_DONT_KNOW, */
                                  /* SC_MONO, or SC_COLOR */
#define VIDEO_FIELD 0x0030        /* The bits to be examined in */
                                  /* the BIOS equipment flag. */
#define MONO_FLAG 0x0030
#define COLOR_80 0x0020
#define COLOR_40 0x0010
int scmode(pmode, pcolumns, papage)
int *pmode, *pcolumns, *papage;
{
    union REGS inregs, outregs;    /* Registers for BIOS calls */

    inregs.h.ah = 15;
    int86(SC_BIOS_INT, &inregs, &outregs);
    *pmode = (outregs.h.al & 0x7f); /* Discard bit 7 (sign bit) */
    *pcolumns = outregs.h.ah;      /* of mode. */
    *papage = outregs.h.bh;

    int86(17, &inregs, &outregs); /* Equipment setting determines */
                                  /* which adapter is used. */

    switch (outregs.x.ax & VIDEO_FIELD)
    {
        case MONO_FLAG:
            b_device = SC_MONO;
            break;
        case COLOR_80:
        case COLOR_40:
        default:
            /* Impossible bit settings. */
            b_device = SC_COLOR;
            break;
    }

    /* Check VGA/MCGA state. */

    inregs.x.ax = 0x1a00;
    int86(SC_BIOS_INT, &inregs, &outregs);
    if (outregs.h.al == 0x1a)
```

```

{
    /* Function 0x1a is supported. */
    switch (outregs.h.bl) /* Handle active device. */
    {
        case 0x07:
        case 0x08: b_vga = b_device; break;

        case 0x0b:
        case 0x0c: b_mega = b_device; break;
    }
    switch (outregs.h.bh) /* Handle alternate device. */
    {
        case 0x07:
        case 0x08:
            b_vga = (b_device == SC_MONO ? SC_COLOR : SC_MONO);
            break;

        case 0x0b:
        case 0x0c:
            if (b_device == SC_MONO)
                b_mega = SC_COLOR;
            break;
    }
}
return b_device;
}

```

## SCMODE4 设置方式 4 色板和背景颜色

```
#include <bsrccons.h>
```

```
int scmode4( int palette,
```

```
            int backgrd);
```

palette      方式 4 下的前景色板:

0    绿/红/黄;

1    淡绿/紫红/白。

backgrd      背景颜色(0-31)。

(返回)      错误代码。可能值包括:

SC\_NO\_ERROR      (0)    成功。

SC\_BAD\_OPT      (1)    当前显示适配器不是在方式 4 下。

SCMODE4 设置显示方式 4 (320x200 4 色图形) 下的画点色板和背景颜色。

在方式 4 下, 点是通过一个值 1、2 或 3 来指定一个前景颜色而显示出来的, 显示的实际颜色由所选的色板决定:

点值:      0      1      2      3

色板 0: (背景) 绿    红    黄

色板 1: (背景) 淡绿 紫红 白

颜色 0 使点变成与背景同样的颜色, 该点被“删除”。

如果当前显示适配器不是在方式 4 下, 则出现一个错误。

源程序(SCMODE4.C):

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

```
#include <bsrccons.h>
```

```

int scmode4 (palette, backgrd)
int palette, backgrd;
{
    union REGS regs;          /* Registers for int86() call. */
    int mode;                 /* Video mode. */
    int cols;                 /* Number of columns on screen. */
    int apage;                /* Active display page. */

    scmode (&mode, &cols, &apage);

    if (mode == 4)
    {
        regs.h.ah = 0x0b;      /* Go do interrupt to set palette. */
        regs.h.bh = 1;
        regs.h.bl = (unsigned char) palette;
        int86 (SC_BIOS_INT, &regs, &regs);

        regs.h.ah = 0x0b;      /* Go do interrupt */
        regs.h.bh = 0;          /* to set background */
        regs.h.bl = (unsigned char) backgrd; /* and intensity. */
        int86 (SC_BIOS_INT, &regs, &regs);

        return (0);
    }
    return (-1);               /* Mode not equal to 4. */
}

```

## SCNEWDEV 选择并设置显示设备，设置字符行数

```
#include <bcreens.h>
```

```
int scnewdev( int mode,
```

```
int num_rows);
```

mode 新方式值。如果使用已有的方式和设备，则为-1。

num\_rows 设备上新的字符行的数目(25、30、43、或50)。

(返回) 错误代码：如果不支持所需的方式或字体，返回-1；若无错误，返回 mode。

SCNEWDEV 选择并重置显示适配器，选择 25-、30-、43-或 50 行内设字体(如果可能)。适配器上所有的显示页被清空。它还选择合适的打印屏幕例程，设置全局变量 b\_device 和 b\_curpage。

可能的方式值和它们的硬件需求列在显示方式表中。

如果安装的硬件不支持这个方式或所需设备不支持所需的字体，则出现一个错误。在这两种情况下不做任何操作，-1 作为函数值返回。

源程序(SCNEWDEV.C):

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

```
#include <bcreens.h>
```

```
int b_vidcopy(int,int);
```

```
/* Internal (see below) */
```

```
#define FROM_BIOS 0 /* Direction flags for copying */
```

```
#define TO_BIOS 1 /* BIOS video state variables. */
```

```
#define VIDEO_FIELD 0x0030 /* The bits to be modified in */
```

```
/* the BIOS equipment word. */
```

```
#define MONO_FLAG 0x0030
```

```

#define COLOR_80      0x0020
#define COLOR_40      0x0010
int b_vidcpy(int,int);          /* Internal (see below) */
#define FROM_BIOS      0          /* Direction flags for copying */
#define TO_BIOS        1          /* BIOS video state variables. */
                                   /* Address of BIOS equipment word. */
#define EQUIP_LOC      (utfar(0x0040,0x0010,unsigned int))
                                   /* Address of BIOS INFO byte. */
#define INFO_LOC       (utfar(0x0040,0x0087,unsigned char))
int scnewdev(mode,num_rows)
int mode,num_rows;
{
    int old_dev,old_mode,old_act_page,old_cols;
    int device;
    union REGS inregs,outregs;

    int our_word;                /* Our copy of equipment word. */
    int old_flag,new_flag;       /* Old and new values of video */
                                   /* device bit field. */
    char old_info;               /* Our copy of former INFO byte */
    char new_info;
    /* Note current state of BIOS. */
    old_dev = scmode(&old_mode,&old_cols,&old_act_page);
    /* Check validity of mode and font requested */
    scequip();                   /* Sense hardware settings. */
    if (mode == -1)
        mode = old_mode;        /* Use existing mode */
    switch (mode)
    {
        /* Determine requested device. */
        /* PCjr color modes: */
        case 8:
        case 9:
        case 10:
            if (b_pmodel != IBM_JR)
                return -1;
                                   /* CGA color modes: */
        case 4:
        case 5:
        case 6:
            if (num_rows != 25)
                return -1;
        case 0:
        case 1:
        case 2:
        case 3:
                                   /* This is a color mode, */
                                   /* so check for color device. */
            if ( b_cga != SC_COLOR && b_pmodel != IBM_JR
                && b_ega != SC_COLOR && b_pgc != SC_COLOR
                && b_vga == SC_ABSENT && b_mega == SC_ABSENT)

```

```

        return -1;                /* Color/Graphics Adapter */
                                   /* functions are not available. */

    device    = SC_COLOR;
    new_flag  = COLOR_80;
    break;

                                   /* EGA-only color modes: */

case 13:
case 14:
    if (num_rows != 25)
        return -1;
case 16:
    if ( num_rows > 43
        || (b_ega != SC_COLOR && b_vga == SC_ABSENT))
        return -1;
    device    = SC_COLOR;
    new_flag  = COLOR_80;
    break;

                                   /* Monochrome modes: */

case 15:
    if ( num_rows > 43
        || (b_ega != SC_MONO && b_vga == SC_ABSENT))
        return -1;
case 7:
    /* This is a monochrome mode, */
    /* so check for monochrome device. */
    if (b_mdpa != SC_MONO && b_ega != SC_MONO && b_vga == SC_ABSENT)
        return -1;
    /* Monochrome Adapter functions */
    /* are not available. */

    new_flag  = MONO_FLAG;    /* Monochrome functions found. */
    device    = SC_MONO;
    break;

                                   /* VGA/MCGA modes: */

case 18:
    if (b_vga == SC_ABSENT)
        return -1;
case 17:
    if ( (num_rows != 30 && num_rows != 60)
        || (b_vga == SC_ABSENT && b_mega == SC_ABSENT))
        return -1;
    device    = SC_COLOR;
    new_flag  = COLOR_80;
    break;

case 19:
    if ( num_rows != 25

```

```

        || (b_vga == SC_ABSENT && b_mega == SC_ABSENT))
        return -1;
    device = SC_COLOR;
    new_flag = COLOR_80;
    break;

default:
    return -1;                /* Unknown mode. */
}

if (device == b_mdpa && mode != 7)
    return -1;
if ( (device == b_cga || device == b_pgc)
    && mode > 6)
    return -1;

switch (num_rows)
{
    case 25:                    /* 25-line mode: */
        break;                /* no further checking needed. */

    case 30:                    /* 30-line and 60-line require */
    case 60:                    /* mode 17 or 18. */
        if (mode != 17 && mode != 18)
            return -1;
        break;

    case 43:                    /* 43-line mode: */
        if ( device == b_mdpa    /* can't run on MDPA */
            || device == b_cga    /* or on CGA */
            || device == b_pgc    /* or on PGC in CGA mode */
            || b_mega != SC_ABSENT /* or if MCGA present; */
            || ( device == b_ega    /* if on EGA */
                && device == SC_COLOR /* in color mode, */
                && b_sw_ega != 0x9    /* must have ECD in */
                && b_sw_cga != 0x3)) /* high-res mode. */
            return -1;
        break;

    case 50:                    /* 50-line mode requires */
                                /* VGA or MCGA. */
        if ( device == b_mdpa
            || device == b_cga
            || device == b_ega
            || (b_vga == SC_ABSENT && b_mega == SC_ABSENT))
            return -1;
        break;

default:
    return -1;                /* Unknown number of lines. */
}

```

```

}
/* Validation complete: now switch devices if necessary */
/* Record current BIOS variables in case we ever need to change */
/* back to the old device. */
if (b_vidcopy(FROM_BIOS,old_dev))
    return -1;
/* If we are switching VGA from color to monochrome or vice */
/* versa, note the change in our global variables. */
if ( device != b_mdpa && device != b_ega && device != b_ega
    && device != b_pgc)
{
    if (b_vga == old_dev)
        b_vga = device;
}

/* For 43-line text modes on VGA, set scan lines to 350. */
/* However, this works only if VGA is the former device. */
if ( device == b_vga
    && device == old_dev
    && ((0 <= mode && mode <= 3) || mode == 7))
{
    inregs.h.ah = 0x12; /* Alternate select. */
    inregs.h.bl = 0x30; /* Scan lines. */
    inregs.h.al = (unsigned char)
        (num_rows == 43 ? 1 /* 350 scan lines. */
         : 2); /* 400 scan lines. */
    int86(SC_BIOS_INT,&inregs,&outregs);
}

/* The BIOS keeps track of the color vs. monochrome distinction */
/* by two bits in the equipment word, so we must poke those bits */
/* to signal the change in device. */
int86(17,&inregs,&outregs); /* Retrieve equipment word. */
our_word = outregs.x.ax;
old_flag = (our_word & VIDEO_FIELD);
if (new_flag != old_flag /* There is a change. */
    && (old_flag != COLOR_40 || new_flag != COLOR_80))
{
    /* (Skip the change if 40-col */
    /* mode is set and we're */
    /* setting color.) */
    /* Install modified equipment */
    /* word. */
    utpokeb(EQUIP_LOC,
        (unsigned char) ((our_word & ~VIDEO_FIELD) | new_flag));
}
b_device = device;

/* Handle cursor compensation bit: */
/* The EGA BIOS adjusts the cursor type based on the lower bit of */

```

```

/* the variable called INFO. When the bit is 0, cursor types are */
/* automatically adjusted to fit in 14 scan lines; when the bit */
/* is 1, cursor types are used as requested. The BIOS does not */
/* maintain this bit. Therefore we set this bit when in 43-line */
/* mode and clear it for 25-line mode. */
if (b_ega == device || b_vga == device || b_mega == device)
{
    /* Fetch INFO byte. */

    old_info = utpeekb(INFO_LOC);
    new_info = old_info;
    if (num_rows >= 43)
        new_info |= 1; /* Set compensation bit */
                        /* (turn compensation off). */
    else
        new_info &= (~1); /* Clear compensation bit */
                        /* (turn compensation on). */
    if (new_info != old_info) /* If there is a change, */
                            /* install the new INFO byte. */
        utpokeb(INFO_LOC, new_info);
}

/* Reset new device */
inregs.h.ah = 0;
inregs.h.al = (unsigned char) mode;
int86(SC_BIOS_INT, &inregs, &outregs);
/* For 43-line text modes on VGA, set scan lines to 350, unless */
/* we were able to do so before resetting the mode. Then reset */
/* the mode again. */
if ( device == b_vga && device != old_dev
    && ((0 <= mode && mode <= 3) || mode == 7)
    && scrows() != num_rows)
{
    inregs.h.ah = 0x12; /* Alternate select. */
    inregs.h.bl = 0x30; /* Scan lines. */
    inregs.h.al = (unsigned char)
        (num_rows == 43 ? 1 /* 350 scan lines. */
         : 2); /* 400 scan lines. */
    int86(SC_BIOS_INT, &inregs, &outregs);

    inregs.h.ah = 0; /* Reset device again. */
    inregs.h.al = (unsigned char) mode;
    int86(SC_BIOS_INT, &inregs, &outregs);
}

/* Load small font if necessary. */
if (num_rows >= 43)
{
    if (mode > 3 && mode != 7)
        inregs.x.ax = 0x1123; /* Graphics mode. */
    else

```



```

        inregs.x.ax = 0x1112;    /* Text mode. */
        inregs.h.bl = 0;
        inregs.h.dl = (unsigned char) num_rows;
        int86(SC_BIOS_INT,&inregs,&outregs);

        inregs.h.ah = 0x12;      /* Select alternate print screen*/
        inregs.h.bl = 0x20;      /* routine. */
        int86(SC_BIOS_INT,&inregs,&outregs);
    }
    sepage(0);
    /* Record new BIOS variables in case SCCHGDEV ever wants to */
    /* change to the new device. */
    b_vidcpy(FROM_BIOS,device);
    return mode;
}

/**
 * Name          b_vidcpy -- Copy BIOS video state variables.
 * Synopsis      ercode = b_vidcpy(dir,device);
 *              int ercode      Result code:
 *
 *                      1 if dir or device invalid or if
 *                      table information is unavailable;
 *                      0 if ok.
 *              int dir          Direction:
 *
 *                      FROM_BIOS means from BIOS to table,
 *                      TO_BIOS  means from table to BIOS.
 *              int device       SC_MONO or SC_COLOR to denote which
 *                      entry of the b_adap_state table to use.
 * Description    This function copies the four blocks of BIOS video state
 *                variables to or from the table called b_adap_state
 *                (defined above).
 *
 *                If dir is TO_BIOS, the b_adap_state table entry is
 *                checked for valid data; if it is marked as invalid, then
 *                1 is returned as an error code.
 *
 *                An error is also reported if dir or device is invalid.
 * Returns        ercode      Result code:
 *
 *                      1 if dir or device invalid or if
 *                      table information is unavailable;
 *                      0 if ok.
 */

/* Internal structure for preserving BIOS video variables */
VIDEO_STATE b_adap_state[2] = {0}; /* One entry for SC_MONO, one */
/* for SC_COLOR. */

int b_vidcpy(dir,device)
int dir,device;
{
    int i;
    int were_on; /* Flag indicating whether interrupts */
                /* were on before we turned them off. */

```

```

/* There are four blocks of BIOS variables that describe the */
/* state of an adapter. Here are tables to help us copy them */
/* into the b_adap_state array: */

static char *save_area[2][4] = /* Pointers to the sections of */
{                               /* the VIDEO_STATE structures */
                               /* in the b_adap_state table */
    {
        b_adap_state[0].pc_area,
        b_adap_state[0].video_area,
        (char *) &b_adap_state[0].save_ptr,
        &b_adap_state[0].prtsc_status},

    {
        b_adap_state[1].pc_area,
        b_adap_state[1].video_area,
        (char *) &b_adap_state[1].save_ptr,
        &b_adap_state[1].prtsc_status}
};

static int save_size[4] = /* Sizes of the four sections */
{                          /* of a VIDEO_STATE structure */
    sizeof(b_adap_state[0].pc_area),
    sizeof(b_adap_state[0].video_area),
    sizeof(b_adap_state[0].save_ptr),
    sizeof(b_adap_state[0].prtsc_status)
};

static char far *bios_ptr[4] = /* Addresses of the four blocks */
{                               /* of BIOS variables */
    uttfar(0x0040,0x0049,char),
    uttfar(0x0040,0x0084,char),
    uttfar(0x0040,0x00a8,char),
    uttfar(0x0040,0x0100,char)
};

if ( (dir != FROM_BIOS && dir != TO_BIOS)
    || (device != SC_COLOR && device != SC_MONO))
    return 1; /* Invalid code. */
if (dir == TO_BIOS && !b_adap_state[device].known)
    return 1; /* Requested table info isn't */
              /* valid. */
were_on = utintoff(); /* Turn interrupts off. */
for (i = 0; i < 4; i++) /* For each of the four blocks, */
{
    if (dir == FROM_BIOS) /* Copy the block of data */
        utmovmem(bios_ptr[i], /* from BIOS*/
            (char far *) save_area[device][i],
            save_size[i]);
    else
        utmovmem((char far *) save_area[device][i], /* to BIOS */
            bios_ptr[i],
            save_size[i]);
}

```

```

if (were_on)
    utinton();                /* Turn interrupts back on.    */
if (dir == FROM BIOS)
{
    b_adap_state[device].curpage = b_curpage;
    b_adap_state[device].known   = 1; /* Flag table info as valid */
}
return 0;                    /* Success.                */
}

```

## SCPAGE 设置当前显示页

```
#include <bscreens.h>
```

```
int scpage (int page);
```

page 待设置成当前页的显示页号。

(返回) 实际设置的当前显示页。

SCPAGE 对指定了当前显示页的全局变量 `b_curpage` 进行设置(在 `bscreen.h` 中声明了 `b_curpage`)，它具有将所有 Turbo C TOOLS 屏幕输出定向到所需页的功能。这个页不必是活动的，也就是说，不必在当前显示。

由于 SCPAGE 不检查当前方式和当前设备，所以您有可能选择了一个不存在的页。用 SCPAGES 可以取得当前显示设备和当前显示方式下可用的页数。

SCAPAGE 显示一个指定的页，SCNEWDEV 或 `sechgdev` 选择一个给定的设备。

这个例程是用宏而不是用外部函数实现的。

源程序(SCPAGE.C):

```
#include <bscreens.h>
```

```
int b_curpage = 0;                /* Video page for Blaise C    */
                                   /* TOOLS screen I/O.          */

```

## SCPAGES 返回显示页的数目

```
#include <bscreens.h>
```

```
int scpages(void);
```

(返回) 显示页的数目。

SCPAGES 报告当前显示适配器在当前方式下能够支持的页的数目。对于不支持多页的显示方式，返回 1。例如在 Monochrome Adapter 上的方式 7。

用 SCAPAGE 可以显示任何指定的页。SCPAGE 能够使 Turbo C TOOLS 屏幕输出定向到任意一页，不论该页在当前是否显示。

源程序(SCPAGES.C):

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

```
#include <bscreens.h>
```

```
int scpages()
```

```

{
    int    pages;
    int    mode, columns, act_page;
    union REGS inregs, outregs;
    struct SREGS sregs;
    struct                                /* mytable: Table of mode    */

```

```

{
    /* information returned by VGA/ */
    /* MCGA function AH=0x1b, BX=0. */

    char        junk1[0x29];
    unsigned char pages;      /* Number of display pages. */
    char        junk2[0x16];
} mytable;

if (!b_know_hw)
    seequip();              /* Sense hardware present. */
pages = 0;                  /* In case of impossible */
                             /* conditions. */
semode(&mode,&columns,&act_page); /* Get VGA/MCGA state. */
if (b_vga == b_device || b_mega == b_device)
    switch (mode)
    {
        case 0:              /* Handle cases where BIOS is */
        case 1:              /* wrong. */
            pages = (scrows() > 43) ? 7 : 8;
            break;
        case 2:
        case 3:
        case 7:
            pages = (scrows() >= 43) ? 4 : 8;
            break;
        default:              /* Ask the BIOS. */
            inregs.h.ah = 0x1b;
            inregs.x.bx = 0;
            inregs.x.di = utoff(&mytable);
            sregs.es = utseg(&mytable);
            int86x(SC_BIOS_INT,&inregs,&outregs,&sregs);
            pages = mytable.pages;
            break;
    }
else if (b_ega == b_device)
    switch (mode)            /* Enhanced Graphics Adapter */
    {
        case 0:
        case 1:
            if (scrows() == 25)
                pages = 8;
            else              /* Presumable 43-line mode */
                pages = ((b_mem_ega == 64) ? 4 : 8);
            break;
        case 2:
        case 3:
        case 7:
            if (scrows() == 25)
                pages = ((b_mem_ega == 64) ? 4 : 8);
            else              /* Presumable 43-line mode */
                pages = b_mem_ega >> 5;
    }

```

```

        break;
    case 4:
    case 5:
    case 6:
        pages = 1;
        break;
    case 13:
        pages = b_mem_ega >> 5;
        break;
    case 14:
        pages = b_mem_ega >> 6;
        break;
    case 16:
        pages = (b_mem_ega == 256) ? 2 : 1;
        break;
    }
else
    switch (mode)
    {
        case 0:
        case 1:
            pages = 8;          /* Color/Graphics Adapter 40 col*/
            break;
        case 2:
        case 3:
            pages = 4;          /* Color/Graphics Adapter 80 col*/
            break;
        case 4:          /* CGA medium-res graphics */
        case 5:          /* CGA medium-res graphics */
        case 6:          /* CGA high-res graphics */
        case 7:          /* Monochrome Adapter */
        case 8:          /* PCjr low-res graphics */
        case 9:          /* PCjr medium-res graphics */
        case 10:         /* PCjr high-res graphics */
            pages = 1;
            break;
    }
return pages;
}

```

## SCPAL1 定义一个 EGA、VGA、或 MCGA 色板颜色

#include <bscreens.h>

int scpall( unsigned reg,

unsigned value);

reg 定义的色板寄存器(0-15。若为 16，则用于边界)。

value 显示的颜色值(见 SCPALETTE 的格式说明)。

(返回) 返回的错误代码。可能值包括:

SC\_NO\_ERROR (0) 成功。

SC\_BAD\_OPT (1) 当前显示适配器不支持这个操作。

SC\_RANGE (2) reg 超出了 0-15 的范围。

SCPAL1 定义特定的属性值或彩色点值所显示的物理颜色。当前显示设备必须是 EGA、VGA 或 MCGA，否则不做任何操作，SC\_BAD\_OPT 作为函数值返回。

请参看 SCPALETTE 中的颜色值格式和标准值表。用 SCPALETTE 可以一次设置整个具有 17 种颜色的色板，SCMODE4 选择两个标准色板中的一个色板及显示方式 4 下的背景颜色。

Enhanced Color Display 仅在 200 线方式下支持非零边界颜色。用 SCBORDER 可以测试这个条件或在非 EGA 显示适配器下设备边界颜色。

源程序(SCPAL1.C):

```
#include <dos.h>
#include <bscreens.h>
#define OK 0 /* Error codes. */
#define INVALID_REQUEST 1
#define REG_RANGE 2

int scpall(reg,value)
unsigned reg,value;
{
    int result;
    int device;
    int mode,act_page,columns;
    union REGS inregs,outregs;

    if (utrange(reg,0,16))
        result = REG_RANGE;
    else
    {
        sctquip(); /* Detect equipment. */

        device = scmode(&mode,&columns,&act_page);
        if (device == b_ega || device == b_vga || b_pcmodel == IBM_JR)
        {
            inregs.h.ah = 0x10;
            inregs.h.al = (unsigned char) ((reg == 16) ? 1 : 0);
            inregs.h.bh = (unsigned char) utlobyte(value);
            /* (BL ignored if AL == 1.) */
            inregs.h.bl = (unsigned char) reg;
            int86(0x10,&inregs,&outregs);
            result = OK;
        }
        else /* Not supported. */
            result = INVALID_REQUEST;
    }

    return result;
}
```

SCPALETTE 定义 EGA、VGA 或 MCGA 颜色的整个色板

```
#include <bscreens.h>
```

int scpalett(const char \*ptable);

ptable       十七颜色值表的地 址(见下面的格式)。

(返回)       错误代码。可能值包括:

SC\_NO\_ERROR   (0)   成功。

SC\_BAD\_OPT    (1)   当前显示适配器不支持这个操作。

SCPALETT 定义属性和彩色点值显示的物理颜色。当前显示适配器必须是 EGA, VGA 或 MCGA 否则不做任何操作, SC\_BAD\_OPT 作为函数值返回。

颜色值由 ptable 所指向的表指定, 该表由十七个单字节值组成。前十六个值对应着可能的属性值和彩色点值, 第十七个值指明边界(“屏幕外”)的颜色。(Enhanced Color Display 仅在 200 线方式下支持非黑色的边界。)

颜色值具有下面的格式:

```
*****位*****
      5      4      3      2      1      0
      R'     G'/I  B'     R      G      B
```

每个值的 0-5 位对应着相应的彩色信号, 在使用 IBM Color Display 时第四位对应高亮度位, 颜色值的第 6、7 位被忽略。

例如, 当适配器重置时, EGA ROM BIOS 定义下面的颜色值:

标准 Enhanced Graphics Adapter 色板

*****位*****								
		5	4	3	2	1	0	
表项	值	R'	G'/I	B'	R	G	B	颜色
0	0x00	0	0	0	0	0	0	黑色
1	0x01	0	0	0	0	0	1	蓝色
2	0x02	0	0	0	0	1	0	绿色
3	0x03	0	0	0	0	1	1	青绿
4	0x04	0	0	0	1	0	0	红色
5	0x05	0	0	0	1	0	1	紫红
6	0x14	0	1	0	1	0	0	棕色
7	0x07	0	0	0	1	1	1	白色
8	0x38	1	1	1	0	0	0	暗色
9	0x39	1	1	1	0	0	1	淡蓝
10	0x3a	1	1	1	0	1	0	淡绿
11	0x3b	1	1	1	0	1	1	淡青
12	0x3c	1	1	1	1	0	0	淡红
13	0x3d	1	1	1	1	0	1	淡紫
14	0x3e	1	1	1	1	0	1	黄色
15	0x3f	1	1	1	1	1	1	增白
16(边界)	0x00	0	0	0	0	0	0	黑色

SCPAL1 设置某一个颜色值, SCPAL1 或 SCBORDER 设置边界颜色, 而 SCMODE4 选择两个标准色板中的一个色板及显示方式 4 下的背景颜色。

源程序(SCPALETT.C):

```
#include <dos.h>
#include <bscreens.h>
#define OK 0 /* Error codes. */
#define INVALID_REQUEST 1
int scpalett(ptable)
const char *ptable;
{
    int result;
```

```

int      device;
int      mode,act_page,columns;
union REGS  inregs,outregs;
struct SREGS segregs;

sccquip();                      /* Detect equipment.      */

device = smode(&mode,&columns,&act_page);
if (device == b_ega || device == b_vga || b_pemodel == IBM_JR)
{
    inregs.x.ax = 0x1002;
    inregs.x.dx = utoff(ptable);
    segregs.es = utseg(ptable);
    int86x(SC_BIOS_INT,&inregs,&outregs,&segregs);
    result = OK;
}
else                            /* Not an EGA.          */
    result = INVALID_REQUEST;

return result;
}

```

## SCPCLR          清除当前显示页

#include <bscreens.h>  
void scpclr(void);  
SCPCLR 清除当前显示页(记录在 b\_curpage 中), 不论它是活动的(可见)还是不活动的。  
屏幕由通常的黑底白字属性填充(前景 7, 背景 0)。  
SCPCLR 必须在字符方式下工作。

源程序(SCPCLR.C):

```

#include <bscreens.h>
void scpclr()
{
    int mode,cols,act_page;
    int save_row,save_col,high,low;
    smode(&mode,&cols,&act_page);

                                /* Save cursor location      */
    securst(&save_row,&save_col,&high,&low);
    securset(0,0);

                                /* Use foreground attribute */
                                /* 0 in graphics mode,          */
                                /* 7 in text mode.                */
    scattrib(((mode > 3 && mode != 7) ? 0 : NORMAL),
              SC_BLACK,
              (char) ' ',
              cols * scrows());    /* Size of screen          */
    securset(save_row,save_col);  /* Restore cursor location */
}

```



```
#include <bscreens.h>
```

```
int scpgecur( int off,
```

```
            int high, int low,
```

```
            int adjust);
```

```
off          光标关闭指示(0=on)。
```

```
high         光标上扫描行。
```

```
low          光标下扫描行。
```

```
adjust       CUR_ADJUST(1)或 CUR_NO_ADJUST (0)。
```

```
(返回)       off 参数的值。
```

SCPGCUR 设置并记录当前显示页的光标尺寸。除非这一页是活动的(即可见的), 否则不做任何操作。

如果 off 不为零, 则光标被关闭(变成不可见); 否则光标可见。

光标尺寸由上下扫描行决定。对于单色显示器, 扫描行值在 0 到 13 之间, 对于 Color/Graphics Adapter, 值的范围是 0 到 7。如果指明了 CUR\_ADJUST, 则超界扫描行的请求将被调整, 使之合于恰当的范围。如果指明了 CUR\_ADJUST, 下列值将在所有环境下有效(即使在 EGA、VGA 和 MCGA 下)。

#### 推荐的光标扫描行范围

光标形状	上扫描行	下扫描行
扫描行在上方	0	1
上半方块	0	3
全方块	0	13
下半方块	8	13
扫描行在底部	12	13

如果指定了 CUR\_NO\_ADJUST, 则使用指定的未加调整的扫描行。当恢复被另一个程序改变的光标时这是很有用的。然而, 不合适的值会导致一个不可见的或不连续的光标。

对于非活动页, BIOS 和 Turbo C TOOLS 不记录光标尺寸或开/关状态。

用 SCCURST 可以取得当前页光标的尺寸开/关状态。

#### 源程序(SCPGCUR.C):

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

```
#include <bscreens.h>
```

```
int scpgecur(off,high,low,adjust)
```

```
int off;
```

```
int high;
```

```
int low;
```

```
int adjust;
```

```
{
```

```
    union REGS inregs,outregs;          /* General registers          */
```

```
    int dev,mode,act_page,columns;
```

```
    unsigned int max_scan_line;
```

```
                                /* Address of BIOS INFO byte.    */
```

```
#define INFO_LOC (utfar(0x0040,0x0087,unsigned char))
```

```
    char info;
```

```
    int changed_info,truncate;
```

```
    dev = scmode(&mode,&columns,&act_page);
```

```
    if (b_curpage == act_page)          /* Change physical cursor if    */
```

```
    {                                    /* current page is active.      */
```

```
        scequip();
```

```
        if (dev == b_mdpa)
```

```

        max_scan_line = 13;
    else if (dev == b_ega || dev == b_vga || b_pcmode == IBM_JR)
        max_scan_line = 7;
    else
    {
        inregs.x.ax = 0x1130;
        inregs.h.bh = 0;
        int86(SC_BIOS_INT,&inregs,&outregs);
        max_scan_line = outregs.x.cx - 1;
    }

    changed_info = 0;
    if (adjust == CUR_NO_ADJUST)
    {
        /* In the unnatural case where there are 43 or more text lines and*/
        /* cursor compensation is enabled, disable the compensation and */
        /* restore it at exit. */
        if (max_scan_line > 7)
        {
            info = utpeekb(INFO_LOC);
            if (0 == (info & 1)) /* If compensation enabled, */
            {
                info |= 1; /* disable compensation. */
                utpokeb(INFO_LOC,info);
                changed_info = 1;
            }
        }
    }
    else
    {
        /* We may need to adjust the requested scan lines to fit into the */
        /* 0-7 range. This is needed if a scan line exceeds 7 and if one */
        /* of the following is true: */
        /*
        /* 1) this is the Color/Graphics Adapter;
        /* 2) this is 43-line mode (there are only 8 scan lines);
        /* 3) EGA cursor compensation is being performed in a 14-scan-
        /* line environment.
        high = utlonyb(high);
        low = utlonyb(low);
        if ( dev != SC_MONO
            || max_scan_line <= 7)
        {
            if ( (dev == b_ega || dev == b_vga || dev == b_mega)
                && max_scan_line > 7)
            {
                /* If EGA, truncate scan lines only if */
                /* BIOS cursor compensation is enabled */
                /* (i.e., bit 1 of INFO is off).
                info = utpeekb(INFO_LOC);

```

```

        truncate = (0 == (info & 1));
    }
    else
        truncate = 1;
    if (truncate)
    {
        if (high > max_scan_line)
            high = (max_scan_line * high) / 13;
        if (low > max_scan_line)
            low = (max_scan_line * low) / 13;
    }
}
}
if (off)
    /* Set bits 4 and 5 if turning */
    high |= 0x0030;
    /* cursor off. */
inregs.h.ah = 1;
inregs.h.ch = (unsigned char) high;
inregs.h.cl = (unsigned char) low;
int86(16,&inregs,&outregs);
if (changed_info)
{
    info &= ~1;
    /* Clear compensation bit (turn */
    /* compensation back on). */
    utpokeb(INFO_LOC,info);
}
}
return(off);
}

```

## SCREAD 从屏幕读取一个显示字符及其属性

```

#include <bscreens.h>
char scread( int *pfore,
             int *pback);
pfore      变量的指针，该变量返回前景属性。
pback      变量的指针，该变量返回背景属性。
(返回)     字符。

```

SCREAD 返回当前显示页上当前光标位置的字符和属性。

当前位置的显示属性字节作为前景和背景显示属性返回。然而，如果屏幕处于图形方式下，这些值无定义。

源程序(SCREAD.C):

```

#include <dos.h>
#include <bscreens.h>
char scread(pfore,pback)
int *pfore;
int *pback;
{
    union REGS inregs,outregs;

```

```

inregs.h.ah = 8;
inregs.h.bh = (unsigned char) b_curpage;

int86(SC_BIOS_INT,&inregs.&outregs);
*pfore = utlonyb(outregs.h.ah);
*pback = uthinyb(outregs.h.ah);

return outregs.h.ah;
}

```

## SCRESTPG 恢复一个显示页

```

#include <bscreens.h>
int screstpg(const PAGE_STATE *ppage_state);
ppage_state 结构的地址 通过该结构恢复显示数据。
(返回) 错误代码。

```

SC\_NO\_ERROR (0) 成功。  
 SC\_RANGE (2) 对于当前显示方式来说，所记录的页尺寸过大。  
 SC\_NO\_MEMORY (3) 无足够内存来完成这个操作。

SCRESTPG 将数据恢复到当前显示页(即通过 SCPAGE 设置的页)上，恢复光标位置。信息通过一个 PAGE\_STATE 结构传递，该结构定义如下：

```

typedef struct /*PAGE STATE 结构:*/
{
    /*当前页的状态。*/
    in curs_row, curs_column; /*光标位置。*/
    char *pimage; /*压缩的 屏幕图象。*/
    int image_length; /*压缩图象的长度。*/
} PAGE_STATE;

```

数据是利用 UTSQZSCN 以压缩格式存放的。当用户不再需要这些数据时(见下面的示例)，可以用标准的 free() 函数释放 ppage\_state pimage 指针。

SCSAVEPG 记录显示的正文数据，SCSETVID 则恢复显示适配器的状态。

源程序(SCRESTPG.C):

```

#include <bscreens.h>
#include <bvideo.h>
int edecr screstpg(ppage_state)
const PAGE_STATE *ppage_state;
{
    int color_or_mono;
    int mode, act_page;
    int rows, columns;
    int raw_image_length;
    char *praw_image;

    /* First get the screen size. */
    color_or_mono = scmode(&mode, &columns, &act_page);
    rows = scrows();
    raw_image_length = columns * rows * 2;

    /* If the current adapter is a CGA and b_vifast is */

```

```

        /* zero, then we need to first uncompress the      */
        /* screen into a temporary buffer, then write it to */
        /* the screen using viwrrct, to prevent snow.      */
        /* Otherwise, uncompress it to the screen directly. */
if (!b_vifast && (b_ega == color_or_mono))
{
    praw_image = malloc(raw_image_length);
    if (praw_image == NIL)
        return(SC_NO_MEMORY);
    if (utunsqz(ppage_state->pimage, praw_image,
                ppage_state->image_length, raw_image_length) !=
        raw_image_length)
    {
        free(praw_image);
        return(SC_RANGE);
    }

    viwrrct(0, 0, rows - 1, columns - 1, praw_image, -1, -1,
            CHAR_ATTR);
    free(praw_image);
}
else
    if (utunsqz(ppage_state->pimage, viptr(0, 0),
                ppage_state->image_length, raw_image_length) !=
        raw_image_length)
    {
        return(SC_RANGE);
    }
    scursel(ppage_state->curs_row, ppage_state->curs_column);
    return(SC_NO_ERROR);
}

```

## SCROWS 返回屏幕的字符行数

```
#include <bcreens.h>
```

```
int scrows(void);
```

(返回) 行的数量。

SCROWS 根据安装的是 EGA、VGA 不是 MCGA 返回屏幕的字符行的数量。

用 SCMODE 可以得到字符列的数量。

源程序(SCROWS.C):

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

```
#include <bcreens.h>
```

```
int scrows()
```

```

{
    union REGS inregs,outregs;          /* Registers for BIOS call */

    if (!b_know_hw)
        scequip();                      /* Look for EGA */
    if (b_ega != SC_ABSENT || b_vga != SC_ABSENT || b_mega != SC_ABSENT)

```

```

{
    inregs.x.ax = 0x1130;
    inregs.h.bh = 0;
    int86(SC_BIOS_INT,&inregs,&outregs);
    return outregs.h.dl + 1;
}

return PC_ROWS;
}

```

## SCSAVEPG 保存一个显示页

```
#include <bcreens.h>
int scsavepg(PAGE_STATE *ppage_state);
ppage_state 结构的地址 该结构记录显示页。
(返回)      错误代码:
```

SC\_NO\_ERROR (0) 成功。  
SC\_NO\_MEMORY (3) 无足够内存来完成这个操作。

SCSAVEPG 记录当前显示页(通过 SCPAGE 设置的页)上的显示数据, 它还记录光标的位置, 信息在 PAGE\_STATE 结构中返回, 该结构定义如下:

```
typedef struct /*PAGE_STATE 结构: 当前页的状态*/
{
    int curs_row, curs_column; /*光标的位置。*/
    char*pimage; /*压缩的 屏幕图象。*/
    int image_length; /*压缩图象的长度。*/
} PAGE_STATE;
```

数据通过 UTSQZSCN 以压缩格式保存起来。如果不再需要这些数据(见 SCRESTPG 下的示例), 可以用标准的 free( ) 函数释放 ppage\_state pimage 指针。

SCRESTPG 恢复显示的正文数据, SCGETVID 则记录显示页的状态。

源程序(SCSAVEPG.C):

```
#include <bcreens.h>
#include <bvideo.h>
int cdecl scsavepg(ppage_state)
PAGE_STATE *ppage_state;
{
    int color_or_mono,
    int mode, act_page;
    int rows, columns;
    int cursor_high, cursor_low;
    int raw_image_length, compressed_image_length;
    char far *praw_image;
    char *pcompressed_image;
    char *pbuffer;
    int allocated_buffer = 0;
    char dummy_char;

    /* First get the screen size and cursor coordinates.*/

```

```

color_or_mono = scinode(&mode, &columns, &act_page);
rows = scrows();
scurst(&(ppage_state->curs_row), &(ppage_state->curs_column),
        &cursor_high, &cursor_low);
raw_image_length = columns * rows * 2;

    /* If the current adapter is a CGA and b_vifast is */
    /* zero, then we need to first read the screen into */
    /* a temporary buffer using virdirect() to prevent */
    /* snow. Otherwise, compress from the screen */
    /* directly. */
if (!b_vifast && (b_cga == color_or_mono))
{
    pbuffer = malloc(raw_image_length);
    if (pbuffer == NIL)
        return(SC_NO_MEMORY);
    allocated_buffer = 1;

    virdirect(0, 0, rows - 1, columns - 1, pbuffer, CHAR_ATTR);
    praw_image = pbuffer;
}
else
    praw_image = viptr(0, 0);
    /* Now determine how much memory the compressed */
    /* screen image will require by passing a dummy */
    /* buffer of length 0 to utsqzsen, then allocate */
    /* space for the compressed image and perform the */
    /* compression. */
compressed_image_length = utsqzsen(praw_image, &dummy_char,
                                    raw_image_length, 0);
pcompressed_image = malloc(compressed_image_length);
if (pcompressed_image == NIL)
    return(SC_NO_MEMORY);
utsqzsen(praw_image, pcompressed_image,
        raw_image_length, compressed_image_length);
ppage_state->pimage = pcompressed_image;
ppage_state->image_length = compressed_image_length;
    /* Now free the temporary buffer if necessary. */
if (allocated_buffer)
    free(pbuffer);
return(SC_NO_ERROR);
}

```

## SCSETVID 恢复整个显示状态

#include <bscreens.h>

int scsetvid(const ADAP\_STATE \*pstate);

pstate 结构的地址，该结构记录适配器的状态。

(返回) 错误代码：如果成功，为 0；如果所需适配器的状态非法，则为-1。

SCSETVID 恢复当前显示适配器的状态(但不恢复显示数据), 信息通过 ADAP\_STATE 结构传递, 该结构定义如下:

```
typedef struct          /*CUR_TYPE 结构: */
{
    int high, low;      /*光标的高低扫描行。*/
}CUR_TYPE;
typedef struct          /*ADAP_STATE 结构: 显示适配器的状态。*/
{
    int mode;           /*当前显示方式。*/
    int act_page;       /*活动(显示)页。*/
    int cur_page;       /*当前页。*/
    int rows, columns;  /* 屏幕尺寸。*/
    int curs_off;       /*光标打开或关闭。*/
    CUR_TYPE curs_size; /*光标高低扫描行。*/
} ADAP_STATE;
```

用 SCGETVID 可以在 ADAP\_STATE 结构中记录显示适配器的状态而 SCSAVEPG 和 SCRESTPG 则保存和恢复显示的正文数据。

源程序(SCSETVID):

```
#include <bcreens.h>
int scsetvid(pstate)
const ADAP_STATE *pstate;
{
    int current_adapter, requested_adapter;
    int current_mode;
    int current_act_page;
    int current_rows, current_columns;
    int current_curs_off;
    CUR_TYPE current_curs_size;
    int row, column;
    current_adapter = smode(&current_mode, &current_columns,
                           &current_act_page);
    requested_adapter = ((pstate->mode == 7) || (pstate->mode == 15))
        ? SC_MONO : SC_COLOR;
    /* First, check to see if we need to switch adapters. If so, try */
    /* to switch without calling scnewdev(). Having switched, call */
    /* smode() again to record the state of the new device. */
    if (current_adapter != requested_adapter)
    {
        if (scchgdev(requested_adapter) != 0)
            if (scnewdev(pstate->mode, pstate->rows) != pstate->mode)
                return(-1);
        current_adapter = smode(&current_mode, &current_columns,
                               &current_act_page);
    }
    current_rows = scrrows();
    current_curs_off = sccurst(&row, &column, &current_curs_size.high,
                              &current_curs_size.low);
    /* Check to see whether we need to call scnewdev() to switch */
    /* modes or rows. */
}
```



```

if ((current_mode != pstate->mode) ||
    (current_rows != pstate->rows) ||
    (current_columns != pstate->columns))
{
    if (screwdev(pstate->mode, pstate->rows) != pstate->mode)
        return(-1);
}
/* Now restore the active (displayed) page and the current page */
/* for this device. */
if (scapage(pstate->act_page) != pstate->act_page)
    return(-1);
if (scpage(pstate->cur_page) != pstate->cur_page)
    return(-1);
/* Finally, check to see if we need to fix the cursor. */
if ((current_curs_off != pstate->curs_off) ||
    (current_curs_size.high != pstate->curs_size.high) ||
    (current_curs_size.low != pstate->curs_size.low))
{
    if (sepgcur(pstate->curs_off, pstate->curs_size.high,
                pstate->curs_size.low, CUR_NO_ADJUST) !=
        pstate->curs_off)
    {
        return(-1);
    }
}
return(0);
}

```

## SCTTYWIN 以 TTY 方式向矩形区域写入一个字符

```

#include <bscreens.h>
void setttywin( int u_row, int u_col,
                int l_row, int l_col,
                char ch,
                int fore, int back,
                int scr_fore, int scr_back);

```

u\_row, u\_col 区域的上行左列。

l\_row, l\_col 区域的下行右列。

ch 待写的字符。

fore 如果保留已有的前景，为-1；一个0到15之间的值指明一个新前景属性。

back 如果保留已有的背景，为-1；一个0到15之间的值指明一个新背景属性。

scr\_fore 滚动时加入空行的前景属性(若为-1，则使用滚动前字符位置的 前景属性)。

scr\_back 滚动时加入空行的背景属性(若为-1，则使用滚动前字符位置的背景属性)。

SCTTYWIN 以电传方式(TTY)向当前显示页上的一个矩形区域写入一个字符。它将回车、换行、退格和响铃作为命令而不作为打印字符对待。如果在最后一行出现换行或最后一行溢出，则该区域滚动。所有的输出和滚动仅限于这个区域。

如果光标当前位于该区域内，则字符写在光标位置；如果光标不在区域内，则字符写在区域的左上角，光标被放置在显示字符的后面。

下面这些字符作为特殊字符看待：

换行('\12') 使光标下移一行，如果光标已经位于区域的底行，则该区域滚动。

回车('\15') 使光标移到最左列。

退格('\10') 使光标左移一列(非破坏性地), 如果它已经位于最左列, 则不做任何操作。响铃字符('\7')在当前页是活动页的条件下使得计算机的扬声器发声。

如果屏幕处于字符方式下, 对 fore 和 back 指定-1 可以分别保留已有的前景或背景属性。不论 option 为何值都是这样。如果屏幕处于图形方式下, 对 fore 指定-1 将使用颜色 1。

如果屏幕在字符方式下滚动, 则新底行的属性字节以下面的方式形成: 如果 xcr\_fore 和/或 xcr\_back 的值不是-1, 则使用它们的原值; 如果两者都是-1, 则前景或背景颜色取自于底行的一个字符, 该字符以下列方式选取: 如果滚动是由于换行造成的, 颜色取自于滚动前最后一个字符的颜色; 如要滚动是由于矩形区域最后一行的溢出引起的, 则颜色取自于区域的左下角(当 scr\_fore 和 scr\_back 均为-1 时, 这与 SCTTYWART 的约定是相同的。)

仅在当前显示页是活动页的情况下 SCTTYWIN 才能在图形方式下滚动。如果屏幕在图形方式下滚动, 滚动的正文颜色就是 scr\_fore 所指定的颜色。如果 scr\_fore 是-1, 则使用颜色。

必要时裁减区域的尺寸使适合于显示器的尺寸。

源程序(SCTTYWIN.C):

```
#include <dos.h>
#include <bscreens.h>
#include <bvideo.h>
#define BEL '\7'
#define BS '\b'
#define CR '\r'
#define LF '\12'

void scttywin(u_row,u_col,l_row,l_col,ch,fore,back,scr_fore,scr_back)
int u_row,u_col,l_row,l_col;
char ch;
int fore,back,scr_fore,scr_back;
{
    union REGS inregs,outregs; /* Registers for BIOS calls */
    int mode,act_page,columns,last_row;
    int oldmask,old_attr;
    int high,low,row,col;

    scmode(&mode,&columns,&act_page);
    last_row = scrows() - 1;
    last_row--;

    utbound(u_row,0,last_row) /* Force reasonable values */
    utbound(u_col,0,columns - 1)
    utbound(l_row,u_row,last_row)
    utbound(l_col,u_col,columns - 1)

    securst(&row,&col,&high,&low);
    if (utrange(row,u_row,l_row) || utrange(col,u_col,l_col))
        secursel((row = u_row),(col = u_col));

    switch (ch)
    {
        case BEL:
            if (b_curpage == act_page)
```

```

        settywrt(ch,0);          /* Don't beep on inactive page */
break;

case BS:
    if (col > u_col)             /* Don't back up past first */
        secureset(row,col - 1); /* column */
    break;

case CR:
    secureset(row,u_col);        /* Beginning of current line */
    break;

default:
    /* First write the character */

    if (fore == -1)              /* "oldmask" will help us */
        oldmask = 0x0f;         /* extract the portion(s) of */
    else                          /* the previous attributes */
        oldmask = 0x00;         /* which must be preserved. */

    if (back == -1)
        oldmask |= 0x0f;

    if (oldmask == 0xff)
    {
        /* Write character only */
        inregs.h.ah = 10;
        inregs.h.al = ch;
        inregs.h.bh = (unsigned char) b_curpage;
    }
    else
    {
        /* Write character with */
        /* attributes */

        if (oldmask != 0x00)
        {
            inregs.h.ah = 8;
            inregs.h.bh = (unsigned char) b_curpage;
            int86(16,&inregs,&outregs);
            old_attr = outregs.h.ah & oldmask;
        }
        else
            old_attr = 0;

        inregs.h.ah = 9;
        inregs.h.al = ch;
        inregs.h.bh = (unsigned char) b_curpage;
        inregs.h.bl = (unsigned char) (old_attr |
            (utnybbyt(back,fore) & ~oldmask));
    }
    inregs.x.cx = 1;
    int86(16,&inregs,&outregs); /* Actually write it */

    inregs.h.ah = 2;

```

```

inregs.h.bh = (unsigned char) b_curpage;
if (++col <= l_col)      /* Check for possible wrap */
{
    /* This fits on current line */
    inregs.h.dh = (unsigned char) row;
    inregs.h.dl = (unsigned char) col;
    int86(16,&inregs,&outregs);
    break;
}

/* Wrap to next line */

col = u_col;
inregs.h.dh = (unsigned char) row;
inregs.h.dl = (unsigned char) col;
int86(16,&inregs,&outregs);
case LF:
    if (row < l_row)
    {
        /* No need to scroll */
        secureset(row + 1,col);
        break;
    }

    /* Obtain attribute with which */
    /* to fill new bottom line. */

    scread(&fore,&back);      /* Obtain attribute from */
                             /* previous cursor position */

    if (scr_fore == -1)
        scr_fore = fore;
    if (scr_back == -1)
        scr_back = back;

    /* Scroll up. */

    viscroll(1,
        utnybbyt(scr_back,scr_fore),
        u_row,u_col,
        l_row,l_col,
        SCR_UP);
    break;
}
}

```

## SCTTYWRT 以 TTY 方式向屏幕写一个字符

#include <bsercons.h>

int settywrt( char ch,

int fore);

ch 待显示的字符。

fore 如果屏幕方式是图形方式，则为使用的前景属性。

(返回) 返回值总是 0。

SCTTYWRT 以通常的 TTY 格式向当前屏幕写一个字符。当前页必须是活动的(即当前显示)。

下面的字符作为特殊字符处理：

换行('\n') 使光标下移一行，如果光标已经位于屏幕的底行，屏幕将发生滚动。

回车('\15') 使光标移到列 0(最左列)。

退格('\10') 使光标左移一列(非破坏性地)。如果光标已经位于列 0 上, 这时不作任何操作。

响铃字符('\7') 使得计算机在当前页是活动页的情况下响铃。

如果屏幕处于图形方式下, 则用 `fore` 来设置写入字符的颜色。另外, 如果 前页是不活动的而屏幕发生滚动, 则 `fore` 用于滚动正文的颜色。在字符方式下 `fore` 被忽略。

如果由于一个换行字符使屏幕在字符方式下滚动, 则新空行的属性取自于滚动前最后一个字符的属性(标准 BIOS 方式)。如果滚动是由于 屏幕最后一行 溢出产生的, 则属性取自于最后一行列 0 处的原属性。

`SCTTYWIN` 用 TTY 方式向屏 幕的一个矩形区域写一个字符。

源程序(SCTTYWRT.C):

```
#include <dos.h>
#include <bcreens.h>
int scttywrt(ch,fore)
char ch;
int fore;
{
    union REGS inregs,outregs;
    inregs.h.ah = 14;
    inregs.h.al = ch;
    inregs.h.bh = (unsigned char) b_curpage;
    inregs.h.bl = (unsigned char) fore;
    int86(SC_BIOS_INT,&inregs,&outregs);
    return(0);
}
```

**SCWRAP**            以 TTY 方式向一个矩形中写入一个字符串, 带有整字换行。

```
#include <bcreens.h>
void scwrap( int u_row, int u_col,
             int l_row, int l_col,
             int num_spaces,
             const char *pbuffer,
             int fore, int back,
             int option);
```

`u_row, u_col`    区域的顶行右列。  
`num_spaces`    待写入字符的个数。如果指定了 `CHARS_ONLY` (见下面的 `option`), 则 `num_spaces` 为零表示缓冲区以 `NUL('\0')` 字符结尾。  
`pbuffer`        待写入的数据。  
`fore`            如果现有的前景要保留, 为-1; 如果指定了 `CHARS_ONLY`, 一个 0 到 15 的值指明新的前景属性。  
`back`            如果现有的背景要保留, 为-1; 如果指定了 `CHARS_ONLY`, 一个 0 到 15 的值指明新背景属性。  
`option`         如果缓冲区包含(char,attr) 9 对, 则为 `CHAR_ATTR(2)`; 如果缓冲区仅包含字符并且 `fore` 和 `back` 用于属性, 则为 `CHARS_ONLY (0)`。

`SCWRAP` 向当前显示页上的一个矩形区域写入字符串而不使字在行间断裂。

一个“字”定义的为一个非空白字符序列, 它被空白字符或缓冲区的开头或末尾所包围。“空白”字符是空格(' '), 制表符('\t'), 响铃('\7'), 退格('\b'), 换行('\12')和回车('\15')。仅当字的长度超过区域的宽度时, 字才会在区域的行之间断裂。如果一个字太长, 无法完整地容纳于当前行, 则它被写到区域的下一行中。必要时屏幕发生滚动。

缓冲区中的空格被逐字写到当前行,直到末尾。一行的开头不写入任何空格。

如果矩形区域的未行出现一个换行符或未行溢出,则该区域发生滚动。新空行属性的选择方式与 SCTTYWRT 相同。

如果光标位于区域以内,字符从光标位置开始写入;如果光标不在区域内,则字符在区域的左上角开始写入,光标置于显示字符串之后。

下列字符作为特殊字符处理:

Tab ('\\t') 被认作一个空白字符。

DEL 字符('\\177') 作为一个空格写入屏幕。然而,它在这种意义上被作为非空字符对等;当整字换行时,非空格符与 DEL 字符的连接保持完整。DEL 写入屏幕的字符值是 0x20(' ')。

NUL ('\\0') 根据 option 和 num\_spaces 值的不同有不同的意义。如果指明了 CHARS\_ONLY 而 num\_spaces 为零,则 NUL 表示字符串的末尾,否则它是一个可打印字符(看起来如同空格)而不是字符串的末尾。NUL 与 DEL 的处理方式相同,但 NUL 写往屏幕的字符值为零。

换行('\\12') 使光标下移一行。如果光标已经位于区域的底行,这时区域发生滚动。

回车('\\15') 使光标移到区域的最左列。

退格('\\10') 使光标左移一列(非破坏性)。如果光标已经位于区域的最左列,则不做任何操作。

● 响铃('\\10') 使光标左移一列(非破坏性)。如果光标已经位于区域的最左列,则不做任何操作。

响铃字符('\\7') 在当前页是活动页的情况下使计算机响铃。

num\_spaces 参数指明实际使用的缓冲区字符的数目。(如果指定了 CHAR\_ATTR,则使用缓冲区字节数的两倍。)

如果屏幕处于字符方式并且指定了 CHARS\_ONLY,则对 fore 或 back 指定-1 分别保留现有的前景或背景属性;如果屏幕处于图形方式并且指定了 CHARS\_ONLY,则对 fore 指定-1 将使用颜色 1。

除非当前页是活动的(即可见),否则该函数在图形方式下不使屏幕滚动。

源程序(SCWRAP.C):

```
#include <dos.h>
#include <string.h>
#include <bscreens.h>
#define range(a,l,h) (((a) < (l) || (a) > (h)))
/* Macro inc_ptr steps a pointer through the character buffer, */
/* returning the new pointer value. */
#define inc_ptr(ptr) (want_attr ? (ptr += 2) : ++ptr)
/* Macro str_len measures the distance between two locations in */
/* the character buffer. */
#define str_len(from,to) ((to - from) >> (want_attr ? 1 : 0))
#define BEL '\\7'
#define BS '\\b'
#define TAB '\\t'
#define CR '\\r'
#define LF '\\12'
#define DEL '\\177'
```

```
void scwrap(u_row,u_col,l_row,l_col,num_spaces,buffer,fore,back,option)
int u_row,u_col,l_row,l_col,num_spaces;
const char *buffer;
int fore,back,option;
{
    int mode,act_page,columns,last_row;
```

```

char      ch;
const char *pendstr;
int      nbytes,nonblank_len;
int      row,col,high,low,i;
int      want_attr;

semode(&mode,&columns,&act_page);
last_row = scrrows() - 1;

utbound(u_row,0,last_row)      /* Force reasonable values */
utbound(u_col,0,columns - 1)
utbound(l_row,u_row,last_row)
utbound(l_col,u_col,columns - 1)

sccurst(&row,&col,&high,&low);
if (range(row,u_row,l_row) || range(col,u_col,l_col))
    sccurst((row = u_row),(col = u_col));

want_attr = ((option & CHAR_ATTR) != 0);
if (!want_attr && num_spaces == 0)
    num_spaces = (int) strlen(buffer);

while (num_spaces > 0)
{
    sccurst(&row,&col,&high,&low);
    switch (ch = *buffer)
    {
        case TAB:
            ch = ' ';      /* Treat TAB as a blank. */
        case ' ':
            if (col == u_col) /* Write the blank only if this */
            {                /* isn't the beginning of the */
                num_spaces--; /* line. */
                inc_ptr(buffer);
                break;
            }
        case CR:      /* SCTTYWIN handles both */
        case LF:      /* ordinary and special */
        case BEL:     /* characters. */
        case BS:
            settywin(u_row,u_col,l_row,l_col,ch,
                want_attr ? (int) utlonyb(*(buffer+1)) : fore,
                want_attr ? (int) uthinyb(*(buffer+1)) : back,
                -1,-1);
            num_spaces--;
            inc_ptr(buffer);
            break;

        default:      /* This is the beginning of a */
                     /* string of one or more */

```

```

/* nonblanks. */

/* First measure the length of */
/* the string of nonblanks. */
nbytes = (want_attr ? (num_spaces < 2) : num_spaces);
pendstr = buffer;
while (pendstr < buffer + nbytes)
    if ((ch = *inc_ptr(pendstr)) == ' '
        || ch == CR
        || ch == LF
        || ch == TAB
        || ch == BEL
        || ch == BS)
        break;
nonblank_len = (int) str_len(buffer, pendstr);

/* If this string won't fit on */
/* this line, */
if (col + nonblank_len > l_col + 1

    /* and the string is small */
    /* enough to fit on a new line, */
    && nonblank_len <= (l_col - u_col + 1)

    /* and this isn't already a */
    /* fresh line, */
    && col > u_col)

    /* Then fill the rest of the */
    /* line with blanks and let */
    /* SCTTYWIN advance us to the */
    /* next line. */
    while (col++ <= l_col)
        settywin(u_row, u_col, l_row, l_col,
            (char) ' ', -1, -1, -1, -1);

    /* Write the string itself. If */
    /* the string is longer than */
    /* the region is wide, then it */
    /* will be split, but at least */
    /* we tried. */
    for (i = nonblank_len; i--;)
    {
        if ((ch = *buffer++) == DEL)
            ch = ' '; /* Print DEL as a blank */
        settywin(u_row, u_col, l_row, l_col, ch,
            want_attr ? (int) utlonyb(*buffer) : fore,
            want_attr ? (int) uthinyb(*buffer) : back,
            -1, -1);
        if (want_attr)

```



```

        buffer++;
    }
    num_spaces -= nonblank_len;

    break;
}
}
}

```

## SCWRITE 在屏幕上显示一个字符的多个拷贝

```
#include <bSCREENS.h>
```

```
int scwrite(char ch,
```

```
unsigned cnt);
```

ch 待显示的字符。

cnt 待显示的 ch 拷贝的个数。

(返回) 返回值总是零。

SCWRITE 在当前显示屏幕上显示 ch 字符的 cnt 个拷贝而不改变原先设置的属性。字符在当前页 (b\_curpage) 当前光标位置上开始写入。

如果写入操作超出屏幕的末端或(在图形方式下)写入操作超出当前正文行,则会引来预料的结果。SCWRITE 不使屏幕滚动。

SCATTRIB 对于图形方式是很合适的,因为它能够控制字符的颜色。SCWRITE 在图形方式下使用颜色 1。

源程序(SCWRITE.C):

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

```
#include <bSCREENS.h>
```

```
int scwrite(ch,cnt)
```

```
char ch;
```

```
unsigned cnt;
```

```

{
    union REGS inregs,outregs;
    if (cnt) /* BIOS treats 0 as 64K. */
    {
        inregs.h.ah = 10;
        inregs.h.al = ch;
        inregs.h.bh = (unsigned char) b_curpage;
        inregs.h.bl = 1; /* Use color 1 in case this is */
                        /* graphics mode */
        inregs.x.cx = cnt;
        int86(16,&inregs,&outregs);
    }

    return(0);
}

```

## VIATRECT 改变屏幕上一个矩形的属性

```
#include <bVIDEO.h>
```

```
int viatrect( int u_row,int u_col,
              int l_row,int l_col,
              int fore, int back);
```

u\_row, u\_col 矩形区域的顶行左列。

l\_row, l\_col 区域的底行右列。

fore, back 前景和背景属性。

(返回) 被改变的物理空间的数量。

VIATRECT 用所给属性填充当前显示页上一个矩形区域而不影响在那儿显示的字符。光标保持不动。

区域的左上角是 (u\_row,u\_col)，这里(0,0) 代表整个屏幕的左上角；区域的右下角是(l\_row,l\_col)，这里(24,79)是 80x25 屏幕的右下角。必要时裁减被填充区域的尺寸使之合于物理屏幕。

这个函数仅在字符方式(0、1、2、3、7)下工作。

源程序(VIATRECT.C):

```
#include <bscreens.h>
#include <bvideo.h>
int viatrect(u_row,u_col,l_row,l_col,fore,back)
int u_row,u_col,l_row,l_col;
int fore,back;
{
    int device,mode,act_page,last_row,columns;
    int height,width;
    int command;
    char far *pscreen;

    device = scmode(&mode,&columns,&act_page);
    if (mode > 3 && mode != 7)
        return 0;
    last_row = scrrows() - 1;

    utbound(u_row,0,last_row)          /* Force reasonable values */
    utbound(l_row,u_row,last_row)
    utbound(u_col,0,columns - 1)
    utbound(l_col,u_col,columns - 1)
    height = l_row - u_row + 1;
    width = l_col - u_col + 1;
    pscreen = viptr(u_row,u_col);
    command = 4;
    if ( b_vifast != 0
        || mode == 7
        || scequip() == IBM_CV
        || device == b_ega
        || device == b_vga
        || device == b_mega)
        command |= 0x8000;          /* Nced not await retrace */
    vidirect(&pscreen,&pscreen,height,width,columns * 2,
             utnybbyt(back,fore),command);
    return height * width;
}
```

## VIDSPMSG 显示一条消息

```
#include <bvideo.h>
```

```
int vidspmsg( int row, int col,  
              int fore, int back,  
              const char *pmsg);
```

row, col 消息的位置。

fore, back 显示属性。

pmsg 待显示的字符串。

(返回) 被显示字符的数量。

VIDSPMSG 用 fore 和 back 显示属性在指定位置显示一条消息，光标保持不动。消息被显示在当前显示页上(见 SCPAGE)，该页不必是活动的(可见的)。

VIDSPMSG 不折转到屏幕的下一行，不滚动屏幕。

由于 VIDSPMSG 直接访问显示内存，所以速度很快，但它不能工作在图形方式下。

VIDSPMSG 是用一个宏实现的，它对某些参数多次求值。由于这个原因，用户在使用参数时应该避免出现副作用。

源程序(BVIDEO.H)；

参见附录 C 的 BVIDEO.H 的宏定义。

## vidirec0 直接从或向视屏适配读取或写入长方形的字符。

```
rcode = vidirec0( ppfrom, ppto, num_rows, row_length,  
                  sen_width, attr, option, gap);
```

rcode 错误代码：如果选项未知则为 1，如果成功则为 0。

ppfrom 指向包含源缓冲区地址的远地的 char 指针。

ppto 指向包含目的缓冲区地址的远地 char 指针。

num\_rows 长方形的行数

row\_length 长方形的列宽

sen\_width 两倍的屏幕宽度

attr 前台和后台属性(仅用于某些选项)

option I/O 的风格和方向；以及是否防止干扰。参见下面的讨论和表。

gap 内存中数据缓冲区连续行间的字符单位(为一些选项忽略)

直接从或向一个彩色/图形适配器、单显适配器以及兼容的适配器的视屏缓冲区读取或写入长方形内的字符。在读取时为最大程度地消除干扰要注意水平和垂直回扫的间隔。字符一行一行地传送。

缓冲区的地址由 ppfrom 和 ppt 所指的远 char 指针中保存。或者两个指针均指向依赖于选项值的视频内存中的物理地址。

几个 I/O 格式是有效的。它们通过选项值来选择。如果设置选项的位 15，则表示传输执行不考虑干扰。选项的其余位可选择如下值：

Option	说明
--------	----

0	不带属性地向屏幕写字符。*ppfrom 指向字符缓冲区。(如同选项当 gap 为 0 时 option 为 13 的情况)。
---	--

1	带属性向屏幕写字符。*ppform 指向*字符，属性)对缓冲区。(和 gap 为 0 时 option 等于 13 一样)。
---	--

2	用 attr 指定的常量属性的字符写向屏幕。*ppfrom 指向一字符缓冲区。(和 gap 为 0，option 等于 15 相同)。
---	---

3	用常量的字符和属性填充屏幕的长方形。*ppfrom 指向字符。 Attr 指定属性。
---	---

- 4 用常量属性填充屏幕的长方形, 不改变显示的字符。  
忽略 ppfrom。
- 5 从屏幕读取没有属性的字符。 \*\*ppto 用字符填充。在最后添加 NUL。(和 gap 为 0 时 option 为 16 一样)。
- 6 读取有属性的屏幕字符。 \*\*ppto 用(字符, 属性)填充。(和 gap 为 0 时 option 等于 17 一样)。
- 7 在同一显示页下向上或向左直接复制有属性的窗口。
- 8 在同一显示页下向上或向左直接复制没有属性的窗口。
- 9 在同一显示页下向下或向右直接复制有属性的窗口。
- 10 在同一显示页下向下或向右直接复制没有属性的窗口。
- 11 复制窗口(带属性)到别一不覆盖的窗口或另一显示页。
- 12 复制窗口(不带属性)到别一不覆盖的窗口或另一显示页。

注意: 前面的选择 u 忽略 gap。

- 13 不带属性地向屏幕输出字符。 \*ppfrom 指向行由 gap 字节分开的字符缓冲区。
- 14 带属性地向屏幕输出字符。 \*ppfrom 指向行由 gap 字节分开的(字符, 属性)对缓冲区。
- 15 向屏幕输出字符上 attr 指定常量属性的字符。 \*ppfrom 指向行由 gap 字节分开的字符缓冲区。
- 16 从屏幕读取不带属性的字符。 \*\*ppto 用来存放字符。(注意不添加尾符 NUL。 \*\*ppto 的行之间相隔 gap 字节。
- 16 从屏幕读取不带属性的字符。 \*\*ppto 用来存放(字符, 属性)对。(注意不添加尾符 NUL。 \*\*ppto 的行之间相隔 gap 对。

注意: 不检查参数的正确性。(option 是否越界被检查除外)。

如果 option 的位 15 被清除, 表明防止干扰, 那么当过程退出时中断仍被允许。

返回      ercode      错误代码: 如果 option 未知, 则为 1; 若成功为 0。  
             \*\*ppto      目的缓冲区的数据。

源程序(VIDIRECO.ASM):

```
video_status_port equ 3DAh
include beginasm.mac
beginMod utamove

; Symbols to aid in referencing arguments on stack
; (Some of these items are also used as local variables.):
a equ 0
b equ a + sizeDPointer
c equ b + sizeDPointer
d equ c + 2
e equ d + 2
f equ e + 2
g equ f + 2
h equ g + 2

ppfrom equ [bp + stkoff + a]
ppto equ [bp + stkoff + b]
num_rows equ byte ptr [bp + stkoff + c]
row_length equ byte ptr [bp + stkoff + d]
```

```

sen_width equ word ptr [bp + stkoff + e]
attr      equ byte ptr [bp + stkoff + f]
option    equ word ptr [bp + stkoff + g]
gap       equ word ptr [bp + stkoff + h]

```

; Format of the table entry for one option:

```

option_entry struc
do_setup      dw      ? ; Address of setup_proc.
do_slow       dw      ? ; Address of slow_proc.
do_fast       dw      ? ; Address of fast_proc.
do_newrow     dw      ? ; Address of newrow_proc.
dirflag       db      ? ; Direction: down or up.
option_entry ends

```

; Macro to wait until the beginning of a horizontal  
; retrace interval. On exit, interrupts are off.

```

await_horiz_retrace macro
    local await_display_on,await_display_off
    sti
    nop
await_display_on:
    in     al,dx
    shr    al,1
    jc     await_display_on
    cli
await_display_off:
    in     al,dx
    shr    al,1
    jnc    await_display_off
endm

```

; Here are macros to define procedures for each style of I/O as  
; specified by "option".

; Each style of I/O requires four procedures:

- ; (1) a specialized setup procedure ("setup\_proc") (which may  
; do nothing)
- ; (2) a procedure ("slow\_proc") to do the entire read or write  
; while avoiding video interference;
- ; (3) a procedure ("fast\_proc") to do the entire read or write  
; without regard to interference (i.e., as fast as possible);
- ; (4) a procedure ("newrow\_proc") to adjust address registers  
; (SI and/or DI) and the column counter (CX) for a new row  
; of the rectangle.

; These four procedures will be addressed through table "option\_table"  
; defined below.

; Since these procedures must therefore be highly regular in form,  
; the following macros aid in defining the beginning and end of

```

;      each.

begin_setup_proc macro o_name ;; Define beginning of setup procedure
o_name&_setup_proc proc near
endm

end_setup_proc macro o_name ;; Define end of setup procedure
ret
o_name&_setup_proc endp
endm

; On entry to each slow_proc,
;
; BX = address of option_table entry;
; CX = number of cells in one row;
; DX = I/O address of video status port.
;
; Also, usually
; DS:SI = address of first source byte;
; ES:DI = address of first target byte.
;
; Additional entry conditions are satisfied
; by the setup_proc.

begin_slow_proc macro o_name,save_bx_flag
o_name&_slow_proc proc near
ifidn <save_bx_flag>,<save_bx>
push bx ;; Some options must use BX as
;; additional storage space, so
;; save BX if requested.
endif
o_name&_single_loop: ;; Begin loop that transfers one
;; character cell.
endm

end_slow_proc macro o_name,save_bx_flag
local done
loop o_name&_single_loop ;; Transfer one more cell
;; in this row.

sti
ifidn <save_bx_flag>,<save_bx>
pop bx ;; Restore BX if it was used for
;; temporary data.
endif
dec num_rows
jz done
call cs:[bx],do_newrow ;; Use BX to invoke
;; proper newrow_proc.
jmp short o_name&_slow_proc ;; Do next row.
done: ret ;; Done.

```

```
o_name&_slow_proc    endp
                    endm
```

```
; Each fast_proc has the same entry
; conditions as slow_proc.
```

```
begin_fast_proc      macro    o_name
o_name&_fast_proc    proc      near
                    endm
```

```
end_fast_proc        macro    o_name
                    local     done

                    dec       num_rows
                    jz        done
                    call      cs:[bx].do_newrow    ;; Use BX to invoke
                                                ;; proper newrow_proc.

                    jmp       short o_name&_fast_proc
done:
                    ret
o_name&_fast_proc    endp
                    endm
```

```
; Each newrow_proc assumes CH == 0.
```

```
begin_newrow_proc    macro    o_name
o_name&_newrow_proc  proc      near
                    mov       cx,row_length
                    endm
```

```
end_newrow_proc      macro    o_name
                    ret
o_name&_newrow_proc  endp
                    endm
```

```
; *****
; *****  DEFINE FOUR PROCEDURES FOR EACH OPTION *****
; *****
```

```
; (Note: many of these definitions just refer to others using "equ".)
```

```
beginCseg vidirec0
```

```
; W_CHARS:    Write characters to the screen without attributes.
```

```
begin_setup_proc     w_chars    ; No setup needed.
end_setup_proc       w_chars
```

```
begin_slow_proc      w_chars
                    await_horiz_retrace
                    movsb
```

```

        inc     di
end_slow_proc      w_chars

begin_fast_proc    w_chars
w_chars_fast_loop:
    movsb
    inc     di
    loop    w_chars_fast_loop
end_fast_proc      w_chars

begin_newrow_proc  w_chars
    sub     di,cx
    sub     di,cx
    add     di,scr_width
end_newrow_proc    w_chars
; W_CHAR_ATTRS: Write characters & attributes to the screen.
w_char_attr_setup_proc equ    w_chars_setup_proc ; No setup needed.
begin_slow_proc    w_char_attr,save_bx
    lodsw                      ; Insufficient time for a
    mov     bx,ax              ; whole "movsw", so do the job
    await_horiz_retrace        ; by a "lodsw" plus a "stosw".
    mov     ax,bx
    stosw
end_slow_proc      w_char_attr,save_bx
begin_fast_proc    w_char_attr
    rep     movsw
end_fast_proc      w_char_attr
w_char_attr_newrow_proc equ    w_chars_newrow_proc
; CONST_ATTR: Write characters to screen with constant attribute
; specified in AH.
begin_setup_proc   const_attr
    mov     ah,attr
end_setup_proc     const_attr
begin_slow_proc    const_attr,save_bx
    lodsb                      ; Insufficient time for "lodsb"
    mov     bl,al              ; plus "stosw", so do the job
    await_horiz_retrace        ; in halves.
    mov     al,bl
    stosw
end_slow_proc      const_attr,save_bx

begin_fast_proc    const_attr
const_attr_fast_loop:
    lodsb
    stosw
    loop    const_attr_fast_loop
end_fast_proc      const_attr

const_attr_newrow_proc equ    w_chars_newrow_proc

```



CONST\_CHAR\_ATTR: Fill rectangle with constant character  
and attribute (stored in SI).

```
begin_setup_proc    const_char_attr
    mov     ah,attr
    mov     al,[si]
    mov     si,ax          ; Keep char & attr in SI for the loop
end_setup_proc      const_char_attr
```

```
begin_slow_proc     const_char_attr
    await_horiz_retrace
    mov     ax,si
    stosw
end_slow_proc        const_char_attr
```

```
begin_fast_proc     const_char_attr
    mov     ax,si
    rep     stosw
end_fast_proc        const_char_attr
```

const\_char\_attr\_newrow\_proc equ w\_chars\_newrow\_proc

; CONST\_ATTR\_ONLY: Change attribute on rectangle to value  
stored in AH.

```
begin_setup_proc    const_attr_only
    mov     ah,attr
end_setup_proc      const_attr_only
```

```
begin_slow_proc     const_attr_only
    await_horiz_retrace
    inc     di
    mov     al,ah
    stosb
end_slow_proc        const_attr_only
```

```
begin_fast_proc     const_attr_only
    mov     al,ah
const_attr_only_fast_loop:
    inc     di
    stosb
    loop    const_attr_only_fast_loop
end_fast_proc        const_attr_only
```

const\_attr\_only\_newrow\_proc equ w\_chars\_newrow\_proc

; R\_CHARS: Read only characters from screen.

r\_chars\_setup\_proc equ w\_chars\_setup\_proc ; No setup needed.

```
begin_slow_proc     r_chars
    await_horiz_retrace
    movsb
```

```

        inc    si
end_slow_proc      r_chars

begin_fast_proc    r_chars
r_chars_fast_loop:
    movsb
    inc    si
    loop   r_chars_fast_loop
end_fast_proc      r_chars

begin_newrow_proc  r_chars
    sub    si,cx
    sub    si,cx
    add    si,scr_width
end_newrow_proc    r_chars

; R_CHAR_ATTRS: Read characters & attributes from screen.
r_char_attrs_setup_proc    equ    w_chars_setup_proc ; No setup needed.

begin_slow_proc    r_char_attrs
    await_horiz_retrace
    movsw
end_slow_proc      r_char_attrs

r_char_attrs_fast_proc    equ    w_char_attrs_fast_proc
r_char_attrs_newrow_proc  equ    r_chars_newrow_proc

; UP_CHAR_ATTRS: Copy characters & attributes upward on the screen
;                  or directly to the left. Start at upper left
;                  corner of rectangle.
up_char_attrs_setup_proc    equ    w_chars_setup_proc ; No setup needed.

begin_slow_proc    up_char_attrs,save_bx
    await_horiz_retrace
    lodsw                      ; Insufficient time for a
    mov    bx,ax              ; whole "movsw", so do the job
    await_horiz_retrace      ; by a "lodsw" plus a "stosw".
    mov    ax,bx
    stosw
end_slow_proc      up_char_attrs,save_bx

up_char_attrs_fast_proc    equ    w_char_attrs_fast_proc
begin_newrow_proc    up_char_attrs
    sub    si,cx
    sub    si,cx
    add    si,scr_width
    sub    di,cx
    sub    di,cx
    add    di,scr_width
end_newrow_proc      up_char_attrs

```

```

; UP_CHARS:      Copy only characters upward on the screen
;                or directly to the left. Start at upper left
;                corner of rectangle.
up_chars_setup_proc    equ    w_chars_setup_proc ; No setup needed.
    begin_slow_proc    up_chars
        await_horiz_retrace
        movsb
        inc    si
        inc    di
    end_slow_proc      up_chars

    begin_fast_proc    up_chars
up_chars_fast_loop:
    movsb
    inc    si
    inc    di
    loop   up_chars_fast_loop
end_fast_proc          up_chars

up_chars_newrow_proc    equ    up_char_attrs_newrow_proc

; DOWN_CHAR_ATTRS: Copy characters & attributes downward on the screen
;                  or directly to the right. Start at lower right
;                  corner of rectangle.

```

```

    begin_setup_proc    down_char_attrs
        mov    al,num_rows    ; Adjust SI and DI to point
        dec    al            ; at the last character in the
        mov    cx,scn_width   ; rectangle instead of the first.
        mul    cl
        mov    cl,row_length
        mov    ch,0
        dec    cx
        add    cx,cx
        add    ax,cx

        add    si,ax
        add    di,ax
    end_setup_proc      down_char_attrs

```

```

down_char_attrs_slow_proc    equ    up_char_attrs_slow_proc
down_char_attrs_fast_proc    equ    up_char_attrs_fast_proc

```

```

    begin_newrow_proc    down_char_attrs
        add    si,cx
        add    si,cx
        sub    si,scn_width
        add    di,cx
        add    di,cx

```

```

        sub    di,scn_width
end_newrow_proc    down_char_attrs

;   DOWN_CHARS:  Copy only characters downward on the screen
;               or directly to the right.  Start at lower right
;               corner of rectangle.
down_chars_setup_proc    equ    down_char_attrs_setup_proc

begin_slow_proc    down_chars
    await_heriz_retrace
    movsb
    dec    si
    dec    di
end_slow_proc    down_chars

begin_fast_proc    down_chars
down_chars_fast_loop:
    movsb
    dec    si
    dec    di
    loop   down_chars_fast_loop
end_fast_proc    down_chars

down_chars_newrow_proc    equ    down_char_attrs_newrow_proc

;   W_CHARS_GAP:  Write characters to the screen without attributes.

w_chars_gap_setup_proc    equ    w_chars_setup_proc ; No setup needed.
w_chars_gap_slow_proc     equ    w_chars_slow_proc
w_chars_gap_fast_proc     equ    w_chars_fast_proc

begin_newrow_proc    w_chars_gap
    sub    di,cx
    sub    di,cx
    add    di,scn_width
    add    si,gap
end_newrow_proc    w_chars_gap

;   W_CHAR_ATTRS_GAP:  Write characters & attributes to the screen.
w_char_attrs_gap_setup_proc equ    w_char_attrs_setup_proc ; No setup needed.
w_char_attrs_gap_slow_proc equ    w_char_attrs_slow_proc
w_char_attrs_gap_fast_proc equ    w_char_attrs_fast_proc

begin_newrow_proc    w_char_attrs_gap
    sub    di,cx
    sub    di,cx
    add    di,scn_width
    add    si,gap
    add    si,gap
end_newrow_proc    w_char_attrs_gap

```

```
;  CONST_ATTR_GAP:  Write characters to screen with constant attribute
;                    specified in AH.
```

```
const_attr_gap_setup_proc  equ    const_attr_setup_proc
const_attr_gap_slow_proc   equ    const_attr_slow_proc
const_attr_gap_fast_proc   equ    const_attr_fast_proc
const_attr_gap_newrow_proc equ    w_chars_gap_newrow_proc
```

```
;  R_CHARS_GAP:  Read only characters from screen.
```

```
r_chars_gap_setup_proc     equ    r_chars_setup_proc ; No setup needed.
r_chars_gap_slow_proc      equ    r_chars_slow_proc
r_chars_gap_fast_proc      equ    r_chars_fast_proc
```

```
begin_newrow_proc  r_chars_gap
    sub    si,cx
    sub    si,cx
    add    si,scn_width
    add    di,gap
end_newrow_proc    r_chars_gap
```

```
;  R_CHAR_ATTRS_GAP:  Read characters & attributes from screen.
```

```
r_char_attrs_gap_setup_proc equ    r_char_attrs_setup_proc ; No setup needed.
r_char_attrs_gap_slow_proc  equ    r_char_attrs_slow_proc
r_char_attrs_gap_fast_proc  equ    r_char_attrs_fast_proc
```

```
begin_newrow_proc  r_char_attrs_gap
    sub    si,cx
    sub    si,cx
    add    si,scn_width
    add    di,gap
    add    di,gap
end_newrow_proc    r_char_attrs_gap
```

```
; *****
; *****  DEFINE THE TABLE OF OPTION INFORMATION  *****
; *****
```

```
;  Values for dirflag:
```

```
down    equ    1          ; Decreasing addresses (up the screen)
up       equ    0         ; Increasing addresses (down the screen)
```

```
;  Macro to create one table entry:
```

```
o_e      macro    o_name,dir
option_entry
<o_name&_setup_proc,o_name&_slow_proc,o_name&_fast_proc,o_name&_newrow_proc,dir>
```

endm

; The table itself follows. It's part of the code segment.

option\_table:

end_of_first	label	option_entry
	o_e	w_chars,up ; 0
	o_e	w_char_attrs,up ; 1
	o_e	const_attr,up ; 2
	o_e	const_char_attr,up ; 3
	o_e	const_attr_only,up ; 4
	o_e	r_chars,up ; 5
	o_e	r_char_attrs,up ; 6
	o_e	up_char_attrs,up ; 7
	o_e	up_chars,up ; 8
	o_e	down_char_attrs,down ; 9
	o_e	down_chars,down ; 10
	o_e	up_char_attrs,up ; 11
	o_e	up_chars,up ; 12
	o_e	w_chars_gap,up ; 13
	o_e	w_char_attrs_gap,up ; 14
	o_e	const_attr_gap,up ; 15
	o_e	r_chars_gap,up ; 16
	o_e	r_char_attrs_gap,up ; 17

; (Note that options 11 and 12 just  
; refer to the procedures used by options  
; 7 and 8, respectively, because the  
; same algorithms suffice.)

end_of_table	label	option_entry
size_of_entry	equ	end_of_first - option_table
table_length	equ	(end_of_table - option_table)/size_of_entry

```
; *****  
; ***** BEGIN THE ACTUAL CODE *****  
; *****
```

beginProc vidirec0

push	bp	; Save the frame pointer
mov	bp,sp	
push	di	
push	si	
push	ds	
push	es	

```

    mov     ax,option
    mov     dx,ax           ; Spare copy of raw option
    and     ax,7fffh       ; Ignore bit 15 of option.

    cmp     ax,table_length ; Check for option value beyond table.
    jb     option_ok

bad_option:
    mov     ax,1           ; Invalid option
    jmp     short exit

option_ok:
    mov     cl,size_of_entry ; Convert option value into address
    mul     cl             ; of proper table entry.
    jc     bad_option
    add     ax,offset option_table
    mov     option,ax       ; Save address of table entry.
    mov     bx,ax           ; Set DF according to dirflag value
    cmp     cs:[bx].dirflag,down; in table.
    je     downward
    cld
    jmp     short have_set_direction

downward:
    std

have_set_direction:
    if      LONGDATA
    lds     bx,dword ptr ppto
    assume  ds:nothing
    else
    mov     bx,ppto
    endif
    les     di,[bx]         ; Put target address into ES:DI.
    assume  es:nothing

    if      LONGDATA
    lds     bx,dword ptr ppfrom
    assume  ds:nothing
    else
    mov     bx,ppfrom
    endif
    lds     si,[bx]         ; Put source address into DS:SI.
    assume  ds:nothing

    mov     bx,option

    call    cs:[bx].do_setup ; Do specialized portion
                                ; of setup.

                                ; Complete the setup.

```

```

        mov     cl,row_length
        mov     ch,0
        test    dh,80h
        mov     dx,video_status_port
        jnz     no_waiting

        call    cs:[bx].do_slow    ; Invoke slow_proc.
        jmp     short done

no_waiting:
        call    cs:[bx].do_fast    ; Invoke fast_proc.

done:
        xor     ax,ax              ; Successful completion

exit:
        sti
        pop     es
        pop     ds
        pop     si
        pop     di

        cld
        pop     bp                ; Get the original frame pointer.
        ret

        endProc vidirec0
        endCseg vidirec0
        endMod  vidirec0

        end

```

## VIHORIZ      在当前显示页上水平滚动正文列

```

#include <bvideo.h>

int vihoriz( int num_cols,
             int attrib,
             int u_row, int u_col,
             int l_row, int l_col,
             int dir);

num_cols    滚动的列数。值 0 消除该区域。
attrib      空列使用的显示属性(见下面)。
u_row, u_col 滚动区域的顶行左列。
l_row, l_col 滚动区域的底行右列。
dir         滚动方向(SCR_LEFT(1) 或 SCR_RIGHT (0)。
(返回)      实际滚动的列数。

```

VIHORIZ 在当前显示页的一个矩形区域中将一些字符列(连同属性)向左或向右移动, 空列由空格和指定属性填充。

attrib 是空列所用的显示属性。



必要时裁减矩形尺寸使之合于物理屏幕。

VIHORIZ NTEDD PB TWF YYAA(0, 1, 2, 3 或 7)下工作。

源程序(VIHORIZ.C):

```
#include <bcreens.h>
#include <bvideo.h>
int vihoriz(num_cols,attr,u_row,u_col,l_row,l_col,dir)
int num_cols,attr,u_row,u_col,l_row,l_col,dir;
{
    int device,mode,last_row,columns,act_page;
    int height,width;
    int command;
    char far *pfrom;
    char far *pto;
    const char blank = ' ';
    const char far * const pblank = &blank;
    int fast_mask;

    device = smode(&mode,&columns,&act_page);
    if (mode > 3 && mode != 7)
        return 0;
    last_row = scrrows() - 1;

    utbound(u_row,0,last_row) /* Force reasonable values */
    utbound(l_row,u_row,last_row)
    utbound(u_col,0,columns - 1)
    utbound(l_col,u_col,columns - 1)
    height = l_row - u_row + 1;
    width = l_col - u_col + 1;
    if (num_cols <= 0 || num_cols > width)
        num_cols = width; /* Do not scroll more columns */
                          /* than there are in the region.*/

    if ( b_vifast != 0
        || mode == 7
        || scequip() == IBM_CV
        || device == b_ega
        || device == b_vga
        || device == b_mega)
        fast_mask = (int) 0x8000; /* Need not await retrace */
    else
        fast_mask = 0x0000;

    if (num_cols < width)
    {
        /* We're not blanking the whole */
        /* region. */
        if (dir == SCR_LEFT)
        {
            /* Leftward */
            command = 7;
            pfrom = viptr(u_row,u_col + num_cols);
```

```

        pto = viptr(u_row,u_col);
    }
    else
    {
        /* Rightward */
        command = 9;
        pfrom = viptr(u_row,u_col);
        pto = viptr(u_row,u_col + num_cols);
    }

    vidirect(&pfrom,&pto,height,width - num_cols,
            columns * 2,0,command | fast_mask);
}
else
    dir = SCR_LEFT;

    /* Shortcut to force */
    /* computation of pto. */

    /* Now blank out the new columns in the region. */
    if (dir == SCR_LEFT)
    {
        /* Leftward */
        /* Address of first column to */
        pto = viptr(u_row,u_col + width - num_cols); /* blank */
    }
    else
    {
        /* Rightward */
        pto = pfrom; /* pfrom already points to */
        /* first column to blank */
    }
    vidirect(&pblank,&pto,height,num_cols,
            columns * 2,attr,
            3 | fast_mask);
    return num_cols;
}

```

## VIPTR 将屏幕位置转换成内存地址

```

#include <bvideo.h>
char far *viptr( int row,
                int col);
row      行 (0 = 屏幕顶行)。
col      列 (0 = 左边沿)。
(返回)   计算所得的地址。如果有错误, 返回 FARNIL。

```

VIPTR 计算当前显示页上指定位置在显示内存中的物理地址。

如果屏幕未处于字符方式(0、1、2、3、7)下或 row、col 超出屏幕的边沿,则出现一个错误,FARNIL 成为返回值。

### 源程序(VIPTR.C)

```

#include <dos.h>
#include <bscreens.h>
#include <bvideo.h>
int b_vifast = 0; /* Nonzero if interference is */

```

```

/* to be ignored - use fastest*/
/* possible method. */

char far *viptr(row,col)
int row,col;
{
    int mode,act_page,columns;
#define CRT_LEN_SEG    0x0000    /* Address of BIOS variable */
#define CRT_LEN_OFFSET 0x044c    /* CRT_LEN which contains */
/* length of video page */
    scmode(&mode,&columns,&act_page);
    if ((mode > 3 && mode != 7)    /* Quit if graphics mode or */
        || col < 0                /* if col or row out of range */
        || col >= columns
        || row < 0
        || row >= scrrows())
        return (char far *) NIL;
    /* Construct address: segment is beginning of screen buffer, */
    /* offset is based on page length, page number, row, and column. */
    /* Note: pg_len may be incorrect for Monochrome Adapter but that */
    /* should have no effect since b_curpage should be 0. */
#define pg_len (*utfar(CRT_LEN_SEG,CRT_LEN_OFFSET,unsigned))
    return utfar( ((mode == 7) ? 0xb000 : 0xb800),
        ((row * columns) + col) * 2 + b_curpage * pg_len,
        char);
}

```

## VIRDRECT 读取屏幕上一个矩形区域中的内容

```

#include <bvideo.h>
int virdirect( int u_row, int u_col,
               int l_row, int l_col,
               char *pbuffer,
               int option);
u_row, u_col  矩形区域的顶行左列。
l_row, l_col  区域的底行右列。
pbuffer       返回的数据所占用的空间。
option        如果缓冲区包含 (char, attr) 9 对, 为 CHAR_ATTR (2); 如果缓冲区只包含字符, 则为
CHARS_ONLY (0)。
(返回)       读取的物理空间的字节数。

```

VIRDRECT 是一个宏, 它读取当前显示页上一个矩形区域的内容。读时可带有也可不带有显示属性, 光标保持不动。数据读取以一行接一行的方式进行。

当前显示适配器必须处于字符方式(0, 1, 2, 3, 7)下。

必要时裁减矩形的尺寸使之能够合于物理屏幕。

如果指定了 CHAR\_ATTR 并且屏幕处于图形方式下, 则不论显示字符的颜色如何, 1 将作为属性值返回。

VIRDRECT 不在缓冲区的末尾加入尾随的 NUL ('\0'), 缓冲区中可以有任何 8 位字符。

源程序(BVIDEO.H);

该宏定义在 BVIDEO.H 中, 所有的头文件都罗列在附录 C 中。

```
#include <bvideo.h>
```

```
int virdsect( int u_row, int u_col,
              int l_row, int l_col,
              char *pbuffer,
              unsigned gap,
              int option);
```

u\_row, u\_col 矩形区域的顶行左列。

l\_row, l\_col 区域的底行右列。

pbuffer 返回数据所占用的空间。

gap 缓冲区中两行之间的空 1 (以空格数计)。

option 如果缓冲区包含 (char, attr) 9 对, 为 CHAR\_ATTR (2); 如果缓冲区只包含字符, 则为 CHARS\_ONLY (0)。

(返回) 读入的物理空间字节数。

VIRDSECT 读取当前显示页上一个矩形区域的内容, 读时带有也可不带有属性, 光标保持不动。数据读取以一行接一行的方式进行。

这个函数与 VIRDRECT 相似, 不同的是它可以处理缓冲区的一段, 该缓冲区代表一个更大的矩形区域。例如, 用连续的缓冲区字节代表一个字符矩形区域, 那么这个函数可以从屏幕读取缓冲区的一个矩形子段, gap 指明读屏幕时缓冲区中行之间需跳过的字符单元数。

当前显示适配器必须处于字符方式 (0、1、2、3、7) 下。

必要时裁减矩形的尺寸使之能够合于物理屏幕。然而, 这样做有可能导致数据的错放。

如果指明了 CHAR\_ATTR 而屏幕处于图形方式下, 则不论显示字符的颜色如何, 1 将作为属性值返回。

VIRDSECT 不在缓冲区的末端加入尾随的 NUL ('\0'), 在缓冲区中可以出现任何 8 位字符。

源程序(VIRDSECT.C):

```
#include <bcreens.h>
```

```
#include <bvideo.h>
```

```
int virdsect(u_row,u_col,l_row,l_col,buffer,gap,option)
```

```
int u_row,u_col,l_row,l_col;
```

```
char *buffer;
```

```
unsigned gap;
```

```
int option;
```

```
{
```

```
    int device,mode,act_page,last_row,columns;
```

```
    int height,width;
```

```
    int command;
```

```
    char far *pbuf;
```

```
    const char far *pscreen;
```

```
    device = scmode(&mode,&columns,&act_page);
```

```
    if (mode > 3 && mode != 7)
```

```
        return 0;
```

```
    last_row = scrows() - 1;
```

```
    utbound(u_row,0,last_row)
```

```
/* Force reasonable values */
```

```
    utbound(l_row,u_row,last_row)
```

```
    utbound(u_col,0,columns - 1)
```

```

atbound(l_col,u_col,columns - 1)
height = l_row - u_row + 1;
width  = l_col - u_col + 1;

pbuf    = buffer;
pscreen = vptr(u_row,u_col);

command = ((option & CHAR_ATTR) ? 17 : 16);
if (    b_vfast != 0
    || mode      == 7
    || seequip() == IBM_CV
    || device    == b_ega
    || device    == b_vga
    || device    == b_mega)
    command |= 0x8000;          /* Need not await retrace */

vidirec0t(&pscreen,&pbuf,height,width,columns * 2,
          0,command,gap);
return height * width;
}

```

## VIScroll 垂直滚动当前显示页上的正文行

```

#include <bvideo.h>
int viscroll( int num_rows,
              int attrib,
              int u_row, int u_col,
              int l_row, int l_col,
              int dir);

```

**num\_rows** 滚动的行数。值 0 清除该区域。  
**attrib** 空行使用的属性(见下面)。  
**u\_row, u\_col** 滚动区域的顶行左列。  
**l\_row, l\_col** 滚动区域的底行右列。  
**dir** 滚动方向 (SCR\_UP (0) 或 SCR\_DOWN (1))。  
**(返回)** 实际滚动的行数。

VIScroll 在当前显示页的一个矩形区域中向上或向下移动字符行(连同其属性)，空白行由空格和指定属性填充。

如果屏幕处于字符方式，attrib 是空行使用的显示属性；如果屏幕处于图形方式并且当前页是不活动的，attrib 是滚动正文的颜色；如果屏幕处于图形方式而当前页是活动的，则 attrib 被忽略。

VIScroll 在图形方式下不能滚动一个非活动页，在这种情况下，函数的返回的值为零。

必要时裁减矩形尺寸使之能够合于物理屏幕。

源程序(BVIDEO.H):

该宏定义在附录 C 的 BVIDEO.H 文件中。

## VIWRRECT 向当前显示页上一个矩形区域中写入数据

```

#include <bvideo.h>
int viwrrect( int u_row, int u_col,

```

```

        int l_row, int l_col,
        const char *pbuffer,
        int fore, int back,
        int option);
u_row, u_col  矩形区域的顶行左列。
l_row, l_col  区域的底行右列。
pbuffer      待写入的数据。
fore, back   前景和背景属性 (见下面)。
option       如果缓冲区包含(char, attr) 9 对, 为 CHAR_ATTR (2); 如果缓冲区仅包含字符而 fore
和 back 用作属性, 则为 CHARS_ONLY (0)。
(返回)       写入的物理空间的字节数。

```

VIWRRECT 是一个宏, 它用缓冲区的字符填充当前显示页上的矩形区域。填充时可以使用也可不使用相应的显示属性。数据一行接一行地写入, 光标保持不动。所有字符, 包括 NUL('\0'), 均被显示。必要时裁减矩形的尺寸使之能够合于物理屏幕。

屏幕必须处于字符方式下。

如果 option 是 CHARS\_ONLY, 则指定 fore 和 back 为-1 可以保留原有的属性。

源程序(BVIDEO.H);

参见附录 C 头文件 BVIDEO.H 中该宏的定义。

## VIWRSECT 显示矩形缓冲区中的一个矩形段

```

#include <bvideo.h>
int viwrsect( int u_row, int u_col,
              int l_row, int l_col,
              const char *pbuffer,
              unsigned gap,
              int fore, int back,
              int option);
u_row, u_col  矩形区域的顶行左列。
l_row, l_col  区域的底行右列。
pbuffer      待写入的数据。
gap          缓冲区两行之间的空 1 (以空格个数计)。
fore, back   前景和背景属性(见下面)。
option       如果缓冲区包含(char, attr) 9 对, 为 CHAR_ATTR (2); 如果缓冲区只包含字符而 fore
和 back 用作属性, 则为 CHARS_ONLY (0)。
(返回)       写入的物理空间的字节数。

```

VIWRSECT 用缓冲区的字符填充当前显示页上的一个矩形区域, 填充时可使用也可不使用相应的显示属性。数据一行一行地写入, 光标保持不动, 所有字符, 包括 NUL('\0'), 均被显示。必要时裁减矩形的尺寸使之合于物理屏幕, 但这有可能导致数据的错放。

这个函数与 VIWRRECT 相似, 不同的是, 该函数可以处理缓冲区中的一段。例如, 用连续的缓冲区字节代表一个矩形区域, 那么该函数能够将这个缓冲区的一个矩形子段写入屏幕, gap 表示写入行之间要跳过的字符单元的数量。

屏幕必须处于字符方式下。

如果 option 是 CHARS\_ONLY, 则指定 fore 和 back 为-1 可以保留原有的属性。

源程序(VIWRSECT.C);

```

#include <bSCREENS.h>
#include <bvideo.h>
int viwrsect(u_row, u_col, l_row, l_col, buffer, gap, fore, back, option)

```

```

int u_row,u_col,l_row,l_col;
const char *buffer;
unsigned gap;
int fore,back,option;
{
    int device,mode,act_page,last_row,columns;
    int height,width;
    int command;
    const char far *pbuf;
    char far *pscreen;

    device = scmode(&mode,&columns,&act_page);
    if (mode > 3 && mode != 7)
        return 0;
    last_row = screws() - 1;

    utbound(u_row,0,last_row)          /* Force reasonable values */
    utbound(l_row,u_row,last_row)
    utbound(u_col,0,columns - 1)
    utbound(l_col,u_col,columns - 1)
    height = l_row - u_row + 1;
    width = l_col - u_col + 1;

    pbuf = buffer;
    pscreen = viptr(u_row,u_col);

    if (option & CHAR_ATTR)
        command = 14;                /* (char,attr) pairs */
    else
        if (fore == -1 && back == -1)
            command = 13;            /* Preserve attributes */
        else
            command = 15;            /* Use fore and back as
                                     /* constant attributes. */

    if ( b_vifast != 0
        || mode == 7
        || seequip() == IBM_CV
        || device == b_ega
        || device == b_vga
        || device == b_mega)
        command |= 0x8000;           /* Need not await retrace */

    vidirec0(&pbuf,&pscreen,height,width,columns * 2,
            utnybbyt(back,fore),command,gap);
    return height * width;
}

```

## 第十三章 字符串处理

所有Turbo C TOOLS字符串函数处理并返回标准的C字符串,即char类型的数组,字符串的末尾跟一个NUL字符('\0')作为标记。如必要,字符串函数使用一个名为tarsize的参数,作为目标字符串的长度。目标字符数组必须至少为tarsize字节长。而由于后续的NUL的缘故,作为结果的字符串的最大长度必须比tarsize小1个字节。

当使用一个整数值指示字符串中的某个位置时,0总是指向字符串的第一个字符,1指示第二个,等等。

当涉及空字符串时,我们指的是长度为零的字符串(即""),请不要与NUL字符('\0')混淆,也不要与指向数据地址0的NULL指针混淆。(在本书的所有源代码中,NULL指针写为“NIL”。

### 字符串函数的种类

#### 对字符串的一部分进行填充

STPJUST 在指定的空间中对一个字符串进行居中或对齐(左对齐或右对齐)操作。

#### 查找

STSCHIND 在一个字符串中查找一个字符的第一次出现,返回它的位置。

#### 字符转换

STPXLATE 用一个翻译表翻译一个字符串的每一个字符。

STPCVT 执行一种或几种类型的转换:转换为大写或小写字母,删除开头、结尾或所有的空白,将多个连续空白压缩为一个空格,删除控制字符等。该函数可以任选地使括在引号(")或省略号( )中的子串不受影响。

#### 处理tab字符

STPEXPAN 将tab字符扩展为空格。

STPTABFY 将连续多个空格压缩为tab字符。

### 字符串处理扩展函数源程序

STPCVT 常用字符串转换

```
#include <bstrings.h>
```

```
char *stpcvt( char *psource,  
              int conv);
```

psource 指向源字符串的指针。

conv 转换码(见下面)。

(返回) 指向转换后的字符串的指针(与psource相同)。

STPCVT根据conv指定的命令对字符串\*psource进行转换,返回指向已转换字符的指针。字符串\*psource在原位置被转换而未被移走。

可能的转换码是:

符号	值	意义
RWHITE	1	删除所有空白字符
RLWHITE	2	删除所有前导空白字符
RTWHITE	4	删除所有尾随空白字符
REDUCE	8	将所有空白字符压缩为一个空格字符



NOQUOTE	16	括在省略号( )或引号(")中的字符不作转换
TOUP	32	将小写字符转换为大写字符
TOLOW	64	将大写字符转换为小写字符
RCONTROL	128	删除所有控制字符

转换码可以通过位操作运算符OR(|)组合起来。除了NOQUOTE之外,转换码按上表所列顺序执行。例如,33(即TOUP RWHITE)取消所有空白字符并将所有小写字符转换为大写字符。转换码49做同样的事情,但不转换括起来的子串。

空白和控制字符由编译器的isspace( )和isctrl( )函数或宏定义,然而,除非编译器的issacii( )函数或宏报告它们是ASCII字符(其码小于128),否则它们也不被认为是空白或控制字符。所以,特殊的具有ASCII码大于127的IBM字符保持不变。

如果出现了不平衡的括起来的子串,则NOQUOTE不能保护尾随的空白字符。

## 源程序(STPCVT.C):

```
#include <ctype.h>                /* Definition of char manipulation */
                                   /* macros. */
#include <bstrings.h>
#define TRUE      1
#define FALSE     0
#define BLANK     ' '

char *stpcvt(psource,conv)
char *psource;
int conv;
{
    char *pfrom      = psource;    /* Next character to get */
                                   /* from source. */
    register char *pto = psource;   /* Next position to fill in */
                                   /* target. */
    register char c;
    char quote_char = '\0';

    /* Convenient flags (not to be changed): */

    int rlwhite      = conv & RLWHITE;
    int rwhite       = conv & RWHITE;
    int reduce       = (!rwhite) && (conv & REDUCE);
    int ckquotes     = conv & NOQUOTE;
    int to_up        = conv & TOUP;
    int to_low       = conv & TOLOW;
    int rcontrol     = conv & RCONTROL;

    /* Flags indicating current state as we process the string */
    /* (these change as we encounter various characters): */

    int in_white     = FALSE; /* Not in a white field yet. */
    int hit_nonwhite = FALSE; /* No nonwhite chars encountered. */
    int quote_on     = FALSE; /* Not in a quote yet. */

    /* This main loop handles everything but trailing white space. */
}
```

```

/* Each pass handles one character c from psource, writing zero */
/* or one character back to psource. We can process psource in */
/* place this way because we're never adding additional chars. */

while ((c = *pfrom++) != '\0')
{
    if (quote_on)
    {
        /* We're inside a quoted */
        *pto++ = c; /* substring. */
        if (c == quote_char)
            quote_on = FALSE; /* Found end of the quote. */
    }
    else if (ckquotes && ((c == '"') || (c == '\')))
    {
        /* Begin new quote. */
        *pto++ = c;
        in_white = FALSE;
        hit_nonwhite = TRUE;
        quote_on = TRUE;
        quote_char = c;
    }
    else if (isspace(c) && isascii(c))
    {
        if (rwhite)
            ; /* Do nothing with character c. */

        else if (rlwhite && !hit_nonwhite)
            ; /* Leading white space to omit-- do nothing */
            /* with character c. */

        else if (reduce)
        {
            if (in_white)
                ; /* We're within white block-- do nothing */
                /* with character c. */

            else
            {
                /* Begin block of white space. */
                *pto++ = BLANK;
                in_white = TRUE;
            }
        }

        else if (rcontrol && iscntrl(c))
            ; /* Remove control characters which are */
            /* defined as whitespace. */

        else
            /* Just copy normally. */
            *pto++ = c;
    }
}

```

```

else if (isctrl(c) && isascii(c))
{
    in_white      = FALSE;
    hit_nonwhite = TRUE;

    if (rcontrol)
        ; /* Don't copy the ctrl char. */

    else
        /* Just copy normally. */
        *pto++ = c;
}
else
{
    /* Simple nonwhite or */
    /* control character. */
    in_white      = FALSE;
    hit_nonwhite = TRUE;

    if (isascii(c))
    {
        if (to_up)
            c = toupper(c);

        if (to_low)
            c = tolower(c);
    }

    *pto++ = c; /* Just copy normally. */
}
}
*pto = '\0';

/* Now handle trailing white space... */

if (conv & RTWHITE)
    for (c = *--pto;
         isspace(c) && isascii(c) && (pto >= psource);
         c = *--pto)
        *pto = '\0';

return (psource);
}

```

## STPEXPAN 将tab字符转换为空格

```

#include <bstrings.h>
char *stpexpan( char *ptarget,
                char *psource,
                int incr,
                int tarsize);

```

ptarget,psource

指向目的和源字符串的指针, 这些字符串不能互相重叠。

incr           tab站位间隔。

tarsize        目的字符串的最大尺寸(计入结尾NUL ('\0')字符)。

(返回)        如果转换完成, 返回NIL; 否则返回指向源中下一个字符的指针。

STPEXPAN将tab字符('\t')用足够的空格(' ')替换, 造成与tab同样的视觉效果。

tab站位定义如下: 如果incr是正值, 则从位置0 (即一行的第一个字符)开始, 每incr个字符一个站位; 如果incr是0或负数, 则不设置tab站位, 源字符串中的任何tab字符原封不动地拷贝到目的字符串中。

STPEXPAN认为psource中的所有字符(除了tab和NUL)是正文字符, 在ptarget中占有一个位置。

用STPTABFY可以将多个空格压缩为tab

源程序(STPEXPAN.C):

```
#include <bstrings.h>
#include <butil.h>
#define BLANK ' '
#define NEWLINE '\n'
#define TAB '\t'
#define NUL '\0'
char *stpexpan(ptarget,psource,incr,tarsize)
register char *ptarget;
char *psource;
int incr,tarsize;
{
    char c;
    register int len;          /* Running total of characters put */
                                /* into ptarget, also index of next */
                                /* char in ptarget. */
    int numspaces;
    len = 0;
    tarsize--;
    while (((c = *psource) != NUL) && (len < tarsize))
    {
        switch (c)
        {
            case TAB:
                if ((incr > 0) &&
                    ((numspaces = (incr - (len % incr))) != 0))
                {
                    if ((len += numspaces) < tarsize)
                    {
                        /* There's enough room. */
                        while (numspaces--)
                            *ptarget++ = BLANK;
                        break;
                    }
                }
                else
                {
                    /* There isn't enough room, */
                    continue; /* so quit. */
                }
            }

            /* Else TAB expansion is not in effect: */
        }
    }
}
```

```

        /* just fall through and copy the TAB to ptarget.          */
        default:
            *ptarget++ = c;
            len++;
            break;
    }
    psource++;
}
*ptarget = '\0';
return (c ? psource : NIL);
}

```

## STPJUST 在域中将一个字符串左右对齐或居中

```

#include <bstrings.h>
char*stpjust( char *ptarget,
              const char *psource,
              char fill,
              int fldsize,
              int code);

```

ptarget 指向目的字符串的指针，该字符串必须至少为(fldsize+1)字节长，以容纳尾随的NUL("\0")。

psource 指向源字符串的指针。

fill 填充字符。

fldsize 待填充的域的尺寸。

code 对齐类型：JUST\_LEFT (-1)、JUST\_CENTER (0) 或JUST\_RIGHT (1)。

(返回) 指向作为结果的目的字符串的指针。

STPJUST在指定尺寸的域中将一个字符串左右对齐或居中，必要时用给定的字符填充。结果字符串的长度为fldsize。

如果源字符串具有比fldsize更多的字符，则源字符串被截断以便装入目的字符串中。源字符串的左部、右部或中部字符根据指定的左对齐、居中、右对齐的不同操作而分别加以使用。

如果源字符串比fldsize字节数少(不计结尾的NUL("\0")), 则剩余的空间用填充字符填充。如果指定了左对齐，填充在右侧进行；如果指定了右对齐，填充在左侧进行；如果是居中，则填充在两侧进行。

源程序(STPJUST.C):

```

#include <string.h>
#include <bstrings.h>
char *stpjust(ptarget,psource,fill,fldsize,code)
register char *ptarget;
const register char *psource;
char fill;
int fldsize,code;
{
    register int diff,i;
    int numleft;
    char *savetarget = ptarget;

```

```

if (fldsize < 0)
    fldsize = 0;

if ((diff = ((int) strlen(psource)) - fldsize) >= 0)
{
    /* Use only a portion of source */
    switch (code)
    {
        case JUST_RIGHT:      /* Skip leftmost characters */
            psource += diff;
            break;
        case JUST_CENTER:    /* Use center characters */
            psource += diff >> 1;
            break;
        case JUST_LEFT:      /* Use leftmost characters */
        default:
            break;
    }
    while (fldsize--)
        *ptarget++ = *psource++;
}
else
{
    /* There's extra space to fill */
    diff = -diff;      /* diff is number of spaces to fill */
    switch (code)
    {
        /* numleft = number of spaces on left */
        case JUST_RIGHT:
            numleft = diff;
            break;
        case JUST_CENTER:
            numleft = diff / 2;
            break;
        case JUST_LEFT:
        default:
            numleft = 0;
            break;
    }
    for (i = numleft; i; i--)
        *ptarget++ = fill;      /* Add the fill chars on the left */
    while (*psource)
        *ptarget++ = *psource++; /* Copy the string itself */
    for (i = diff - numleft; i; i--)
        *ptarget++ = fill;      /* Add the fill chars on the right */
}
*ptarget = '\0';
return savetarget;
}

```

## STPTABFY      用tab字符替换空格

```
#include <bstrings.h>
```

```
char *stptabfy (    char *psource,
                  int incr);
```

psource        指向源字符串的指针。

incr            tab站位间隔。

(返回)        指向已转换的源字符串的指针。

STPTABFY将几组连续空格(' ')替换成tab字符('\t'), 这样有可能减少内存中一个字符串的长度, 结果字符串写回到源字符串。tab字符的使用服从最少原则: 如果用一个空格就能达到目的, 则使用空格而不用tab。

tab站位定义如下: 如果incr为正, 则从位置0(即一行的第一个字符)开始, 每隔incr个字符一个站位。如果incr为零或负值, 则每个位置一个站位。在这种情况下, 原封不动地拷贝源字符串中的所有空格和tab。

原有的tab字符不受影响, 按照前进到下一个tab站位处理。

STPTABFY将psource中的所有字符(除了tab和NUL)看作图形字符, 这些字符在结果字符串中各占据一个位置。

用STPEXPAN可以将tab扩展为空格。

源程序(STPTABFY.C):

```
#include <bstrings.h>
#define BLANK ' '
#define NEWLINE '\n'
#define TAB '\t'
char *stptabfy(psource,incr)
char *psource;
int incr;
{
    char c;
    register int col = 0;          /* Column counter (module incr). */
    register int numblanks = 0; /* Number of blanks we've saved up. */

    char *pfrom = psource;
    register char *pto = psource;

    do
    {
        switch (c = *pfrom++)
        {
            case BLANK:
                numblanks++;
                col++;
                if ((incr <= 0) ||
                    (col % incr == 0))
                {
                    *pto++ = (char) ((numblanks > 1) ? TAB : BLANK);
                    numblanks = 0;
                }
                break;

            case TAB:
                col =
```

```

        numblanks = 0;          /* Discard the saved blanks */
        *pto++ = TAB;
        break;

    default:
        col++;
        for (; numblanks; numblanks--)
            *pto++ = BLANK;      /* Spill any saved blanks */
        *pto++ = c;
        break;
    }
} while (c);

return (psource);
}

```

## STPXULATE 用翻译表翻译一个字符串

```

#include <bstrings.h>
char *stpulate(    char *psource,
                  const char *ptable,
                  const char *ptrans);

```

psource 指向源字符串的指针。

ptable 指向待翻译字符串表的指针。

ptrans 指向翻译字符串的指针。

(返回) 指向已翻译字符串的指针(与psource相同)。

STPXULATE根据\*ptable和\*ptrans指定的命令翻译字符串\*psource, 结果放在与源字符串相同的位置。psource的拷贝作为函数值返回。

如果\*psource是空字符串(""), 则返回空字符串, 否则源字符串中的每个字符psource[k]按照下面的方法转换:

如果psource[k]不在ptable中, 保持不变;

如果ptable中psource[k]的位置长度比ptrans的大, psource[k]由空格(' ')替换; 否则psource[k]被trans[j]所替换, 此处j是ptable中psource[k]的位置。

源程序(STPXULATE.C):

```

#include <stdio.h>          /* For definition of NULL. */
#include <string.h>
#include <bstrings.h>
#define BLANK ' '
#define NUL '\0'
char *stpulate(psource,ptable,ptrans)
register char *psource;
const char *ptable;
const char *ptrans;
{
    register int tindex, i;
    int len_trans;
    char ch, *ptemp;

```



```

len_trans = (int) strlen (ptrans);      /* Length of ptrans.      */
ptemp     = psource;

for (i = 0; ((ch = *psource) != NUL); i++)
{
    tindex = stschind (ch, ptable);
    if (tindex >= 0)                /* psource[i] is in ptable. */
        if ((tindex + 1) > len_trans)/* Indices start from zero. */
            *psource = BLANK;
        else
            *psource = ptrans[tindex];
    psource++;
}

return (ptemp);
}

```

## STSCHIND 查找字符串中的一个字符，返回它的位置

```
#include <bstrings.h>
int stschind(    char check,
                const char *psearch);
```

check 待查找的字符。psearch指向被查找字符串的指针。

(返回) 在\*psearch中查到的位置(如果在开头，为0；如果未找到，为-1)。

STSCHIND查找字符串\*psearch中的一个字符，返回其第一次出现的位置。如果该字符未出现，返回-1。

STSCHIND 能区分大小写字母。例如，'C'与'c'不同。

源程序(STSCHIND.C):

```

#include <stdio.h>      /* For definition of NULL.      */
#include <string.h>
#include <bstrings.h>
int stschind (check, psearch)
char        check;
const char *psearch;
{
    const char *p;

    if ((p = strchr (psearch, check)) == NULL)
        return (-1);

    return ((int) (p - psearch));
}

```

## 第十四章 实用函数和宏

实用函数和宏提供了很有价值的各种程序所需的服务, 包括对指针、地址和数据项的处理, 对程序的操作环境信息以及对时钟和扬声器的访问等。

### 实用函数的种类

标记有(R)的函数和宏在本手册中均有详细的使用说明, 其它未标记的函数和宏在`buil.h`中定义。

### 指针和地址

NIL		是指向数据地址0的指针, 用于表示一个非法指针值。它与NULL相同。
FARNIL		是一个指向0:0的双字指针。
UTOFF	(R)	返回任意地址或指针表达式的偏移部分。
UTSEG	(R)	返回任意地址或指针表达式的段部分。
UTTOFAR	(R)	对于给定的段和偏移地址及数据类型构造一个双字指针。
UTTOFARU	(R)	对任意指针进行标准化, 使之具有0到15的偏移值。
UTPLONG	(R)	将任意指针转换为一个20位物理指针。

### 内存传送

UTPEEKB	(R)	读取任意给定地址位置的数据字节。
UTPOKEB	(R)	在给定地址放一个数据字节。
UTPEEKW	(R)	读取任意给定位置的数据字。
UTPOKEW	(R)	在给定地址存放一个数据字。
UTPEEKN	(R)	将任意给定地址位置的多个数据字节读入到一个局部变量或数组中。
UTPOKEN	(R)	将多个数据字节放入内存中的任意位置。
UTMOVMEM	(R)	不受限制地从或向内存任意位置拷贝数据。
UTCOPY	(R)	拷贝一个给定类型的变量或结构。
UTSAFCPY	(R)	拷贝内存中的任意区域, 但如果该区域经过段界回绕到段的起始位置, 则不拷贝。

### 探测指针错误

UTNULCHK	(R)	检测中否将数据地址0作为存放数据的位置。
UTCHKNIL	(R)	检查数据地址0。如果这个地址被用于存放数据, 则使程序夭折。这个宏需要 <code>stdio.h</code> 。

### 数据压缩

UTSQZSCN	(R)	压缩一个屏幕图象。
UTUNSQZ	(R)	还原一个由UTSQZSCN压缩的屏幕图象。

### ANSI.SYS支持

UTANSI	(R)	检测 <code>ansi.sys</code> 的存在, 可任选地使之无效或有效。
--------	-----	--

### 程序环境信息

UTSPSEG		返回程序段前缀的段地址。这个宏需要的 <code>dos.h</code> 或 <code>process.h</code> 。
UTMODEL	(R)	报告IBM型号(PC、XT、AT、PS/2、PCjr等)、子型号和BIOS版本

		级别。
UTCRT	(R)	用DOS临界段标志的地址装载全局变量b_critad, 以使UTDOSRDY使用。
UTDOSRDY	(R)	报告DOS服务是否可用。
UTCTLBRK	(R)	报告或修改DOS的Ctrl-Break和Ctrl-C的检查状态。(Ctrl-Break标志控制是在每个DOS功能调用还是仅在进行键盘或控制台I/O时检查Ctrl-Break和Ctrl-C。
UTDOSVER		将DOS版本号作为一个unsigned int返回。这个宏需要dos.h。
UTDOSMAJOR		返回DOS版本号的主要部分。这个宏需要的dos.h。
UTDOSMINOR		返回DOS版本的次要部分。这个宏需要dos.h。

## 口I/O

UTINP	从一个给定的硬件I/O口读入一个数据字节。这个宏需要dos.h。
UTOUTP	将一个数据字节输出到一个给定的硬件I/O口。这个宏需要dos.h。

## 时钟访问

UTGETCLK	(R)	返回自午夜以来BIOS的时钟脉冲计数。
UTSLEEP	(R)	挂起一个程序的执行, 直至给定的时钟脉冲计数出现。
UTTK2TIM	(R)	将自午夜以来的时钟脉冲计数转换为24时制时间。
UTTIM2TK	(R)	将24制时间转换为对应的自午夜以来的时钟脉冲计数。

## 关闭或开放中断

UTINTFLG	(R)	关闭或打开硬件中断, 返加中断标志的原状态。
UTINTOF		关闭硬件中断, 返回中断标志的原状态。
UTINTON		打开硬件中断, 返加中断标志的原状态。

## 扬声器控制

UTSPKR	(R)	以指定的频率打开扬声器或关闭扬声器。
UTSPKON		以指定的频率打开扬声器。
UTSPKOFF		关闭扬声器。
UTSOUND		以给定的持续时间产生指定频率的声音。
UTALARM		以450赫兹的频率发声1/2秒。

## 算术计算

UTMAX		返回两个数中的较大者。
UTMIN		返回两个数中的较小者。
MAX		返回两个数的较大者。(如果max()已被定义为一个宏, butil.h不重新定义它。)
MIN		返回两个数的较小者。(如果min()已被定义为一个宏, 则butil.h不再定义它。)
UTABS		返回一个数的绝对值。
UTSIGN		返回一个数的符号: -1、0或1。

## 在长字节、字、字节和半字节之间转换

UTHIWORD	返回一个长整数值的高字部分。
UTLOWORD	返回一个长整数值的低字部分。
UTWDLONG	用两个字值构造一个长整数值。
UTHIBYTE	返回一个长整数值的高字节部分。
UTLOBYTE	返回一个值的低字节。
UTBYWORD	用两个字节值构造一个字。

UTHINYB	返回一个值的低字节的高半字节(四位)。
UTLONYB	返回一个值的低半字节(低四位)。
UTNYBBYT	用两个低半字节(四位)值构造一个字节值。

## 设置范围界限

UTRANGE	报告一个值是否超出给定范围。
UTBOUND	强制一个值进入给定范围。
UTUPLIM	防止一个值超出给定界限。
UTLOWLIM	强制一个值等于或超过给定的下限。

注意: UTBOUND、UTUPLIM和UTLOWLIM不返回值, 它们扩展为一个或多个简单或组合的语句。

## 数据类型操作

UTALLOC	利用calloc()为一个给定类型的变量分配空间, 返回指向该变量的指针。
---------	---------------------------------------

## 各种输出

UTSKIP	对表达式printf("/n")求值, 返回其值。(也就是说, UTSKIP向标准输出设备写入一个换行符。)这个宏需要stdio.h。
--------	---

## 实用函数源程序

### UTAMOVE 复制内存

```
utamove (source, target, num, direction);
```

source           复制的源

target           复制的目的地。

num              复制内存的量(以字节计)。

direction        MVFORWARD (0) 和MVBACKWARD方向标志。

该函数从内存中的任一位置拷贝内存到另外的任一位置。这并不是一个快速移动的子程序; 它不能处理覆盖——使用UTMOVMEM来完成。

返回            无

源程序(UTAMOVE.ASM):

```
include beginasm.mak
beginProg utamove

    MVFORWARD equ 0           ; Direction flag settings
    MVBACKWARD equ 1

source equ    dword ptr [bp + stkoff + 0]
target equ    dword ptr [bp + stkoff + 4]
num equ      word  ptr [bp + stkoff + 8]
direct equ    word  ptr [bp + stkoff + 10]
```

```
    push    bp
    mov     bp, sp
    push    ds
    push    es
    push    si
```

```

        push    di

        lds     si, source
        les     di, target
        mov     cx, num
        cld
        cmp     direct, MVFORWARD
        jz      dset
        std

dset:
        rep     movsb
        cld
        pop     di
        pop     si
        pop     cs
        pop     ds
        pop     bp
        ret

        endProg utamove
end

```

## UTANSI 检测、关闭或重新开放ANSI.SYS

```

#include <butil.h>
int utansi(int option);

```

option 待执行的操作:

UT\_ANS\_DETECT (2) 只报告ansi.sys的状态。  
 UT\_ANS\_DISABLE (0) 如果ansi.sys活动, 关闭之。  
 UT\_ANS\_ENABLE (-1) 如果存在ansi.sys并被关闭, 重新使之打开。

(返回) ansi.sys的原状态或遇到的错误:

UT\_ANS\_ABSENT (-2) 未检测到。  
 UT\_ANS\_ENABLE (-1) 检测到并且是活动的。  
 UT\_ANS\_DISABLE (0) 检测到但被关闭。  
 UT\_ANS\_HANDLES (3) 错误: 文件句柄用尽。  
 UT\_BAD\_OPT (1) 不支持option所包含的值。

UTANSI检测ansi.sys是否已安装、是否是活动的, 并可选地打开或关闭它。

警告和限制:

当ansi.sys被关闭时不要使程序夭折或从内存删除、重新定位程序。

如果其它程序已经关闭ansi.sys, UTANSI无法检测或重新使之打开。

如果控制台已被CTTY命令设置为其它的设备, 则UTANSI不能工作。它只能在当前控制台是标准IBM显示和键盘时, 即设备con时才能工作。

如果标准错误设备(即文件柄20)被重新定向, 则UTANSI不能工作。

UTANSI使用DOS, 包括DOS中断12。

源程序(UTANSI.C):

```

#include <conio.h>          /* For kbhit().          */
#include <dos.h>             /* For REGS, SREGS, intdos(), */
                             /* intdosx().             */

```

```

#include <io.h>                                /* For close(), dup(), dup2(), */
                                              /* write(). */

#include <bintrupt.h>
#include <bcreens.h>
#include <butil.h>

int b_ansdis = 0;                             /* Nonzero if we disabled */
                                              /* ANSI.SYS. */

#define MSGSIZE(sizeof(return_cursor_position) - 1)

static void dos_kb_flush(void);

int utansi(option)
int option;
{
    int      old_state;
    int      new_stdin;
    int      mode,act_page,columns;
    int      save_page;
    int      save_row,save_col,high,low;
    int      fore,back,col;
    static char return_cursor_position[] = "\33[6n";
    char      save_chars[MSGSIZE];
    static void far *psave_vec;
    static unsigned char replacement_code[] =
    {
        0x50,          /* push ax */
        0x56,          /* push si */
        0x57,          /* push di */
        0x55,          /* push bp */
        0xbb,0x07,0x00, /* mov bx,7h */
        0xb4,0x0e,      /* mov ah,0eh */
        0xcd,0x10,      /* int 10h */
        0xd,           /* pop bp */
        0xf,           /* pop di */
        0xe,           /* pop si */
        0x58,          /* pop ax */
        0xcf           /* iret */
    };

    switch (option)
    {
        case UT_ANS_DETECT:
        case UT_ANS_ENABLE:
        case UT_ANS_DISABLE:
            break;
        default:
            return UT_BAD_OPT;
    }
}

```

```

}

if (b_ansdis)
    old_state = UT_ANS_DISABLE;
else
    /* Check presense of ANSI.SYS. */
    {
        if (-1 == (new_stdin = dup(0))) /* Save spare copy of stdin. */
            old_state = UT_ANS_HANDLES/* Quit if failure. */
        else
            {
                dup2(2,0); /* Force handle 0 to refer to */
                           /* current console. */

                /* Save cursor position and */
                /* initial screen data. */

                semode(&mode,&columns,&act_page);
                save_page = b_curpage;
                scpage(act_page);
                securst(&save_row,&save_col,&high,&low);
                for (col = 0; col < MSGSIZE; col++)
                {
                    secursset(0,col);
                    save_chars[col] = scread(&fore,&back);
                }
                secursset(0,0);

                dos_kb_flush(); /* Clear keyboard input. */

                /* Write string to console. */
                write(2,return_cursor_position,MSGSIZE);

                if (kbhit()) /* Look for response from ANSI. */
                {
                    dos_kb_flush(); /* Discard the response. */
                    old_state = UT_ANS_ENABLE; /* ANSI.SYS is active. */
                }
                else
                {
                    /* No response; restore screen. */
                    for (col = 0; col < MSGSIZE; col++)
                    {
                        secursset(0,col);
                        scwrite(save_chars[col],1);
                    }
                    old_state = UT_ANS_ABSENT;
                }
                secursset(save_row,save_col); /* Clean up. */
                scpage(save_page);
                dup2(new_stdin,0);
                close(new_stdin);
            }
    }
}

```

```

    }

    /* Presence test is complete; */
    switch (option) /* now do any work. */
    {
        case UT_ANS_DISABLE:
            if (old_state == UT_ANS_ENABLE)
            {
                /* Save previous 0x29 handler.
                psave_vec = isgetvec(0x29);
                /* Install our 0x29 handler.
                isputvec(0x29,replacement_code);
                b_ansdis = 1;
            }
            break;

        case UT_ANS_ENABLE:
            if (b_ansdis)
            {
                /* Restore previous 0x29 handler*/
                isputvec(0x29,psave_vec);
                b_ansdis = 0;
            }
            break;
    }

    return(old_state);
}

static void dos_kb_flush() /* dos_kb_flush(): Flush */
{ /* standard input via DOS. */
    union REGS inregs,outregs;

    inregs.x.ax = 0x0c06;
    inregs.h.dl = 0xff;
    intdos(&inregs,&outregs);
}

```

## UTCHKNIL 报告无效指针赋值，使程序夭折

```

#include <stdio.h> /* UTCHKNIL需要*/
#include <butil.h>
int utchkni(void);

```

(返回) 如果未发现变化，为零；如果发现了变化，则不为零。

UTCHKNIL是一个宏，它用UTNULCHK检查缺省数据段的前几个字节，看看有无值的变化。(如果数据区发生了变化，这意味着有可能使用了一个非法指针值来存储数据。)如果发现变化，则向控制台报告源文件和行号，该程序被夭折，其ERRORLEVEL为1。

通常的技术是在main()函数的开始调用UTCHKNIL，然后在程序运行时的不同时刻激活UTCHKNIL。

宏UTCHKNIL 提供了两个版本，缺省版本向上面所述那样工作。程序中如果在包含butil.h或任何其它Turbo C TOOLS 头文件之前包含过下面的语句：



```
#define BNOCHKNIL 1
```

则UTCHKNIL将扩展成为空操作的表达式，于是减小了代码的尺寸。用户应该在完全测试和调试了他的程序之后再这么做。

源程序(BUTIL.H):

参见附录C的宏定义。

## UTCRTIT          取得DOS临界段标志的地址

```
#include <butil.h>
```

```
void utcrit(void);
```

UTCRTIT          将UTDOSRDY和其它函数所具有的DOS临界段标志的地址装入到全局变量b\_critad中。

DOS临界段标志(也称为DOS忙标志)是一个一字节值，指示是否有一个DOS功能调用正在进行，中断处理程序是否有一个DOS服务例程可用。当有一个DOS服务可用时，该字节值为零。

UTCRTIT本身激活DOS，所以不要在硬件中断服务例程中或DOS忙碌时使用它。

源程序(UTCRTIT.C):

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

```
#include <butil.h>
```

```
unsigned char far *b_critad = FARNIL;
```

```
unsigned char far *utcrit()
```

```
{
```

```
    union REGS regs;
```

```
    struct SREGS sregs;
```

```
    regs.x.ax = 0x3400;          /* DOS function number.          */
```

```
    int86x (UTDOSINT, &regs, &regs, &sregs);
```

```
    b_critad = uttfar (sregs.es, regs.x.bx, unsigned char);
```

```
    return (b_critad);
```

```
}
```

## UTCTLBRK          设置或返回Ctrl-Break检查的状态

```
#include <butil.h>
```

```
int utctlbrk(      int set,
```

```
                 int new_state);
```

set                CBRK\_SET      (1)    设置状态，

                  CBRK\_GET      (0)    返回状态。

new\_state        如果set是CBRK\_SET，则new\_state是新的Ctrl-Break状态。

                  CBRK\_ON       (1)    打开检查；

                  CBRK\_OFF      (0)    关闭检查。

(返回)           原Ctrl-Break状态；

                  CBRK\_ON       (1)    打开检查；

                  CBRK\_OFF      (0)    检查关闭。

UTCTLBRK 返回DOS Ctrl-Break检查的状态，任选地设置那个状态。

如果状态为“off”(即检查被关闭),则仅当DOS对标准输入、输出、打印或辅助设备执行I/O时,才检查Ctrl-Break或Ctrl\_C。如果状态是“on”,则只要调用DOS功能,DOS就要检查Ctrl-Break。

即使程序已经中止,对Ctrl-Break状态的改变也将一直保持下去。

源程序(UTCTLBRK.C):

```
#include <dos.h>
#include <butil.h>
int utctlbrk(set,new_state)
int set,new_state;
{
    union REGS regs;
    int      old_state;

    regs.x.ax = 0x3300;          /* DOS function 0x33, return */
                                /* CTRL-BREAK state.      */

    intdos(&regs,&regs);
    old_state = regs.h.dl;

    if (set != CBRK_GET && new_state != old_state)
    {
        regs.h.al = 0x01;        /* DOS function 0x33, set */
                                /* CTRL-BREAK state.      */

        regs.h.dl = (unsigned char) new_state;
        intdos(&regs,&regs);
    }

    return(old_state);
}
```

## UTDOSRDY 报告DOS服务是否可用

#include <butil.h>

int utdosrdy(void);

(返回) 如果DOS就绪,为1;若DOS忙碌,则为0。

UTDOSRDY检查DOS的临界段标志,报告DOS服务是否可用。

UTDOSRDY使用全局变量b\_critad。在激活UTDOSRDY之前应该利用UTCRT至少装载b\_critad一次,否则UTDOSRDY将返回0,就好象DOS未就绪一样。

在中断0x28处理程序内部没有必要使用UTDOSRDY,因为允许它使用除了功能1到12(控制台和打印机I/O)之外的任何DOS功能。

源程序(BUTIL.H):

参见附录C的头文件BUTIL.H中的宏定义。

## UTGETCLK 报告自午夜以来BIOS计时脉冲的个数

#include <butil.h>

int utgetclk(long \*pcount);

pcount 自午夜以来计时脉冲的个数。

(返回) 如果最后的计时脉冲引起午夜回复,为1;否则为0。

int utintflg(int flag);

UTGETCLK返回自午夜以来BIOS的四字节计时脉冲的计数。对于标准的IBMPC, 计时脉冲每秒出现1,193,180/65,536(约 18.2)次。

如果函数值为1, 则出现午夜回复, 但 尚未清除回复标志。UTGETCLK不清除回复标志, 这就允许其它程序(假设DOS)对回复作出反应并清除它。

用UTTK2TIM可以将最终的计时脉冲计数转换为24小时制时间。

源程序(UTGETCLK.C):

```
#include <butil.h>
int utgetclk(pcount)
long *pcount;
{
    int  were_on;          /* Stores previous interrupt state.      */
    unsigned char rollover; /* Copy of BIOS's rollover flag.    */

    were_on = utintoff();   /* Turn interrupts off.          */

    utpeekn ((char far *) UTCLOCKADDR,
             (void *) pcount,
             sizeof (*pcount));

    rollover = utpeekb (UTROLLADDR);

    if (were_on)
        utinton();          /* Turn interrupts back on.      */

    return (rollover != 0);
}
```

## UTINTFLG 打开或关闭硬件中断

#include <butil.h>

flag 如果开放中断, 为UT\_INTON (1); 如果关闭中断, 为UT\_INTOFF (0)。

(返回) 如果在调用前中断打开, 为UT\_INTON (1); 如果未打开, 为UT\_INTOFF (0)。

UTINTFLG 打开或关闭除了NMI (不可屏蔽中断)外的所有硬件中断, 返回中断标志的原状态。如果中断关闭, 则以后的中断被推迟, 直至重新打开。一旦中断打开, 任何在中断关闭期间的中断请求立即得到服务。

不要使中断关闭无限长的时间。

源程序(UTINTFLG.ASM):

```
include beginasm.mac      ; Specifies the compiler and memory model
beginProg utintflg

want_on equ    word ptr [bp + stkoﬀ]
push    bp
mov     bp, sp
pushf                                ; Get flags in AX to examine later--
pop     ax                          ; Bit 9 of flags is Interrupt flag.

mov     bx, want_on
```

```

        cmp     bx, 0
        jz      setoff

seton:
        test    ah, 2
        jnzwereon
        mov     ax, 0                ; Interrupts were disabled.
        sti                     ; Enable them.
        jmp     short done

wereon:
        mov     ax, 1                ; Interrupts were already enabled.
        jmp     short done

setoff:
        test    ah, 2
        jz      wereoff
        mov     ax, 1                ; Interrupts were enabled.
        cli                     ; Disable them.
        jmp     short done

wereoff:
        mov     ax, 0                ; Interrupts were already disabled.
        jmp     short done

done:
        cld
        pop     bp
        ret

        end

```

## UTMODEL 报告IMB型号和子型号及BIOS版本

```
#include<butil.h>
```

```
char utmodel(void);
```

(返回) IBM型号码(见下表中的可能值)。

UTMODEL返回一个代码,说明程序运行在的IBM个人码。)子型号(如果有的话)和BIOS版本号(如果有的话)也被返回。

型号码作为函数值返回,它也保存在全局变量b\_pcmodel中。子型号码(如果有的话)在b\_submod中返回,BIOS版本号(如果有的话)在b\_biosrev中返回。所有这些全局变量在butil.h中声明。

可能值如下:

型号	子型号	BIOS版本	计算机
0xff	-	-	PC
0xfe	-	-	PC-XT或便携式PC
0xfb	0x00	0x01	带有256/640母板的PC-XT
0xfb	0x00	0x02	带有256/640母板的PC-XT
0xfd	-	-	PCjr
0xfc	-	-	PC-AT
0xfc	0x00	0x01	PC-AT
0xfc	0x01	0x00	PC-AT

0xfc	0x02	0x00	XT模型286
0xf9	0x00	0x00	PC便携式
0xfa	0x01	0x00	PS/2模型25
0xfa	0x00	0x00	PS/2模型30
0xfc	0x09	0x00	PS/2模型30 286
0xfc	0x04	0x00	PS/2模型50
0xfc	0x04	0x03	PS/2模型50z
0xfc	0x05	0x00	PS/2模型 60
0xf8	0x09	0x00	PS/2模型70类型1
0xf8	0x04	0x00	PS/2模型70类型2
0xf8	0x01	0x00	PS/2模型80

未列在上表的型号码被认为是非法的IBM型号。如果遇到了未知型号的代码或得不到型号及BIOS版本号, 则b\_submond和b\_biosrev被置为缺省值0xff。

源程序(UTMODEL.ASM):

```
include beginasm.mac
beginMod utmodel

; IBM model cod :
IBM_PC equ 0ffh ; IBM PC.
IBM_XT equ 0feh ; IBM C-XT or Portabne
IBM_JR equ 0fdh ; IBM PC-JR.
IBM_AT equ 0feh ; IBM PC-AT, XT model 286,
; PS/2 models 50, 60
IBM_X2 equ 0fbh ; IBM PC-XT with 256/640 motherboard.
IBM_CV equ 0f9h ; IBM PC Co vertible.
IBM_30 equ 0fah ; IBM PS/2 model 25 or 30.
IBM_80 equ 0f8h ; IBM PS/2 model 80.
```

BIOSREV struc

```
BIOSREV_length dw ?
BIOSREV_model db ?
BIOSREV_submodel db ?
BIOSREV_rev db ?
BIOSREV_feature db 5 dup (?)
```

BIOSREV ends

```
beginDSeg ; define & initialize global variables.
pubSym b_pmodel, <db 0>
pubSym b_submod, <db 0ffh>
pubSym b_biosrev, <db 0ffh>
endDSeg
```

```
beginCseg utmodel
beginProc utmodel
```

```
push bp
mov bp, sp
```

```
push ds
mov ax, seg b_pmodel@
```

```

mov     ds,ax

cmp     ds:b_pcmmodel@,0      ; Skip if we've already done this.
jne     done

mov     ax,0ffffh             ; Fetch model code from 0ffffh:0eh.
mov     es,ax
assume  es:nothing
mov     al,es:[byte ptr 0eh]
mov     ds:b_pcmmodel@,al

cmp     al,IBM_80              ; Skip if model code is outside known
jb     done                    ; range.

mov     ah,0c0h                ; Request further BIOS version info.
int     15h
jc     done                    ; Quit if function unavailable.

mov     al,es:[bx.BIOSREV_submodel] ; Fetch submodel.
mov     ds:b_submod@,al
mov     al,es:[bx.BIOSREV_rev]      ; Fetch BIOS release.
mov     ds:b_biosrev@,al

done:
mov     al,ds:b_pcmmodel@        ; Return model code
cbw                                  ; as a signed char.

pop     ds
pop     bp
ret

endProc utmqdel
endCseg utmodel
endMod utmodel

end

```

## UTMOVME 不受限制地从内存或向内存任何位置拷贝数据

```

#include <butil.h>
void utmovmem( const char far *psource,
               char far *ptarget,
               unsigned int length);
psource      源数据区地址(near或far)。
ptarget      目的区地址(near或far)。
length       需拷贝的字节数。

```

UTMOVME从内存任意位置拷贝数据到任意另一个位置，源和目的位置可以叠盖。如果发生了叠盖，须选择拷贝方向，使得每个数据字节只拷贝一次。

psource和ptarget可以是near指针。如果确是near指针，它们应该向下面的示例那样转换为far指针。如果包括进了头文件butil.h,则编译器自动执行转换。

用户还可以用UTPEEKB、UTPEEKW、UTPOKEB 和UTPOKEW 来传送字节或字。如果源或目的地址值无法预先确定，例如从中断向量中得到，则应使用UTSAFCY。

```
/**
 *
 * Name          UTMOVMEM - Copy memory from a far pointer to another.
 *
 * Synopsis      utmovmem (source, target, amount);
 *
 *               const char far *source    A far pointer to the bytes
 *                                           to be moved.
 *
 *               char far *target          A far pointer to the place the
 *                                           bytes are to be moved to.
 *
 *               unsigned int amount       The number of bytes to move.
 *
 * Description    UTMOVMEM copies memory from a source region to a target
 *               region.  It is a smart memory move routine, meaning that
 *               it can be used for overlapping memory moves.
 *
 *               Cases that UTMOVMEM can deal with are:
 *
 *               1.  SourceSourceSource
 *
 *                           TargetTargetTarget
 *
 *               2.          SourceSourceSource
 *
 *                           TargetTargetTarget
 *
 *               3.          TargetTargetTarget
 *
 *                           SourceSourceSource
 *
 *               4.  TargetTargetTarget
 *
 *                           SourceSourceSource
 *
 * Returns        Nothing.
 *
 * Version        6.00 (C)Copyright Blaise Computing Inc.    1987-1989
 */
```

```
#include <butil.h>
```

```
void utmovmem (source, target, num)
const char far *source;
char far *target;
register unsigned int num;
{
    const char huge *snum;
    char huge *tnum;

    /* Normalize source and target pointers so we can compare them. */
    source = utnorm (source, char);
    target = utnorm (target, char);

    /* Adjust to end of source and target regions, and normalize. */
    snum = utnorm ((source + num), char);
    tnum = utnorm ((target + num), char);

    if ((num != 0) && ((char huge *) source != (char huge *) target))
    {
```

```

if (
    /* Source completely before target. */
    (snum <= (char huge *) target) ||
    /* Source completely after target. */
    ((char huge *) source >= tnum) ||
    /* Target covers source left-to-right. */
    (((char huge *) source > (char huge *) target) &&
     ((char huge *) source <= tnum)))

    utamove (source, target, num, UT_MVFORWARD);

else
{
    /* Source covers target left-to-right. */
    target += num - 1;
    source += num - 1;

    utamove (source, target, num, UT_MVBACKWARD);
}
}

```

## UTNORM 使一个指针具有最小的偏移值

```
#include <butil.h>
```

```
type_name far *utnorm( void far *ptr,
                       type_name);
```

ptr 任意地址表达式(near或far)。

type\_name 数据类型名。

(返回) 作为结果的指向给定数据类型的far指针。

UTNORM是一个宏，构造指向一个给定地址的far指针。地址被调整，使得偏移部分的值为0到15，作为结果的指针值能够可靠地与任何其它用同样方法标准化的指针相比较。

用UTPLONG可以将任何地址表示式转换成20位物理地址。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTNULCHK 检测无效的指针赋值

```
#include <butil.h>
```

```
int utnulchk(void);
```

(返回) 如果未发生变化，为零；如果发生了变化，为非零值。

UTNULCHK 检查缺省数据段的前几个字节，看看值有无变化。如果数据区被改变，这意味着很有可能使用了一个无效的指针值来存储数据。

UTNULCHK在第一次被调用时记录缺省数据段的前几个字节的内容。第一次调用时它总是返回0 (“没有变化”)。每次调用之后，UTNULCHK记录新数据值，使得以后的改变可以被检测到。

通常的方法是在main()函数中把调用UTNULCHK作为一个起始步骤，然后在程序运行过程中的不同时刻检查UTNULCHK返回的值。非零值表示使用了一个非法地址表达式来存储数据。

当探测到一个无效的指针赋值时，UTCHKNIL自动使程序夭折，发出一条消息。



源程序(UTNULCHK.C):

```
#include <dos.h>
#include <butil.h>
#define BCHKNULSIZE 0x10 /* #of bytes to save from low memory. */
                          /* Flag to indicate whether or not */
                          /* UTNULCHK has been called before. */
static unsigned char b_chkdid = 0;

                          /* Buffer to store low memory bytes for */
                          /* later comparison. */
static char b_chkbyt [BCHKNULSIZE];

unsigned int utnulchk()
{
    char far *pLOW;          /* Pointer to low memory. */
    unsigned int changed=0; /* Flag to indicate memory changed. */
    unsigned int i;          /* Counter/array index. */
    unsigned int seg;        /* Segment to read in low memory. */
    seg = utseg (&_psp);
    if (b_chkdid)            /* If this is not the first call to */
    {                          /* UTNULCHK... */

        /* Check each byte to make sure it is */
        /* the same as it was. */
        for (pLOW = uttofar (seg, 0, char), i = 0;
             (i < BCHKNULSIZE) && (!changed);
             pLOW++, i++)

            if (utpeekb (pLOW) != b_chkbyt [i])
                changed = 1;
    }

    /* If it was changed or this is the */
    /* first call to UTNULCHK... */
    if (changed || (!b_chkdid))
    {
        /* Read current low memory into buffer. */
        utpeekn (uttofar (seg, 0, char), b_chkbyt, BCHKNULSIZE);

        b_chkdid = 1; /* Set flag to indicate we read it. */
    }

    return (changed);
}
```

UTOFF            返回一个地址的偏移部分

```
#include <butil.h>
```

unsigned int utoff (void far \*ptr);  
ptr           任意地址表达式(near 或 far)。  
(返回)       地址的偏移部分。  
UTOFF        是一个宏, 它返回任意地址的偏移部分。

用UTSEG可以提取地址的段部分, UTTOFAR或UTOFARU可以将段和偏移值结合起来, 形成一个双字地址(即far 指针)。

源程序(BUTIL.H):

参见附录C的BUTIL.H的定义。

## UTPEEKB        从任意地址读取一个字节

```
#include <butil.h>
unsigned char utpeekb(const void far *ptr);
ptr           任意的地址表达式(near或far)
(返回)       从*ptr取得的八位值。
```

UTPEEKB 是一个宏, 从内存任意位置读取数据的一个字节。它可以接受任何地址表达式。

用UTPOKEB可以向内存任意位置存贮数据的一个字节, UTPEEKW和UTPEEKN可以读取更多的数据。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPEEKN        从任意地址读取多个字节的数据

```
#include <butil.h>
void utpeekn(   const void far *psource,
               void far *ptarget,
               unsigned int length);
psource        指向源数据的指针(near或far)。
ptarget        指向目的区的指针(near 或far)。
length        需拷贝的字数。
```

UTPEEKN是一个宏, 可以从内存的任意位置读取任意字节数的数据。它可以接受任何地址表达式。

用UTPOKEN可以在内存中的任位置存放数据, UTPEEKB 和UTPEEKW可以比UTPEEKN更为有效地读取数据的字节和字。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPEEKW        从任意地址读取一个字

```
#include <butil.h>
unsigned int utpeekw(const void far *ptr);
wordval        从*ptr 读取的十六位值。
ptr           任意的地址表达式(near或far)。
```

UTPEEKW是一个宏, 从内存任意位置读取数据的一个字(两字节)。它可以接受任意的地址表达式。

用UTPOKEW可以在内存的任位置存放一个字的数据, 而UTPEEKB 和UTPEEKN可以读取多个

字节的数据。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPLONG      将一个指针转换为指向20位物理地址的指针

#include <butil.h>

unsigned long utplong (void \*ptr);

ptr            任意的地址表达式(near或far)。

(返回)        表达为20位物理地址的ptr的值。

UTPLONG 是一个宏, 它将任意的地址表达式转换 为一个20位物理地址。

作为结果的unsigned long 值可以可靠地同其它已转换成同样形式的地址相比较。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPOKEB      在任意地址存放一个字节的的数据

#include <butil.h>

void utpokeb( void far \*ptr,  
              unsigned char byteval);

byteval        在 \*ptr 处写入的八位值。

ptr            任意的地址表达式 (near 或 far)。

UTPOKEB是一个宏, 在内存的任意位置存放一个字节的的数据。它可以接受任何地址表达式。

用UTPEEKB可以从内存任意位置读取数据的一个字节, 而UTPOKEW 和 UTPOKEN能够写入多个字节的数据。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPOKEN      在任意地址存放多个字节的数据

#include <butil.h>

void utpoken( const void far \*psource,  
              void far \*ptarget,  
              unsigned int length);

psource        指向源数据区的指针 (near 或 far)。

ptarget        指向目的数据区的指针 (near 或 far)。

length        需拷贝的字节数。

UNPOKEN是一个宏, 把数据的任意个字节存放在内存的任意位置。它可以接受任意的地址表达式。

UTPEEKN可以从内存的任意位置读取数据, 而UTPOKEB 和UTPOKEW可以更方便地存放数据的单个字节或字。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTPOKEW 在任意位置写入一个字的数据

```
#include <butil.h>
```

```
void utpokew( void far *ptr,  
              unsigned char wordval);
```

wordval      需存放在 \*ptr 处的十六位值。

ptr          任意的地址表达式(near 或 far)。

UTPOKEW是一个宏,将数据的一个字(两字节)存放在内存的任意位置。它可以接受任何地址表达式。

UTPEEKW可以从内存任意位置读取数据的一个字,而UTPOKEB 和 UTPOKEN可以存放数据的一个或多个字节。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTSAFCPY 以确保不跨越段界的方式拷贝数据

```
#include <butil.h>
```

```
int utsafcpy( const void far *psource,  
              void far *ptarget,  
              unsigned int length);
```

psource      源数据区的地址 (near 或 far)。

ptarget      目的数据区域地址 (near 或 far)。

length      需拷贝的字节数。

(返回)      错误代码:

0                                  数据拷贝成功;  
UT\_SRC\_STRADDLE (1)      源数据跨越段界;  
UT\_TGT\_STRADDLE (2)      目的数据跨越段界;  
UT\_2\_STRADDLE (3)      源和目的缓冲区都跨越段界。

UTSAFCPY从内存任意位置向任意其它位置拷贝数据,但仅当源和目的缓冲区均完全处于一个段中(但不必在同一段中)时才能完成拷贝,这就避免了 80286 上的段越界异常(中断0x0e)。

源和目的位置可以互相叠盖。如果发生叠盖,须选择拷贝方向使得数据的每个字节仅拷贝一次。

如果两个缓冲区中的某一个跨越段界,则 UTSAFCPY 返回一个非零数,不拷贝任何数据。这个条件并不必然意味着数据的尾端折转到段的开始位置,相反,这可能意味着选择的段地址不合适,使得段没有包含整个缓冲区,也意味着地址值不正确。用 UTNORM可以改正任何地址,使之覆盖最多达 65,521 字节的缓冲区。

参见UTOFF中的第二个示例,该示例介绍了如何直接执行地址检查。

源程序(UTSAFCPY.C):

```
#include <butil.h>
```

```
int utsafcpy(psource,ptarget,amount)
```

```
const void far *psource;
```

```
void far *ptarget;
```

```
unsigned int amount;
```

```
{
```

```
    int result = 0;
```

```
    if (amount > 1)
```

```
    {
```

```

        if (utoff(psource) >= (unsigned) 1 - amount)
            result = UT_SRC_STRADDLE;
        if (utoff(ptarget) >= (unsigned) 1 - amount)
            result += UT_TGT_STRADDLE;
    }
    if (0 == result)                /* Copy only if safe.          */
        utmovmem(psource,ptarget,amount);

    return result;
}

```

## UTSEG 返回任意地址的段部分

```

#include <butil.h>
unsigned int utseg(void far *ptr);
ptr          任意的地址表达式(near 或 far)。
(返回)       地址的段部分。

```

UTSEG是一个宏，它返回任意地址的段部分。

用UTOFF可以提取任意地址的偏移部分。UTTOFAR 或 UTTOFARU 可以将段和偏移值结合起来，产生一个双字地址（即一个 far 指针。）

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTSLEEP 暂停处理直至经过几个计时脉冲之后

```

#include <butil.h>
unsigned utsleep(unsigned period);
period      处理过程挂起的计时脉冲数。
(返回)      程序实际被挂起的计时脉冲数。

```

UTSLEEP至少使处理过程Q停所需的脉冲数，并把实际脉冲数作为函数值返回。由于函数检查系统时钟以决定所需时间是否已过，所以实际时间比所需时间长是完全有可能的。

在标准IBM PC上，计时脉冲每秒出现 1,193,180/65,536 (约18.2)次。

UTSLEEPQ时打开硬件中断，但返回前恢复中断标志的原状态。

源程序(UTSLEEP.C):

```

#include <butil.h>
unsigned utsleep(period)
unsigned period;
{
    long    initclk;          /* Initial clock count.          */
    long    nowclk;           /* Moving clock count.           */
    unsigned elpticks;        /* Elapsed tick count.          */
    int     ints_were_on;     /* Whether interrupts were on   */
                                /* already.                      */
    /* Find out the current clock count and wait until period
    /* counts have passed.
    ints_were_on = utinton();

```

```

    utgetclk(&initclk);
    for (elpticks = 0;
        elpticks < period;
        elpticks = (unsigned) (nowclk - initclk))
    {
        utgetclk(&nowclk);
        if (nowclk < initclk) /* Must have wrapped past midnight. */
            nowclk += 0x1800b0L;
    }

    if (!ints_were_on)
        utintoff();

    return(elpticks);
}

```

## UTSPKR 打开或关闭扬声器

`#include <butil.h>`  
 void utspkr(unsigned freq);  
 freq 以赫兹为单位的声音的频率 (至少 19)。若为0, 则关闭扬声器。  
 如果freq非零, UTSPKR打开扬声器, 产生频率为 freq 的声波, 直至另一个对UTSPKR 的调用改变 freq。freq 以赫兹为单位 (每秒周期数), 必须至少为 19。  
 如果 freq 为零, UTSPKR 关闭扬声器。

源程序(UTSPKR.C);

```

#include <conio.h>
#include <dos.h>
#include <butil.h>
/* The following are hardware port addresses needed to access the */
/* 8255 (speaker) and 8253 (timer) chips. */

#define PB8255 0x61 /* Port B of the 8255 chip. */
#define LTC8253 0x42 /* Latch register 8253 (timer)*/
#define CMD8253 0x43 /* Command register of 8253. */

#define CLKFREQ 1193180L /* System clock frequency. */

void utspkr(freq)
unsigned freq;
{
    unsigned latch;

    if (freq != 0)
    {
        /* Enable speaker data and timer gate signals on PortB. */
        utoutp (PB8255, utinp (PB8255) | 3);
        /* Now program channel 2 of the 8253 timer to generate a */
        /* square wave of frequency "freq" Hz. */
    }
}

```

```
/* Divide clock frequency by requested speaker frequency to */
/* generate value for the timer latch. */
```

```
utoutp (CMD8253, 0x0b6);
```

```
latch = (unsigned int) (CLKFREQ / freq);
```

```
utoutp (LTC8253, utlobyte(latch));
```

```
utoutp (LTC8253, uthibyte(latch));
```

```
}
```

```
else /* Clear the lower two bits of Port B. */
```

```
utoutp(PB8255, utinp (PB8255) & ~3);
```

```
}
```

## UTSQZSCN 压缩一个屏幕图象

```
#include <butil.h>
```

```
int utsqescn( const char far *psource,
              char *ptarget,
              int sresize,
              int tarsize);
```

psource 屏幕图象的地址。

ptarget 容纳压缩图象的目的缓冲区的地址。

sresize 以字节计的源图象的尺寸 (即数据字符的两倍)。

tarsize 以字节计的目的缓冲区的尺寸, 不附加尾随的 NUL('\0')。

(返回) 压缩图象所需的尺寸 (即使 tarsize 不够用)。

UTSQZSCN压缩一个屏幕图象, 返回压缩图象所需的字节数。

假设源图象是一系列字符/属性对 (正象字符方式下显示内存的格局)。然而, 源缓冲区中可以提供任何格式的数据。

如果 tarsize 不足以容纳压缩图象, 则只目的缓冲区被填充到tarsize字节。不论哪种情况, 所需字节数作为函数值返回。

(当把压缩图象传送给另一个程序时, 要记住告知压缩图象尺寸, 因为UTUNSQZ 需要这个尺寸。)

压缩后的屏幕图象是一个描述屏幕图象每个字符和属性的数据字节流, 在这个流中插入了一些特殊代码, 用于表示一些有相同属性字节的相 5 字符及重复的字符序列。下面是确切的格式:

```
x(A)x(C)Len x(A)A1 Charstr x(A)A2...Charstr. x(A)An...
```

这里 x(A) 是一个值为 '\376' 的字节 ("属性单元", 有时称为 "属性键");

x(C) 是一个值为 '\377' 的字节 ("字符单元", 有时称为 "字符键");

"Len" 是一个字, 包含有扩展屏幕图象的长度 (即 sresize, 以字节计),

而 A1、A2、...、An是属性字节。

"Charstr"的格式如下:

```
C2C2...Cn...x(C)RC'.....
```

这里 C1、C2、Cn、...、C'是字符;

R 是一个字, 包含有字符 C' 的重复计数。

在某些情况下, 压缩图象可能比原图象更大。见下面的第一个示例。

UTUNXQZ 是压缩的逆过程, 它重建原图象的一个拷贝。用 SCSSAVEPG 和 SCRESTPG可以直接从物理显示器上保存图象。

源程序(UTSQZSCN.C);

```

#include <butil.h>
static void dump_characters(char *, int *, int, int, char,
                           char *, int *);
int utsqzscn(psource, pdestination, source_length,
             destination_length)
const char far *psource;
char          *pdestination;
int           source_length;
int           destination_length;
{
    int         overflow = 0;
    int         source_index, destination_index;
    int         character_count;
    char        current_character = '\0';
    char        previous_character;
    unsigned char current_attribute = 0;
    unsigned char previous_attribute;

    if ((psource == NIL) || (pdestination == NIL))
        return(0);

    /* First set the control information in the compressed image.

    if (destination_length >= 4)
    {
        pdestination[0] = UT_ATTR_KEY;
        pdestination[1] = UT_CHAR_KEY;
        utpoken(&pdestination[2], source_length);
    }
    else
        overflow = 1;

    destination_index = 4;
    character_count   = 0;
    previous_character = psource[0];
    previous_attribute = 0;
    source_index      = 0;

    /* Now pass through the screen image.  If there is an attribute
    /* change, output the attribute key and attribute after
    /* outputting any character string built.  If there is a
    /* character change, output the character string built.  If the
    /* compressed image is greater than the target space, set
    /* overflow to 1, but keep recording how big the compressed
    /* image would be.

    while (source_index < source_length)
    {
        current_character = psource[source_index];

```



```

current_attribute = psource[source_index + 1];
if (current_attribute != previous_attribute)
{
    dump_characters(pdestination, &destination_index,
                    destination_length, character_count,
                    current_character, &previous_character,
                    &overflow);
    character_count = 0;
    if (overflow)
        destination_index += 2;
    else
        if ((destination_index + 2) > destination_length)
        {
            overflow = 1;
            destination_index += 2;
        }
        else
        {
            pdestination[destination_index++] =
                UT_ATTR_KEY;
            pdestination[destination_index++] =
                current_attribute;
        }
        previous_attribute = current_attribute;
    }
    if ((current_character != previous_character) &&
        (character_count != 0))
    {
        dump_characters(pdestination, &destination_index,
                        destination_length, character_count,
                        current_character, &previous_character,
                        &overflow);
        character_count = 1;
    }
    else
        character_count++;
    source_index += 2;
}

/* Output the last character or character string. */

dump_characters(pdestination, &destination_index,
                destination_length, character_count,
                current_character,
                &previous_character, &overflow);

return(destination_index);
}

/**

```

```

*
* Name          DUMP_CHARACTERS Place waiting character/count pair
*                  in compressed screen buffer.
*
* Synopsis      dump_characters(pdestination, pdestination_index,
*                  destination_length, pcharacter_count,
*                  current_character, pprevious_character,
*                  poverflow)
*
*              char *pdestination      Destination (output) buffer.
*              int *pdestination_index  Current buffer index.
*              int destination_length   Length of buffer.
*              int character_count      Number of identical
*                  characters accumulated.
*              char current_character   The current source buffer
*                  character.
*              char *pprevious_character The previous character in the
*                  source buffer.
*              int *poverflow           Whether target buffer has
*                  overflowed.
*
* Description    This function will output a waiting character or a
*                  coded sequence indicating that a number of identical
*                  characters were encountered in the source buffer,
*                  placing the character or coded sequence in the
*                  destination buffer. This is done whenever
*                  UTSQZSCN encounters a change of attribute or
*                  character in the source buffer.
*
*                  If *poverflow is non-zero, no characters will actually
*                  be placed in the buffer, but *pdestination_index and
*                  *pprevious_character will contain the same values
*                  they would have if the buffer had not overflowed. This
*                  allows UTSQZSCN to compute the size the compressed image
*                  will require, even though it does not fit in the current
*                  buffer.
*
* Returns        *pdestination_index  New index into destination
*                  buffer.
*                  *pprevious_character New previous character.
*                  *poverflow          Non-zero if the destination
*                  buffer has overflowed,
*                  unchanged otherwise.
**/

static void dump_characters(pdestination, pdestination_index,
                           destination_length, character_count,
                           current_character, pprevious_character,
                           poverflow)

```

```

char *pdestination;
int *pdestination_index;
int destination_length;
int character_count;
char current_character;
char *pprevious_character;
int *poverflow;
{
    if (character_count == 0)
    {
        (*pprevious_character) = current_character;
        return;
    }

    if (*poverflow)
    {
        if (((*pprevious_character) == (char) UT_ATTR_KEY) ||
            ((*pprevious_character) == (char) UT_CHAR_KEY) ||
            (character_count > 1))
        {
            (*pdestination_index) += 3;
        }
        else
            (*pdestination_index)++;
    }
    else
        if ((character_count == 1) &&
            ((unsigned char) (*pprevious_character) != UT_ATTR_KEY) &&
            ((unsigned char) (*pprevious_character) != UT_CHAR_KEY))
        {
            if ((*pdestination_index) >= destination_length)
                (*poverflow) = 1;
            else
                pdestination[(*pdestination_index)] =
                    (*pprevious_character);
        }
        else
        {
            if (((*pdestination_index) + 4) > destination_length)
            {
                (*poverflow) = 1;
                (*pdestination_index) += 3;
            }
            else
            {
                pdestination[(*pdestination_index)++] = UT_CHAR_KEY;
                utpokenw(&pdestination[(*pdestination_index)],
                    character_count);

                (*pdestination_index) += 2;
            }
        }
    }
}

```

```

        pdestination[( *pdestination_index)] =
            ( *pprevious_character);
    }
}

( *pprevious_character) = current_character;
( *pdestination_index)++;
}

```

## UTTIM2TK 将时间转换为计时脉冲计数

```

#include <butil.h>
unsigned long uttim2tk(    int hours,
                          int mins,
                          int secs,
                          int hunds);
hours        自午夜以来的小时数(0-23)。
mins        分钟数(0-59)。
secs        秒数 (0-59)。
hunds        百分之一秒数 (0-99)。
(返回)      作为结果的自午夜以来的计时脉冲数。

```

UTTIM2TK将24小时制的小时、分种、秒、百分之一秒数转换为相应的计时脉冲数。在标准IBM PC上, 计时脉冲每秒出现 1,193,180/65,536 (约 18.2) 次。

UTTIM2TK不使用浮点算法。

如果hours、mins、secs和hunds在指定范围内, 则结果精度在两个脉冲以内。

用UTTK2TIM可以把计时脉冲计数转换为小时、分种、秒和百分之一秒。

源程序(UTTIM2TK.C):

```

#include <butil.h>
#define UL (unsigned long)
unsigned long uttim2tk (hrs, mins, secs, hunds)
int hrs, mins, secs, hunds;
{
    return  (hrs ? ((UL hrs * 65543L) + UL (hrs / 3) ) : 0L)
        + (mins ? ((UL mins * 1092L) + UL ((mins < 1) / 5)) : 0L)
        + (secs ? ((UL secs * 18L) + UL (secs / 5) ) : 0L)
        + (UL hunds / 5L);
}

```

## UTTK2TIM 将计时脉冲计数转换为 24 小时制时间

```

#include <butil.h>
void uttk2tim(    unsigned long ticks,
                int *phrs,
                int *pmins,
                int *psecs,
                int *phunds);
ticks        计时脉冲的计数。

```

phours, pmins, psecs, phunds

变量的地址。这些变量返回小时 (0-23)、分钟 (0-59) 和百分之一秒 (0-99)，它们必须指向不同的位置。

UTTK2TIM 将一个计时脉冲计数转换为相应的小时、分钟、秒和百分之一秒数。

24小时制的结果时间在 phours、pmins、psecs 和 phunds 所指向的单元中返回。这四个位置必须互不相同。

在标准 IBM PC 上，计时脉冲每秒出现 1,193,180/65,536 (约 18.2) 次。

UTTK2TIM 不使用浮点算法。

如果脉冲计数不超过 1,573,066 个 (一天内的计时脉冲数)，则结果精度在 1.4 秒内。

用 UTTIM2TK 可以将 24 小时制时间转换为计时脉冲计数。

源程序(UTTK2TIM.C):

```
#include <butil.h>
#define UL (unsigned long)
void uttk2tim (ticks, phrs, pmins, psecs, phunds)
unsigned long ticks;
int *phrs, *pmins, *psecs, *phunds;
{
    unsigned long fudge;

    *phrs = (int) (ticks / 65543L);
    fudge = (*phrs)
        ? ((UL (*phrs) * 33494L) / 100000L)
        : (0L);
    ticks -= UL (*phrs) * 65543L;
    ticks = (ticks > fudge) ? (ticks - fudge) : (0L);
    *pmins = (int) (ticks / 1092L);
    fudge = (*pmins)
        ? ((UL (*pmins) * 388915L) / 1000000L)
        : (0L);
    ticks -= UL (*pmins) * 1092L;
    ticks = (ticks > fudge) ? (ticks - fudge) : (0L);
    *psecs = (int) (ticks / 18L);
    fudge = (*psecs)
        ? ((UL (*psecs) * 206482L) / 1000000L)
        : (0L);
    ticks -= UL (*psecs) * 18L;
    ticks = (ticks > fudge) ? (ticks - fudge) : (0L);
    *phunds = (int) (ticks * 4L);
}
```

UTTOFAR 用段和偏移构造一个双字指针

```
#include <butil.h>
type_name far *uttofar( unsigned int segment,
                        unsigned int offset,
                        type_name);
segment, offset
```

物理地址的段和偏移部分。

type\_name 任意数据类型名。

(返回) 作为结果的指向给定数据类型的far 指针。

UTTOFAR 是一个宏，它用段和偏移值构造指向一个特定数据类型的 far 指针。

用UTTOFARU 可以构造一个泛数据指针，即指向空的指针。UTNORM 对任何地址表达式进行标准化，使它具有0到15的偏移，而UTPLONG 可以把任何地址表达式转换为 20 位物理地址。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTTOFARU 用一个段和偏移构造一个泛双字指针

```
#include <butil.h>
```

```
void far *uttofaru(    unsigned int segment,  
                      unsigned int offset);  
segment, offset
```

物理地址的段和偏移部分。

(返回) 指向任意数据类型的 far 指针。

UTTOFARU是一个宏，它从段和偏移值中构造一个 far 指针，作为结果的指针是一个泛数据指针，即指向空的指针。

用UTTOFAR 可以构造指向一个数据类型 的指针，UTNORM可以对任何地址表达式进行标准化，使它具有 0 到15 的偏移值，而UTPLONG 把地址表达式转换成一个 20 位地址。

源程序(BUTIL.H):

参见附录C的BUTIL.H的宏定义。

## UTUNSQZ 还原一个压缩屏幕图象

```
#include <butil.h>
```

```
int utunsqz(    const char *psource,  
               char far *ptarget,  
               int srsize,  
               int tarsize);  
psource 压缩屏幕图象的地址。  
ptarget 用于容纳展开图象的目的缓冲区的地址。  
srsize 以字节计的压缩源图象的尺寸。  
tarsize 以字节计的目的缓冲区的尺寸。不附加尾随的('\0')。  
(返回) 展开图象所需的尺寸(即使tarsize不够用)。
```

UTUNSQZ展开由UTSQZSCN压缩的屏幕图象，返回还原压缩图象所需的字节数。最后展开的图象是一系列字符/属性对(就象字符方式下显示内存的格局)。

如果tarsize不足以容纳展开的图象，则目的缓冲区最多填充tarsize个字节。不论哪种情况，需要的字节数作为函数值返回。

请参见UTSQZSCN中有关压缩图象格式的说明。UTUNSQZ将压缩图象有前两个字节分别解释为属性单元。

用UTSQZSCN 可以压缩一个屏幕图象，SCSAVEPG和SCRESTPG则直接从物理显示器上保存图象。

源程序(UTUNSQZ.C):

```

#include <butil.h>
int utunsqz(psource, pdestination, source_length, destination_length)
const char *psource;
char far *pdestination;
int source_length;
int destination_length;
{
    unsigned char char_key;
    unsigned char attr_key;
    int source_index, destination_index;
    int character_count;
    int expanded_len;
    unsigned char current_attr = 0;
    unsigned char current_byte;

    if ((psource == NIL) || (pdestination == NIL))
        return(0);

    current_attr = 0;
    attr_key = psource[0];
    char_key = psource[1];
    expanded_len = *((int *) &psource[2]);
    source_index = 4;
    destination_index = 0;

    /* Now pass through the compressed image. If an attribute key is */
    /* found, the current attribute is altered to the attribute value.*/
    /* If a character key is found, the number of copies of the */
    /* character is output (together with the current attribute). If */
    /* a key is found at the end of the source, it is treated as a */
    /* normal character. */

    while (source_index < source_length)
    {
        current_byte = psource[source_index];

        if (current_byte == attr_key)
        {
            if (source_index < (source_length - 1))
            {
                source_index++;
                current_attr = psource[source_index];
            }
            else
            {
                pdestination[destination_index++] = current_attr;
                pdestination[destination_index++] = current_byte;
            }
        }
        else
    }
}

```

```

    {
        if ((current_byte == char_key) && ((source_index + 3)
            < source_length))
        {
            source_index++;
            character_count = *((int *) &psource[source_index]);
            source_index += sizeof(int);
            current_byte = psource[source_index];
        }
        else
            character_count = 1;

        while (character_count--)
        {
            if (destination_index >= (destination_length - 1))
                return(expanded_len);

            pdestination[destination_index++] = current_byte;
            pdestination[destination_index++] = current_attr;
        }
        source_index++;
    }

    return(expanded_len);
}

```

### 实用程序函数设计示例

NOANSI.C      终止并驻留，以关闭和开启ANSI.SYS的功能。

/\*\* The command line format is as follows:

\*      noansi [-r]

\* NOANSI is a self-removing TSR. If it is invoked with no arguments, it

\* installs itself (if not already installed). The -r option will cause

\* it to remove itself if it is installed.

\*/

#include <dos.h>

#include <stdio.h>

#include <bintrv.h>

#include <bintrpt.h>

#include <bkeybrd.h>

#include <bkeys.h>

#include <butil.h>

#include <ctype.h>

/\* The declaration of the intervention function.      \*/

/\* It will be called when the hot key is pressed.      \*/

void keyhandler(TV\_EVENT \*);

#define TRUE 1

#define FALSE 0

#define NUL '\0'



```

#define STKSIZE 2000
#define OK      0
#define FAL'_OUT 101
#define NOT_FOUND 1
#define ERROR_DISABLING2
#define ERROR_REMOVING3
#define ALREADY_INSTALLED 4
#define NO_ANSI_INSTALLED 5
        /* Signature for this version of CTRLANSI. */
#define NOANSI_SIGN "NOANSI03/31/89"
        /* Allocation of stack space for the intervention */
        /* scheduler and user function. */
char schedst[STKSIZE];
        /* This is the data structure to pass to the */
        /* scheduler so that it will give control every */
        /* time ALT A is pressed. */
IV_KEY keytab [] =
{
    {KB_C_A_A,    KB_S_A_A,    IV_KY_SERVICE}, /* Activate ANSI */
    {KB_C_A_D,    KB_S_A_D,    IV_KY_SERVICE} /* Disable ANSI */
};
#define k(c) kbscanof(c)
        /* Internal functions -- install and remove */
        /* intervention function. */
int install_iv(void);
int remove_iv(void);
void main(int, char **);
void main(argc, argv)
int    argc;
char *argv[];
{
    if (argc == 1)
        exit(install_iv());
    if ((argc == 2) && ((argv[1][0] == '-') || (argv[1][0] == '/')))
        if (toupper(argv[1][1]) == 'R')
            exit(remove_iv());
    printf("usage: ctrlansi [-r]\n");
    exit(0);
}

/*
 * Name      INSTALL_IV -- Install interrupt vectors for NOANSI,
 *            then terminate and stay resident.
 * Synopsis  ret = install_iv();
 *            int ret      Return code from IVINSTAL if an
 *            error condition was encountered.
 * Description This function installs NOANSI if another copy
 *            is not already installed, and ANSI.SYS is installed.
 * Returns   ret      ALREADY_INSTALLED (4)
 *            A copy of CTRLANSI is already installed.

```

```

*          NO_ANSI_INSTALLED (5)
*          ANSI.SYS is not installed.
*          FALL_OUT (101)
*          ISRESEXT() failed.
**
int install_iv()
{
    int         ercode;
    IV_VECTORS vecs;
        /* Check to see if CTRLANSI already installed.          */
    ivvecs(IV_RETVEC, &vecs);
    if (ivsense(&vecs, NOANSI_SIGN) != FAPNIL)
    {
        puts ("NOANSI already installed.");
        return(ALREADY_INSTALLED);
    }
        /* Check to see if ANSI.SYS is installed.              */
    if (UT_ANS_ABSENT == utansi(UT_ANS_DETECT))
    {
        puts("ANSI.SYS is not installed.\n");
        return(NO_ANSI_INSTALLED);
    }
        /* Install the intervention routine.                      */
    ercode = ivinstal(keyhandler, NOANSI_SIGN, schedstk, STKSIZE,
        keytab, sizeof(keytab) / sizeof(IV_KEY),
        NIL, 0,
        /* utansi() uses DOS functions, including 1-12.          */
        IV_DOS_NEED | IV_DKEY_NEED | IV_NO_FLOAT_NEED);
    if (ercode != 0)
    {
        printf("IVINSTAL error %d.\n", ercode);
        return(ercode);
    }
        /* Terminate and stay resident.                          */
    isrcsext(OK);
        /* Should never get here.                                */
    return(FALL_OUT);
}
/**
* Name      REMOVE_IV -- Remove a previously installed copy of
*            NOANSI.
* Synopsis  ret = remove_iv();
*            int ret      Return code.
*            NO_ERROR(0)-
*                No error encountered.
*            NOT_FOUND (1)-
*                NOANSI not found.
*            ERROR_DISABLING (2)-
*                NOANSI could not be disabled.
*            This error should *never* be

```

```

*                                     seen.
*                                     ERROR_REMOVING (3)—
*                                     NOANSI could not be removed (most
*                                     likely overwritten MALLOC
*                                     pointers).
* Description   This function removes a currently-active copy of NOANSI
*               from memory, restoring interrupt vectors and freeing
*               memory in the process.
* Returns       ret (nonzero if error—see above).
**/

```

```

int remove_iv()
{
    IV_VECTORS vecs;
    IV_CTRL far *pivctrl;
    /* Check to see if NOANSI installed. */
    ivvecs(IV_RETVEC, &vecs);
    if ((pivctrl = ivsense(&vecs, NOANSI_SIGN)) == FARNIL)
    {
        puts("NOANSI not found.");
        return(NOT_FOUND);
    }
    /* Make sure to leave ANSI.SYS enabled on exit. */
    utansi(UT_ANS_ENABLE);
    if (ivdisabl(pivctrl))
    {
        puts("Error disabling NOANSI.");
        return(ERROR_DISABLING);
    }
    if (isremove(pivctrl->psp))
    {
        puts("Error removing NOANSI.");
        return(ERROR_REMOVING);
    }
    return(0);
}

```

```

/**
* Name          KEYHANDLER — Handle hot key events.
* Synopsis      (To be called only by intervention code scheduler)
*               keyhandler(pevent);
*               IV_EVENT *pevent      Pointer to intervention event
*                                     structure. The structure
*                                     contains the hot key which
*                                     was detected (if any), as
*                                     well as other data.
* Description   This function accepts control from the scheduler
*               every time a defined hot key is pressed. ALT-A will
*               disable ANSI.SYS; ALT-D will disable it.
* Returns       None.
**/

```

```

void keyhandler(pevent)
IV_EVENT *pevent;
{
    int error;

    /* Do a key action, if one exists. */
    switch (pevent->key.action)
    {
        /* If the user typed a "service" key, perform the
        /* appropriate action. */
        case IV_KY_SERVICE:
            if (pevent->key.keycode == KB_S_A_A)
            {
                error = utansi(UT_ANS_ENABLE);
                if (error != UT_ANS_ENABLE)
                    printf("NOANSI: Error enabling ANSI.SYS.\n");
            }
            else
            {
                error = utansi(UT_ANS_DISABLE);
                if (error != UT_ANS_DISABLE)
                    printf("NOANSI: Error disabling ANSI.SYS.\n");
            }
            break;
    }
}

```

## 第十五章 窗口系统高级程序设计

### 窗口功能概述

Turbo C TOOLS 窗口函数提供了各种字符窗口应用程序所必需的功能和灵活性。系统自动处理许多高级效果,用户为标准应用程序建立简单的窗口也非常容易。每个窗口“出生”时带有一组缺省的特性,用户可以安全地省略许多复杂设置。

下面是 Turbo C TOOLS 窗口所具有的基本特性:

窗口在显示时可带有多种边界和标题或根本不带有边界。

窗口是虚拟的,也就是说,它可以超出它的显示边界的尺寸,用户可以借助鼠标器或键盘浏览一个更大的数据区。不论窗口是否是当前显示的,它们都具有一组灵活的 I/O 函数用于向窗口写或从窗口读出数据。除了用于窗口的 printf() 之外,这些函数还可以控制颜色和属性。

窗口可以被删除,删除时恢复原始屏幕的内容。然而,也可以指定窗口为不可删除的,以便于节省本来用来记录原屏幕数据的内存空间。窗口可以相互迭盖,系统自动处理迭盖窗口中的输入和输出操作。

Turbo C 的字符窗口可以与 Turbo C TOOLS 窗口联合使用。

窗口的显示和处理可以在多个显示页上进行。

一个窗口的输出可以被延迟,这样挂起的改变内容可以在一瞬间出现。

每个窗口可以具有自己独立的可控制的光标。

### 窗口函数的种类

#### 建立和释放窗口结构

- |          |                                     |
|----------|-------------------------------------|
| WNCREATE | 分配并建立一个 BWINDOW 结构,包括窗口数据区的初始拷贝。    |
| WNDSTROY | 释放一个 BWINDOW 结构以及有关的内容数据结构。(不改变屏幕。) |

#### 显示和删除窗口

- |          |  |
|----------|--|
| WNDSPLAY | 用可选的边界显示一个窗口并选择这个窗口为“当前窗口”,用于以后的 I/O 请求。它还选定该窗口(对于它的显示页上的其它窗口来说)具有活动光标。如果相关的话, Turbo C 字符窗口被设置为与新窗口的视口相匹配。 |
| WNREMOVE | 从显示页上取消一个窗口,恢复屏幕的原内容,窗口不再与它的显示位置相关联。如果窗口具有活动光标,光标即被关闭。如果相关的话, Turbo C 字符窗口被置为整个显示屏幕,光标被关闭。                 |
| WNREDRAW | 重现显示页上的所有窗口并恢复光标。(如果其它进程影响了屏幕,这个功能是很有用的。)  |

#### 虚拟窗口

- |          |  |
|----------|--|
| WNVDISP  | 在视口中显示一个窗口,该视口可以比窗口数据区小。这个函数与 WNDSPLAY 很相似。                |
| WNORIGIN | 在视口中移动窗口,显示数据区的各个部分。                                       |
| WNSHOBLE | 在视口中移动窗口。如果可能的话,强制显示数据区中需要的那一部分。                           |
| WNREAD   | 允许用户借助于鼠标器或键盘在视口中移动窗口。如果窗口未显示, WNREAD 即在视口中显示这个窗口,事后取消该窗口。 |

#### 窗口输出

- |          |                                     |
|----------|-------------------------------------|
| WNPRINTF | 以 printf() 的方式格式化一个字符串,将该字符串写入当前窗口。 |
| WNWRAP   | 以 TTY 方式将一个字符串写入当前视口,带有整字换行。        |

WNWRTTY	以 TTY 方式向当前窗口写入一个字符。
WNWRSTR	以 TTY 方式向当前窗口写入一个字符串。
WNWRSTRN	以 TTY 方式向任意窗口写入一个字符串, 带有几个可选项。
WNWRBUF	将缓冲区写入当前窗口。它不带有整字换行, 不滚动屏幕, 对字符不做特殊处理。
WNWRRECT	向窗口中一个矩形区域做写入操作。

## 窗口输入

WNQUERY	返回来自用户的输入, 在当前窗口回显所有的按键。
WNRDBUF	从当前窗口(不是从用户)读取数据, 写入到缓冲区中。
WNFIELD	允许用户编辑窗口中的一个域。要想进一步了解有关的内容, 请参见有关域编辑一章。

## 滚动和清除

任意一个滚动函数都可以清除一个区域。	
WNSCROLL	垂直滚动当前窗口的数据区。
WNHORIZ	水平滚动当前窗口的数据区。
WNSCRBLK	垂直或水平滚动窗口中的一个矩形块。

## 控制属性

WNATTR	修改当前窗口数据区的属性。(它不修改窗口的缺省属性, 而 WNSETOPT 可以修改。)
WNATRBK	修改窗口中一个矩形块的属性。
WNATRSTR	修改窗口中一系列连续位置的属性。

## 控制光标

WNCURMOV	移动当前窗口的光标。
WNCURPOS	返回当前窗口光标相对于该窗口的位置。
WNCURSOR	选择一个窗口(对于显示页上所有的窗口来说)使之具有活动光标。

## 控制窗口任选项

WNSELECT	选择一个窗口作为“当前窗口”, 用于以后的 I/O 请求。如果相关, 该窗口也被作为 Turbo C 字符窗口而建立。
WNUPDATE	在可能的情况下完成任何挂起的输出请求。
WNSETOPT	设置窗口任选项, 如下所示: 光标外观; 窗口数据区的缺省属性; 窗口输出是“延迟的”还是“立即的”; 窗口是否是“可删除物”; 虚拟窗口是否自动移动, 使窗口光标在视口中保持可见(“自动跟踪”视口间隙。
WNSETOPT	也可以迫使窗口数据结构在使用之前被分配。
WNGETOPT	返回窗口任选项的值或窗口状态的某些参数, 如下所示: 尺寸、属性或边界; 位于窗口显示位置的视口及被遮盖区域的尺寸; 光标尺寸或位置(相对于窗口数据区或屏幕); 窗口是否是“延迟的”、“可删除的”, 窗口是否被指定为“当前的”; 虚拟窗口是否移动使窗口光标在视口中保持可见(“自动跟踪”);

	视口间隙;
WNSETBUF	强行分配 WNPRINTF 所使用的缓冲区。
WNEROR	在全局变量 b_wncrr 中记录窗口的错误代码。

## 使用 Turbo C 的字符窗口

WNREVUPD      用当前显示在屏幕上的数据更新当前窗口数据区已保存的拷贝,这对于使 Turbo C TOOLS 窗口与通过 Turbo C 字符窗口或其它手段显示的数据相同步是很有用的。

## 为 WNREAD 进行窗口输入做准备

WNSCRLBR	设置一个窗口的滚动箭头, 这些滚动箭头可以指向水平方向(左右移动)或垂直方向(上下移动)或指向水平垂直两个方向。
WNINITEV	建立一个 WNREAD 所识别的键盘和鼠标器事件的缺省表。
WNCHGEVN	插入或修改事件表中的一个事件。
WNREMEVN	从事件表中删除一项。
WNZAPEVN	删除整个事件表。

## 使用高级窗口特性

### 显示和更新窗口

在某些应用程序中,向窗口填充信息的过程可能是比较缓慢的,用户于是更希望使输出延迟,直到某个时刻。下面的语句可以做到这一点:

```
wnsetopt(pwindow, WN_DELAYED, 1);
下面的语句使窗口成为“立即的”;
wnsetopt(pwindow, WN_DELAYED, 0);
用 WNUUPDATE 可以迫使任何挂起的输出送往屏幕。
```

### 虚拟窗口

用户可以在比窗口数据区小的视口中显示任意的窗口。(视口是显示屏幕上的一个矩形区域,用于显示窗口数据区的一部分。)要在可能比窗口小的视口中显示一个窗口,需使用 WNVDISP。(WVNDSPY 与 WNVDISP 相同,不过 WNDSPY 使用与窗口同尺寸的视口。)

按照缺省,窗口数据区的显示部分是自动控制的,窗口自动移动,使得光标位置保持可见(即使窗口光标被关闭)。这个特性称为自动光标跟踪或自动跟踪。下面的语句可以使这个特性失效:

```
wnsetopt(pwin, WN_CUR_TRACK, 0);
在关闭之后若想重建自动跟踪,可使用下面的语句:
wnsetopt(pwin, WN_CUR_TRACK, 1);
```

通过使用视口间隙用户还可以提高自动跟踪的性能。间隙使窗口在移动时让光标位置与视口边沿保持一个间隙。用户可能通过 WNSETOPT 分别控制四个边沿的间隙。当所有四个边沿间隙的初始值是 0 时,间隙不起作用。当自动跟踪关闭时,间隙只影响 WNSHOBK、MNREAD 和 MNLREAD。

### 不可删除的窗口

通常窗口是可删除的,即在任何时候都可以利用 WNREMOVE 从屏幕上取消某个窗口,恢复屏幕的原内容。每个 BWINDOW 结构包含着一些信息,这些信息保存着屏幕的原内容。

然而,为了节省内存空间和时间,用下面的语句可以随时指定一个窗口为不可删除窗口:

```
wnsetopt(pwindow, WN_REMOVABLE, 0);
```

这将释放用于记录原屏幕图象的空间。为节省大部分的内存,可以在窗口显示之前发出这个函数调用。

## 控制内存分配

窗口函数利用标准的内存分配函数 malloc()和 calloc()来建立数据结构和缓冲区,这对于中止并驻留的程序来说是比较危险的。为此,用户也许希望确保在程序中止并驻留之前使窗口数据结构和缓冲区得到分配。为了做到这一点,在程序中止之前可以对每个窗口执行下面的语句

```
wnsetopt(pwindow,WN_PREV_ALLOC, 1);
```

如果使用 WNPRINTF, 在程序中止之前应激活 WNSSETBUF; 如果使用 WNREAD, 在程序中止之前需激活 WNINITEV 和 WNSCRLBR。这些步骤将分配必需的信息项, 因而 WNDSPY, WNPRINTF 和 WNREAD 就不必这样做了。

如果使用了 Turbo C TOOLS 的选单或帮助系统, 请参见“选单函数(MN)”和“帮助系统(HL)”下的“控制内存分配”一节。

## 窗口扩展函数源程序

### WNATRBLK 修改窗口中一个矩形块的属性

```
#include <bwindow.h>
```

```
BWINDOW *wnatrbk( BWINDOW *pwin,
                  int r1,int c1,
                  int r2, int c2,
                  int fore, int back,
                  int option);
```

pwin 待修改的 BWINDOW 结构的地址。  
r1, c1 块左上角的行和列(相对于窗口左上角(0,0))。  
r2,c2 块右下角的行和列。  
fore,back 新前景和背景属性(如果不改变,则为-1)。  
option 为下列值之一:

符号	值	意义
WN_UPDATE	0	除非窗口被“延迟”, 否则更新窗口。
WN_NO_UPDATE	4	不更新窗口。

(返回) 改变后的 BWINDOW 结构的地址。如果失败, 返回 NIL。

WNATRBLK 用给定属性填充一个矩形块而不影响已显示的字符, 光标保持不动。如果 fore 或 back 是-1, 则前景或背景属性分别保持不变。

WNATRBLK 不影响窗口的缺省属性, WNSETOPT 会产生影响, 而 WNATRSTR 可以修改窗口中一系列连续位置的属性。

如果块尺寸超过窗口数据区, 这时出现一个错误。

源程序(WNATRBLK.C):

```
#include <bwindow.h>
#define H (wndata_h(pwin))
#define W (wndata_w(pwin))
#define LENGTH ((c2 - c1) + 1)
#define START ((W * r) + c1)
BWINDOW *wnatrbk (pwin, r1, c1, r2, c2, fore, back, option)
BWINDOW *pwin;
int r1, c1;
int r2, c2;
int fore, back;
int option;
{
    int endpos, i, r;
```



```

CELL *pdata;
char oldmask = '\0';
char attr;

/* Validate window data structures. */
wnvalidw (pwin)

/* Make sure coordinates make sense. */
if (utrange (r1, 0, H - 1) || utrange (r2, 0, H - 1) ||
    utrange (c1, 0, W - 1) || utrange (c2, 0, W - 1) ||
    (c1 > c2) || (r1 > r2))
    wnretterr (WN_ILL_DIM);

/* "oldmask" will help us extract the portions of
/* the existing attributes that we want to keep. */
if (fore == -1)
    oldmask = 0x0f;
if (back == -1)
    oldmask |= 0xf0;

attr = (char) (utnybbyt (back, fore) & ~oldmask);
pdata = pwin->img.pdata;

/* Stuff the new attribute into the window.
for (r = r1; r <= r2; r++)
{
    endpos = (START + LENGTH);
    for (i = START; i < endpos; i++)
        pdata[i].attr = (pdata[i].attr & oldmask) | attr;
}

/* Write whole window unless delayed or not shown. */
return (wnupblk (pwin, r1, c1, r2, c2, option));
}

```

## WNATRSTR 改变窗口中一片连续位置的属性

```

#include <bwindow.h>
BWINDOW *wnatrstr( BWINDOW *pwin,
                  int row, int col,
                  int count,
                  int fore, int back,
                  int option);

```

pwin 待修改的 BWINDOW 结构的地址。  
 row, col 待修改的第一个位置的行和列(相对于窗口数据区左上角(0,0))。  
 count 待修改的位置的数目。  
 fore 新的前景属性(如果不改变前景, 则为-1)。  
 back 新的背景属性(如果不改变背景, 则为-1)。  
 option 为下列值之一:

符号	值	意义
WN_UPDATE	0	除非窗口被“延迟”, 否则更新窗口
WN_NO_UPDATE	4	不刷新窗口。

(返回) 修改后的 BWINDOW 结构的地址。如果失败, 返回 NIL。

WNATRSTR 用给定属性填充窗口中一系列连续位置, 被改变区域可以在窗口内折转到另外的行上, 直到到达窗口末端。已显示的字符保持不变, 窗口不发生滚动, 光标也不移动。

如果 fore 或 back 是 -1, 前景或背景属性保持不变。

如果 count 超过窗口的末端, 则被改变区域以窗口末端为限。

WNATRSTR 不影响窗口的缺省属性, WNSETOPT 会产生影响, 而 WNATRBLK 修改窗口中一个矩形的属性。

如果 row 或 col 所指定的位置超出了窗口数据区, 这时出现一个错误。

### 源程序(WNATRSTR.C):

```
#include <bwindow.h>
#define H      (wndata_h(pwin))
#define W      (wndata_w(pwin))
#define AREA   (H * W)
#define START  ((W * row) + col)
BWINDOW *wnatrstr (pwin, row, col, count, fore, back, option)
BWINDOW *pwin;
int      row, col;
int      count;
int      fore, back;
int      option;
{
    int      endpos, i;
    CELL *pdata;
    char      oldmask = '\0';
    char      attr;

    /* Validate window data structure. */
    wnvalidw (pwin)

    /* Make sure (row,col) is in window. */
    if (utrange (row, 0, H - 1) ||
        utrange (col, 0, W - 1))
        wnreterr (WN_ILL_DIM);

    /* "oldmask" will help us extract the portions of
       /* the existing attributes that we want to keep. */
    if (fore == -1)
        oldmask = 0x0f;
    if (back == -1)
        oldmask |= 0xf0;

    attr = (char) (utnybbyt (back, fore) & ~oldmask);
    endpos = ((START + count > AREA) ? (AREA - START) : (count))
        + START - 1;
    pdata = pwin->img.pdata;
    for (i = START; i <= endpos; i++)
        pdata[i].attr = (pdata[i].attr & oldmask) | attr;
        /* Update just a section of this line if we can get
           /* away with it. */
    if (row == (int) endpos / W)
        return(wnnupblk (pwin, row, col, row, col + (i - START) - 1,
            option));
        /* Otherwise update all lines changed. */
    return(wnnupblk (pwin, row, 0, (int) endpos / W, W - 1, option));
}
```

## WNATTR 改变当前窗口的属性

```
#include <bwindow.h>
```

```
BWINDOW *wnattr ( int fore,
                  int back);
```

fore 新前景属性(如果不改变前景, 为-1)。

back 新背景属性(如果不改变背景, 为-1)。

(返回) 改变后的 BWINDOW 结构的地址。如果失败, 返回 NIU。

WNATTR 是一个宏, 它用给定的属性填充当前窗口的数据区而不影响已显示的字符。WNATTR 不移动光标。

如果 fore 或 back 是-1, 则前景或背景属性保持不变。

WNATTR 不影响窗口的缺省属性, WNSETOPT 会产生影响, 而 WNATRBLK 或 WNASTRSTR 修改窗口某一部分的属性。

如果不存在当前窗口, 这时出现一个错误。

### 源程序(BWINDOW.H):

该宏定义在头文件 HWINDOW.H 中。参见附录 C。

## WNCHGEVN 加入或修改 WNREAD 认可的一个用户响应

```
#include <bwindow.h>
```

```
int wnchgevn( WN_EVENT_LIST **ppfirst,
              const WN_EVENT *pevent
              WN_ACTION action);
```

ppfirst 指针的地址 该指针指向认可的用户响应表的第一项。

pevent 待拷贝到表中或待修改的 WN\_EVENT 结构的地址。

action 对于已在表中的用户响应, 这是新动作码, 表示这个用户响应的结果。见 WNREAD 中的值表。

(返回) 错误代码。可能值包括:

WN\_NO\_ERROR (0) 成功。

WN\_NO\_MEMORY (1) 无足够动态内存来分配给新表成员。

WN\_NUL\_PTR (3) ppfirst 或 pevent 为 NIL, 不执行任何操作。

WNCHGEVN 加入或修改 WNREAD 认可的响应表中的一项。如果表中已有的项与\*pevent 匹配, 则其动作改变为 action; 否则\*pevent 的拷贝加入到表中。见 WNREAD 中有关 WN\_EVENT 和 WN\_ACTION 数据类型的说明。

如果表最初为空(即\*ppfirst 为 NIL), 则\*ppfirst 改指向新加入的表成员。

WNCHGEVN 不改变全局窗口错误代码 b\_wnerr。

### 源程序(WNCHGEVN.C):

```
#include <bwindow.h>
```

```
#include <bkeys.h>
```

```
#include <butil.h>
```

```
#include <string.h>
```

```
int wnchgevn(pplist_head, pevent, action)
```

```
WN_EVENT_LIST **pplist_head;
```

```
WN_EVENT *pevent;
```

```
WN_ACTION action;
```

```

{
    WN_EVENT      *pfound_event;
    WN_EVENT_LIST *pwin_event_node;
    if ((pplist_head == NIL) || (pevent == NIL))
        return(WN_NULL_PTR);
    pfound_event = wnretnv(*pplist_head, pevent);
    if (pfound_event == NIL)
    {
        /* The specified event is not in the list; add it. */
        pwin_event_node = (WN_EVENT_LIST *)
            malloc(sizeof(WN_EVENT_LIST));
        if (pwin_event_node == NIL)
            return(WN_NO_MEMORY);

        pwin_event_node->signature = BEVENT_SIGN;
        pwin_event_node->win_event = *pevent;

        /* Now make a copy of the event's pdata. */
        pwin_event_node->win_event.pdata = malloc(pevent->data_size);
        if (pwin_event_node->win_event.pdata == NIL)
            return(WN_NO_MEMORY);
        memcpy(pwin_event_node->win_event.pdata, pevent->pdata,
            pevent->data_size);

        if (*pplist_head == NIL)
        {
            pwin_event_node->pnext = pwin_event_node;
            pwin_event_node->pprev = pwin_event_node;
            *pplist_head = pwin_event_node;
        }
        else
        {
            pwin_event_node->pprev = (*pplist_head)->pprev;
            pwin_event_node->pnext = *pplist_head;
            if ((*pplist_head)->pprev != NIL)
                (*pplist_head)->pprev->pnext = pwin_event_node;
            (*pplist_head)->pprev = pwin_event_node;
            *pplist_head = pwin_event_node;
        }
    }
    else
    {
        /* Change the action for the specified event. */
        *((WN_ACTION *) pfound_event->pdata) = action;
    }
    return(WN_NO_ERROR);
}

```

**WNCREATE**      建立一个窗口结构

```
#include <bwindow.h>
```

```
BWINDOW *wncreate( int heigh, int width,  
                   int attr);
```

height      数据区的行数。

width        数据区的列数。

attr        数据区的缺省属性。

(返回)      新建立的 BWINDOW 结构的地址。如果失败，返回 NIL。

WNCREATE 为 BWINDOW 结构分配空间，用缺省值填充该空间，这包括分配窗口的数据区，用具有指定属性的空格(' ')初始化该数据区。

窗口数据区的总面积(高乘宽)不能超过 32,767。

如果内存分配失败，则出现一个错误。

WINDSTROY 废弃 BWINDOW 结构及相关的内部数据结构。

## 源程序(BWINDOW.H):

该宏定义在附录 C 的 BWINDOW.H 中。

**WNCOVER**      如果被一给定的长方形区域遮盖，则标为低层窗口。

```
presult = wncover (pnode, pcorner, pdim);
```

presult      指向检查的最高层窗口节点，如果失败则指向 NIL。

pnode        指向检查的最高层窗口节点。

pcorner      指向说明长方形区域左上角的结构。

pdim        指向说明长方形区域尺寸的结构。

该函数比较一给定区域窗口下的所有窗口，并标志那些窗口被区域覆盖。

如果有活动光标的窗口被覆盖，则使光标变成非活动态，并关闭屏幕的光标。

返回        presult      指向检查的最高层窗口节点。如果失败则指向 NIL。

             b\_pactnode[]    有活动光标的窗口

             b\_werr        可能值:

             (不改变)    成功

             WN\_BAD\_NODE    无效的 pnode.

             WN\_BAD\_WIN    内部错误 Internal error.

## 源程序(WNCOVER.C):

```
#include <bwindow.h>
```

```
WIN_NODE *wncover (pnode, pcorner, pdim)
```

```
WIN_NODE *pnode;
```

```
const LOC *pcorner;
```

```
const DIM *pdim;
```

```
{
```

```
    WIN_NODE *presult;
```

```
    BWINDOW *pwin;
```

```
    int      row, col, high, low;
```

```
    wnvalidn (pnode)
```

```
    pwin = pnode->pwin;
```

```
    wnvalidw (pwin)
```

```
    if (wnovrlap (pcorner,            /* If rectangle overlaps this    */
```

```
                 pdim,                /* window,                       */
```

```
                 &(pwin->where_shown.corner),
```

```
                 &(pwin->view_size)))
```

```

{
    /* mark this window as covered. */
    pwin->internals.frozen =
    pwin->internals.any_data_covered =
    pwin->internals.cur_frozen = 1;

    /* If this window's cursor is
    /* activated, turn physical cursor
    /* off and deactivate window. */
    if (pnode == b_pactnode[pwin->where_shown.dev][pwin->where_shown.page])
    {
        sccurst (&row, &col, &high, &low);
        sepgeur (1, high, low, CUR_NO_ADJUST);
        b_pactnode[pwin->where_shown.dev][pwin->where_shown.page] = NIL;
    }
}
result = pnode;
if (pnode->below != NIL) /* Continue with lower windows. */
    if (wncover (pnode->below, pcorner, pdim) == NIL)
        result = NIL;

return (result);
}

```

## WNCREAT0 -- 分配和创建一窗口结构。

```

result = wncreat0( height,width,attr,
                  win_sign,node_sign);

```

result 指向新创建的 BWINDOW 结构，如果失败则指向 NIL。  
height 数据区的行数。  
width 数据区的列数。  
attr 数据区的缺省属性。  
win\_sign 窗口结构的标签值。  
node\_sign 窗口节点的标签字。

函数为窗口结构分配空间，并以缺省值填充。包括分配和用指定属性的空格窗口数据区。

如果窗口尺寸大于屏幕最大尺寸或内存分配失败，则产生错误。

用 WNDSTROY 释放 BWINDOW 结构，与相关的内部数据结构。

该函数由 wncreate() 宏调用，后者提供标签值。相同的标签值用宏 wncalwi0() 和 wncaln0d() 各自提供给 WNCALWI0() 和 WNCALNOD0()。通过使用这些宏窗口提供了比较的标签值，以便检测 BWINDOW 的不兼容版本。

返回	result	指向新创建的 BWINDOW 结构，如果失败则指向 NIL。
	b_wnerr	可能值：
		(不改变) 成功
		WN_ILL_DIM 无效的尺寸
		WN_NO_MEMORY 内存不够。

## 源程序(WNCREAT0.C):

```

#include <stdlib.h>
#include <bwindow.h>
int b_wnerr = WN_NO_ERROR; /* Most recent error (WN_NO_ERROR */
                          /* if none). */

```

```

BWINDOW *wncreat0(height,width,attr,win_sign,node_sign)
int height,width,attr;
unsigned win_sign,node_sign;
{
    BWINDOW *pwin;
    CELL *ptr;
    int area,i;

    /* Make sure the window's dimensions are legal. */
    if (((unsigned long) height) * ((unsigned long) width)
        * 2L) > ((unsigned long) 0xffff))
        wnreterr (WN_ILL_DIM);
    /* Allocate space for window structure. */
    if ((pwin = utalloc (BWINDOW)) == NIL)
        wnreterr (WN_NO_MEMORY);
    pwin->img.dim.h = height;
    pwin->img.dim.w = width;
    /* Allocate space for node structure. */
    if ((pwin->pnode = utalloc (WIN_NODE)) == NIL)
    {
        free ((char *) pwin);
        wnreterr (WN_NO_MEMORY);
    }
    pwin->pnode->pwin = pwin;
    /* Allocate space for memory image. */
    if ((pwin->img.pdata =
        (CELL *) calloc ((unsigned int)
                        (wndata_h(pwin) * wndata_w(pwin)),
                        sizeof (CELL))) == NIL)
    {
        free ((char *) pwin->pnode);
        free ((char *) pwin);
        wnreterr (WN_NO_MEMORY);
    }

    /* Initialize window to all blanks with standard */
    /* attribute for this window. */
    area = height * width;
    for (ptr = pwin->img.pdata, i = 0; i < area; i++, ptr++)
    {
        ptr->ch = ' ';
        ptr->attr = (char) attr;
    }

    /* Initialize fields to default values (except */
    /* where 0 is an acceptable default value). */
    pwin->attr = attr;
    pwin->where_shown.dev = ABSENT;
    pwin->cur_type.high = 12;
    pwin->cur_type.low = 13;
    pwin->options.removable = 1;

```

```

pwin->options.cur_track = 1;

/* Distinctive signatures. */
pwin->signature = win_sign;
pwin->pnode->signature = node_sign;

return (pwin);
}

```

## WNCURMOV 移动当前窗口的光标

```
#include <bwindow.h>
BWINDOW *wncurmov(int row,int col);
row, col    新光标位置的行和列(相对于窗口数据区左上角(0,0))。
(返回)      当前 BWINDOW 结构的地址。若失败, 返回 NIL。
```

WNCURMOV 移动当前窗口的光标。如果“自动跟踪”有效, 窗口在其视口中移动以保持光标位置在视口间隙中可见。

除非通过 WNCURSOR 使活动光标保持在显示页上, 否则发生的变化不可见。如果窗口被“延迟”则新光标位置也不可见, 直到窗口被更新。

如果程序与 NW\_???OBJ(“原始窗口”)文件链接, Turbo C 字符窗口也被更新。

如果位置超出了窗口的界限或不存在“当前”窗口, 这时出现一个错误。在这种情况下, b\_wncerr 存放错误代码, NIL 作为函数值返回。

### 源程序(WNCURMOV.C):

```

#include <conio.h>
#include <bwindow.h>
#define CUR_MARGIN_LEFT (b_pcurwin->cur_left_marg)
#define CUR_MARGIN_RIGHT (b_pcurwin->cur_right_marg)
#define CUR_MARGIN_TOP (b_pcurwin->cur_top_marg)
#define CUR_MARGIN_BOTTOM (b_pcurwin->cur_bottom_marg)
/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (b_pcurwin->data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(b_pcurwin) - 1)
#define VISIBLE_TOP (b_pcurwin->data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(b_pcurwin) - 1)
BWINDOW *wncurmov (row, col)
int row, col;
{
    int old_origin_row;
    int old_origin_col;
    wnvalidw(b_pcurwin);
/* Validate requested position. */
    if (utrange (row, 0, wndata_h(b_pcurwin) - 1) ||
        utrange (col, 0, wndata_w(b_pcurwin) - 1))
        wnreterr (WN_ILL_DIM);

/* If window is not displayed, note new position */
/* and exit. */
    if ((b_pcurwin->where_shown.dev != SC_MONO) &&

```



```

    (b_pcurwin->where_shown.dev != SC_COLOR))
{
    b_pcurwin->cur_loc.row = row;
    b_pcurwin->cur_loc.col = col;
    return(b_pcurwin);
}

    /* Remember the data area origin coordinates. */
old_origin_row = b_pcurwin->data_origin.row;
old_origin_col = b_pcurwin->data_origin.col;
    /* Now perform cursor auto tracking. */
wncurtrk(b_pcurwin, row, col);
    /* Now update the window's image if necessary. */
if ((old_origin_row != b_pcurwin->data_origin.row) ||
    (old_origin_col != b_pcurwin->data_origin.col))
{
    if (NIL == wnupblk(b_pcurwin, 0, 0, wndata_h(b_pcurwin),
                      wndata_w(b_pcurwin), WN_UPDATE))
        return(NIL);
}

    /* Now set the physical cursor's location and state.*/
wncurset(b_pcurwin);

return (b_pcurwin);
}

```

## WNCUPRPOS      返回当前窗口的光标位置

```
#include <bwindow.h>
```

```
BWINDOW *presult = wncurpos( int *prow,
                             int *pcol);
```

prow      返回光标行位置的变量的地址(0 = 数据区的顶行)。

pcol      返回光标列位置的变量的地址(0 = 数据区的左列)。

(返回)      当前 BWINDOW 结构的地址。若失败, 返回 NIL。

WNCURPOS 返回相对于窗口数据区左上角的当前窗口的光标位置。

如果不存在当前窗口, 则出现一个错误。在这种情况下, b\_wnerr 保存一个错误代码, NIL 作为函数值返回。

### 源程序(WNCURPOS.C):

```
#include <bwindow.h>
```

```
BWINDOW *wncurpos(prow, pcol)
```

```
int *prow, *pcol;
```

```

{
    wnvalidw (b_pcurwin)
    *prow = b_pcurwin->cur_loc.row;
    *pcol = b_pcurwin->cur_loc.col;
    return (b_pcurwin);
}

```

## WNCURSOR      激活一个窗口光标

```
#include <bwindow.h>
```

```
BWINDOW*wncursor (BWINDOW *pwindow);
```

pwindow      待选择的 BWINDOW 结构的地址。

(返回)      新选择的 BWINDOW 结构的地址。若失败，返回 NIL。

WNCURSOR 激活窗口的光标(虽然光标可能是不可见的)，在那个显示页上其它窗口的光标被关闭。

如果 pwindow 未指向一个合法的窗口结构、窗口不是当前显示的或窗口数据区的某部分被另一个窗口覆盖，这时出现一个错误。

### 源程序(WNCURSOR.C:

```
#include <conio.h>
```

```
#include <bwindow.h>
```

```
/* The window (on each page) selected to      */  
/* have a visible cursor.                      */
```

```
WIN_NODE *(b_pactnode[MAX_DEVICES][MAX_PAGES]) = {0};
```

```
BWINDOW *wncursor (pwin)
```

```
BWINDOW *pwin;
```

```
{
```

```
int old_dev, old_page, old_npage;
```

```
int mode, columns, act_page;
```

```
BWINDOW *presult;
```

```
/* Validate window and window's node data structures*/
```

```
wnvalidw (pwin)
```

```
wnvalidn (pwin->pnode)
```

```
if (pwin->internals.any_data_covered)
```

```
    wnreterr (WN_COVERED);
```

```
if (pwin->options.hidden)
```

```
    wnreterr (WN_NOT_SHOWN);
```

```
/* Note former device; note current page on former */
```

```
/* device; select and validate device and page.      */
```

```
old_dev = scmode (&mode, &columns, &act_page);
```

```
old_page = b_curpage;
```

```
if (wnseldev (&pwin->where_shown, &pwin->view_size, &old_npage))
```

```
    wnreterr (WN_NOT_SHOWN);
```

```
/* If a window on this page is currently designated */
```

```
/* to control the cursor, deactivate it.              */
```

```
if (b_pactnode[pwin->where_shown.dev][pwin->where_shown.page]
```

```
    != NIL)
```

```
    b_pactnode[pwin->where_shown.dev][pwin->where_shown.page]
```

```
        ->pwin->internals.cur_frozen = 1;
```

```
/* Activate this window instead.                      */
```

```
b_pactnode[pwin->where_shown.dev][pwin->where_shown.page] =
```

```
    pwin->pnode;
```

```

pwin->internals.cur_frozen = 0;
        /* Move the cursor, and set the dimensions.          */
presult = wncurtrk(pwin, pwin->cur_loc.row, pwin->cur_loc.col);
wncurset(pwin);
        /* Restore current page on new device; restore old    */
        /* device; restore current page on old device.        */
sepage (old_npage);
sechgdev (old_dev);
sepage (old_page);

return (presult);
}

```

## WNDSPLAY 在同尺寸视口中显示一个窗口

```
#include <bwindow.h>
```

```

BWINDOW *wndsplay( BWINDOW *pwin,
                   const WHERE *pwhere,
                   const BORDER *pbord);

```

pwin 待显示的 BWINDOW 结构的地址。

pwhereWHERE 结构的地址, 该结构指明了设备、显示页及窗口显示位置的坐标。

pbordBORDER 结构的地址, 该结构指明了边界和放置在视口周围的标题。

(返回) 显示后的 BWINDOW 结构的地址。若失败, 返回 NIL。

WNDSPLAY 在给定显示设备及显示页上显示一个窗口, 在其四周加上一个边界。(边界类型可以是“无边界”或包含上/下标题。)该窗口显示在与其数据区尺寸相匹配的视口中, 同时它被置为当前窗口(选定为 I/O 的窗口)。窗口的光标被激活(虽然光标可能是不可见的), 那一页上其它窗口的光标被关闭。如果程序与 NW\_???BJ 文件链接, 则 Turbo C 字符窗口设置成与视口相匹配。

由 pwhere 所指向的 WHERE 结构指明显示地址, 该结构在 bwindow.h 中定义如下:

```

typedef struct    /*LOC 结构: */
{
    /*相对于矩形区域的行和列。*/
    int row,col;
}LOC;
typedef struct    /*WHERE 结构: 显示窗口所在的设备、页和位置。*/
{
    int dev;      /*显示设备: SC_COLOR (1)、SC_MONO (0)。如果不*/
                  /* 显示, 为 SC_ABSENT (-2)。*/
    int page;     /*显示页号: 0 到 7*/
    LOC corner;   /*显示窗口数据区的左上角(不包括界)。*/
} WHERE;

```

由 pbord 所指向的 BORDER 指明窗口的边界, 该结构在 bwindow.h 中定义如下:

```

typedef struct    /*BORDER 结构: 窗口边界的描述。*/
{
    int type;     /*边界类型(见下面)。*/
    int attr;     /*属性(颜色)。*/
    char ch;      /*如果类型是 BBRD_CHAR, 则为边界字符。*/
    char_dumy;    /*使结构尺寸在任何情况下都是 偶数的 哑元。*/
    char *pttitle; /*上标题。*/
    char *pbtitle; /*下标题。*/
    int ttattr;    /*上标题的属性。*/
    int btattr;    /*下标题的属性。*/
}

```

```

    BJOINT *pjoins; /*保留用于以后使用。*/
} BORDER;

```

边界类型的说明如下所述。我们首先选择下列值之一指明要画的字符。

符号	值	意义
BBRD_NO_BORDER	0x000	无边界, 忽略其它位。
BBRD_CHAR	0x01f	字符 ch 用于边界。
BBRD_SSSS	0x001	(用 IBM 方框字符画方框; 这些值指明 SCBOX 使
BBRD_SSSD	0x002	用的方框类型, 但比 SCBOX 所用的类型值大 1。)
BBRD_DSSS	0x003	
BBRD_DSDD	0x004	
BBRD_SDSS	0x005	
BBRD_SDSD	0x006	
BBRD_DDSS	0x007	
BBRD_DDSD	0x008	
BBRD_SSDS	0x009	
BBRD_SSDD	0x00a	
BBRD_DSDD	0x00b	
BBRD_DSDD	0x00c	
BBRD_SDDS	0x00d	
BBRD_SDDD	0x00e	
BBRD_DDDS	0x00f	
BBRD_DDDD	0x010	

接着将所选的值与下列值的零个、任意个或全部结合起来, 这些值指明是否在主边界字符上定上标题:

符号	值	意义
BBRD_NO_TITLE	0x000	无标题。
BBRD_TLT	0x020	左上标题
BBRD_TCT	0x040	中上标题
BBRD_TRT	0x080	右上标题。
BBRD_BLT	0x100	左下标题。
BBRD_BCT	0x200	中下标题。
BBRD_BRT	0x400	右下标题。

如果视口的数据区与屏幕边沿相邻接(也就是说没有地方放置边界), 则不画出边界。

WNVDISP 在视口中显示一个窗口, 该视口可以比窗口的数据区小。

如果 pwin 未指向一个合法的窗口结构或窗口的显示位置不可能, 则出现一个错误。例如, 请求了一个未知的设备或视口的某个角超出了物理屏幕。

### 源程序(BWINDOW.H):

该宏定义在附录 C 的头文件 BWINDOW.H 中。

### WNCURTRK 移动窗口的光标, 如果必要的活调整数据区源。

```

presult = wncurtrk(pwin, row, col);
presult 指向指定的窗口结构, 如果失败则指向 NIL。
pwin     要移动的窗口。
row,col  新的光标相对于窗口左上角的行和列。

```

移动指定的窗口的光标, 如果窗口是光标自动跟踪的活, 则调整数据区源坐标, 不改变窗口的屏幕图象。

如果位置超出窗口边界则产生错误。

返回	presult	指向当前 BWINDOW 结构; 或者如果失败指向 NIL。
	pwin->cur_loc	当前 BWINDOW 光标位置, 如果失败不改变其值。
	b_wncrr	值如下:
		(不改变)            成功
	WN_ILL_DIM	位置超出范围
	WN_BAD_WIN	窗口无效

### 源程序(WNCURTRK.C):

```
#include <bwindow.h>
#define CUR_MARGIN_LEFT (pwin->cur_left_marg)
#define CUR_MARGIN_RIGHT (pwin->cur_right_marg)
#define CUR_MARGIN_TOP (pwin->cur_top_marg)
#define CUR_MARGIN_BOTTOM (pwin->cur_bottom_marg)
/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (b_pcurwin->data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(b_pcurwin) - 1)
#define VISIBLE_TOP (b_pcurwin->data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(b_pcurwin) - 1)
BWINDOW *wncurtrk(pwin, row, col)
BWINDOW *pwin;
int row, col;
{
    int old_data_row, old_data_col;
    wnvalidw(pwin);
    /* Validate requested position. */
    if (utrange (row, 0, wndata_h(pwin) - 1) ||
        utrange (col, 0, wndata_w(pwin) - 1))
        wnreterr(WN_ILL_DIM);
    /* Save the data area origin coordinates. */
    old_data_row = pwin->data_origin.row;
    old_data_col = pwin->data_origin.col;
    /* If the window is designated WN_CUR_TRACK and its */
    /* cursor is not frozen, adjust the coordinates of */
    /* the data area which appear in the viewport's */
    /* origin. */
    if ((pwin->options.cur_track) && !(pwin->internals.cur_frozen))
    {
        /* Check to see if cursor is now outside of margin */
        /* area. */
        if (row < VISIBLE_TOP + CUR_MARGIN_TOP)
            pwin->data_origin.row = row - CUR_MARGIN_TOP;
        if (row > VISIBLE_BOTTOM - CUR_MARGIN_BOTTOM)
            pwin->data_origin.row = row + CUR_MARGIN_BOTTOM -
                wnview_h(pwin) + 1;
        if (col < VISIBLE_LEFT + CUR_MARGIN_LEFT)
            pwin->data_origin.col = col - CUR_MARGIN_LEFT;
        if (col > VISIBLE_RIGHT - CUR_MARGIN_RIGHT)
            pwin->data_origin.col = col + CUR_MARGIN_RIGHT -
                wnview_w(pwin) + 1;
```

```

        /* Make sure data corner coordinates make sense. */
        utbound(pwin->data_origin.row, 0,
                wndata_h(pwin) - wnview_h(pwin));
        utbound(pwin->data_origin.col, 0,
                wndata_w(pwin) - wnview_w(pwin));
        /* Now mark the window as dirty if the window's
        /* data origin has changed. */
        if ((old_data_row != b_pcurwin->data_origin.row) ||
            (old_data_col != b_pcurwin->data_origin.col))
        {
            pwin->internals.dirty = 1;
        }
    }

    /* Assign the cursor coordinates. */
    pwin->cur_loc.row = row;
    pwin->cur_loc.col = col;
    return(pwin);
}

```

## WNDSTROY      废弃一个窗口结构

`#include <bwindow.h>`  
**int wndstroy(BWINDOW \*pwindow);**  
**pwindow**      待废弃的 BWINDOW 结构的地址。  
**(返回)**      错误代码。如果无错误，返回 WN\_NO\_ERROR。  
**WNDSTROY** 彻底废弃一个 BWINDOW 结构，它不影响任何屏幕显示页。  
 用 **WNDSTROY** 废除一个窗口结构后便不能再使用它(或它的任何组成部分)，它的内存已被释放。  
 如果 **pwindow** 未指向一个合法结构，则出现一个错误。  
 用 **WNDSTROY** 废除窗口之前可以用 **WNREMOVE** 恢复原屏幕内容。

### 源程序(WNDSTROY.C):

```

#include <string.h>
#include <bwindow.h>
int wndstroy (pwin)
BWINDOW *pwin;
{
    /* Validate window pointer. */
    if (wnvalwin (pwin) == NIL)
        wnreturn (WN_BAD_WIN);

    /* If it's showing, abandon border & location. */
    if (pwin->where_shown.dev == SC_MONO ||
        pwin->where_shown.dev == SC_COLOR)
        if (NIL == wnforget (pwin))
            return (b_wnerr);

    /* Free the image of previous screen data. */
    if (pwin->prev.pdata != NIL)
    {

```

```

    free ((char *) pwin->prev.pdata);
    pwin->prev.pdata = NIL;
}

    /* Free the window's event list, if any. */
wnzapevn(&(pwin->pevent_list));
    /* Free the memory copy of the window's data area. */
if (pwin->img.pdata != NIL)
{
    free ((char *) pwin->img.pdata);
    pwin->img.pdata = NIL;
}

    /* Mark the node abandoned and free it. */
pwin->pnode->signature = BNODE_DEAD;
pwin->pnode->pwin = NIL;
free ((char *) pwin->pnode);
pwin->pnode = NIL;

    /* Mark the window abandoned and free it. */
pwin->signature = BWIN_DEAD;
free ((char *) pwin);

return (WN_NO_ERROR);
}

```

## WNERROR          记录窗口或选单的系统错误

```

#include <bwindow.h>
int wnerror (int ercode);
ercode          指示错误类型的代码(如果无错, 为 WN_NO_ERROR (0)).
(返回)          ercode 的值。

```

WNERROR 将 Turbo C TOOLS 全局窗口错误指示器 b\_wncrr 设置为一个给定值。用 WNERROR 可以报告所有窗口和选单的错误。(b\_wncrr 在 bwindow.h 中声明, 在 wncreate.c 中初始化为 WN\_NO\_ERROR (0)。任何时候都可以检查 b\_wncrr 来确定最近的窗口或选单错误。)

源程序(WNERROR.C):

```

#include <bwindow.h>
#if WN_EXT_ERROR
#undef wnerror                      /* Un-define macro version. */
#endif

int wnerror (ercode)
int ercode;
{
    return (b_wncrr = ercode);
}

```

## WNFIELD          对窗口中的一个域进行编辑

```

#include <bedit.h>
#include <bstrings.h>              /*用于 STPCVT 中的常量。*/

```

```
#include <bwindow.h>
```

```
int wnfield( BWINDOW *pwin,
             const char *pinitstr,
             char *pretstr,
             int retsize,
             const ED_CONTROL *petrol,
             KEY_SEQUENCE *pfinal);
```

pwin 窗口地址。

pinitstr 最初显示在域中的字符串的地址 以 NUL 字符('\0')结尾。

pretstr 由来自用户的编辑数据填充的缓冲区的地址

retsize pretstr 指向的缓冲区的尺寸, 计入尾随的 NUL('\0')字符所占用的一个字节。

petrl 结构的地址, 该结构控制编辑过程。

pfinal 结构的地址, 该结构返回最后的按键。如不需要报告最后按键, 则为 NIL。

(返回) 错误代码, 可能值包括:

WN\_NO\_ERROR (0) 未探测到错误。

ED\_USER\_ABORT (302) 最后按键是一个夭折请求, 不返回 数据。

其它值 见 WN/MN/HL 错误。

WNFIELD 是一个宏, 它允许用户编辑窗口中的一个域。如果视口数据区的任何部分被另一个窗口遮盖, 则出现错误。

ED\_CONTROL 结构指明多个任选项, 包括数据在窗口中的显示区域。该结构在 bedit.h 头文件中定义为:

```
typedef struct /*LOC 结构: */
{
    /*相对于一个矩形区域的行和列。*/
    in row,col;
} LOC;

typedef struct /*DIM 结构: */
{
    /*矩形区域的高和宽。*/
    int h,w;
} DIM;

typedef struct /*CUR_TYPE 结构: */
{
    int high,low; /*光标的高低扫描行。*/
} CUER_TYPE;

typedef struct /*ED_CONTROL 结构: */
{
    /*编辑缓冲区的位置及其它信息。*/
    LOC ul_corner; /*左上角。*/
    DIM dimensions; /*域尺寸。*/
    int attribute; /*显示属性。*/
    CUR_TYPE replace_cursor; /*替换和插入方式下的光标尺寸。*/
    CUR_TYPE insert_cursor;
    PKEY_CONTROL control_function; /*键控制函数。*/
    unsigned int control_flags; /*wnfield()控制标 帜*/
} ED_CONTROL;
```

应按如下方式装载 ED\_CONTROL, 定义编辑域:

ul\_corner 指明相对于窗口数据区的域的位置, dimensions 指定域的高和宽。域的高度可以是 1 行或多行。retsize 限制了用户可以输入的字符数。

attribute 指明显示属性。设置 attribute 为 0 使用窗口的缺省属性。

replace\_cursor 和 insert\_cursor 指明当用户处于替换或插入方式时显示的光标尺寸。high、low 值为 -1 表示使用窗口的现有光标, 系统将根据当前屏幕方式的不同对扫描行界限加以调整。参见 SCPGCUR。



control\_function 是每次查询键盘后调用的一个键控制函数的地址。键控制函数接受 ED\_BUFFER 结构的地址作为 KB\_DATA 结构中的 pfunction\_data 成员。见 EDBUFFER 函数中有关 ED\_BUFFER 结构的说明。

对 control\_flags 可以使用下面的任选位(有些错自于 STPCVT)。对于不想使用的特性,请使用零位。

符号	值	意义
ED_LEFTJUST	0x8000	退出时左对齐。
ED_CENTERJUST	0x4000	退出时居中。
ED_RIGHTJUST	0x2000	退出时右对齐。
ED_ZERO_FILL	0x1000	退出时用零填充。
ED_BEEP_END	0x0800	到达文件末尾时鸣叫。
ED_AUTOSKIP	0x0400	在域尾时传送。
ED_INSERT_MODE	0x0200	最初处于插入状态。
ED_WORD_WRAP	0x0100	连续整字换行:不容许两个相邻的空白。
RCONTROL	0x0080	退出时废弃全部控制字符。
TOLOW	0x0040	退出时将大写字母转换为小写字母。
TOUP	0x0020	退出时将小写字母转换为大写字母。
NOQUOTE	0x0010	退出时不改变括在省略号(')或引号(')中的字符。
REDUCE	0x0008	退出时将所有空白缩减为一个空白。
RTWHITE	0x0004	退出时废弃所有尾随空白。
RLWHITE	0x0002	退出时废弃所有前导空白。
RWHITE	0x0001	废弃所有空白。

除非 pfinal 是一个 NIL 指针,否则 WNFIELD 函数返回最后一个按键。pfinal 是 KEY\_SEQUENCE 结构的地址,该结构在 bkeybrd.h 中定义如下:

typedef struct

```
{
    unsigned char character_code;
    unsigned char key_code;
} KEY_SEQUENCE;
```

ENINITKY、EDRETKEY、EDCHGKEY、DREMKEY 和 EDZAPKEY 维护已认可的按键命令表,影响它们的功效。

## 源程序(BWINDOW.H):

该宏定义在附录 C 的 BWINDOW.H 中。

## WNFORGET 从屏幕位置脱离窗口,但并不清除屏幕。

```
presult = wnforget (pwin);
```

presult 指向刚删除的 BWINDOW 结构,如果失败则指向 NIL。

pwin 指向要删除的 BWINDOW 结构。

函数从视频显示页删除一窗口,但不实际改变显示的字符。也就是说,先前的屏幕数据不被替换。

如果 pwin 指向一个无效的窗口或窗口不是当前显示的,则发生错误。

当从屏幕删除窗口时,使用 WNREMOVE 清除屏幕。用 WNDSTROY 放径整个窗口结构。

返回 presult 指向刚删除的 BWINDOW 结构,如果失败则指向 NIL。

\*pwin 修改几个域

b\_pactnode[] 有活动光标的窗口

b\_wncrr 值如下:

(不改变) 成功

WN\_BAD\_WIN \*pwin 无效

WN\_NOT\_SHOWN 不立即显示

WN_BAD_NODE	内部错误
WN_BAD_PAGE	内部错误
WN_BAD_DEV	内部错误

### 源程序(WNFORGET.C):

```
#include <conio.h>
#include <bwindow.h>
BWINDOW *wnforget (pwin)
BWINDOW *pwin;
{
    BWINDOW *presult;
    int mode, columns, act_page;
        /* Validate window pointer. */
    wnvalidw (pwin)
        /* Ensure it's showing. */
    if (pwin->where_shown.dev != SC_MONO &&
        pwin->where_shown.dev != SC_COLOR)
        wnreterr (WN_NOT_SHOWN);
        /* If window is current, mark no window current. */
        /* Also set the native window size to the entire */
        /* screen. */
    if (b_pcurwin == pwin)
    {
        b_pcurwin = NIL;
        (void) semode(&mode, &columns, &act_page);
        wnsetwin(0, 0, columns - 1, B_NATIVE_MAX_ROWS - 1);
    }
    if (b_pactnode[pwin->where_shown.dev][pwin->where_shown.page]
        ->pwin == pwin)
        /* Then this window is the one with active cursor. */
        b_pactnode[pwin->where_shown.dev][pwin->where_shown.page] = NIL;

        /* Now no window has an active cursor. (But don't */
        /* touch the visible cursor.) */
    pwin->internals.cur_frozen = 1;
        /* Remove from linked list. */
    if ((presult = wnpgrem (pwin)) != NIL)
    {
        pwin->where_shown.dev =
        pwin->where_prev.dev = ABSENT;
        pwin->internals.dirty =
        pwin->options.hidden = 0;
    }
        /* Correct internal flag in case it remains set. */
    pwin->internals.temp_hid = 0;
    return (presult);
}
```

**WNGETIMG** 读取屏幕长方形区域的图象。

```
presult = wngetimg (pimage, pwhere);
```

presult           指向读取的 IMAGE 结构, 如果失败则指向 NIL。

pimage            指向用人屏幕读取的数据的 IMAGE 结构。

pwhere            指向表示视频设备、显示页和屏幕位置的 WHERE 结构。

函数把显示屏幕长方形区域的内容读进由 IMAGE 结构指定的缓冲区。

函数也选择指定设备和显示页。

如果发生下述情况, 则出现错误:

(1) 如果 pimage 指向的 IMAGE 结构没有包含一个指向数据缓冲区的指针;

(2) 如果 pimage->dim 指定的尺寸和 pwhere->corner 指向的位置相对显示设备来说是不正确的。

返回	presult	指向读取的 IMAGE 结构, 失败时指向 NIL。
	b_device	视频设备
	b_curpage	当前显示页
	b_wncerr	值如下:
		(不改变)           成功
	WN_NULL_PTR	pimage->pdata 无效
	WN_BAD_DEV	无效的显示设备、显示页和尺寸。
	WN_ILL_DIM	内部错误

### 源程序(WNGETIMG.C):

```
#include <bvideo.h>
#include <bwindow.h>
#include <bmouse.h>
IMAGE *wngetimg (pimage, pwhere)
IMAGE *pimage;
const WHERE *pwhere;
{
    int old_npage;
    int num_read;
    if (pimage->pdata == NIL)
        wnreterr (WN_NULL_PTR);
        /* Validate and select device and page (also */
        /* validate dimensions). */
    if (wnseldev (pwhere, &pimage->dim, &old_npage))
        wnreterr (WN_BAD_DEV);
        /* Temporarily keep the mouse cursor out of the */
        /* window. */
    mohide(MO_HIDE);
        /* Read the screen image. */
    num_read = virdirect (pwhere->corner.row,
                          pwhere->corner.col,
                          pwhere->corner.row + pimage->dim.h - 1,
                          pwhere->corner.col + pimage->dim.w - 1,
                          (char *) pimage->pdata, CHAR_ATTR); /* Re-enable the
mouse cursor. */
    mohide(MO_SHOW);

    if (num_read != pimage->dim.h * pimage->dim.w)
        wnreterr (WN_ILL_DIM);
```

```

    return (pimage);
}

```

## WNGETOPT 读取窗口信息项或状态

```

#include <bwindow.h>
BWINDOW *wngetopt ( BWINDOW *pwin,
                    int item,
                    int *pvalue);

```

pwin          窗口地址。  
 item          标识待返回数据的代码号。  
 pvalue        接收数据值的变量的地址。  
 (返回)        窗口地址。若失败，则为 NIL。

WNGETOPT 读取有关窗口特性或状态信息的某一项。下面是可得到的信息:

项符号项值意义

WN_ROWS	14	1-32767	数据区高度
WN_COLS	15	1-32767	数据区宽度
WN_ATTR	20	0-255	缺省属性
WN_BORDER_TYPE	22		边界类型:见 WNDSPRAY。
WN_BORDER_CHAR	23	0-255	边界字符。
WN_BORDER_ATTR	24	0-255	边界属性。
WN_BORDER_TTATTR	32	0-255	上标题属性。
WN_BORDER_BTATTR	33	0-255	下标题属性。
WN_CUR_OFF	6	1	光标关闭。
	0		光标打开。
WN_CUR_HIG	7	0-13	光标高扫描行。
WN_CUR_LOW	8	0-13	光标低扫描行。
WN_ROW_REL	16	0-32767	相对于窗口数据区的光标位置。
WN_COL_REL	17	0-32767	
*WN_ROW_ABS	-18	0-499	屏幕上的绝对光标位置。
*WN_COL_ABS	-19	0-79	

窗口的显示位置:

WN\_DEVICE      1      窗口所在的设备(SC\_COLOR (1)、 SC\_MONO (0)。  
 如果未显示，则为 SC\_ABSENT (-2))。

*WN_PAGE	-2	0-7	窗口所在的屏幕页。
*WN_ROW_CORNER	-3	0-79	视口数据区左上角的位置。
*WN_COL_CORNER	-4	0-49	
*WN_HT_VIEW	-39	1-50	视口数据区的高度。
*WN_WID_VIEW	-40	1-50	视口数据区的宽度。

被视口和边界所覆盖的屏幕区域:

*WN_ROW_OVERALL	-27	0-49	被覆盖区域(包括边界)左上角的位置。
*WN_COL_OVERALL	-28	0-79	
*WN_HT_OVERALL	-29	1-80	被覆盖区域的高度。
*WN_WID_OVERALL	-30	1-50	被覆盖区域的宽度。

用户可控制的任选项:

WN_DELAYED	9	1	输出延迟，直到 WNUPDATE。
	0		输出可以立即执行。
WN_REMOVABLE	10	1	窗口可删除
	0		窗口不可删除。

WN_IS_CURRENT	25	1	是当前窗口。
	0		不是当前窗口。
*WN_ACTIVE_CUR	-26	1	该窗口光标是活动的。
	0		该窗口光标不是活动的。
WN_PREV_ALLOC	31	1	缓冲区已分配给原屏幕数据。
	0		缓冲区未分配给原屏幕数据。
WN_CUR_TRACK	34	1	在视口中显示的区域自动移动, 显示出光标(“自动跟踪”)。
	0		光标可能移出视口。
			自动跟踪过程中使用的间隙:
WN_CUR_T_MARG	37	0-49	在上方的行数。
WN_CUR_B_MARG	38	0-49	在下方的行数。
WN_CUR_L_MARG	35	0-79	在左边的列数。
WN_CUR_R_MARG	36	0-79	在右边的列数。
			多窗口的内部效果:
WN_FROZEN	11	1	窗口不能更新。
	0		窗口可以更新。
WN_DIRTY	12	1	输出被挂起。
	0		窗口保持最新。
WN_ANY_DATA_COVERED			
	13	1	某些窗口覆盖了该窗口数据区的某一部分。
	0		窗口数据区重新显现。
*WN_HIDDEN	-5	0	窗口是“隐藏的”: 已挂接到屏幕位置上但不可见(即使未被遮盖)。
		1	窗口不是隐藏的。

如果在上表中某项有星号(\*)标记, 则在窗口未显示时调用这一项便出现错误。如果所需项未知或 pwin 未指向一个合法窗口, 也出现一个错误。

## 源程序(WNGETOPT.C):

```
#include <bwindow.h>

/* Macro to build a case statement which does a simple */
/* lookup or calculation. */
#define easycase(item_num, data) \
    case (item_num): \
        *pvalue = (data); \
        presult = pwin; \
        break;

/* Macro to build a case statement whose success depends on */
/* the window being currently displayed (although perhaps */
/* hidden). (Currently a stub.) */
#define dispcase(item_num,data) easycase (item_num, data)

#define ABS_ROW_LOC (((pwin->cur_loc.row < pwin->data_origin.row) || \
    (pwin->cur_loc.row > (pwin->data_origin.row + \
    wnview_h(pwin) - 1)) \
    ? (-1) \
    : (pwin->cur_loc.row - pwin->data_origin.row)))

#define ABS_COL_LOC (((pwin->cur_loc.col < pwin->data_origin.col) || \
    (pwin->cur_loc.col > (pwin->data_origin.col + \
```

```

        wnview_w(pwin) - 1))
    ? (-1)
    : (pwin->cur_loc.col - pwin->data_origin.col)))

#define PNode (b_pactnode[pwin->where_shown.dev][pwin->where_shown.page])

BWINDOW *wngetopt (pwin, item, pvalue)
BWINDOW *pwin;
int item, *pvalue;
{
    BWINDOW *presult;
    wnvalidw (pwin)
    switch (item)
    {
        /* Dimensions of data area. */
        easycase (WN_ROWS, wndata_h(pwin))
        easycase (WN_COLS, wndata_w(pwin))
        /* Where the window is displayed. */
        easycase (WN_DEVICE, pwin->where_shown.dev)
        dispcase (WN_PAGE, pwin->where_shown.page)
        dispcase (WN_ROW_CORNER, pwin->where_shown.corner.row)
        dispcase (WN_COL_CORNER, pwin->where_shown.corner.col)
        /* Area covered by window & border. */
        dispcase (WN_ROW_OVERALL, pwin->where_prev.corner.row)
        dispcase (WN_COL_OVERALL, pwin->where_prev.corner.col)
        dispcase (WN_HT_OVERALL, wnbord_h(pwin))
        dispcase (WN_WID_OVERALL, wnbord_w(pwin))
        /* Whether it's hidden. */
        dispcase (WN_HIDDEN, pwin->options.hidden)
        /* Cursor on/off state & scan lines. */
        easycase (WN_CUR_OFF, pwin->options.cur_off)
        easycase (WN_CUR_HIGH, pwin->cur_type.high)
        easycase (WN_CUR_LOW, pwin->cur_type.low)
        /* User-controllable options. */
        easycase (WN_DELAYED, pwin->options.delayed)
        easycase (WN_REMOVABLE, pwin->options.removable)
        /* Default attributes. */
        easycase (WN_ATTR, pwin->attr)

        /* Cursor auto-tracking on/off. */
        easycase (WN_CUR_TRACK, pwin->options.cur_track)

        /* Auto-track margins. */
        easycase (WN_CUR_L_MARG, pwin->cur_left_marg)
        easycase (WN_CUR_R_MARG, pwin->cur_right_marg)
        easycase (WN_CUR_T_MARG, pwin->cur_top_marg)
        easycase (WN_CUR_B_MARG, pwin->cur_bottom_marg)

        /* Internal effects of displaying windows. */
    }
}

```

```

    easycase (WN_FROZEN,      pwin->internals.frozen)
    easycase (WN_DIRTY,      pwin->internals.dirty)
    easycase (WN_ANY_DATA_COVERED, pwin->internals.any_data_covered)
        /* Cursor location (relative to data area). */
    easycase (WN_ROW_REL,     pwin->cur_loc.row)
    easycase (WN_COL_REL,     pwin->cur_loc.col)
        /* Absolute cursor location on screen. */
    dispcase (WN_ROW_ABS,     ABS_ROW_LOC)
    dispcase (WN_COL_ABS,     ABS_COL_LOC)
        /* Border. */
    easycase (WN_BOR_TYPE,    pwin->bord.type)
    easycase (WN_BOR_CHAR,    (int) pwin->bord.ch)
    easycase (WN_BOR_ATTR,    pwin->bord.attr)
    easycase (WN_BOR_TTATTR,  pwin->bord.ttattr)
    easycase (WN_BOR_BTATTR,  pwin->bord.btattr)

        /* Whether this is the current window. */
    easycase (WN_IS_CURRENT, (pwin == b_pcurwin))

        /* Whether this window (among all the windows
        /* displayed on its page) is the one whose cursor
        /* is active. */
    dispcase (WN_ACTIVE_CUR,
        ((pwin->pnode == PNode) && (pwin == PNode->pwin)))

        /* Whether the window previous screen buffer has
        /* already been allocated. */
    easycase (WN_PREV_ALLOC, (pwin->prev.pdata == NIL) ? 0 : 1)
        /* Unknown item requested. */
    default:
        wnretterr (WN_BAD_OPT);
}
return (presult);
}

```

## WNHIDE 删除窗口但仍与视频显示设备和显示页联连。

presult = wnhide(pwin);

presult 指向刚隐藏的 BWINDOW 结构，如果失败则指向 NIL。

pwin 指向要删除的 BWINDOW 结构。

函数使当前显示的窗口不可见并恢复先前的屏幕内容。窗口的位置和边界不变，其后可通过 WNUNHIDE 重新显示。

如果 pwin 没有指向一有效的窗口结构，或者窗口被指定为不可删除，或者窗口当前并不显示，则都会产生错误。

使用 WNREMOVE 删除并脱离窗口位置和边界。使用 WNFORGET 永久性地从窗口位置和边界脱离。

返回 presult 指向刚隐藏的 BWINDOW 结构，如果失败则指向 NIL。

\*pwin 修改了几个域。

b\_pactnode[1] 有活动光标的窗口节点。

b\_wnerr 可能的值如下：

(不改变)	成功
WN_BAD_WIN	*pwin 无效
WN_NOT_SHOWN	不是当前显示的。
WN_CANT_HIDE	不可删除
WN_BAD_NODE	内部错误
WN_BAD_DEV	内部错误
WN_ILL_DIM	内部错误
WN_NULL_PTR	内部错误

### 源程序(WNHIDE.C):

```
#include <conio.h>
#include <bwindow.h>

/* Internal functions defined below: */

/* Hide covering windows. */
static WIN_NODE *undisp(WIN_NODE *,LOC *,DIM *);

/* Mark lower windows uncovered. */
static WIN_NODE *uncovr(WIN_NODE *,LOC *,DIM *);

/* Check for covering windows. */
static int upcovr(WIN_NODE *,LOC *,DIM *);

/* Redisplay covering windows. */
static WIN_NODE *redisp(WIN_NODE *);
BWINDOW *wnhide(pwin)
BWINDOW *pwin;
{
    int old_dev, old_page, old_npage;
    int mode, columns, act_page;
    BWINDOW *presult;
    int row, col, high, low;
    WIN_NODE *pnode;
    wnvalidw(pwin)
        /* Quit if already invisible. */
    if (pwin->options.hidden)
        return(pwin);
        /* Error if not removable. */
    if (!pwin->options.removable)
        wnreterr(WN_CANT_HIDE);
    old_page = b_curpage;
    old_dev = scmode(&mode, &columns, &act_page);

    /* Validate and select device and page. */
    if (wnseldev(&pwin->where_prev, &pwin->prev.dim, &old_npage))
        wnreterr(WN_NOT_SHOWN);

    presult = pwin;
```



```

        /* Temporarily remove higher windows covering this */
        /* window. */
if (undisp(pwin->nnode,
        &pwin->where_prev.corner,
        &pwin->prev.dim) == NIL)
    presult = NIL;

        /* Correct internal flag: this window was un- */
        /* displayed by UNDISP but NOT temporarily. */
pwin->internals.temp_hid = 0;
pwin->options.hidden = 1;
        /* Mark lower windows if uncovered. */
if (uncovr(pwin->nnode,
        &pwin->where_prev.corner,
        &pwin->prev.dim) == NIL)
    presult = NIL;
        /* If this window has the active cursor... */
if (b_pactnode[pwin->where_shown.dev][pwin->where_shown.page]
    ->pwin == pwin)
{
    /* Set so no window has an active cursor; retrieve */
    /* cursor size; turn cursor off; set native text */
    /* window to entire screen. */
    b_pactnode[pwin->where_shown.dev][pwin->where_shown.page] = NIL;
    sccurst(&row, &col, &high, &low);
    scpgcur(1, high, low, CUR_NO_ADJUST);
}

        /* If this is the current window, and the native */
        /* text window matches it, then reset the native */
        /* text window. */
if ((b_pcurwin == pwin) && wnchkdm(pwin))
{
    wnsetwin(0, 0, columns - 1, B_NATIVE_MAX_ROWS - 1);
}

pwin->internals.cur_frozen = 1;

        /* Redisplay higher windows. (also mark lower */
        /* windows if covered). */
if (redisp(pwin->nnode) == NIL)
    presult = NIL;

        /* Since we may have just exposed some windows, */
        /* update each lower window on this display page, */
        /* except those that are delayed. */
for (pnode = b_wnlist[pwin->where_shown.dev][pwin->where_shown.page];
    pnode != NIL;
    pnode = pnode->below)

    if (!pnode->pwin->options.delayed)

```

```

        if (wnnupblk(pnode->pwin, 0, 0,
                    wndata_h(pnode->pwin) - 1,
                    wndata_w(pnode->pwin) - 1, WN_UPDATE) == NIL)
        {
            presult = NIL;
            break;
        }

        /* Restore current page on new device; restore old */
        /* device; restore current page on old device. */
        scpage (old_npage);
        sechgdev (old_dev);
        scpage (old_page);

        return(presult);
    }
}

```

**undisp**      暂时删除(不显示)屏幕上由一长方形区域遮盖的窗口。

presult = \*undisp(pnode, pcorner, pdim);

presult          pnode 的复本, 如果失败则为 NIL。

pnode            指向可能删除的最低层的窗口。

pcorner          指向包含区域左上角的结构。

pdim            指向包含区域飞寸的结构。

函数查找所有覆盖指定的长方形区域的窗口, 并设法暂时删除窗口。这些被删除的窗口皆被标志。隐藏窗口不受影响。

不可删除窗口不实际删除。

返回            presult          指向刚隐藏的 BWINDOW 结构, 如果失败则指向 NIL。

b\_wncrr

可能值如下:

(不改变)

成功

WN\_BAD\_NODE      坏的 pnode 或内部错误。

WN\_BAD\_DEV        内部错误

WN\_NULL\_PTR       内部错误

WN\_ILL\_DIM        内部错误

```
static WIN_NODE *undisp(pnode, pcorner, pdim)
```

```
WIN_NODE          *pnode;
```

```
LOC               *pcorner;
```

```
DIM               *pdim;
```

```
{
    WIN_NODE *presult;
    BWINDOW *pwin;
```

```
wnvalidn(pnode)
```

```
presult = pnode;
```

```
pwin      = pnode->pwin;
```

```
/* First un-display higher windows that cover this */
```

```
/* rectangle. */
```

```
if (pnode->above != NIL)
```

```
    if (undisp(pnode->above,
```

```

        pcorner,
        pdim) == NIL)
presult = NIL;

/* If this window covers the rectangle, undisplay. */
if ((!pwin->options.hidden) &&
    (!pwin->internals.temp_hid) &&
    wnoverlap(pcorner, pdim,
              &pwin->where_prev.corner, &pwin->prev.dim))
{
    /* First un-display covering windows. */
    if (pnode->above != NIL)
        if (undisp(pnode->above,
                  &pwin->where_prev.corner,
                  &pwin->prev.dim) == NIL)
            presult = NIL;
    /* Replace previous screen data. */
    if (pwin->options.removable)
        if (wnresprv(pwin) == NIL)
            presult = NIL;
    pwin->internals.temp_hid = 1; /* Temporarily hidden. */
    pwin->internals.dirty = 0; /* No longer out-of-date. */
}
return(presult);
}

```

**uncovr** – 标志非覆盖的窗口，这些窗口在给定的窗口下并有被一长方形的区域而非其它窗口覆盖。

presult = \*uncovr(pnode,pcorner,pdim);

presult      pnode 的副本，如果失败则为 NIL。

pnode      指向可能要标志的最上层窗口的节点。

pcorner      指向含区域左上角的结构。

pdim      指向区域尺寸的结构。

函数标志那些在给定的窗口以下的并被一指定的区域而非其它窗口覆盖的窗口。

返回      presult      指向刚隐藏 BWINDOW 的结构，如果失败则为 NIL。

b\_werr      值如下：

(不改变)

成功

WN\_BAD\_NODE      坏的 pnode 功内部错误。

static WIN\_NODE \*uncovr(pnode, pcorner, pdim)

WIN\_NODE      \*pnode;

LOC      \*pcorner;

DIM      \*pdim;

```

{
    WIN_NODE *presult;
    BWINDOW *pwin;
    wnvalidn(pnode)
    pwin = pnode->pwin;

```

/\* If rectangle overlaps this window, \*/

if (wnoverlap(pcorner,

```

        pdim,
        &pwin->where_shown.corner,
        &pwin->view_size))

        /* (unless higher windows overlap this one)          */
if (pnode->above == NIL ||
    !upcovr(pnode->above,
            &pwin->where_shown.corner,
            &pwin->view_size))

        /* mark this window "uncovered".                      */
pwin->internals.frozen =
pwin->internals.any_data_covered = 0;

presult = pnode;
if (pnode->below != NIL) /* Continue with lower windows.      */
    if (uncovr(pnode->below, pcorner, pdim) == NIL)
        presult = NIL;

return(presult);
}

```

### **upcovr** 检查高层窗口是否覆盖长方形区域。

result = \*upcovr(pnode, pcorner, pdim);

presult pnode 的复本，如果失败则为 NIL。

pnode 指向窗口节点。

pcorner 指向包含区域左上角的结构。

pdim 指向包含区域尺寸的结构。

给出一窗口和由\*pwin 和\*pcorner 定义的区域，该函数指向该窗口或高层窗口是否覆盖该区域。

返回 presult 指向刚隐藏的 BWINDOW 结构，如果失败则 指向 NIL。

b\_wncrr 值如下：

(不改变) 成功

WN\_BAD\_NODE 坏的 pnode 或内部错误。

static int upcovr(pnode, pcorner, pdim)

WIN\_NODE \*pnode;

LOC \*pcorner;

DIM \*pdim;

{

int result;

if (wnvalnod(pnode) == NIL)/\* Quit if invalid node. \*/

wnreterz(WN\_BAD\_NODE);

if (!pnode->pwin->options.hidden &&

wnovrlap(pcorner, /\* Check for rectangle overlap. \*/

pdim,

&(pnode->pwin->where\_prev.corner),

&(pnode->pwin->prev.dim)))

result = 1; /\* Overlap. \*/

/\* No overlap, so look higher. \*/

else

```

    if (pnode->above != NIL)
        result = upcovr(pnode->above, pcorner, pdim);
    else
        /* No higher windows to check, so */
        result = 0; /* answer is "no". */
return(result);
}

```

**redisp** - 重新显示指定窗口以上的被暂时隐藏删除的窗口。

presult = redisp(pnode, pcorner, pdim);

presult          pnode 的复本，如果失败则为 NIL。

pnode          指向要重新显示的最低层窗口的节点。

函数重新显示被暂时隐藏窗口以上的窗口。

隐藏的窗口不受影响。

如果被标志为暂时删除则不可删除窗口被重新显示。

返回          presult          指向刚隐藏的 BWINDOW 结构，如果失败则指向 NIL。

              b\_wncrr          值如下：

(不改变)          成功

WN\_BAD\_NODE          坏的 pnode 内部错误

WN\_BAD\_DEV          内部错误

WN\_NULL\_PTR          内部错误

WN\_ILL\_DIM          内部错误

```
static WIN_NODE *redisp(pnode)
```

```
WIN_NODE          *pnode;
```

```
{
    BWINDOW *pwin;
```

```
    wnvalidn(pnode)
```

```
    pwin = pnode->pwin;
```

```
        /* Skip if hidden by user or if not temp. hidden. */
```

```
if ((!pwin->options.hidden) &&
```

```
    (pwin->internals.temp_hid))
```

```
{
```

```
        /* Read previous screen contents. */
```

```
if (pwin->options.removable)
```

```
    if (wngetimg(&pwin->prev,
```

```
        &pwin->where_prev) == NIL)
```

```
        return(NIL);
```

```
        /* Write border. */
```

```
wnputbor(&pwin->view_size,
```

```
    &pwin->bord,
```

```
    &pwin->where_shown);
```

```
        /* Write data. */
```

```
if (wnpimblk(pwin, 0, 0,
```

```
    wndata_h(pwin) - 1, wndata_w(pwin) - 1) == NIL)
```

```
    return(NIL);
```

```
pwin->internals.dirty = /* Output is now up-to-date.*/
```

```
pwin->internals.temp_hid = 0; /* No longer hidden. */
```

```
if (pnode->below != NIL)/* Mark lower windows as covered. */
```

```

        if (wncover(pnode->below,
                    &pwin->where_shown.corner,
                    &pwin->prev.dim) == NIL)
            return(NIL);
    }

    if (pnode->above != NIL) /* Continue with higher windows. */
        if (redisp(pnode->above) == NIL)
            return(NIL);

    return(pnode);
}

```

## WNHORIZ 水平滚动当前窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnhoriz( int num_cols,
                  int fore, in back,
                  int dir);
```

num\_cols 滚动的列数。值 0 清除窗口。

fore 新空列的前景属性。(-1 指定窗口的缺省前景颜色。)

back 新空列的背景属性。(-1 指定窗口的缺省背景颜色。)

dir 滚动方向(SCR\_LEFT (1) 或 SCR\_RIGHT (0))。

(返回) 滚动后的 BWINDOW 结构的地址。若失败，返回 NIL。

WNHORIZ 在当前窗口中将字符列(连同其属性)向左或向右滚动，空列由空格和指定属性填充。

WNHORIZ 修改窗口的整个数据区但不移动窗口的光标。用 WNORIGIN 可以在视口中显示窗口数据区的各个部分。

如果窗口不是当前窗口，这时出现错误。

## 源程序(WNHORIZ.C):

```

#include <bwindow.h>
#define H (wndata_h(b_pcurwin))
#define W (wndata_w(b_pcurwin))
BWINDOW *wnhoriz (num_cols, fore, back, dir)
int num_cols, fore, back, dir;
{
    /* Validate window data structures. */
    wnvalidw (b_pcurwin)
    /* Force num_cols to be 0 (clear window) if not in
     * range. */
    if (utrange (num_cols, 1, W))
        num_cols = 0;

    if (dir == SCR_LEFT)
        return (wnscrblk (b_pcurwin,
                        0, 0,
                        H - 1, W - 1,
                        fore, back, WNSCR_LEFT, num_cols, WN_UPDATE));

    return (wnscrblk (b_pcurwin,

```

```

0,      0,
H - 1, W - 1,
fore, back, WN_SCR_RIGHT, num_cols, WN_UPDATE));
}

```

## WNINITEV 为 WNREAD 安装缺省的窗口事件

```
#include <bwindow.h>
```

```
int wninitv(BWINDOW *pwin);
```

pwin 待初始化的 BWINDOW 结构的地址。

(返回) 错误代码, 可能值包括:

WN\_NO\_ERROR (0) 成功。

WN\_NO\_MEMORY (1) 无足够动态内存来分配给所有的表成员。

WNINITEV 为一个窗口安装 WNREAD 认可的缺省响应。

WNREAD 自动执行 WNINITEV, 因此用户不必调用 WINITEV, 除非他想修改缺省响应表。利用 WNCHGEVN、WNREMEVN 和 WNZAPEVN 可以修改这个表。

WNINITEV 不修改全局窗口错误代码 b\_wnerr。

## 源程序(WNINITEV.C):

```
#include <butil.h>
```

```
#include <bkeys.h>
```

```
#include <bwindow.h>
```

```
#include <bmouse.h>
```

```
/* Default window key records. They are placed in the */
```

```
/* linked list of window events at initialization. */
```

```
WN_KEY b_winkeys[] =
```

```

{
    {WN_SCROLL_LEFT, {0x00, 0x4B}}, /* Move left - pad 4. */
    {WN_SCROLL_LEFT, {0xE0, 0x4B}}, /* Move left - left arrow. */
    {WN_SCROLL_RIGHT, {0x00, 0x4D}}, /* Move right - pad 6. */
    {WN_SCROLL_RIGHT, {0xE0, 0x4D}}, /* Move right - right arrow. */
    {WN_SCROLL_UP, {0x00, 0x48}}, /* Move up - pad 6. */
    {WN_SCROLL_UP, {0x38, 0x48}}, /* Move up - up arrow. */
    {WN_SCROLL_DOWN, {0x00, 0x50}}, /* Move down - pad 6. */
    {WN_SCROLL_DOWN, {0x32, 0x50}}, /* Move down - down arrow. */
    {WN_PAGE_LEFT, {0x00, 0x73}}, /* Page left - C/pad 4. */
    {WN_PAGE_LEFT, {0xE0, 0x73}}, /* Page left - C/left arrow. */
    {WN_PAGE_RIGHT, {0x00, 0x74}}, /* Page right - C/pad 6. */
    {WN_PAGE_RIGHT, {0xE0, 0x74}}, /* Page right - C/right arrow. */
    {WN_PAGE_UP, {0x00, 0x49}}, /* Page up - pad 9. */
    {WN_PAGE_UP, {0xE0, 0x49}}, /* Page up - Page Up. */
    {WN_PAGE_DOWN, {0x00, 0x51}}, /* Page down - pad 3. */
    {WN_PAGE_DOWN, {0xE0, 0x51}}, /* Page down - Page Down. */
    {WN_LEFT_EDGE, {0x00, 0x77}}, /* Go to left edge - C/pad 7. */
    {WN_LEFT_EDGE, {0xE0, 0x77}}, /* Go to left edge - C/Home. */
    {WN_RIGHT_EDGE, {0x00, 0x75}}, /* Go to right edge - C/pad 1. */
    {WN_RIGHT_EDGE, {0xE0, 0x75}}, /* Go to right edge - C/End. */
    {WN_TOP_EDGE, {0x00, 0x47}}, /* Go to top edge - pad 7. */
    {WN_TOP_EDGE, {0xE0, 0x47}}, /* Go to top edge - Home. */
    {WN_BOTTOM_EDGE, {0x00, 0x4F}}, /* Go to bottom edge - pad 1. */
}

```

```

{WN_BOTTOM_EDGE, {0xE0, 0x4F}}, /* Go to bottom edge - End. */
{WN_ABORT, {0x1B, 0x01}}, /* Abort read - Esc. */
{WN_TRANSMIT, {0x0D, 0x1C}}, /* Finish read - Enter. */
{WN_TRANSMIT, {0x0D, 0xE0}}, /* Finish read - pad Enter. */
{WN_INVALID_ACTION, {KBNDEF, KBNDEF}} /* End of window key entries.*/
};

```

```
};
```

```
WN_MOUSE b_winmouse[] =
```

```
{
```

```

{WN_SCROLL_LEFT,
 {MO_LEFT | MO_CLICK, /* Left click/left arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_LEFT_ARROW}},
{WN_SCROLL_LEFT,
 {MO_LEFT | MO_HOLD, /* Left hold/left arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_LEFT_ARROW}},
{WN_PAGE_LEFT,
 {MO_RIGHT | MO_CLICK, /* Right click/left arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_LEFT_ARROW}},
{WN_LEFT_EDGE,
 {MO_LEFT | MO_DCLICK, /* Left double click/left arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_LEFT_ARROW}},

```

```

{WN_SCROLL_RIGHT,
 {MO_LEFT | MO_CLICK, /* Left click/right arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_RIGHT_ARROW}},
{WN_SCROLL_RIGHT,
 {MO_LEFT | MO_HOLD, /* Left hold/right arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_RIGHT_ARROW}},
{WN_PAGE_RIGHT,
 {MO_RIGHT | MO_CLICK, /* Right click/right arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_RIGHT_ARROW}},
{WN_RIGHT_EDGE,
 {MO_LEFT | MO_DCLICK, /* Left double click/right arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_RIGHT_ARROW}},

```

```

{WN_SCROLL_UP,
 {MO_LEFT | MO_CLICK, /* Left click/up arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_UP_ARROW}},
{WN_SCROLL_UP,
 {MO_LEFT | MO_HOLD, /* Left hold/up arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_UP_ARROW}},
{WN_PAGE_UP,
 {MO_RIGHT | MO_CLICK, /* Right click/up arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_UP_ARROW}},
{WN_TOP_EDGE,
 {MO_LEFT | MO_DCLICK, /* Left double click/up arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_UP_ARROW}},

```

```

{WN_SCROLL_DOWN,
 {MO_LEFT | MO_CLICK, /* Left click/down arrow. */

```



```

    MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_DOWN_ARROW}},
{WN_SCROLL_DOWN,
 {MO_LEFT | MO_HOLD,          /* Left hold/down arrow.          */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_DOWN_ARROW}},
{WN_PAGE_DOWN,
 {MO_RIGHT | MO_CLICK,        /* Right click/down arrow.    */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_DOWN_ARROW}},
{WN_BOTTOM_EDGE,
 {MO_LEFT | MO_DCLICK,        /* Left double click/down arrow. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_DOWN_ARROW}},
{WN_TRANSMIT,
 {MO_LEFT | MO_DCLICK,        /* Left double click/inside window. */
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_IN_WINDOW}},
{WN_ABORT,
 {MO_LEFT | MO_DCLICK,        /* Left double click/outside window.*/
  MO_RSHIFT|MO_LSHIFT|MO_CSHIFT|MO_ASHIFT, WN_OUT_WINDOW}},

{WN_INVALID_ACTION, {0, 0, 0}} /* End of window mouse entries. */
};

```

```

int wninitv(pwin)
BWINDOW *pwin;
{
    int i;
    WN_EVENT_LIST *new_event_node;
    if (pwin->pevent_list != NIL)
        wnzapevn(&pwin->pevent_list);
        /* Initialize the list of default window events.          */
        /* for each key in the default key table, a new            */
        /* record is allocated, initialized and placed at          */
        /* the head of the event list.                               */
    i = 0;
    while (b_winkeys[i].action != WN_INVALID_ACTION)
    {
        new_event_node = malloc(sizeof(WN_EVENT_LIST));
        if (new_event_node == NIL)
            return(WN_NO_MEMORY);
        new_event_node->signature = BEVENT_SIGN;
        new_event_node->win_event.event_type = WN_KB_EVENT;
        new_event_node->win_event.event.keystroke =
            b_winkeys[i].key_sequence;
        new_event_node->win_event.pdata =
            malloc(sizeof(WN_ACTION));
        new_event_node->win_event.data_size = sizeof(WN_ACTION);
        if (new_event_node->win_event.pdata == NIL)
            return(WN_NO_MEMORY);
        *((WN_ACTION *) new_event_node->win_event.pdata) =
            b_winkeys[i].action;
        if (pwin->pevent_list != NIL)
        {

```

```

    pwin->pevent_list->pprev = new_event_node;
    new_event_node->pnext     = pwin->pevent_list;
    new_event_node->pprev     = NIL;
    pwin->pevent_list        = new_event_node;
}
else
{
    new_event_node->pnext = NIL;
    new_event_node->pprev = NIL;
    pwin->pevent_list    = new_event_node;
}
i++;
}
i = 0;
while (b_winmouse[i].action != WN_INVALID_ACTION)
{
    new_event_node = malloc(sizeof(WN_EVENT_LIST));
    if (new_event_node == NIL)
        return(WN_NO_MEMORY);
    new_event_node->signature = BEVENT_SIGN;
    new_event_node->win_event.event_type = WN_MOUSE_EVENT;
    new_event_node->win_event.event.mouse = b_winmouse[i].mouse;
    new_event_node->win_event.pdata = malloc(sizeof(WN_ACTION));
    new_event_node->win_event.data_size = sizeof(WN_ACTION);
    if (new_event_node->win_event.pdata == NIL)
        return(WN_NO_MEMORY);
    *((WN_ACTION *) new_event_node->win_event.pdata) =
        b_winmouse[i].action;
    if (pwin->pevent_list != NIL)
    {
        pwin->pevent_list->pprev = new_event_node;
        new_event_node->pnext     = pwin->pevent_list;
        new_event_node->pprev     = NIL;
        pwin->pevent_list        = new_event_node;
    }
    else
    {
        new_event_node->pnext = NIL;
        new_event_node->pprev = NIL;
        pwin->pevent_list    = new_event_node;
    }
    i++;
}
return(WN_NO_ERROR);
}

```

## WNCMOVE 移动窗口的光标。

wncmove(pwin, row, col);

pwin 指向光标被移动的窗口。

row, col 相对于窗口数据区的窗口光标的新坐标。

pwin 指向的窗口的光标被移动。如果预处理器符号 B\_NATIVE\_WINDOW 等于 0, 编译器的 native 文本窗口不被支持, 并用 secureset() 设置光标位置。如果 B\_NATIVE\_WINDOW 等于 1, 则支持编译器的 native 窗口。此时如果 pwin 是当前选择的窗口, 编译器的设置光标位置的库子程将用来移动光标; 否则使用 secureset()。

光标位置的坐标相对于窗口的左上角(0,0)。函数假设调用子程序已验证了 pwin 的正确性并确保正确物理地移动光标。不进行边界检查, 也不影响逻辑光标位置。

返回 无

### 源程序(WNNATVWN.C):

```
#include <conio.h>
#include <bwindow.h>
#include <bgenwin.h>
void wncmove(pwin, row, col)
BWINDOW *pwin;
int row;
int col;
{
    #if B_NATIVE_WINDOWS
        if (pwin == b_pcurwin)
            gotoxy((col - pwin->data_origin.col) + 1,
                (row - pwin->data_origin.row) + 1);
        else
    #endif
        secureset((row - pwin->data_origin.row) +
            pwin->where_shown.corner.row,
            (col - pwin->data_origin.col) +
            pwin->where_shown.corner.col);
}
```

### WNSETWIN 设置编译器 native 文本窗口的尺寸。

wnsetwin(left, top, bottom, right);  
 left, top, 编译器 native 文本窗口的左、上坐标。  
 bottom, right 编译器 native 文本窗口的右、下坐标。

如果预处理器符号 B\_NATIVE\_WINDOWS 等于 1, 则支持编译器的 native 文本窗口, 并且 native 文本窗口的尺寸被设置为指定的坐标。如果 B\_NATIVE\_WINDOWS 等于 0, 该子程序没有作用。

屏幕的左上角被设为 0。不进行给定值的范围检查。

返回 无

```
void wnsetwin(left, top, bottom, right)
int left;
int top;
int bottom;
int right;
{
    #if B_NATIVE_WINDOWS
        window(left + 1, top + 1, bottom + 1, right + 1);
    #endif
    /* Temporarily disable "Parameter never used" warning*/
    #pragma warn -par
```

```

}
#pragma warn .par

```

## WNCHKDM 检查编译器的 native 文本窗口的尺寸和大小。

match = wnchkdm(pwin);

match 表明该 native 文本窗口是否匹配视区的代码。

pwin 窗口的视区大小与编译器的 native 文本窗口相比较。

如果预处理器符号 B\_NATIVE\_WINDOWS 等于 1, 则支持编译器的 native 文本窗口, 并且 native 文本窗口的尺寸和位置与指定窗口的视区相比较。如果匹配则返回非零值; 否则返回 0。如果 B\_NATIVE\_WINDOWS 等于 0, 不进行检查, 并返回非零值。函数只在指定的窗口是显示的情况下, 才有意义。

返回 match 如果 native 文本窗口不匹配视区则为 0; 如果匹配或者 B\_NATIVE\_WINDOWS 在编译时等于 0(此时不支持 native 文本窗口)则为非零。返回值只有在指定窗口是显示的才有效。

int wnchkdm(pwin)

BWINDOW \*pwin;

```

{
#ifdef B_NATIVE_WINDOWS

    struct text_info textinfo;

    gettextinfo(&textinfo);
    return((((textinfo.wintop - 1) == pwin->where_shown.corner.row) &&
            ((textinfo.winleft - 1) == pwin->where_shown.corner.col) &&
            ((textinfo.winbottom) == (pwin->where_shown.corner.row +
                                     wnview_h(pwin))) &&
            ((textinfo.winright) == (pwin->where_shown.corner.col +
                                     wnview_w(pwin)))));

#else

    return(1);
#endif

    /* Temporarily disable "Parameter never used" warning*/
#pragma warn -par
}
#pragma warn .par

```

## WNGETATR 得到当前 native 文本窗口的属性。

wngetatr(pwin);

pwin 窗口的缺省数据区的属性设置成 native 文本窗口缺省属性。

如果预处理器符号 B\_NATIVE\_WINDOWS 等于 1, 则支持编译器的 native 文本窗口, 并且窗口的缺省数据区的属性设置成 native 文本窗口缺省属性。如果匹配则返回非零值; 否则返回 0。如果 B\_NATIVE\_WINDOWS 等于 0, 不修改窗口的缺省数据区的属性

返回 无

void wngetatr(pwin)

BWINDOW \*pwin;

```

{
#ifdef B_NATIVE_WINDOWS
    struct text_info textinfo;
    gettextinfo(&textinfo);
    pwin->attr = textinfo.attribute;

```

```
#endif
/* Temporarily disable "Parameter never used" warning*/
#pragma warn -par
}
#pragma warn .par
```

## WNSETATR – 设置当前 native 文本窗口的属性。

wnsetatr(attribute);

attribute 文本窗口的新属性。

如果预处理器的字符 B\_NATIVE\_WINDOWS 等于 1，则支持文本窗口，属性用作文本窗口的新属性。如果 B\_NATIVE\_WINDOWS 等于 0，不修改窗口的缺省属性。

返回 无

void wnsetatr(attribute)

int attribute;

```
{
#if B_NATIVE_WINDOWS
```

textattr(attribute);

```
#endif
/* Temporarily disable "Parameter never used" warning*/
#pragma warn -par
}
#pragma warn .par
```

## WNNEEDUP 标志一窗口为无用，如果可能进行更新。

presult = wnneedup (pwin);

presult 指向更新的 BWINDOW 结构，如果失败则指向 NIL。

pwin 指向欲更新的结构。

如果有等待，并且窗口是不延迟、不冻结的，则函数向指定的窗口写 I/请求。不进行错误检查。

返回 presult 指向刚隐藏的 BWINDOW 结构，如果失败则指向 NIL。

\*pwin 修改了几个域。

b\_wncrr 值如下：

(不改变)	成功
WN_BAD_WIN	*pwin 无效
WN_BAD_DEV	内部错误
WN_ILL_DIM	内部错误
WN_NULL_PTR	内部错误

源程序(WNNEEDUP.C):

```
#include <bwindow.h>
```

```
BWINDOW *wnneedup (pwin)
```

```
BWINDOW *pwin;
```

```
{
    pwin->internals.dirty = 1;

    return (wnputimg(pwin));
}
```

**WNNUPBLK** 标志一窗口为无用, 如果可能更新一部分。

presult = wnnupblk(pwin, r1, c1, r2, c2, option);

presult 指向更新的 BWINDOW 结构。如果失败则指向 NIL。

pwin 指向要更新的 BWINDOW 结构。

r1, c1 欲更新的块的左上角的行和列。

r2, c2 欲更新的块的右下角的行和列。

option WN\_UPDATE 为更新窗口(在屏幕上); WN\_NO\_UPDATE 在内存在更新。

如果有等待, 且窗口不是延迟、冻结的, 则函数向指定的窗口在(r1,c1)-(r2,c2)的长方形区域内写与任何 I/O 请求。

返回 presult 指向刚更新的 BWINDOW 结构, 如果失败则指向 NIL。

\*pwin 修改了几个域。

b\_wncrr 可能的值如下:

(不改变)	成功
WN_BAD_WIN	*pwin 无效
WN_NOT_SHOWN	不立即显示
WN_ILL_DIM	坏的坐标
WN_BAD_DEV	内部错误
WN_NULL_PTR	内部错误

源程序(WNNUPBLK.C):

```
#include <bwindow.h>
```

```
BWINDOW *wnnupblk(pwin, r1, c1, r2, c2, option)
```

```
BWINDOW *pwin;
```

```
int r1, c1, r2, c2;
```

```
int option;
```

```
{
```

```
    int old_dev, old_page, old_npage;
```

```
    int mode, columns, act_page;
```

```
    BWINDOW *presult;
```

```
        /* Validate window data structures. */
```

```
    wnvalidw(pwin)
```

```
    if ((pwin->where_shown.dev != SC_MONO) &&
```

```
        (pwin->where_shown.dev != SC_COLOR))
```

```
        /* Not shown anywhere. */
```

```
        pwin->where_shown.dev = ABSENT;
```

```
        /* Quit if invisible, delayed, frozen, covered, not */
```

```
        /* shown, or if option is WN_NO_UPDATE. */
```

```
    if (pwin->options.hidden ||
```

```
        pwin->options.delayed ||
```

```
        pwin->internals.frozen ||
```

```
        pwin->internals.any_data_covered ||
```

```
        (pwin->where_shown.dev == ABSENT) ||
```

```
        (option & WN_NO_UPDATE))
```

```
    {
```

```
        pwin->internals.dirty = 1;
```

```
        return(pwin);
```

```
    }
```

```
        /* Now apply cursor autotracking. */
```

```
    if (wncurtrk(pwin, pwin->cur_loc.row, pwin->cur_loc.col) != pwin)
```

```

return(NIL);

/* Update the physical cursor. */
if (!pwin->internals.cur_frozen &&
    (b_pactnode[pwin->where_shown.dev][pwin->where_shown.page] ==
     pwin->pnode))
{
    wncurset(pwin);
}

/* If this window was already dirty or its origin
   /* has changed, do a full update. */
if (pwin->internals.dirty != 0)
{
    r1 = c1 = 0;
    r2 = wndata_h(pwin) - 1;
    c2 = wndata_w(pwin) - 1;
}

/* Otherwise update just a block. */
pwin->internals.dirty = 1;

/* Note former device, current page on former device*/
old_dev = scmode(&mode, &columns, &act_page);
old_page = b_curpage;

/* Validate and select device and page. */
if (wnseldev(&pwin->where_shown, &pwin->view_size, &old_npage))
    wnreterr(WN_NOT_SHOWN);

/* Put block of image onto physical screen. */
presult = pwin;
if (wnpimblk(pwin, r1, c1, r2, c2) == NIL)
    presult = NIL;
else
    pwin->internals.dirty = 0;

/* Restore current page on new device; restore old
   /* device; restore current page on old device. */
scpage(old_npage);
scchgdev(old_dev);
scpage(old_page);

return(presult);
}

```

## WNORIGIN 在视口中称动窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnorigin ( BWINDOW *pwin,
                    int row, int col,
                    int option);
```

pwin 待移动窗口的地址。

row, col 为移到视口左上角的窗口数据区的行和列值。指定的 row 和 col 值均相对于窗口数据区左上角(0,0)。

option WN\_UPDATE (0) 立即更新视口;  
WN\_NO\_UPDATE (4) 只在内部记录这个变化。

(返回) BWINDOW 结构的地址。若失败, 返回 NIL。

WNORIGIN 在视口中移动一个窗口, 显示窗口数据区的其它部分。

如果自动跟踪有效, 则窗口的移动会受到一些影响。因此, 使用 WNORIGIN 时也许要关闭自动跟踪。

如果需展示的部分不能合于视口或自动跟踪间隙中, 这时可以使用 WNSHOBK。

如果 row 和 col 迫使视口显示窗口数据区以外的数据, 则出现错误。

### 源程序(WNORIGIN.C):

```
#include <bwindow.h>
BWINDOW *wnorigin(pwin, row, col, option)
BWINDOW *pwin;
int row;
int col;
int option;
{
    int old_row;
    int old_col;

    wnvalidw (pwin);          /* Validate window structure.      */

                                /* Confirm that the data area coordinates make sense. */
                                /* sense. */
    if ((row < 0) || (col < 0) ||
        (wnview_w(pwin) > wndata_w(pwin) - col) ||
        (wnview_h(pwin) > wndata_h(pwin) - row))
    {
        wnreterr(WN_ILL_DIM);
    }

                                /* Remember the origin coordinates. */
    old_row = pwin->data_origin.row;
    old_col = pwin->data_origin.col;
                                /* Set the origin coordinates. */
    pwin->data_origin.row = row;
    pwin->data_origin.col = col;
                                /* Now apply cursor autotracking. */
    if (wncurtrk(pwin, pwin->cur_loc.row, pwin->cur_loc.col) != pwin)
        return(NIL);

                                /* Check to see if we actually need to do anything. */
    if (!pwin->internals.dirty &&
        (old_row == pwin->data_origin.row) &&
        (old_col == pwin->data_origin.col))
    {
        return(pwin);
    }

                                /* Update real screen if possible and requested. */
    return(wnnupblk(pwin, 0, 0, wndata_h(pwin) - 1,
```



```

        wndata_w(pwin) - 1, option));
}

```

## WNOVRLAP 报告两个长方形区域是否覆盖。

result = wnovrlap (pcorner1, pdim1, pcorder2, pdim2);

result 如果覆盖则为 1；否则为 0。

\*pcorder1, \*pcorder2 指向表示两个区域的左上角的结构。

\*pdim1, \*pdim2 指向包含两个区域尺寸的结构。

函数比较两长方形区域(假设在同一显示屏幕上)，并报告是否覆盖。

函数假设所有的参数都是正确的，不进行数据检查。

返回 result 如果覆盖则为 1；否则为 0。

源程序(WNOVRLAP.C):

```

#include <bwindow.h>
#define BOT_ROW1 (pcorder1->row + pdim1->h - 1)
#define BOT_ROW2 (pcorder2->row + pdim2->h - 1)
#define RT_COL1 (pcorder1->col + pdim1->w - 1)
#define RT_COL2 (pcorder2->col + pdim2->w - 1)
int wnovrlap (pcorder1, pdim1, pcorder2, pdim2)
const LOC *pcorder1;
const DIM *pdim1;
const LOC *pcorder2;
const DIM *pdim2;
{
    return (BOT_ROW1 >= pcorder2->row &&
            BOT_ROW2 >= pcorder1->row &&
            RT_COL1 >= pcorder2->col &&
            RT_COL2 >= pcorder1->col);
}

```

## WNPAGADD 增加窗口到在一设备一或显示页上显示的窗口的链表上。

presult = wnpagadd (pwindow, dev, page);

\*presult 指向刚创建的 WIN\_NODE 结构，如果失败则指向 NIL。

\*pwindow 增加到列表的窗口。

dev 视频显示设备(彩显或单色)

page 视频显示页。

函数增加窗口到控制显示在一给定设备和视频页上的窗口的链表。

若窗口结构无效或显示设备或显示页非法则产生错误。

返回 presult 指向刚创建的 WIND\_NODE 结构，如果失败则指向 NIL。

b\_wndlist[] 如果成功修改链表。

pwindow->pnode 指向新创建的 WIN\_NODE 结构，如果失败则为 NIL。

b\_wnderr 值如下：

(不改变) 成功

WN\_BAD\_WIN \*pwin 无效

WN\_BAD\_DEV 未知的设备。

WN\_BAD\_PAGE 内部错误。

```

#include <bwindow.h>

```

```

/* List of windows displayed on each page (NIL if */

```

```

/* none). */

```

```

WIN_NODE *(b_wndlist[MAX_DEVICES][MAX_PAGES]) = {0};

```

```

WIN_NODE *wnpgadd (pwindow, dev, page)
BWINDOW *pwindow;
int dev,page;
{
    wnvalidw (pwindow)
    wnvalidn (pwindow->pnode)
        /* Make sure dev & page are within range of array */
        /* subscripts. */
    if (dev != SC_MONO && dev != SC_COLOR)
        wnreterr (WN_BAD_DEV);

    if (utrange (page, 0, MAX_PAGES - 1))
        wnreterr (WN_BAD_PAGE);

        /* Attach node to window. */
    pwindow->pnode->pwin = pwindow;
        /* Attach to former topmost window. */
    pwindow->pnode->below = b_wnlist[dev][page];
        /* Attach former topmost to this window. */
    if (pwindow->pnode->below != NIL)
        pwindow->pnode->below->above = pwindow->pnode;

        /* New head of list. */
    return (b_wnlist[dev][page] = pwindow->pnode);
}

```

**WNPGRM** 从显示在一设备或显示页上的窗口链表上删除。

presult = wnpgrem (pwindow);

\*presult 指向新删除的 BWINDOW 结构，如果失败则指向 NIL。

\*pwindow 从链表中删除的窗口。

函数从控制视频设备和显示页上的窗口的链表上删除一窗口。

若窗口结构无效或窗口不与视频显示设备相联，或者如果没有无效的节点则都会产生错误。

返回 presult 指向刚删除的 BWINDOW 结构，如果失败则指向 NIL。

b\_wnlist[] 如果成功从链表中删除。

pwindow->pnode 如果成功则为 IL。

b\_wnerr 值如下：

(不改变) 成功

WN\_BAD\_WIN \*pwindow 无效

WN\_BAD\_NODE \*pwindow 不与一有效的节点相联。

WN\_BAD\_DEV 不显示

WN\_BAD\_PAGE 内部错误

**源程序(WNPGRM.C):**

```
#include <bwindow.h>
```

```
BWINDOW *wnpgrem (pwindow)
```

```
BWINDOW *pwindow;
```

```
{
```

```
    WIN_NODE *pnode;
```

```
    wnvalidw (pwindow)
```

```
    pnode = pwindow->pnode;
```

```

wvalidn (pnode)
    /* Make sure window is displayed. */
if (pwindow->where_shown.dev != SC_MONO &&
    pwindow->where_shown.dev != SC_COLOR)
    wnreterr (WN_BAD_DEV);
    /* Make sure window's page is within reasonable
    /* limits. */
if (utrange (pwindow->where_shown.page, 0, MAX_PAGES - 1))
    wnreterr (WN_BAD_PAGE);
    /* If this is top window, link root of list to next */
    /* window. */
if (b_wnlist[pwindow->where_shown.dev][pwindow->where_shown.page]
    == pnode)
    b_wnlist[pwindow->where_shown.dev][pwindow->where_shown.page]
    = pnode->below;
    /* Link next higher window to next lower window. */
if (pnode->above != NIL)
    pnode->above->below = pnode->below;
    /* Link next lower window to next higher window. */
if (pnode->below != NIL)
    pnode->below->above = pnode->above;
pnode->above = /* Detach from next higher */
               /* window. */
pnode->below = NIL; /* Detach from next lower */
return (pwindow);
}

```

## WNPIMBLK 输出窗口的一部分到屏幕。to screen.

presult = wnpimblk (pwin, r1, c1, r2, c2);

\*presult 指向已写写的窗口结构，如果失败则指向 NIL。

\*pwin 指向写向屏幕的窗口结构。

r1, c1 更新部分的图象在窗口数据中的左上角坐标。

r2, c2 更新部分的图象在窗口数据中的右下角坐标。

函数和显示屏幕输出指定窗口数据区的内容。

函数还选择指定的设备和显示页。

如果出现下列情况则产生错误：

- (1) 如果 pwin 指向的 BWINDOW 结构不包含有效的指向窗口的指针；
- (2) 如果 pwin->where\_shown 指明的设备和显示页无效；
- (3) 如果 pwin->image.dim 指定的尺寸和 pwin->where.corner 指定的位置相对显示设备不正确；
- (4) 给定的角的坐标不正确。

返回 presult 指向写出的 BWINDOW 结构，如果失败则指向 NIL。

b\_device 视频设备

b\_curpage 当前显示页

b\_wnerr 可能值如下：

(不改变) 成功

WN\_NULL\_PTR pimage->pdata 无效

WN\_BAD\_DEV 无效的设备、显示页或尺寸。

WN\_ILL\_DIM 无效的坐标或内部错误。

源程序(WNPIMBLK.C):

```

#include <stdio.h>

#include <bvideo.h>
#include <bwindow.h>
#include <bmouse.h>

/* Screen coordinates of window's viewport. */
#define VIEW_LEFT (pwin->where_shown.corner.col)
#define VIEW_RIGHT (VIEW_LEFT + wnview_w(pwin) - 1)
#define VIEW_TOP (pwin->where_shown.corner.row)
#define VIEW_BOTTOM (VIEW_TOP + wnview_h(pwin) - 1)

/* Absolute screen coordinates of window's visible data area. */
#define DATA_LEFT (VIEW_LEFT + pwin->data_origin.col)
#define DATA_RIGHT (DATA_LEFT + wndata_w(pwin) - 1)
#define DATA_TOP (VIEW_TOP + pwin->data_origin.row)
#define DATA_BOTTOM (DATA_TOP + wndata_h(pwin) - 1)

/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (pwin->data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(pwin) - 1)
#define VISIBLE_TOP (pwin->data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(pwin) - 1)

BWINDOW *wnpimblk (pwin, r1, c1, r2, c2)
BWINDOW *pwin;
int r1, c1, r2, c2;
{
    int old_npage;
    int num_written;
    int error = 0;
    CELL *pfrom;
    int gap;
    wnvalidw (pwin) /* Validate window structure. */
        /* Validate device, page, and dimensions and select */
        /* device and page. */
    if (wnseldev (&pwin->where_shown, &pwin->view_size, &old_npage))
        wnreterr (WN_BAD_DEV);
        /* Make sure all coordinates are reasonable. */
    if (utrange (r1, 0, wndata_h(pwin) - 1) ||
        utrange (r2, 0, wndata_h(pwin) - 1) ||
        utrange (c1, 0, wndata_w(pwin) - 1) ||
        utrange (c2, 0, wndata_w(pwin) - 1) ||
        (c1 > c2) || (r1 > r2))
        wnreterr (WN_ILL_DIM);
        /* Make sure requested area is visible. */
    if ((r2 < VISIBLE_TOP) || (r1 > VISIBLE_BOTTOM) ||
        (c2 < VISIBLE_LEFT) || (c1 > VISIBLE_RIGHT))
    {
        return(pwin);
    }
}

```

```

}

/* Clip the requested coordinates to fit within the */
/* viewport. */
utbound(r1, VISIBLE_TOP, VISIBLE_BOTTOM);
utbound(r2, VISIBLE_TOP, VISIBLE_BOTTOM);
utbound(c1, VISIBLE_LEFT, VISIBLE_RIGHT);
utbound(c2, VISIBLE_LEFT, VISIBLE_RIGHT);

/* Temporarily keep the mouse cursor out of the */
/* window. */
mohide(MO_HIDE);
pfrom = pwin->img.pdata + ((r1 * wdata_w(pwin)) + c1);
gap = wdata_w(pwin) - ((c2 - c1) + 1);
num_written = viwrsect(VIEW_TOP + (r1 - VISIBLE_TOP),
                       VIEW_LEFT + (c1 - VISIBLE_LEFT),
                       VIEW_TOP + (r2 - VISIBLE_TOP),
                       VIEW_LEFT + (c2 - VISIBLE_LEFT),
                       (char *) pfrom, gap, 0, 0, CHAR_ATTR);

if (num_written != ((r2 - r1 + 1) * (c2 - c1 + 1)))
    error = 1;

/* Re-enable the mouse cursor. */
mohide(MO_SHOW);

if (error)
    wnreterr (WN_ILL_DIM);
return (pwin);
}

```

## WNPOLL 查询属于特定集的键盘或鼠标事件一次。

```

presult = wnpoll (pevent_list, pvert, phoriz, pevent);
*presult 指向发现的事件，如果未找到则为 NIL。
*pvert 指向找到的鼠标事件存储空间的行
*phoriz 指向找到的鼠标事件存储空间的行
*pevent 指向被删除删除事件的结构。

```

函数接受一指向事件表的指向，顺序查询鼠标和键盘检查是否有指定的事件发生。如果一事件被删除，\*pevent 被正确填充。注意\*pevent 和\*presult 很不同，\*pevent 是一被删除的事件，而\*presult 是事件表匹配\*pevent 的成员，如果有这样的成员存在的话。注意返回的鼠标事件应认为是候补事件；也就是说，及\*pvert 和\*phoriz 应由调用者检查是否有实际意义。

返回                      presult                      指向发现的候补事件，如果失败则指向 NIL。  
                              \*pevent                      指向 删除的事件，如果没有则不改变。

### 源程序(WNPOLL.C):

```

#include <bwindow.h>
#include <bmouse.h>
WN_EVENT *wnpoll(pevent_list, pvert, phoriz, pevent)
WN_EVENT_LIST *pevent_list;
unsigned *pvert;

```

```

unsigned *phoriz;
WN_EVENT *pevent;
{
    int            error;
    unsigned long  mouse_event;
    WN_EVENT_LIST *pcurrent_event_node;
    WN_EVENT       find_event;
    int            keystroke_found;

    if ((moequip() > 0) && (pevent_list != NIL))
    {
        pcurrent_event_node = pevent_list;
        do
        {
            if (pcurrent_event_node->win_event.event_type ==
                WN_MOUSE_EVENT)
            {
                error = mocheck(
                    pcurrent_event_node->win_event.event.mouse.event,
                    pcurrent_event_node->win_event.event.mouse.ignore,
                    MO_CLEAR, &mouse_event, pvert, phoriz);
                if (mouse_event && (error == MO_OK))
                {
                    if (pevent != NIL)
                    {
                        *pevent = pcurrent_event_node->win_event;
                        pevent->pdata = NIL;
                        pevent->data_size = 0;
                    }
                    return(&(pcurrent_event_node->win_event));
                }
            }
            pcurrent_event_node = pcurrent_event_node->pnext;
        } while ((pcurrent_event_node != NIL) &&
            (pcurrent_event_node != pevent_list));
    }

    /* No mouse installed or no mouse event found;          */
    /* check the keyboard.                                    */
    find_event.event_type = WN_KB_EVENT;
    find_event.pdata      = NIL;
    find_event.data_size  = 0;
    keystroke_found = kbpoll(b_key_ctrl, pevent_list,
                            &find_event.event.keystroke,
                            KB_REMOVE_KEY);

    if (pevent != NIL)
        *pevent = find_event;
    if (keystroke_found != 0)
        return(wrtecvn(pevent_list, &find_event));
    else
        return(NIL);
}

```

}

## WNPRINTF 向当前窗口写入一个格式化的字符串

```
#include <bwindow.h>
```

```
int wnprintf(const char *pfmt, ...);
```

pfmt 包含格式说明的字符串，以 NUL ('\0')结尾。

(返回) 写入的字符数。如有错误，则返回 EOF (EOF 在 stdio.h 中定义)。

WNPRINTF 以标准的printf()函数的方式格式化一个字符串，将该字符串以电传方式(TTY)写入当前窗口中。必要时正文从一行折转到下一行，窗口滚动。

新行字符 ('\n') 作为回车和换行对待，响铃字符 ('\a')引起鸣叫。

如果这是一个虚拟窗口，则当字符放在窗口中时在视口中显示的窗口部分将跟随光标移动。

WNPRINTF 第一次被激活时利用 calloc()分配一个内部字符串缓冲区，将缓冲区的地址存放在 b\_wnprbf中。如果想控制缓冲区的尺寸或迫使缓冲区在更早的时候分配，可以使用 WNSETBUF。

如果没有一个窗口被指定为当前窗口或内部缓冲区溢出，则出现一个错误。在出现错误的情况下不执行输出。

WNWRSTR或WNWRSTRN能够避开内部的printf()机制，WNWRSTRN可以避免光标跟踪，而 WNWRAP则能够显示一个字符串而不使一个字在窗口的行之间断开。

### 源程序(WNPRINTF.C):

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <stdarg.h>
```

```
#include <bwindow.h>
```

```
/* Buffer to VSPRINTF into. It is NIL the first */  
/* time WNPRINTF is called, unless the user called */  
/* WNSETBUF. */
```

```
char *b_wnprbf = NIL;
```

```
/* Size of buffer. Zero to start with. */
```

```
unsigned int b_wnbfsz = 0;
```

```
int wnprintf (const char *fmt,...)
```

```
{
```

```
    int result;
```

```
    unsigned int size;
```

```
    va_list parg;
```

```
/* Exit if current window is not valid. */
```

```
if (NIL == wnvalwin (b_pcurwin))
```

```
{
```

```
    wncror (WN_BAD_WIN);
```

```
    return (EOF);
```

```
}
```

```
/* Allocate the buffer for VSPRINTF to stuff */
```

```
/* characters into. */
```

```
if (b_wnprbf == NIL)
```

```
{
```

```
    size = wnsetbuf (B_WNPRBF_SIZE);
```

```

        if (size != B_WNPRBF_SIZE)
            wncerror (WN_NO_MEMORY);
    }

    /* Make parg point to the parameter just before the */
    /* beginning of the variable argument list.          */
    va_start (parg, fmt);
    /* Put a NUL at the end of the VSPRINTF buffer so    */
    /* we know if it overflowed.                          */
    b_wnprbf [b_wnbfsz - 1] = '\0';
    /* Do the actual interpretation of the format        */
    /* string into B_WNPRBF.                             */
    result = vsprintf (b_wnprbf, fmt, parg);
    /* Done using the variable argument list-discard it.*/
    va_end (parg);
    /* Check for overflow.                                */
    if (b_wnprbf [b_wnbfsz - 1] != '\0')
    {
        wncerror (WN_PR_OVERFLOW);
        return (EOF);
    }

    /* If VSPRINTF didn't get an error, send the string */
    /* to the window.                                     */
    if (result > 0)
        wnwrstrn (b_pcurwin, b_wnprbf, result, -1, -1, WN_UPDATE);
    return (result);
}

```

## WNPUTBOR 沿着长方形区域显示一边界。

result = wnputbor (pdim, pborder, pwhere);

result 如果边界实际写的话，否则为0。

\*pdim 指向表示环绕的长方形的尺寸的结构。

\*pwhere 指向表示设备、显示页和长方形区域的左上角的位置的指针。

\*pbord 指向表示环绕的边线的类型的BORDER结构。

函数沿着长方形的区域在给定的视频和显示页上显示一边界。

设备和显示页被选为当前(用来输出)。

如果视频设备、显示页或长方形的尺寸是非法的，不显示边线函数返回值1。

返回 result 如果边线已实际写出则为0；否则为1。

b\_device 当前显示设备。

b\_curpage 当前显示页。

## 源程序(WNPUTOPT.C):

```
#include <string.h>
```

```
#include <bvideo.h>
```

```
#include <bwindow.h>
```

```
#include <bmouse.h>
```

```
/* Height and width of window data area.          */
```

```
#define H (pdim->h)
```

```
#define W (pdim->w)
```

```
/* Absolute screen ordinates for one-above top and */
```



```

        /* one-below bottom of window data area. */
#define TOP_ROW    (pwhere->corner.row - 1)
#define BOTTOM_ROW (TOP_ROW + H + 1)
        /* Absolute screen ordinates for one-to-left and */
        /* one-to-right of window data area. */
#define LEFT_COLUMN (pwhere->corner.col - 1)
#define RIGHT_COLUMN (LEFT_COLUMN + W + 1)
        /* Display a string on screen, using absolute */
        /* coordinates and length. */
#define dispstr(row, col, pbuf, len, attr) \
        (viwrrrect((row),(col),(row),(col)+len-1, \
        (pbuf), \
        utlonyb (attr), uthiayb (attr), \
        CHARS_ONLY))
        /* Display a string on screen, using absolute */
        /* coordinates and length. */
#define dispchar(row, col, pchar, attr) \
        (viwrrrect((row),(col),(row),(col), \
        (pchar), \
        utlonyb (attr), uthinyb (attr), \
        CHARS_ONLY))
        /* This is TRUE if all of the bits set in B are set */
        /* in A. */
#define IFBIT(a,b) if ((a & b) == b)
int wnputbor (pdim, pborder, pwhere)
const DIM *pdim;
const BORDER *pborder;
const WHERE *pwhere;
{
    int old_npage;
    int mode, columns, act_page;
    int bordertype = pborder->type & BBRD_TYPE;
    int tiletype = pborder->type & BBRD_TITLE;
    int tlen;
    int error;
    BJOINT *pjoint;
        /* Select device and page. */
    if (wnseldev (pwhere, pdim, &old_npage))
        return (1);
        /* Exit if writing the border is impossible due to */
        /* placement considerations. */
    if (bordertype == 0 || pwhere->corner.row <= 0 ||
        pwhere->corner.col <= 0 || BOTTOM_ROW >= scrow ||
        (smode (&mode, &columns, &act_page), (RIGHT_COLUMN >= columns)))
        return (1);
        /* No border actually written. */
        /* Temporarily keep the mouse cursor out of the */
        /* window's border. */
    mhide(MO_HIDE);
        /* Write the border's box. */
    error = scbox (TOP_ROW, LEFT_COLUMN,

```

```

        BOTTOM_ROW, RIGHT_COLUMN,
        (bordertype == BBRD_CHAR) ? (-1) : (bordertype - 1),
        pborder->ch,
        pborder->attr);

if (error)
{
    /* Re-enable the mouse cursor. */
    mhide(MO_SHOW);
    return (1);
}

if ((bordertype != BBRD_CHAR) || (titletype != BBRD_TITLE))
{
    /* Write out joints if any. */
    IFBIT (pborder->type, BBRD_HASJOINTS)
    {
        for (pjoint = pborder->pjoints;
             pjoint != NIL;
             pjoint = pjoint->next)
            dispchar (pjoint->row + TOP_ROW,
                      pjoint->col + LEFT_COLUMN,
                      &(pjoint->jtype), pborder->attr);
    }

    /* Write out Top Left Title, if specified. */
    IFBIT (titletype, BBRD_TLT)
    {
        tlen = strlen (pborder->ptitle);
        tlen = (tlen > W) ? W : tlen;
        dispstr (TOP_ROW, LEFT_COLUMN + 1,
                 pborder->ptitle, tlen, pborder->tattr);
    }

    /* Write out Top Centered Title, if specified. */
    IFBIT (titletype, BBRD_TCT)
    {
        tlen = strlen (pborder->ptitle);
        tlen = (tlen > W) ? W : tlen;
        dispstr (TOP_ROW,
                 LEFT_COLUMN +
                 (((RIGHT_COLUMN - LEFT_COLUMN + 1) >> 1) -
                  (tlen >> 1)),
                 pborder->ptitle, tlen, pborder->tattr);
    }

    /* Write out Top Right Title, if specified. */
    IFBIT (titletype, BBRD_TRT)
    {
        tlen = strlen (pborder->ptitle);
        tlen = (tlen > W) ? W : tlen;
        dispstr (TOP_ROW, RIGHT_COLUMN - tlen,
                 pborder->ptitle, tlen, pborder->tattr);
    }
}

```

```

        /* Write out Bottom Left Title, if specified.          */
IFBIT (titletype, BBRD_BLT)
{
    tlen = strlen (pborder->pbtitle);
    tlen = (tlen > W) ? W : tlen;
    dispstr (BOTTOM_ROW, LEFT_COLUMN + 1,
             pborder->pbtitle, tlen, pborder->btattr);
}

        /* Write out Bottom Centered Title, if specified.      */
IFBIT (titletype, BBRD_BCT)
{
    tlen = strlen (pborder->pbtitle);
    tlen = (tlen > W) ? W : tlen;
    dispstr (BOTTOM_ROW,
             LEFT_COLUMN +
             (((RIGHT_COLUMN - LEFT_COLUMN + 1) >> 1) -
              (tlen >> 1)),
             pborder->pbtitle, tlen, pborder->btattr);
}

        /* Write out Bottom Right Title, if specified.         */
IFBIT (titletype, BBRD_BRT)
{
    tlen = strlen (pborder->pbtitle);
    tlen = (tlen > W) ? W : tlen;
    dispstr (BOTTOM_ROW, RIGHT_COLUMN - tlen,
             pborder->pbtitle, tlen, pborder->btattr);
}
}

        /* Re-enable the mouse cursor.                          */
mohide(MO_SHOW);

return (0);
}

```

### WNPUTSEN 显示窗口的sensor表。

result = wnputsen(pwin);

result 表示操作成功与失败的错误代码。

\*pwin 指向显示sensor的窗口。

函数试图显示指定窗口的sensor项。

窗口的设备和显示页被选择用来输出。

如果显示设备或页或坐标或项的尺寸非法，不显示该项，函数返回值1。

返回	result	下列值之一：
		WN_NO_ERROR - 成功
		WN_NOT_SHOWN - 窗口不显示
		WN_ILL_DIM - Sensor的大小非法
		WN_ILL_VALUE - Sensor的相对域包含一损坏的值。

b_device	当前显示设备
b_curpage	当前显示页

### 源程序(WNPUTSEN.C):

```
#include <string.h>
#include <bvideo.h>
#include <bwindow.h>
#include <bmouse.h>
int wnputsen(pwin)
BWINDOW *pwin;
{
    int      old_npage;
    int      mode, cols, act_page;
    int      sensor_row, sensor_col;
    WN_SENSOR *psensor;
    int      bord_row, bord_col;
            /* Select device and page. */
    if (wnseldev(&pwin->where_shown, &pwin->view_size, &old_npage))
        return(WN_NOT_SHOWN);
    scmode(&mode, &cols, &act_page);
    bord_row = pwin->where_prev.corner.row;
    bord_col = pwin->where_prev.corner.col;
            /* Walk the list of window sensors. */
    for (psensor = pwin->psensors; psensor != NIL;
        psensor = psensor->pnext)
    {
        switch(psensor->relative)
        {
        case WN_VIEWPORT:
            sensor_row = pwin->where_prev.corner.row;
            sensor_col = pwin->where_prev.corner.col;
            break;

        case WN_DATA:
            if (utrange(psensor->size.h, 0, wndata_h(pwin)) ||
                utrange(psensor->size.w, 0, wndata_w(pwin)) ||
                utrange(psensor->corner.row,
                    0, (wndata_h(pwin)) - psensor->size.h) ||
                utrange(psensor->corner.col,
                    0, (wndata_w(pwin)) - psensor->size.w))
            {
                return(WN_ILL_DIM); /* No sensor actually written. */
            }

            if (psensor->pimage != NIL)
            {
                /* Temporarily keep the mouse cursor out of the */
                /* window. */
                mohide(MO_HIDE);

                wnwrrect(pwin, psensor->corner.row, psensor->corner.col,
```

```

        psensor->corner.row + psensor->size.h - 1,
        psensor->corner.col + psensor->size.w - 1,
        (char *) psensor->pimage, -1, -1, CHAR_ATTR);

    /* Re-enable the mouse cursor. */
    mohide(MO_SHOW);
}

continue;

case WN_SCREEN:
    sensor_row = 0;
    sensor_col = 0;
    break;

default:
    return(WN_ILL_VALUE); /* No sensor actually written. */
}

sensor_row += psensor->corner.row;
sensor_col += psensor->corner.col;

    /* Exit if writing the sensor is impossible due to */
    /* placement considerations. */
    if (utrange(psensor->size.h, 0, wnbord_h(pwin)) ||
        utrange(psensor->size.w, 0, wnbord_w(pwin)) ||
        utrange(sensor_row, bord_row,
            (bord_row + wnbord_h(pwin)) - psensor->size.h) ||
        utrange(sensor_col, bord_col,
            (bord_col + wnbord_w(pwin)) - psensor->size.w))
    {
        return(WN_ILL_DIM); /* No sensor actually written. */
    }

    /* Write out the sensor. */
    if (psensor->pimage != NIL)
    {
        /* Temporarily keep the mouse cursor out of the */
        /* sensor. */
        mohide(MO_HIDE);
        viwrrrect(sensor_row, sensor_col,
            sensor_row + psensor->size.h - 1,
            sensor_col + psensor->size.w - 1,
            (char *) psensor->pimage, -1, -1, CHAR_ATTR);
        /* Re-enable the mouse cursor. */
        mohide(MO_SHOW);
    }
}
return(WN_NO_ERROR);
}

```

## WNQUERY 返回经窗口得到的来自用户的字符串。

```
#include <bwindow.h>
```

```
char wnquery( char *presp,  
              int resp_size,  
              int *pkey);
```

presp 放置用户输入字符的缓冲区的地址。

resp\_size 以字节计的输入缓冲区的尺寸 (包括用于尾随 NUL('\0') 的一个字节)。

pkey 结束输入的按键键码。

(返回) 结束输入的按键 ASCLL (字符)值。

WNQUERY返回来自用户的字符串, 回显这些按键并将所有 I/O 限制在当前窗口。

即使窗口可能被指定为“延迟窗口”, 以前挂起的输出仍在窗口显示出来, 所有按键也回显在窗口中。如果窗口在调用 WNQUERY 之前被“延迟”, 则在退出时恢复这个状态。

用户可以象通常那样键入字符, 但下列按键之一将终止结束输入: ENTER、Ctrl-M、Ctrl-j、一个功能键或扩展的键序列。字符被回显, 光标移过整个窗口; 到达窗口的最右列时, 光标移到下一行; 光标到达窗口右下角时输入即停止, 缓冲区满时输入也停止。

一旦输入停止(光标到达窗口的末尾或缓冲区已满), 光标将停留在最后输入的字符之后, 以后的按键(除了退格符和结束键)将引起鸣叫并被废A。退格符象通常那样工作。在ENTER或其它结束键被按下之前函数不返回 (即使输入已经停止)。

结束字符不作为输入的一部分返回, 结束按键的ASCLL码作为函数值返回, 其键码(扫描码)放在 \*pscan中。

下列按键作为输入的一部分时具有特殊的效果:

退格或Ctrl-H('\b') 删除以前输入的字符并将光标移到那个光标位置。在输入刚刚开始时键入退格或Ctrl-H将引起鸣叫。

NUL (ASCLL 0)被忽略, 并被废A。

Ctrl-G引起鸣叫, 但不作为输入的一部分返回。

ENTER、Ctrl-M('\15')、Ctrl-J('\12')或特殊的非 ASCLL IBM 键(象功能键和ALT键)结束输入。

所有其它的 ASCLL 字符包括控制字符均被回显。

(控制字符作为其本身回显, 也就是说没有上箭头('^')前缀。)

如果没有一个窗口被指定为当前窗口、当前窗口数据区的任意部分被另外一个窗口覆盖或当前窗口未显示, 这时出现一个错误。函数在出现错误的情况下立即返回而不显示信息, 错误代码保存在b\_wnerr中, 0 在 \*pkey 中作为函数值返回, \*presp 被置成空字符串("")。

### 源程序(WNQUERY.C):

```
#include <string.h>
```

```
#include <bkeybrd.h>
```

```
#include <bwindow.h>
```

```
#define BEL ((int) '\7') /* Bell, CTRL-G */
```

```
#define BS ((int) '\b') /* Backspace, CTRL-H */
```

```
#define LF ((int) '\12') /* Linefeed, CTRL-J */
```

```
#define ENTER ((int) '\15') /* ENTER, CTRL-M */
```

```
#define NUL '\0'
```

```
#define wnretnul(errnum)
```

```
{  
    wnerror (errnum);  
    return (NUL);  
}
```

```
char wnquery (presponse, resp_size, pscan)
```

```
char *presponse;
```

```

int    resp_size, *pscan;
{
    int        was_delayed;
    unsigned char ch;
    int        is_char;
    int        max_row,max_col,row,col;
    int        num_have;
    const char  blank = ' ';
    KEY_SEQUENCE kseq;

        /* Zeros in case of error. */
    *presponse = NUL;
    *pscan      = 0;
        /* Validate window. */
    if (wnvalwin (b_pcurwin) == NIL)
        wnretnul (WN_BAD_WIN)
        /* Make sure it's visible. */
    if (b_pcurwin->where_shown.dev != SC_MONO &&
        b_pcurwin->where_shown.dev != SC_COLOR)
        wnretnul (WN_NOT_SHOWN)
        /* Make sure we can write to it. */
    if (b_pcurwin->internals.frozen)
        wnretnul (WN_COVERED)
        /* Complete pending writes, and allow immediate I/O.*/
    was_delayed = b_pcurwin->options.delayed;
    if (wnupdate (b_pcurwin) == NIL)
        return ("0");
    b_pcurwin->options.delayed = 0;
    max_row = wndata_h(b_pcurwin) - 1;
    max_col = wndata_w(b_pcurwin) - 1;
    num_have = 0;
    do        /* Collect the response. */
    {
        /* Await a keystroke. */
        kseq = kbwait(b_key_ctrl,b_pcurwin);
        *pscan = kseq.key_code;
        ch     = kseq.character_code;
        if (ch == 0xc0)
            is_char = (kseq.key_code == 0);
        else
            is_char = ch;

        if (is_char)
        {
            /* Got an ASCII character. */
            wncurpos (&row, &col);

            switch (ch)
            {
                case LF:
                case ENTER:

```

```

        break;          /* This is a terminating          */
                        /* character:  handle it at        */
                        /* loop exit.                      */

case '\0':             /* Ignore NULs.                      */
    break;

case BEL:              /* BEL:  just beep.                      */
    wnwrty ((char) BEL, -1, -1);
    break;

default:
    /* If buffer full or end of window          */
    if ((num_have >= resp_size - 1) ||
        (row >= max_row && col >= max_col))
        wnwrty ((char) BEL, -1, -1);

    else
    {
        /* Echo char & advance cursor.          */
        wnwrty (ch, -1, -1);
        /* Update response buffer.              */
        num_have++;
        *presponse++ = ch;
    }
    break;

case BS:
    if (num_have)
    {
        /* Move cursor to last char            */
        /* entered.                            */

        if (col)/* Back up one column.          */
            col--;

        else /* Back up to previous row.        */
        {
            row--;
            col = max_col;
        }

        /* Erase previous character.            */
        wnwrbuf (row, col, 1, &blank, -1, -1,
            CUR_BEG | CHARS_ONLY | MOVE_CUR);
        num_have--;
        presponse--;
    }
    else
        /* Can't delete past beginning.          */
        wnwrty((char) BEL, -1, -1);
    break;
}

```



```

    }
} while (is_char && (ch != (char) ENTER) && (ch != (char) LF));

*response = '\0';           /* Terminate response string.          */

                               /* Restore "delayed" state.          */
b_pcurwin->options.delayed = was_delayed;

return ch;                   /* Successful exit.          */
}

```

## WNRDBUF 读取当前窗口中一片连续位置的内容

```
#include <bwindow.h>
```

```
int wnrdbuf( int row, int col,
             int num_spaces,
             char *pbuffer,
             int option);
```

row,col 开始读取的行和列(相对于窗口的数据区)。

num\_spaces 读取的字符数。

pbuffer 放置数据的空间。

option 有三位, 指明缓冲区是如何建立的以及读完数据后光标放置在何处:

符号	值	意义
CUR_BEG	0	将光标放在(row, col)。
CUR_AFTER	1	将光标放在字符串的末尾。
CHARS_ONLY	0	缓冲区仅包含字符。
CHAR_ATTR	2	缓冲区包含(char,attr) 9 对。
MOVE_CUR	0	根据 CUR_BEG/CUR_AFTER 放置光标。
NO_MOVE_CUR	4	保持光标位置不变。

(返回) 实际读取的字符数。

WNRDBUF从当前窗口读取 5 接的字符, 可带有或不带有相应的显示属性。它可以将窗口光标放在读取空间的开始或末尾。

必要时字符的读取将延续至后续的行, 但在窗口的末尾停止。字符的实际读取数目作为函数值返回。字符从窗口数据区读取而不涉及显示窗口的视口。

如果字符的读取包括窗口的右下角, 则当指定MOVE\_CUR和CUR\_AFTER时光标被放置在右下角。

缓冲区的末尾不添加尾随的NUL('\0')。

如果没有活动的窗口或row、col超出窗口的尺寸, 则出现一个错误。不论在何种情况下错误码均存放在b\_wncerr中, 零作为函数值返回。

### 源程序(WNRDBUF.C):

```

#include <bwindow.h>
#define H (wndata_h(b_pcurwin))
#define W (wndata_w(b_pcurwin))
#define AREA (H * W)
int wnrdbuf (row, col, num_spaces, buffer, option)
int row, col, num_spaces;
char *buffer;
int option;
{

```

```

int    want_attr, i, offset, last;
CELL *pdata;
int old_row, old_col;
        /* Validate window structure. */
if (wnvalwin(b_pcurwin) == NIL)
    wnreterz (WN_BAD_WIN);
        /* Validate row, col. */
if (utrange (row, 0, H - 1) ||
    utrange (col, 0, W - 1))
    wnreterz (WN_ILL_DIM);
        /* Be sure there's valid data. */
if (b_pcurwin->img.pdata == NIL)
    wnreterz (WN_NULL_PTR);
        /* Cell at which to start. */
offset = (row * W) + col;

        /* Pointer to first cell. */
pdata = b_pcurwin->img.pdata + offset;

        /* Don't go beyond end of window. */
utuplim (num_spaces, (AREA - offset))

want_attr = ((option & CHAR_ATTR) != 0);

for (i = 0; i < num_spaces; i++)
{
    *buffer++ = pdata[i].ch;
    if (want_attr)
        *buffer++ = pdata[i].attr;
}
if (!(option & NO_MOVE_CUR))
{
    old_row = b_pcurwin->data_origin.row;
    old_col = b_pcurwin->data_origin.col;
    if (option & CUR_AFTER)
    {
        /* Leave cursor after last read. */
        last = offset + num_spaces;
        /* Don't go beyond end of window. */
        utuplim(last, AREA - 1)
        wncurtrk(b_pcurwin, last / W, last % W);
    }
    else
        wncurtrk(b_pcurwin, row, col);
    if ((old_row != b_pcurwin->data_origin.row) ||
        (old_col != b_pcurwin->data_origin.col))
    {
        wnnupblk(b_pcurwin, 0, 0, wndata_m(b_pcurwin),
            wndata_w(b_pcurwin), WN_UPDATE);
    }
    wncurset(b_pcurwin);
}

```

```

    }
    return (num_spaces);
}

```

## WNREAD 允许用户在虚拟窗口中浏览

#include <bwindow.h>

```

WN_EVENT *wnread(BWINDOW *pwin,
                  const WHERE *pwhere,
                  int view_ht, int view_wid,
                  int org_row, int org_col,
                  const BORDER *pbord,
                  WN_EVENT *pfinal,
                  int option);

```

pwin 显示给用户的BWINDOW结构的地址。

pwhere 如果窗口未显示, 这是WHERE结构的地址, 该结构指明设备、显示页和窗口显示位置的坐标。

view\_ht, view\_wid 如果窗口未显示, 它们是视口的行数和列数。

org\_row, org\_col 如果窗口未显示, 它们是出现在视口左上角的窗口数据区的行和列(相对于(0,0))。

pbord 如果窗口未显示, 它是BORDER结构的地址, 该结构指明放在视口周围的边界和标题。

pfinal 结构的地址, 该结构接受用户执行的最后事件的拷贝。若不需要这个拷贝, 则为NIL。

option 下列位值的零个或任意个的组合:

符号	值	意义
WN_KNOWN_EVENTS	0x00	忽略未认可的事件。
WN_UNKNOWN_TRANSMIT	0x01	按下未定义的按键时返回。
WN_ALL_TRANSMIT	0x03	在每次按键或认可的鼠标器事件发生后返回。
WN_KBIGNORE	0x04	忽略并废弃所有的按键。
WN_USE_MOUSE	0x00	如果有鼠标器, 使之有效。
WN_NO_MOUSE	0x08	即使有鼠标器也忽略之。

(返回) pfinal的一个拷贝。若失败, 则为NIL。

WNREAD允许用户通过键盘或鼠标器在鼠标器在虚拟窗口中浏览。用户可按下箭头键或在滚动箭头上按动鼠标器查看窗口数据区的不同部分。完成查看窗口的操作后, 按Enter或Esc, WNREAD即返回调用程序。

如果窗口未显示, WNREAD在视口中显示这个窗口。有关描述视口的pwhere、view\_ht、view\_wid、org\_row、org\_col和pbord的详细说明, 请参看WINDSPY和WNVDISP。

(如果窗口已显示而用户想使用滚动箭头, 您可以用WNSCRLBR来安装滚动箭头。)

如果pfinal不是NIL, 则WNREAD用WN\_EVENT结构来填充\*pfinal, 这个结构描述了用户的最后的鼠标器或键盘事件。WN\_EVENT结构在bwindow.h中定义如下:

```

typedef struct /*MOUSE_EVENT 结构: */
{
    unsigned long event; /*需注意和忽略的 鼠标器特性。*/
    unsigned long ignore;
    unsigned location; /* 位置代码。*/
} MOUSE_EVENT;

typedef struct /*WN_EVENT 结构: */
{
    unsigned event_type; /*WN_KB_EVENT或WN_MOUSE_EVENT。*/
    union /*事件联合: */
    {
        /*键盘或 鼠标器事件的基本数据。*/

```

```

        KEY_SEQUENCE keystroke;
        MOUSE_EVENT mouse;
    } event;
    void *pdata;           /*与事件有关的数据。*/
    unsigned data_size;    /*pdata指向的数据尺寸。*/
} WN_EVENT;

```

KEY\_SEQUENCE 在bkeybrd.h.中定义如下:

```

typedef struct
{
    unsigned char character_code;
    unsigned char character_code;
    unsigned char key_code;
} KEY_SEQUENCE;

```

对WN\_EVENT 结构解释如下:

event\_type成员告诉最后的事件是按键还是鼠标器操作。如果最后事件是按键, event.keystroke成员记录按下的键; 如果最后事件是鼠标器操作, 则event.mouse成员记录发现的鼠标器事件。参见MOCHECK中有关鼠标器事件的说明。

pdata成员是WN\_ACTION 变量的地址, 该变量可以是下列值之一:

```

typedef enum          /*WN_ACTION:*/
{
    /*WN_EVENT动作域的值。*/
    WN_NULL,           /*无动作。*/
    WN_SCROLL_LEFT,    /*左移。*/
    WN_SCROLL_RIGHT,   /*右移。*/
    WN_SCROLL_UP,      /*上移。*/
    WN_SCROLL_DOWN,    /*下移。*/
    WN_PAGE_LEFT,      /*左移一页。*/
    WN_PAGE_RIGHT,     /*右移一页。*/
    WN_PAGE_UP,        /*上移一页。*/
    WN_PAGE_DOWN,      /*下移一页。*/
    WN_LEFT_EDGE,      /*移到左边沿。*/
    WN_RIGHT_EDGE,     /*移到右边沿。*/
    WN_TOP_EDGE,       /*移到上边沿。*/
    WN_BOTTOM_EDGE,    /*移到下边沿。*/
    WN_ABORT,          /*中止读。*/
    WN_TRANSMIT        /*完成读。*/
} WN_ACTION;

```

根据pdata指向的值可以推测用户是想进一步查看窗口中的数据还是完成了查看窗口的操作。

有关MO\_HOLD的特别提示: 通常当一个或多个鼠标器按钮处于指定位置后, 不论经历的时间多么短, 这个鼠标器事件均能被识别。然而, WNREAD在经过指定的计时脉冲数之前忽略MO\_HOLD事件, 这个延迟值保存在全局变量b\_hold中, 其缺省值是4个脉冲。要想在WNREAD下对MO\_HOLD立即响应, 可以设置b\_win\_hold为零。

“自动跟踪”对于WNREAD不起作用。

有关的例程: 通过WNINITEV、WNCHGEVN、WNREMEVN和WNZAPEVN可以修改用户响应表, 利用WNSCRLBR可以将滚动箭头放在任何窗口上。如果pwin未指向一个合法的窗口结构或视口超出窗口数据区及屏幕尺寸, 则出现一个错误。如果窗口显示的位置不可能, 这时也出现一个错误。例如请求了一个未知的设备。

### 源程序(WNREAD.C):

```

#include <bwindow.h>
#include <bmouse.h>

```

```

unsigned b_win_delay = 0;      /* Amount of time to wait between */
                               /* window polls. */
long b_win_hold = 4L;         /* Amount of time to wait before */
                               /* reporting MO_HOLD events. */
typedef struct                 /* CLEANUP_ITEMS items to */
{                               /* restore on exit. */
    /*                               */
    int MOCATCH_was_off;       /* Must remove MOCATCH. */
    int sensors_made;          /* Must free sensors. */
    int events_made;           /* Must free event list. */
    int autotrack_state;       /* Former state of autotrack. */
    int cursor_state;          /* Former state of cursor. */
} CLEANUP_ITEMS;
static const WN_EVENT *poll_event(const BWINDOW *, int, int);
static void clean_up(BWINDOW *, const CLEANUP_ITEMS *);
static int wnchxkit(const WN_EVENT_LIST *, int);
WN_EVENT *wnread (pwin, pwhere, view_height, view_width,
                  origin_row, origin_col, pborder, pfinal, option)
BWINDOW *pwin;
const WHERE *pwhere;
int view_height;
int view_width;
int origin_row;
int origin_col;
const BORDER *pborder;
WN_EVENT *pfinal;
int option;
{
    int done = 0;
    int error;
    int row, col;
    const WN_EVENT *pevent;
    int scrollbars = 0;
    CLEANUP_ITEMS cleanup = {0, 0, 0, 0, 0};
    int old_dev, old_page, old_npage;
    int mode, columns, act_page;
    /* First, make sure the window is valid. */
    wnvalidw(pwin);
    /* Remember the state of the cursor & whether auto */
    /* tracking is on, then shut both of them off. */
    wngetopt(pwin, WN_CUR_TRACK, &cleanup.autotrack_state);
    wngetopt(pwin, WN_CUR_OFF, &cleanup.cursor_state);
    wnsetopt(pwin, WN_CUR_TRACK, 0);
    wnsetopt(pwin, WN_CUR_OFF, 1);
    wncursor(pwin);
    /* If the window event list has not yet been */
    /* initialized, do so. */
    if (pwin->pevent_list == NIL)
    {
        if (wnnitivev(pwin) != WN_NO_ERROR)

```

```

        return(NIL);
        cleanup.events_made = 1;
    }

        /* Check for an escape route. */
error = wnchkxit(pwin->pevent_list, option);
if (error != WN_NO_ERROR)
{
    clean_up(pwin, &cleanup);
    return(NIL);
}

        /* Now make the sensor list and display the window */
        /* if necessary. */
if ((pwin->where_shown.dev != SC_MONO) &&
    (pwin->where_shown.dev != SC_COLOR))
{
    if ((moequip() > 0) && (pwin->psensors == NIL) &&
        (pborder != NIL) && (pborder->type != BBRD_NO_BORDER) &&
        !(option & WN_NO_MOUSE))
    {
        if (wndata_h(pwin) > view_height)
            scrollbars = WN_VERTICAL;
        if (wndata_w(pwin) > view_width)
            scrollbars |= WN_HORIZONTAL;
        if (NIL == wnscribr(pwin, view_height, view_width,
                            pborder->attr, scrollbars))
            return(NIL);

        cleanup.sensors_made = 1;
    }

    if (NIL == wnvdsp(pwin, pwhere, view_height, view_width,
                     origin_row, origin_col, pborder))
    {
        clean_up(pwin, &cleanup);
        return(NIL);
    }
}

        /* Remember whether mouse was installed already. */
cleanup.MOCATCH_was_off = !b_mocatch;
if (!(option & WN_NO_MOUSE))
    mohide(MO_SHOW);

        /* We need to know the mode of the device the */
        /* window is displayed on so that we can determine */
        /* what character cells mouse events fall in. */
old_dev = scmode(&mode, &columns, &act_page);
old_page = b_curpage;

        /* Validate and select device and page. */
if (wnseldev(&pwin->where_shown, &pwin->view_size, &old_npage))
    wnreterr(WN_NOT_SHOWN);

```

```

semode(&mode, &columns, &act_page);
    /* Restore current page on new device; restore old */
    /* device; restore current page on old device. */
sepage(old_npage);
scchgdev(old_dev);
sepage(old_page);

row = pwin->data_origin.row;
col = pwin->data_origin.col;
    /* Main polling loop. */
do
{
    do
    {
        pevent = poll_event(pwin, columns, option);
    }
    while (pevent == NIL);

    /* Now delay for a user-specified amount of time. */
    utsleep(b_win_delay);

    /* Check to see if we recieved an unknown event. If */
    /* so, check to see if we should transmit. */
    if (pevent->event_type == WN_ABSENT_EVENT)
        if (option & WN_UNKNOWN_TRANSMIT)
        {
            if ((pfinal != NIL) && (pevent != NIL))
            {
                pfinal->event_type = pevent->event_type;
                pfinal->event      = pevent->event;
                pfinal->pdata      = malloc(pevent->data_size);
                if (pfinal->pdata == NIL)
                {
                    clean_up(pwin, &cleanup);
                    wnterror(WN_NO_MEMORY);
                    return(NIL);
                }
                pfinal->data_size = pevent->data_size;
                memmove(pfinal->pdata, pevent->pdata,
                    pevent->data_size);
            }

            clean_up(pwin, &cleanup);
            return((pfinal && pevent) ? pfinal : NIL);
        }
        else
            continue;

        switch(*((WN_ACTION *) pevent->pdata))
        {

```

```

case WN_NULL:
break;
case WN_SCROLL_UP:
    row--;
break;
case WN_SCROLL_DOWN:
    row++;
break;
case WN_SCROLL_LEFT:
    col--;
break;
case WN_SCROLL_RIGHT:
    col++;
break;
case WN_PAGE_UP:
    row -= (wnview_h(pwin) - pwin->cur_bottom_marg);
break;
case WN_PAGE_DOWN:
    row += (wnview_h(pwin) - pwin->cur_top_marg);
break;
case WN_PAGE_LEFT:
    col -= (wnview_w(pwin) - pwin->cur_right_marg);
break;
case WN_PAGE_RIGHT:
    col += (wnview_w(pwin) - pwin->cur_left_marg);
break;
case WN_TOP_EDGE:
    row = 0;
break;
case WN_BOTTOM_EDGE:
    row = wndata_h(pwin) - wnview_h(pwin);
break;
case WN_LEFT_EDGE:
    col = 0;
break;
case WN_RIGHT_EDGE:
    col = wndata_w(pwin) - wnview_w(pwin);
break;
case WN_TRANSMIT:
case WN_ABORT:
    done = 1;
break;
}

    /* Now clip the row and column to be within bounds. */
    utbound(row, 0, wndata_h(pwin) - wnview_h(pwin));
    utbound(col, 0, wndata_w(pwin) - wnview_w(pwin));

wnorigin(pwin, row, col, WN_UPDATE);
    /* Check to see if we should transmit after every */
    /* event. */

```



```

        if (option & WN_ALL_TRANSMIT)
            done = 1;
    } while (!done);
    if (!(option & WN_NO_MOUSE))
        mhide(MO_HIDE);

    if ((pfinal != NIL) && (pevent != NIL))
    {
        pfinal->event_type = pevent->event_type;
        pfinal->event      = pevent->event;
        pfinal->pdata      = malloc(pevent->data_size);
        pfinal->data_size  = pevent->data_size;
        memmove(pfinal->pdata, pevent->pdata,
                pevent->data_size);
    }

    clean_up(pwin, &cleanup);
    return((pfinal && pevent) ? pfinal : NIL);
}

```

## POLL\_EVENT - 查询窗口的鼠标或键盘事件一次。

pevent = poll\_event(pwin, columns, option)  
 \*pevent 指向找到的事件，如果失败指向NIL。  
 \*pwin 指向为事件而被检查的窗口。  
 columns 窗口显示的页的列数。  
 option 和下列值相与：

WN\_KNOWN\_EVENTS: 注意事件表中的事件。  
 WN\_ALL\_TRANSMIT: 接收每个事件。  
 WN\_UNKNOWN\_TRANSMIT: 接收未知事件。  
 WN\_KBIGNORE: 忽略所有的键盘事件。

函数接受一个指向窗口的指针并检查是否有事件发生。返回值是指向事件的指针。如果没有事件则指向NIL。如果option是WN\_UNKNOWN\_TRANSMIT，则pevent也许指向不在事件表中的事件，此时pevent->pdata是NIL。

返回 pevent 指向找到的事件，如果没有事件则指向NIL。

```

/* Screen coordinates of window's viewport. */
#define VIEW_LEFT (pwin->where_shown.corner.col)
#define VIEW_RIGHT(VIEW_LEFT + wnview_w(pwin) - 1)
#define VIEW_TOP (pwin->where_shown.corner.row)
#define VIEW_BOTTOM (VIEW_TOP + wnview_h(pwin) - 1)
/* Screen coordinates of window's border. */
#define BORD_LEFT (pwin->where_prev.corner.col)
#define BORD_RIGHT(BORD_LEFT + pwin->prev.dim.w - 1)
#define BORD_TOP (pwin->where_prev.corner.row)
#define BORD_BOTTOM (BORD_TOP + pwin->prev.dim.h - 1)
#define TICKS_PER_DAY 0x1800b0L
static const WN_EVENT *poll_event(pwin, columns, option)
BWINDOW const *pwin;
int option;
{
    static WN_EVENT unknown_event;

```

```

const WN_EVENT      *pcandidate_event;
const WN_EVENT      *pfound_event;
WN_EVENT            check_event;
unsigned            location;
WN_SENSOR           *psensor;
unsigned            vert, horiz;
int                 sensor_row, sensor_col;
int                 found = 0;
static const WN_EVENT *psave_event = NIL;
static long         hold_start;
long                current_time;
long                time_difference;

unknown_event.event_type = WN_ABSENT_EVENT;
unknown_event.pdata = NIL;
unknown_event.data_size = 0;
pcandidate_event = wnpoll(pwin->pevent_list, &vert, &horiz,
                        &unknown_event);

if (pcandidate_event == NIL)
    if ((unknown_event.event_type != WN_ABSENT_EVENT) &&
        ((option & WN_ALL_TRANSMIT) ||
         (option & WN_UNKNOWN_TRANSMIT)))
    {
        return(&unknown_event);
    }
    else
    {
        psave_event = NIL;
        return(NIL);
    }

    /* If the event was a keyboard event, return it now.*/
if (pcandidate_event->event_type == WN_KB_EVENT)
{
    if ((option & WN_KBIGNORE) && !(option & WN_ALL_TRANSMIT))
        return(NIL);
    else
        return(pcandidate_event);
}

    /* Event was a mouse event, so decide what to do */
    /* with it. Note that the number of horizontal */
    /* pixels per character depends on the number of */
    /* columns currently being displayed. */
vert >>= 3;
horiz >>= (columns == 80) ? 3 : 4;

    /* Determine if the candidate event is on a sensor. */
if (pwin->psensors != NIL)
    fo. (psensor = pwin->psensors;
        (psensor != NIL) && !found;
        psensor = psensor->pnext)

```

```

{
    switch(psensor->relative)
    {
    case WN_VIEWPORT:
        sensor_row = pwin->where_prev.corner.row;
        sensor_col = pwin->where_prev.corner.col;
        break;

    case WN_DATA:
        /* Sensor is data area relative, so check to see if it */
        /* is visible. */
        if ((psensor->hot.ul.row < pwin->data_origin.row) ||
            (psensor->hot.lr.row >
             pwin->data_origin.row + wndata_h(pwin) - 1) ||
            (psensor->hot.ul.col < pwin->data_origin.col) ||
            (psensor->hot.lr.col >
             pwin->data_origin.col + wndata_w(pwin) - 1))
        {
            continue;
        }

        sensor_row = pwin->where_shown.corner.row -
                     pwin->data_origin.row;
        sensor_col = pwin->where_shown.corner.col -
                     pwin->data_origin.col;
        break;

    case WN_SCREEN:
        sensor_row = 0;
        sensor_col = 0;
        break;
    }

    /* Now check to see if the detected event was in */
    /* the sensor. */
    if (!lurange(ver, sensor_row + psensor->hot.ul.row,
                 sensor_row + psensor->hot.lr.row) &&
        !lurange(horiz, sensor_col + psensor->hot.ul.col,
                 sensor_col + psensor->hot.lr.col))
    {
        location = psensor->type;
        found = 1;
    }

    if (psensor->pnext == pwin->psensors)
        break;
}

/* If it was not on a sensor, check its location */

```

```

        /* relative to the inside of the viewport. */
if (!found)
{
    if (!utrange(ver, VIEW_TOP, VIEW_BOTTOM) &&
        !utrange(horiz, VIEW_LEFT, VIEW_RIGHT))
    {
        location = WN_IN_WINDOW;
    }
    else
        /* If it was not in the viewport, see if it was on
        /* the border.
        if ((pwin->bord.type != BBRD_NO_BORDER) &&
            !utrange(ver, BORD_TOP, BORD_BOTTOM) &&
            !utrange(horiz, BORD_LEFT, BORD_RIGHT))
        {
            location = WN_ON_BORDER;
        }
        else
            location = WN_OUT_WINDOW;
    }
    check_event = *pcandidate_event;
    check_event.event.mouse.location = location;
    pfound_event = wnretn(pwin->pevent_list, &check_event);
        /* Check to see if the user is holding the button
        /* down on an event whose action is MO_HOLD.
    if (pfound_event && (pfound_event->event_type == WN_MOUSE_EVENT) &&
        (pfound_event->event.mouse.event & MO_HOLD))
    {
        if (pfound_event == psave_event)
        {
            /* Check to see if the user has been holding this
            /* button down for at least b_win_hold clock ticks.
            utgetclk(&current_time);
            if (current_time < hold_start)
                time_difference = (TICKS_PER_DAY - hold_start) +
                                current_time;
            else
                time_difference = current_time - hold_start;
            if (time_difference < b_win_hold)
                return(NIL);
        }
        else
        {
            utgetclk(&hold_start);
            psave_event = pfound_event;
            return(NIL);
        }
    }
    else
    {

```

```

    psave_event = NIL;
}
return(pfound_event);
}
static void clean_up(pwin, pdirt)
BWINDOW *pwin;
const CLEANUP_ITEMS *pdirt;
{
    if (pdirt->sensors_made)
        wnzapsen(pwin);
    if (pdirt->events_made)
        wnzapevn(&(pwin->pevent_list));
    if (pdirt->MOCATCH_was_off)
        mopreclk(MO_REMOVE);
    wnsetopt(pwin, WN_CUR_TRACK, pdirt->autotrack_state);
    wnsetopt(pwin, WN_CUR_OFF, pdirt->cursor_state);
}

/**
 *
 * Name          wnhckxit -- Check whether there is any way to
 *                  exit the window.
 *
 * Synopsis      ret = wnhckxit(pevent, option);
 *
 *               int      ret          Zero if there is an exit
 *                                   route (i.e. no error);
 *                                   WN_BAD_EVENT if a bad
 *                                   event entry encountered;
 *                                   WN_NO_EXIT if there is no
 *                                   exit key.
 *
 *               const WN_EVENT_LIST *pevent Pointer to the start of
 *                                   the window's event list.
 *
 *               int option          WNREAD options including
 *                                   the following bits:
 *                                   WN_KBIGNORE
 *                                   WN_ALL_TRANSMIT
 *                                   WN_UNKNOWN_TRANSMIT
 *
 * Description    WNCHKXIT checks the option value and the window's event
 *                 list to see if there is any way for the menu to be
 *                 exited.
 *
 * Returns       ret      Zero if there is an exit route (i.e. no error);
 *                       WN_BAD_EVENT if a bad event entry encountered;
 *                       WN_NO_EXIT if there is no exit event.
 *
 *               b_wnerr Possible values:
 *                       (No change) Success.
 *                       WN_NO_EXIT No way to exit the window.

```

```

*          WN_BAD_EVENT: bad event entry encountered.
**/

static int wnchkxit(pevent, option)
const WN_EVENT_LIST *pevent;
int                  option;
{
    if (option & (WN_ALL_TRANSMIT | WN_UNKNOWN_TRANSMIT))
        return(0);

    if (!(option & (WN_KBIGNORE | WN_NO_MOUSE)))
        for (; pevent != NIL; pevent = pevent->pnext)
        {
            /* Check event signature. */
            if (pevent->signature != BEVENT_SIGN)
                wnreturn(WN_BAD_EVENT);

            /* If the event is a keyboard event and option
             * includes WN_KBIGNORE, just skip the event. */
            if ((option & WN_KBIGNORE) &&
                (pevent->win_event.event_type != WN_KB_EVENT))
            {
                continue;
            }

            if (*(WN_ACTION *) pevent->win_event.pdata) &
                (WN_TRANSMIT | WN_ABORT))
                return(0);
        }

    wnreturn (WN_NO_EXIT);
}

```

## WNREDRAW 重现显示在当前显示页上的全部窗口

```

#include <bwindow.h>

int wnredraw( int dev,
              int page);

dev          屏幕显示设备(SC_COLOR (1)或SC_MONO(0))。
page         屏幕显示页。
(返回)       错误代码。如果没有错误,则为WN_NO_ERROR(0)。

```

当窗口目前被记录在内部数据结构中时, WNREDRAW重现所有当前显示在屏幕显示页上的窗口。如果已指定一个显示窗口具有活动光标, 则该光标被恢复。

指定的设备和页成为当前设备和当前页。

请记住, 可删除窗口保持屏幕内容的一个拷贝, 当显示这些内容时该拷贝即发挥作用。由于 WNREDRAW只是重新显示任何可删除窗口, 所以它不改变这些原屏幕内容的拷贝。(如果以后想删除可删除窗口, 利用这些窗口最初显示之前所显示的数据可以恢复屏幕。)

如果dev不是当前显示设备或page指定了一个不可用的显示页, 则出现一个错误。

## 源程序(WNREDRAW.C):

```
#include <bwindow.h>
```

```
int redraw (WIN_NODE *);          /* Internal function (see below). */
int wnredraw (dev, page)
int dev, page;
{
    int      result;
    WIN_NODE *pnode;
        /* Make sure dev & page are at least within range */
        /* for table lookups. */
    if (utrange (dev, 0, MAX_DEVICES))
        wnreturn (WN_BAD_DEV);
    if (utrange (page, 0, MAX_PAGES))
        wnreturn (WN_BAD_PAGE);

    result = (WN_NO_ERROR);
        /* If this page has windows, */
    if (NIL != (pnode = b_wnlist[dev][page]))
    {
        /* then redraw the windows. */
        result = redraw(pnode);

        /* If that was successful, and if a window has an */
        /* active cursor, then activate the correct cursor. */
        if ((WN_NO_ERROR == result) &&
            (NIL != (pnode = b_pactnode[dev][page])))

            if (NIL == wncursor (pnode->pwin))
                result = (b_wnerr);
    }

    return (result);
}
```

```
/**
```

```
* Name      redraw -- Redisplay all windows currently shown
*              on a display page at or below a given
*              window.
* Synopsis   result = redraw (pnode);
*              int      result  Error code, or WN_NO_ERROR if ok.
*              WIN_NODE *pnode Node of uppermost window to redisplay.
* Description This function redisplay a window currently shown on a
*              video display page (and all windows below it) as they
*              are currently recorded in internal data structures.
*              Hidden windows retain their borders and positions but
*              are not displayed.
*              The device and page corresponding to this node are made
*              current.
*              As removable windows are redisplayed, their recorded
*              copies of the previous screen contents are NOT changed.
```

```

* Returns      result      Possible values:
*
*              WN_NO_ERROR  Success.
*              WN_BAD_DEV   Internal error.
*              WN_BAD_PAGE  Internal error.
*              WN_BAD_NODE  Internal error.
*              WN_BAD_WIN   Internal error.
*              WN_COVERED   Internal error.
*              WN_NOT_SHOWN Internal error.
*              WN_ILL_DIM   Internal error.
*              WN_NULL_PTR  Internal error.
*
*              b_wncrr      Possible values:
*
*              (No change)   Success.
*              (Same as "result" if internal error.)
**/
static int redraw (pnode)
WIN_NODE *pnode;
{
    int      result;
    BWINDOW *pwin;
    if (wnvalnod (pnode) == NIL)
        wnreturn (WN_BAD_NODE);
    pwin = pnode->pwin;
    if (wnvalwin (pwin) == NIL)
        wnreturn (WN_BAD_WIN);
        /* Clean up temp. flag. */
    pwin->internals.temp_hid=0;
        /* Redraw lower windows before handling this one. */
    if (pnode->below != NIL)
        if (WN_NO_ERROR != (result = redraw(pnode->below)))
            /* Propagate error back upward. */
            return (result);
    if (!pwin->options.hidden)
    {
        /* Write border. */
        wnputbor (&pwin->view_size, &pwin->bord, &pwin->where_shown);

        /* Write data. */
        if (NIL == wnputimg (pwin))
            return (b_wncrr);
    }

        /* Output is now up-to-date. */
    pwin->internals.dirty = 0;

    return (WN_NO_ERROR);
}

```

## WNREMEVN 删去WNREAD接受的一个用户响应

```

#include <bwindow.h>
int wnremevn( WN_EVENT_LIST **ppfirst,
              const WN_EVENT *pevent);

```



ppfirst 指针的地址, 该指针指向用户响应的第一项。  
 pevent 待从表中删除的WN\_EVENT结构的地址。  
 (返回) 错误代码。可能值包括:  
     WN\_NO\_ERROR 成功。  
     WN\_NO\_EVENT (16) 表中未发现对应的事件。

WNREMEVN 从WNREAD接受的响应表中删去一项。如果删去的项是表中唯一的项, 则WNREMEVN将\*ppfirst设置为NIL。

见WNREAD中有关WN\_EVENT 数据类型的说明。

WNREMEVN不修改全局窗口错误代码b\_wncrr。

### 源程序(WNREMEVN.C):

```
#include <bwindow.h>
int wnremevn(pplist_head, pevent)
WN_EVENT_LIST **pplist_head;
const WN_EVENT *pevent;
{
    WN_EVENT *pfound_event;
    WN_EVENT_LIST *pfound_node;

    pfound_event = wnretev(*pplist_head, pevent);
    if (pfound_event == NIL)
        /* The event is not in the window event list. */
        return(WN_NO_EVENT);
    /* Locate the window event list node corresponding to the */
    /* window event given. */
    for (pfound_node = *pplist_head;
        (pfound_node != NIL) &&
        (&pfound_node->win_event != pfound_event);
        (pfound_node = pfound_node->pnext))
        ;
    /* If the member is the first and only element of the list */
    /* just make *pplist_head point to NIL. Otherwise, */
    /* surgically remove it from the event list by altering the */
    /* pnext and pprev pointers of the nodes surrounding it. */
    if ((*pplist_head == pfound_node) &&
        (((*pplist_head)->pnext == *pplist_head) ||
        ((*pplist_head)->pnext == NIL)))
    {
        *pplist_head = NIL;
    }
    else
    {
        if (pfound_node->pprev != NIL)
            pfound_node->pprev->pnext = pfound_node->pnext;
        if (pfound_node->pnext != NIL)
            pfound_node->pnext->pprev = pfound_node->pprev;
    }
    pfound_node->pnext = pfound_node->pprev = NIL;

    pfound_node->signature = BEVENT_DEAD;
}
```

```

    if (pfound_event->pdata != NIL)
        free(pfound_event->pdata);
    free(pfound_node);
    return(WN_NO_ERROR);
}

```

## WNREMOVE 从屏幕上取消一个窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnremove(BWINDOW *pwindow);
```

pwindow 待取消的BWINDOW结构的地址。

(返回) 取消后的BWINDOW结构的地址。若失败，返回NIL。

WNREMOVE取消一个当前显示的窗口，恢复屏幕的原内容。

如果窗口光标不是活动的，则WNREMOVE 关闭光标，使窗口光标失效。此外，如果程序已同一个NW\_???OBJ文件链接，那么字符窗口属性虽然未被改变或恢复，Turbo C字符窗口仍被设置成全屏幕。

如果pwindow未指向一个合法的窗口结构，窗口已被指定为“不可删除”或窗口当前未显示，这时出现一个错误。

源程序(WNREMOVE.C);

```
#include <bwindow.h>
```

```
BWINDOW *wnremove (pwindow)
```

```
BWINDOW *pwindow;
```

```

{
    wvalidw (pwindow)
        /* Ensure it's showing. */
    if (pwindow->where_shown.dev != SC_COLOR &&
        pwindow->where_shown.dev != SC_MONO)
        wnreterr (WN_NOT_SHOWN);

        /* Remove from physical screen. */
    if (NIL == wnhide (pwindow))
        return (NIL);

        /* Remove from linked list. */
    if (NIL == wnpgrem (pwindow))
        return (NIL);

        /* Mark as not shown. */
    pwindow->where_shown.dev =
    pwindow->where_prev.dev = ABSENT;
    pwindow->options.hidden = 0;

        /* If this window is current, mark no window current*/
    if (b_pcurwin == pwindow)
        b_pcurwin = NIL;
    return (pwindow);
}

```

## WNRESRVR 恢复窗口先前的屏幕内容。

```
presult = wnresprv (pwin);
```

\*presult 指向恢复的BWINDOW结构，如果失败则指向NIL。

\*pwin 指向要恢复其先前屏幕内容的BWINDOW结构。

该函数输出指定窗口先前屏幕图象(也就是说在显示窗口之前的屏幕图象),有效地恢复成原来屏幕的状态。

该函数还选择窗口的where\_prev域指定的设备和显示页。

如果pwin->prev指向的IMAGE结构不包指向数据缓冲区的指针;或者如果where\_prev表示的设备和显示页无效;或者如果pwin->prev.dim指定的尺寸和pwin->where\_prev.corner相对于显示设备来说是不正确的,则产生错误。

返回	presult	指向输出的BWINDOW结构,如果失败则为NIL。
	b_device	视频设备
	b_curpage	当前显示页。
	b_werr	可能值如下:
		(不改变) 成功
	WN_NULL_PTR	pwin->prev.pdata无效
	WN_BAD_DEV	无效的设备、显示或尺寸。
	WN_ILL_DIM	内部错误

源程序(WNRESRVC.C):

```
#include <bvideo.h>
#include <bwindow.h>
#include <bmouse.h>
#define BORDER_TOP (pwin->where_prev.corner.row)
#define BORDER_LEFT (pwin->where_prev.corner.col)
#define BORDER_BOTTOM(BORDER_TOP + wnbord_h(pwin) - 1)
#define BORDER_RIGHT (BORDER_LEFT + wnbord_w(pwin) - 1)
BWINDOW *wnresprv(pwin)
BWINDOW *pwin;
{
    int old_npage;
    int num_written;

    /* Check window for validity. */
    wnvalidw(pwin);

    /* Validate and select device and and page (also
    /* validate dimensions.
    if (wnseldev (&pwin->where_prev, &pwin->prev.dim, &old_npage))
        wnreterr (WN_BAD_DEV);
    /* Temporarily keep the mouse cursor out of the
    /* window.
    mohide(MO_HIDE);
    /* Write the image.
    num_written = viwrect(BORDER_TOP, BORDER_LEFT,
        BORDER_BOTTOM, BORDER_RIGHT,
        (char *) pwin->prev.pdata,
        0, 0, CHAR_ATTR);
    /* Re-enable the mouse cursor.
    mohide(MO_SHOW);
    if (num_written != wnbord_h(pwin) * wnbord_w(pwin))
        wnreterr (WN_ILL_DIM);
    return (pwin);
}
```

WNRETEVN 从窗口列表中返回一窗口事件记录。

presult = wnretevn(plist\_head, pevent)

\*presult           指向定位的事件记录，如果失败则指向NIL。

\*plist\_head        指向检查的事件记录一的头。

\*pevent            返回的事件。

该函数查找指定的事件表，匹配WN\_EVENT给出的记录、指定的击键或鼠标事件。如果找到，返回指向它的指针。如果未匹配则返回NIL。

返回            WN\_EVENT \*presult 指向窗口事件列表中匹配的窗口事件记录。

源程序(WNRETEVN.C):

```
#include <butil.h>
```

```
#include <bwindow.h>
```

```
WN_EVENT *wnretevn(plist_head, pevent)
```

```
WN_EVENT_LIST *plist_head;
```

```
const WN_EVENT *pevent;
```

```
{
    WN_EVENT_LIST *pcurrent;
    int found = 0;
    /* Exit immediately if we have no event list or no
    /* event.
    if ((plist_head == NIL) || (pevent == NIL))
    {
        return(NIL);
    }
    pcurrent = plist_head;
    /* Do a sequential search of the linked list until the
    /* event is found, or the list is exhausted.
    do
    {
        wnvalide(pcurrent);
        if (pevent->event_type == WN_KB_EVENT)
        {
            if ((pcurrent->win_event.event_type == WN_KB_EVENT) &&
                (pcurrent->win_event.event.keystroke.character_code ==
                 pevent->event.keystroke.character_code) &&
                (pcurrent->win_event.event.keystroke.key_code ==
                 pevent->event.keystroke.key_code))
            {
                found = 1;
            }
        }
        else
        {
            if ((pcurrent->win_event.event_type == WN_MOUSE_EVENT) &&
                (pcurrent->win_event.event.mouse.event ==
                 pevent->event.mouse.event) &&
                (pcurrent->win_event.event.mouse.ignore ==
                 pevent->event.mouse.ignore) &&
                (pcurrent->win_event.event.mouse.location ==
                 pevent->event.mouse.location))
            {

```

```

        found = 1;
    }
}
if (!found)
    pcurrent = pcurrent->pnext;

} while ((pcurrent != NIL) && (pcurrent != plist_head) && !found);

/* If the event was found, pcurrent points to it. */
if (found)
    return(&pcurrent->win_event);
else
    return(NIL);
}

```

### WNRETINF - 返回窗口当前光标和尺寸的信息。

```

error = edretinf( pwin, pinfo, pfield_loc, field_height,
                 field_width);
error
    返回的错误代码。
* pwin
    指向其信息被返回的窗口。
* pinfo
    指向填充的结构。
* pfield_loc
    指向正确的域位置。
field_height
    在窗口显示的域高
field_width
    在窗口显示的域宽

```

WNRETINF返回有一编辑域被显示的窗口的光标和尺寸。信息放在pinfo指向的区域。指定的窗口成为当前窗口。

返回	error	可能值如下:
		WN_NO_ERROR 没有错误发生
		WN_BAD_WIN- 窗口结构是坏的
		WN_ILL_DIM - 对窗口 来说域太大
		WN_NOT_SHOWN - 窗口是不显示的
		WN_COVERED 数据区被覆盖
	b_wncrr	如果无错则不变, 否则和error同值。

源程序(WNRETINF.C):

```

#include <bcreens.h>
#include <bedit.h>
int wnretinf(pwin, pinfo, pfield_loc, field_height, field_width)
BWINDOW *pwin;
CUR_INFO *pinfo;
const LOC *pfield_loc;
int field_height;
int field_width;
{
    if (wnvalwin(pwin) == NIL)
        wnreturn(WN_BAD_WIN);

    /* Make sure it's visible. */
    if (pwin->where_shown.dev != SC_MONO &&
        pwin->where_shown.dev != SC_COLOR)
    {

```

```

        wnreturn(WN_NOT_SHOWN);
    }

    /* Make sure we can write to it.
    if (pwin->internals.frozen)
        wnreturn(WN_COVERED);

    pinfo->off    = pwin->options.cur_off;
    pinfo->row     = pwin->cur_loc.row;
    pinfo->col     = pwin->cur_loc.col;
    pinfo->size    = pwin->cur_type;
    if (pwin != wnselect(pwin))
        return(b_wnerr);
    if (utrange(pfield_loc->row, 0, wndata_h(pwin) - field_height) ||
        utrange(pfield_loc->col, 0, wndata_w(pwin) - field_width))
    {
        wnreturn(WN_ILL_DIM);
    }
    return(WN_NO_ERROR);
}

```

## WNREVUPD 用显示的数据更新已保存的窗口图象

```
#include <bwindow.h>
```

```
BWINDOW *wnrevupd(void);
```

(返回) 当前BWINDOW结构的地址，若失败，返回NIL。

WNREVUPD用当前显示在屏幕上的数据更新当前窗口数据区的已保存的拷贝，这对于Turbo TOOLS窗口与通过Turbo C 字符窗口或通过其它手段显示的数据保持同步是很有用的。

如果程序与一个NW???.OBJ(“原始窗口”)文件相链接，则窗口的缺省属性被设置为与Turbo C字符窗口的缺省性相匹配。

如果没有一个窗口被指定为“当前”窗口，当前窗口未显示或窗口的视口被另一个窗口遮盖，则出现一个错误；如果程序与一个nw\_???obj文件相链接而视口尺寸与当前Turbo C字符窗口不相适合，这时也出现一个错误。只要出现这些错误，b\_wnerr就保存一个错误代码，NIL作为函数值返回。

源程序(WNREVUPD.C):

```
#include <conio.h>
```

```
#include <bwindow.h>
```

```
#include <bvideo.h>
```

```
#include <bmouse.h>
```

```
static int read_image(BWINDOW *);
```

```
BWINDOW *wnrevupd()
```

```

{
    int old_dev, old_page;
    int mode, columns, act_page;
    int cursor_x, cursor_y;
    int cursor_high, cursor_low;

```

```
        /* Validate window data structures.

```

```
        */

```

```
    wnvalidw (b_pcurwin)

```

```

        /* Quit if window is hidden or covered. */
if (b_pcurwin->options.hidden ||
    (b_pcurwin->where_shown.dev != SC_COLOR) &&
    (b_pcurwin->where_shown.dev != SC_MONO))
{
    wnretterr(WN_NOT_SHOWN);
}
if (b_pcurwin->internals.any_data_covered)
    wnretterr(WN_COVERED);
    /* Make sure the native text window matches current */
    /* window. */
if (!wnchkdm(b_pcurwin))
    wnretterr(WN_ILL_DIM);
    /* Note former device, current page on former device*/
old_dev = semode (&mode, &columns, &act_page);
old_page = b_curpage;

if (read_image(b_pcurwin) != 0)
{
    /* Restore old device; restore current page on old */
    /* device. */
    sechgdev (old_dev);
    sepage (old_page);
    wnretterr(b_wnerr);
}

    /* Mark window as being clean. */
b_pcurwin->internals.dirty = 0;
    /* Check to see if cursor is in the current window. */
securst(&cursor_y, &cursor_x, &cursor_high, &cursor_low);
    /* Adjust coordinates to be viewport relative. */
cursor_y -= b_pcurwin->where_shown.corner.row;
cursor_x -= b_pcurwin->where_shown.corner.col;
    /* Check to see if cursor is inside viewport. */
if (!utrange(cursor_x, 0, wnview_w(b_pcurwin) - 1) &&
    !utrange(cursor_y, 0, wnview_h(b_pcurwin) - 1))
{
    /* Synchronize the cursor position & size. */
    b_pcurwin->cur_loc.col = cursor_x +
        b_pcurwin->data_origin.col;
    b_pcurwin->cur_loc.row = cursor_y +
        b_pcurwin->data_origin.row;
    b_pcurwin->cur_type.high = cursor_high;
    b_pcurwin->cur_type.low = cursor_low;
}
wngetatr(b_pcurwin);
    /* Restore old device; restore current page on old */
    /* device. */
    sechgdev (old_dev);
    sepage (old_page);
    return (b_pcurwin);

```

```

}
/* Screen coordinates of window's viewport. */
#define VIEW_LEFT (pwin->where_shown.corner.col)
#define VIEW_RIGHT(pwin->where_shown.corner.col + wnview_w(pwin) - 1)
#define VIEW_TOP (pwin->where_shown.corner.row)
#define VIEW_BOTTOM (pwin->where_shown.corner.row + wnview_h(pwin) - 1)
/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (pwin->data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(pwin) - 1)
#define VISIBLE_TOP (pwin->data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(pwin) - 1)

/**
 *
 * Name READ_IMAGE - Read the specified window's image from the
 * screen.
 *
 * Synopsis error = read_image(pwin);
 *
 * int error 0 if no errors occur, 1 otherwise.
 * BWINDOW *pwin Pointer to the window to read.
 *
 * Description The specified window's image is read from the screen
 * into the window's data buffer. Only the area inside
 * the window's viewport is updated.
 *
 * Returns error 0 if no errors occur, 1 otherwise.
 */
static int read_image(pwin)
BWINDOW *pwin;
{
    int num_read;
    int error = 0;
    CELL *pto;

    /* Temporarily keep the mouse cursor out of the */
    /* window. */
    mhide(MO_HIDE);
    /* Write out each line-block of information. */
    pto = pwin->img.pdata +
        ((VISIBLE_TOP * wdata_w(pwin)) + VISIBLE_LEFT);
    num_read = virdsect(VIEW_TOP, VIEW_LEFT, VIEW_BOTTOM, VIEW_RIGHT,
        (char *) pto, wdata_w(pwin) - wview_w(pwin),
        CHAR_ATTR);
    if (num_read != (VIEW_BOTTOM - VIEW_TOP + 1) *
        (VIEW_RIGHT - VIEW_LEFT + 1))
        error = 1;
    /* Re-enable the mouse cursor. */
    mhide(MO_SHOW);

```



```

    return(error);
}

```

## WNSCRBLK 在窗口中以任意方向滚动一个矩形区域

```
#include <bwindow.h>
```

```

BWINDOW *wnscrblk(BWINDOW *pwin,
                  int r1,int c1,
                  int r2, int c2,
                  int fore, int back,
                  int dir,
                  int count,
                  int option);

```

pwin 待改变的BWINDOW结构的地址。

r1,c1 块左上角的行和列(相对于窗口数据区左上角(0,0))。

r2,c2 块右下角的行和列。

fore,back 新前景和背景属性(如果使用缺省值,则为-1)。

dir 滚动方向(WNSCR\_UP (0), WNSCR\_DOWN(1), WNSCR\_RIGHT (2) 或WNSCR\_LEFT (3))。

count 待滚动的行数或列数。值0清除该矩形区域。

option 下列值之一:

符号	值	意义
WN_UPDATE	0	除非窗口是“延迟”的,否则更新窗口。
WN_NO_UPDATE	4	不更新窗口。

(返回) 滚动后的BWINDOW结构的地址。若失败,返回NIL。

WNSCRBLK 在窗口中上下左右移动指定矩形区域的字符行或字符列(连同其属性),空行或空列用空格和指定属性填充。

count 值为零则指示清除窗口中的这一块。

WNSCRBLK 修改窗口数据区的一部分而不涉及视口中显示的内容,它也不移动窗口光标。用 WNORIGIN可以在视口中显示窗口数据区的各个部分。

源程序(WNSCRBLK.C):

```
#include <bwindow.h>
```

```
#define CELLSIZE (sizeof (CELL))
```

```
#define H (wndata_h(pwin)) /* Height and width of data area */
```

```
#define W (wndata_w(pwin))
```

```
#define HR ((r2 - r1) + 1) /* Height and width of region. */
```

```
#define WR ((c2 - c1) + 1)
```

```
BWINDOW *wnscrblk (pwin, r1, c1, r2, c2, fore, back, dir, count, option)
```

```
BWINDOW *pwin;
```

```
int r1, c1, r2, c2, fore, back, dir, count, option;
```

```
{
```

```
    int attr, r, i;
```

```
    int rt = r1, rb = r2, cl = c1, cr = c2;
```

```
    unsigned int amt;
```

```
    CELL *pto, *pfrom;
```

```
    wnvalidw (pwin)
```

```
        /* Check corners for in window, upper-left is up */
```

```
        /* and to left of lower-right, etc. */
```

```
    if ((c1 > c2) || (r1 > r2) ||
```

```

    utrange (r1, 0, H - 1) || utrange (r2, 0, H - 1) ||
    utrange (c1, 0, W - 1) || utrange (c2, 0, W - 1))
    wnretterr (WN_ILL_DIM);
    attr = wndoattr (pwin->attr, fore, back);
        /* If request would scroll the region completely, */
        /* set up to clear region. */
if (((dir == WNSCR_UP || dir == WNSCR_DOWN) &&
    utrange (count, 1, HR - 1) ||
    ((dir == WNSCR_LEFT || dir == WNSCR_RIGHT) &&
    utrange (count, 1, WR - 1)))
    count = 0;
else
    switch (dir)
    {
        case WNSCR_UP:
            amt = ((c2 - c1) + 1) * CELLSIZE;
            pto = pwin->img.pdata + (W * r1) + c1;
            pfrom = pwin->img.pdata + (W * (r1 + count)) + c1;
            for (r = r1;
                r <= r2 - count;
                r++, pto += W, pfrom += W)
                utmovmem ((char far *) pfrom,
                    (char far *) pto,
                    amt);
            rt = (r2 - count) + 1;
            break;
        case WNSCR_DOWN:
            amt = ((c2 - c1) + 1) * CELLSIZE;
            pto = pwin->img.pdata + (W * r2) + c1;
            pfrom = pwin->img.pdata + (W * (r2 - count)) + c1;
            for (r = r2;
                r >= r1 + count;
                r--, pto -= W, pfrom -= W)
                utmovmem ((char far *) pfrom,
                    (char far *) pto,
                    amt);
            rb = (r1 + count) - 1;
            break;
        case WNSCR_LEFT:
            amt = (((c2 - c1) - count) + 1) * CELLSIZE;
            pto = pwin->img.pdata + (W * r1) + c1;
            pfrom = pwin->img.pdata + (W * r1) + c1 + count;
            for (r = r1;
                r <= r2;
                r++, pto += W, pfrom += W)
                utmovmem ((char far *) pfrom,
                    (char far *) pto,
                    amt);
            cl = (c2 - count) + 1;
            break;
    }

```

```

case WNSCR_RIGHT:
    amt = (((c2 - c1) - count) + 1) * CELLSIZE;
    pto = pwin->img.pdata + (W * r1) + c1 + count;
    pfrom = pwin->img.pdata + (W * r1) + c1;
    for (r = r1;
        r <= r2;
        r++, pto += W, pfrom += W)
        utmovmem ((char far *) pfrom,
                  (char far *) pto,
                  amt);
    cr = (c1 + count) - 1;
    break;
}

/* Clear the block left over (or the whole block */
/* requested, if count was zero). */
pto = pwin->img.pdata + (W * rt) + cl;

for (r = rt;
    r <= rb;
    r++, pto += W)
    for (i = 0; i < (cr - cl) + 1; i++)
    {
        pto[i].ch = ' ';
        pto[i].attr = (char) attr;
    }

/* Update real screen if possible and requested. */
return (wnnupblk (pwin, r1, c1, r2, c2, option));
}

```

## WNSCRLBR 向窗口加入一个滚动箭头

```
#include <bwindow.h>
```

```
BWINDOW *wnscrlbr(BWINDOW *pwin,
                  int view_ht, int view_wid,
                  unsigned attr,
                  int option);
```

pwin 待修改的BWINDOW结构的地址。

view\_ht, view\_wid

视口的行数和列数。

attr 滚动箭头的属性(颜色)。

option 可用位运算符OR( )组合起来的两个相关位:

WN\_HORIZONTAL (1) 在视口边界下方的左右滚动箭头。

WN\_VERTICAL (2) 在视口边界右沿的上下滚动箭头。

(返回) 修改后的窗口的地址。若失败, 则返回NIL。

WNSCRLBR在窗口中安装水平和/或垂直滚动箭头。

WNSCRLBR 不显示滚动箭头, 所以应该在显示窗口之前调用WNSCRLBR。

不要对显示时没有边界(即边界类型为BBRD\_NO\_BORDER(0))的窗口使用WNSCRLBR。

要修改滚动箭头的位置和外观, 可修改源模块wnscrlbr.c。

源程序(WNSCRLBR.C):

```

#include <bwindow.h>
static void free_list(WN_SENSOR *);

/* The following are character constants to use */
/* when constructing the four scroll arrows. */

#define LEFT_ARROW_CHAR '\33'
#define RIGHT_ARROW_CHAR '\32'
#define UP_ARROW_CHAR '\30'
#define DOWN_ARROW_CHAR '\31'

BWINDOW *wnscribr(pwin, view_height, view_width, attr, option)
BWINDOW *pwin;
int view_height;
int view_width;
unsigned attr;
int option;
{
    WN_SENSOR *pcurrent = NIL;
    WN_SENSOR *phead = NIL;
    WN_SENSOR *ptemp;

    /* First, make sure the window is valid. */
    wnvalidw(pwin);
    if (option & WN_HORIZONTAL)
    {
        /* Construct the horizontal scrollbars. */
        pcurrent = (WN_SENSOR *) calloc(1, sizeof(WN_SENSOR));
        if (pcurrent == NIL)
            wnreterr(WN_NO_MEMORY);
        phead = pcurrent;
        pcurrent->type = WN_LEFT_ARROW;
        pcurrent->relative = WN_VIEWPORT;
        pcurrent->hot.ul.row = view_height + 1;
        pcurrent->hot.lr.row = view_height + 1;
        pcurrent->hot.ul.col = 1;
        pcurrent->hot.lr.col = 1;
        pcurrent->corner.row = pcurrent->hot.ul.row;
        pcurrent->corner.col = pcurrent->hot.ul.col;
        pcurrent->size.w = 1;
        pcurrent->size.h = 1;
        pcurrent->pimage = (CELL *) calloc(1, sizeof(CELL));
        if (pcurrent->pimage == NIL)
        {
            free_list(phead);
            wnzapscn(pwin);
            wnreterr(WN_NO_MEMORY);
        }
        pcurrent->pimage->ch = LEFT_ARROW_CHAR;
        pcurrent->pimage->attr = attr;

        pcurrent->pnext = (WN_SENSOR *) calloc(1, sizeof(WN_SENSOR));
        if (pcurrent->pnext == NIL)
        {

```

```

    free_list(phead);
    wnzapsen(pwin);
    wnreterr(WN_NO_MEMORY);
}
pcurrent->pnext->pprev = pcurrent;
pcurrent
    = pcurrent->pnext;
pcurrent->type
    = WN_RIGHT_ARROW;
pcurrent->relative
    = WN_VIEWPORT;
pcurrent->hot.ul.row
    = view_height + 1;
pcurrent->hot.lr.row
    = view_height + 1;
pcurrent->hot.ul.col
    = view_width;
pcurrent->hot.lr.col
    = view_width;
pcurrent->corner.row
    = pcurrent->hot.ul.row;
pcurrent->corner.col
    = pcurrent->hot.ul.col;
pcurrent->size.w
    = 1;
pcurrent->size.h
    = 1;
pcurrent->pimage
    = (CELL *) calloc(1, sizeof(CELL));
if (pcurrent->pimage == NIL)
{
    free_list(phead);
    wnzapsen(pwin);
    wnreterr(WN_NO_MEMORY);
}
pcurrent->pimage->ch = RIGHT_ARROW_CHAR;
pcurrent->pimage->attr = attr;
pcurrent->pnext = NIL;
}

if (option & WN_VERTICAL)
{
    /* Construct the vertical scrollbars. */
    if (pcurrent == NIL)
    {
        pcurrent = (WN_SENSOR *) calloc(1, sizeof(WN_SENSOR));
        if (pcurrent == NIL)
        {
            free_list(phead);
            wnzapsen(pwin);
            wnreterr(WN_NO_MEMORY);
        }
        phead = pcurrent;
    }
    else
    {
        pcurrent->pnext = (WN_SENSOR *) calloc(1, sizeof(WN_SENSOR));
        if (pcurrent->pnext == NIL)
        {
            free_list(phead);
            wnzapsen(pwin);
            wnreterr(WN_NO_MEMORY);
        }
    }
}

```

```

    }
    pcurrent->nnext->pprev = pcurrent;
    pcurrent
        = pcurrent->nnext;
}
pcurrent->type          = WN_UP_ARROW;
pcurrent->relative      = WN_VIEWPORT;
pcurrent->hot.ul.row    = 1;
pcurrent->hot.lr.row    = 1;
pcurrent->hot.ul.col    = view_width + 1;
pcurrent->hot.lr.col    = view_width + 1;
pcurrent->corner.row    = pcurrent->hot.ul.row;
pcurrent->corner.col    = pcurrent->hot.ul.col;
pcurrent->size.w        = 1;
pcurrent->size.h        = 1;
pcurrent->pimage        = (CELL *) calloc(1, sizeof(CELL));
if (pcurrent->pimage == NIL)
{
    free_list(phead);
    wnzapsen(pwin);
    wnreterr(WN_NO_MEMORY);
}
pcurrent->pimage->ch    = UP_ARROW_CHAR;
pcurrent->pimage->attr = attr;

pcurrent->nnext = (WN_SENSOR *) calloc(1, sizeof(WN_SENSOR));
if (pcurrent->nnext == NIL)
{
    free_list(phead);
    wnzapsen(pwin);
    wnreterr(WN_NO_MEMORY);
}
pcurrent->nnext->pprev = pcurrent;
pcurrent
    = pcurrent->nnext;
pcurrent->type          = WN_DOWN_ARROW;
pcurrent->relative      = WN_VIEWPORT;
pcurrent->hot.ul.row    = view_height;
pcurrent->hot.lr.row    = view_height;
pcurrent->hot.ul.col    = view_width + 1;
pcurrent->hot.lr.col    = view_width + 1;
pcurrent->corner.row    = pcurrent->hot.ul.row;
pcurrent->corner.col    = pcurrent->hot.ul.col;
pcurrent->size.w        = 1;
pcurrent->size.h        = 1;
pcurrent->pimage        = (CELL *) calloc(1, sizeof(CELL));
if (pcurrent->pimage == NIL)
{
    free_list(phead);
    wnzapsen(pwin);
    wnreterr(WN_NO_MEMORY);
}

```

```

    pcurrent->pimage->ch    = DOWN_ARROW_CHAR;
    pcurrent->pimage->attr = attr;
    pcurrent->pnext = NIL;
}

    /* Now attach the scrollbar(s) to the window. if */
    /* window already has sensors, add them to the list.*/
if (phead != NIL)
    if (pwin->psensors == NIL)
    {
        pwin->psensors = phead;
        phead->pprev = NIL;
    }
    else
    {
        for (ptemp = pwin->psensors; ptemp->pnext != NIL;
            ptemp = ptemp->pnext)
        ;

        ptemp->pnext = phead;
        phead->pprev = ptemp;
    }
return(pwin);
}

/**
 * Name      FREE_LIST - Free a window sensor list.
 * Synopsis  free_list(plist);
 *           WN_SENSOR *plist Pointer to the sensor list to free.
 * Description This function accepts a pointer to a window sensor list
 *           and frees each element in the list.
 * Returns   nothing.
 **/
static void free_list(plist)
WN_SENSOR *plist;
{
    WN_SENSOR *ptemp;
    while(plist != NIL)
    {
        ptemp = plist;
        plist = plist->pnext;
        if (ptemp->pimage != NIL)
            free(ptemp->pimage);
        ptemp->pnext = NIL;
        ptemp->pprev = NIL;
        free(ptemp);
    }
    return;
}

```

## WNSCROLL 垂直滚动当前窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnscroll( int num_rows,  
                   int fore,int back,  
                   int dir);
```

num\_rows 滚动的行数。值0清除该窗口。

fore 新空行的前景属性。(-1指定窗口的缺省前景颜色。)

back 新空行的背景属性。(-1指定窗口的缺省背景颜色。)

dir 滚动方向(SCR\_UP\* (0)或SCR\_DOWN(1))。

(返回) 滚动后的BWINDOW结构的地址。若失败，返回NIL。

WNSCROLL在当前窗口内上下移动字符行(连同其属性)，空行用空格和指定属性填充。

WNSCROLL修改窗口的整个数据区而不涉及视口中显示的内容，它也不移动窗口的光标。用WNORIGHIN可以在视口中显示窗口数据区的各个部分。

如果不存在当前窗口，则出现一个错误。

用WNSCRBLK可以在窗口中滚动一个矩形区域。

源程序(WNSCROLL.C);

```
#include <bwindow.h>
```

```
#define H (wndata_h(b_pcurwin))
```

```
#define W (wndata_w(b_pcurwin))
```

```
BWINDOW *wnscroll (num_rows, fore, back, dir)
```

```
int num_rows, fore, back, dir;
```

```
{
```

```
    /* Validate window data structures. */
```

```
    wnvalidw (b_pcurwin)
```

```
    /* Force num_rows to be 0 (clear window) if not in */
```

```
    /* range. */
```

```
    if (utrange (num_rows, 1, H))
```

```
        num_rows = 0;
```

```
    if (dir == SCR_UP)
```

```
        return (wnscriblk (b_pcurwin,
```

```
                           0, 0,
```

```
                           H - 1, W - 1,
```

```
                           fore, back, WNSCR_UP, num_rows, WN_UPDATE));
```

```
    return (wnscriblk (b_pcurwin,
```

```
                      0, 0,
```

```
                      H - 1, W - 1,
```

```
                      fore, back, WNSCR_DOWN, num_rows, WN_UPDATE));
```

```
}
```

## WNSELECT 选择用于I/O的窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnselect(BWINDOW *pwin);
```

pwin 用于I/O的窗口。

(返回) 新选择的BWINDOW 结构的地址。若失败，返回NIL。

WNSELECT将一个窗口指定为“当前的”，以后的窗口I/O请求被定向到这个窗口。全局变量



b pcurwin保存pwin的拷贝。

窗口不必是当前显示的。如果窗口是当前显示的，它的设备和页便成为当前屏幕和当前页(用于Turbo C TOOLS的屏幕I/O)。

窗口光标未被激活。该显示页上的另一个窗口可以通过WNCURSOR取得活动光标。

如果窗口光标是活动的并且程序与NW\_???.OBJ链接，则Turbo C字符窗口被设置为与窗口的视口相匹配。

如果pwin未指向一个合法的窗口结构，则出现一个错误。

源程序(WNSELECT.C):

```
#include <conio.h>
#include <bwindow.h>
BWINDOW *b_pcurwin = NIL; /* Window selected for I/O. */
BWINDOW *wnselect (pwin)
BWINDOW *pwin;
{
    int old_npage;
    /* Validate window data structures. */
    wnvalidw (pwin)
    /* Select and validate device, page, and dimensions.*/
    if ((pwin->where_shown.dev == SC_MONO) ||
        (pwin->where_shown.dev == SC_COLOR))
    {
        if (wnseldev (&pwin->where_shown, &pwin->view_size, &old_npage))
            wnreterr (WN_NOT_SHOWN);
        /* Set the native text window to match the window's */
        /* viewport. */
        if (!pwin->options.hidden &&
            !pwin->internals.frozen &&
            !pwin->internals.any_data_covered)
        {
            wnsetwin(pwin->where_shown.corner.col,
                    pwin->where_shown.corner.row,
                    pwin->where_shown.corner.col + wnview_w(pwin) - 1,
                    pwin->where_shown.corner.row + wnview_h(pwin) - 1);
            wnsetatr(pwin->attr);
        }
    }
    return (b_pcurwin = pwin);
}
```

## WNSETBUF 为WNPRINTF分配内部缓冲区

```
#include <bwindow.h>
```

```
unsigned wnsetbuf(unsigned size);
```

size 缓冲区的新尺寸(如果要释放该缓冲区，则为零)。

(返回) 分配后的缓冲区的尺寸。如果没有足够的内存可用，则为零。

WNSETBUF分配WNPRINTF所用的缓冲区，将它保存在全局变量b\_wnprbf中。该全局变量在bwindow.h中声明。

若调用WNPRINTF时尚未分配缓冲区，则WNPRINTF为缓冲区分配B\_WNPRBF\_SIZE (1024)字节。

将WNSSETBUF 用于一般目的是没有必要的。如果想控制缓冲区的尺寸或想在特定时间分配缓冲区，比如在中止运行程序并保持驻留之前或在分配大量内存之前分配缓冲区，这个函数是很有用的。可以多次调用WNSSETBUF来释放缓冲区或调整缓冲区的尺寸。如果出现错误，零将作为函数值返回。然而WNSSETBUF不调用WNERORR,不改变b\_wnerr。

源程序(WNSSETBUF.C);

```
#include <stdio.h>
#include <bwindow.h>
unsigned wnsetbuf (size)
unsigned size;
{
    /* If requested size is same as existing size, OK. */
    if (size == b_wnbfsz)
        return (size);
    /* Free the buffer if it exists. */
    if (b_wnprbf != NIL)
        free (b_wnprbf);
    /* Allocate a new buffer -- register a window error */
    /* if there is a problem. */
    if ((size != 0) &&
        ((b_wnprbf = calloc (size, sizeof (char))) == NULL))
        wnreterz (WN_NO_MEMORY);
    /* Return requested size. */
    return (b_wnbfsz = size);
}
```

**WNSETCUR** 设置当前窗口的状态、尺寸和位置。

error = wnsetcur(pwin, off, pcursor, row, col);

error 返回的错误代码。

\*pwin 指向光标被移动的窗口。

off 光标的新状态(0 = on, 1 = off).

\*pcursor 指向包含光标尺寸的结构。

row,col 相对于窗口左上角的新光标位置的行和列。

移动指定的光标，设置其大小为\*pcursor所指的值。在窗口图象被更新前看不见修改。

返回	error	返回的错误代码:
		WN_NO_ERROR成功
		WN_BAD_WIN- 坏的窗口指针
		WN_ILL_VALUE - 光标尺寸无效。
		WN_ILL_DIM - 光标位置无效。
	b_wnerr	如果没有错误不改变，否则与error同值。

源程序(WNSETCUR.C);

```
#include <bwindow.h>
int wnsetcur(pwin, off, pcursor, row, col)
BWINDOW *pwin;
int off;
CUR_TYPE *pcursor;
int row;
int col;
{
    if (wnvalwin(pwin) == NIL)
```

```

    wnerror(WN_BAD_WIN);
pwin->cur_type = *pcursor;
if ((pwin != wnsetopt(pwin, WN_CUR_HIGH, pcursor->high)) ||
    (pwin != wnsetopt(pwin, WN_CUR_LOW, pcursor->low )) ||
    (pwin != wnsetopt(pwin, WN_CUR_OFF, off)))
{
    return(b_wnerr);
}
if (utrange(row, 0, wndata_h(pwin) - 1) ||
    utrange(col, 0, wndata_w(pwin) - 1))
{
    wnerror(WN_ILL_DIM);
}
pwin->cur_loc.row = row;
pwin->cur_loc.col = col;
return(WN_NO_ERROR);
}

```

## WNSETOPT 设置窗口控制项

```
#include <bwindow.h>
```

```
BWINDOW *wnsetopt(BWINDOW *pwin,
                  int item,
                  int value);
```

pwin 窗口地址。  
 item 控制项的标识代码号。  
 value 新数据值。  
 返回) 窗口地址。若失败，则返回NIL。

WNSETOPT用下表的值修改窗口的某个特性：

项	符号项	值	意义
WN_CUER_OFF	6	1	关闭光标。0 打开光标。
WN_CUR_HIGH	7	0-13	光标高扫描行。
WN_CUR_LOW	8	0-13	光标低扫描行。
WN_ATTR	20	0-255	设置缺省前景和背景属性(颜色)。
WN_REMOVABLE	10	1	如果可能，使之可删除。
		0	使之不可删除，废A原屏幕数据。
WN_DELAYED	9	1	延迟输出，直至WNUPDATE。
		0	允许立即输出。
WN_PREV_ALLOC	31	1	为原屏幕数据分配缓冲区。
		0	释放原屏幕数据缓冲区。
WN_CUR_TRACK	34	1	打开自动光标跟踪。
		0	关闭自动光标跟踪。
自动跟踪时使用的间1：			
WN_CUR_T_MARGIN	7	0-49	在上方的行数。
WN_CUR_B_MARGIN	8	0-49	在下方的行数。
WN_CUR_L_MARGIN	5	0-79	在左方的行数。
WN_CUR_R_MARGIN	6	0-79	在右方的行数。

WNSETOP 设置内部的控制项值但不屏作任何可见的操作。这样，当通过设置控制项 WN\_DELAYED为零使窗口立即输出时，一定要调用WNUPDATE 使任何挂起的输出写到屏幕上。类似地，在调用WNUPDATE之前对光标尺寸或自动光标跟踪状态的修改可能是不可见的。

如果所需项未列在表中, 该项不能修改或pwin未指向一个合法窗口, 这时出现一个错误。

源程序(WNSETOPT.C):

```
#include <bwindow.h>
#define FALSE 0
#define TRUE (!FALSE)
    /* Macro to build a case statement which tests the new value */
    /* and performs the assignment if the value meets the specified */
    /* condition. */
#define easycase(item_num,cond,data,calc)
    case (item_num):
        if (cond)
        {
            (data) = (calc);
            presult = pwin;
        }
        else
        {
            wnerror(WN_ILL_VALUE);
            presult = NIL;
        }
        break;
    /* Macro to build a case which tests and assigns a value for a */
    /* bitfield. */
#define bitcase(item_num,data,calc)
    easycase(item_num,(value==0||value==1),data,calc)
    /* Macro to build a case which assigns the new value without */
    /* testing its validity. */
#define anycase(item_num,data,calc)
    case (item_num):
        (data) = (calc);
        presult = pwin;
        break;
BWINDOW *wnsetopt (pwin, item, value)
BWINDOW *pwin;
int item, value;
{
    BWINDOW *presult;
    /* Validate window data structure. */
    wnvalidw (pwin);
    /* This kind of item is meaningless unless window */
    /* displayed. */
    if (item < 0)
        if (pwin->where_shown.dev != SC_MONO &&
            pwin->where_shown.dev != SC_COLOR)
            wnreterr (WN_NOT_SHOWN);
    switch (item)
    {
```

```

/* Cursor on/off state */
bitcase( WN_CUR_OFF, pwin->options.cur_off,
         value)
/* Cursor scan lines */
easycase( WN_CUR_HIGH, (value >= 0 && value <= 13),
         pwin->cur_type.high,
         value)
easycase( WN_CUR_LOW, (value >= 0 && value <= 13),
         pwin->cur_type.low,
         value)

/* User-controllable options */
bitcase( WN_DELAYED, pwin->options.delayed,
         value)

case WN_REMOVABLE:
    if (value == 0)
    {
        /* Making non-removable. */
        if (pwin->options.removable)
        {
            /* Discard copy of previous
             * screen data.
             */
            if (pwin->prev.pdata != NIL)
            {
                free((char *) pwin->prev.pdata);
                pwin->prev.pdata = NIL;
            }
            pwin->options.removable = value;
        }
        presult = pwin;
    }
    else if (value == 1)
    {
        /* Making removable. */
        if ((!pwin->options.removable) &&
            (!pwin->options.hidden) &&
            (pwin->where_shown.dev == SC_MONO ||
             pwin->where_shown.dev == SC_COLOR))
        {
            /* Too late: already displayed */
            /* and non-removable.
             */
            wnerror(WN_ILL_VALUE);
            presult = NIL;
        }
        else
        {
            /* Easy to make it removable.
             */
            pwin->options.removable = value;
            presult = pwin;
        }
    }
    else
    {
        wnerror(WN_ILL_VALUE);
    }

```

```

        presult = NIL;
    }
    break;

/* Default attributes */
anycase( WN_ATTR, pwin->attr,
        value)

/* Cursor auto-tracking on/off */
bitcase( WN_CUR_TRACK, pwin->options.cur_track,
        value)

/* Auto-track margins */
anycase( WN_CUR_L_MARG, pwin->cur_left_marg, value);
anycase( WN_CUR_R_MARG, pwin->cur_right_marg, value);
anycase( WN_CUR_T_MARG, pwin->cur_top_marg, value);
anycase( WN_CUR_B_MARG, pwin->cur_bottom_marg, value);
case WN_PREV_ALLOC:
    presult = pwin;
    switch (value)
    {
        case 1:
            /* User wants to pre-allocate the memory */
            /* for the previous screen contents. */
            /* If the memory is already allocated, quit */
            /* with success. */
            if (pwin->prev.pdata != NIL)
                ;
            else
            {
                /* We cannot do it if the window is already */
                /* displayed. */
                if (!(pwin->options.hidden) &&
                    (pwin->where_shown.dev == SC_MONO ||
                     pwin->where_shown.dev == SC_COLOR))
                    wnreterr (WN_ILL_VALUE);
                /* Attempt to allocate, return failure if */
                /* call to CALLOC fails. */
                if ((pwin->prev.pdata =
                    (CELL *) calloc ((unsigned int)
                                     ((wndata_h(pwin) + 2) *
                                      (wndata_w(pwin) + 2)),
                                     sizeof (CELL))) == NIL)
                    wnreterr (WN_NO_MEMORY);
            }
            break;
        case 0:
            /* User wants to de-allocate the memory */
            /* for the previous screen contents. */

            /* If the memory is not already allocated, */
            /* quit with success. */
            if (pwin->prev.pdata == NIL)

```

```

else
{ /* We cannot deallocate if the window is      */
  /* being shown and it is a removable window.*/
  if ((pwin->options.removable)      &&
      (!pwin->options.hidden)        &&
      (pwin->where_shown.dev == SC_MONO ||
       pwin->where_shown.dev == SC_COLOR))
    wnretterr (WN_ILL_VALUE);

    /* Do the deallocation.                      */
  else
  {
    free ((char *) pwin->prev.pdata);
    pwin->prev.pdata = NIL;
  }
  break;
default:
  wnretterr (WN_ILL_VALUE);
}
break;
case WN_HIDDEN:
case WN_ROW_REL:
case WN_COL_REL:
case WN_ROW_ABS:
case WN_COL_ABS:
case WN_IS_CURRENT:
case WN_ACTIVE_CUR:
case WN_BOR_TYPE:
case WN_BOR_CHAR:
case WN_BOR_ATTR:
case WN_DEVICE:
case WN_PAGE:
case WN_ROW_CORNER:
case WN_COL_CORNER:
case WN_FROZEN:
case WN_DIRTY:
case WN_ANY_DATA_COVERED:
case WN_ROWS:
case WN_COLS:
case WN_ROW_OVERALL:
case WN_COL_OVERALL:
case WN_HT_OVERALL:
case WN_WID_OVERALL:
case WN_HT_VIEW:
case WN_WID_VIEW:
  wnerror(WN_HARD);
  presult = NIL;

```

```

        break;

    default:
        wnerror(WN_BAD_OPT);    /* Unknown item requested.    */
        presult = NIL;
        break;
    }
    return (presult);
}

```

## WNSHOBK 在视口间隙中显示一个窗口数据块

```

#include <bwindow.h>
BWINDOW *wnshobk(BWINDOW *pwin,
                 const REGION *pblock,
                 const LOC *phot1, const LOC *phot2,
                 int option);

pwin      待移动的窗口的地址。
pblock    窗口中待显示块的两个对角(见下面)。
phot1, phot2 期望显示的窗口中的两个位置(见下面)。
option    WN_UPDATE      (0) 立即更新视口;
          WN_NO_UPDATE (4) 仅在内部记录变化。
(返回)    BWINDOW 结构的地址。若失败, 返回NIL。

```

WNSHOBK 在视口中移动窗口, 尽可能多地显示给定的窗口数据区矩形块。它对列在\*phot1和\*phot2中的两个“热点”给予了优先权, 即在试图将矩形块移入视口后, WNSHOBK迫使\*phot2接着是\*hot1进入视口。这样, 即使矩形块不能适合于视口中, 这两个热点也有可能显示; \*phot1必然能够显示(除非自动跟踪使之失效)。

通过建立REGION结构可以指明待显示的块, 这个结构在(bwindow.h中定义如下, 其中所有的行列值均相对应至于窗口左数据区上角(0,0):

```

typedef struct          /*LOC 结构:*/
{
    /*相对于一个矩形区域的行和列。*/
    int row,col;
}LOC;

typedef struct          /*REGION结构:*/
{
    /*屏幕上的矩形区域。*/
    LOC ul;             /*左上角。*/
    LOC lr;             /*右下角*/
} REGION;

```

用LOC结构指明热点。如果两个热点中有一个值为(-1,-1), 则该热点即为块的左上角。

WNSHOBK 试图将块和两个热点装入当前视口间1中(如果有的话)。它对块和热点进行调整, 使之能够一齐装到视口中。非零间1会产生迫使块和热点倾向于视口某一段的效果, 但间1不能迫使矩形块或热点离开视口。

如是自动跟踪有效, 它将对把块和热点装入视口的尝试产生影响, 因此, 使用WNSHOBK时有可能需要关闭自动跟踪。

源程序(WNSHOBK.C):

```

#include <bwindow.h>

/* Relative coordinates of window's visible data area.    */
#define VISIBLE_LEFT (data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(pwin) - 1)

```



```

#define VISIBLE_TOP (data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wview_h(pwin) - 1)
    /* Relative coordinates of area inside window's margins. */
#define MARGIN_LEFT (VISIBLE_LEFT + pwin->cur_left_marg)
#define MARGIN_RIGHT (VISIBLE_RIGHT - pwin->cur_right_marg)
#define MARGIN_TOP (VISIBLE_TOP + pwin->cur_top_marg)
#define MARGIN_BOTTOM (VISIBLE_BOTTOM - pwin->cur_bottom_marg)
static void shift_win(BWINDOW *, LOC *,
                      int, int, int, int, const REGION *);
BWINDOW *wnshoblk(pwin, pblock, phot1, phot2, option)
BWINDOW *pwin;
const REGION *pblock;
const LOC *phot1;
const LOC *phot2;
int option;
{
    REGION hot1_region;
    REGION hot2_region;
    LOC data_origin;

    wnvalidw (pwin) /* Validate window structure. */

    /* Now set up REGION structures to keep track of
       the two hot spots. */

    hot1_region.ul.row = (phot1->row == -1) ? pblock->ul.row
                                           : phot1->row;
    hot1_region.ul.col = (phot1->col == -1) ? pblock->ul.col
                                           : phot1->col;
    hot1_region.lr.row = (phot1->row == -1) ? pblock->ul.row
                                           : phot1->row;
    hot1_region.lr.col = (phot1->col == -1) ? pblock->ul.col
                                           : phot1->col;

    hot2_region.ul.row = (phot2->row == -1) ? pblock->ul.row
                                           : phot2->row;
    hot2_region.ul.col = (phot2->col == -1) ? pblock->ul.col
                                           : phot2->col;
    hot2_region.lr.row = (phot2->row == -1) ? pblock->ul.row
                                           : phot2->row;
    hot2_region.lr.col = (phot2->col == -1) ? pblock->ul.col
                                           : phot2->col;

    /* Confirm that the block area coordinates and the
       "hot spots" make sense. */
    if ((pblock->ul.row > pblock->lr.row) ||
        (pblock->ul.col > pblock->lr.col) ||
        (pblock->ul.row < 0) ||
        (pblock->lr.row > (wndata_h(pwin) - 1)) ||
        (pblock->ul.col < 0) ||
        (pblock->lr.col > (wndata_w(pwin) - 1)))

```

```

    utrange(hot1_region.ul.row, pblock->ul.row, pblock->lr.row) ||
    utrange(hot1_region.ul.col, pblock->ul.col, pblock->lr.col) ||
    utrange(hot2_region.ul.row, pblock->ul.row, pblock->lr.row) ||
    utrange(hot2_region.ul.col, pblock->ul.col, pblock->lr.col))
{
    wnretterr(WN_ILL_DIM);
}
data_origin = pwin->data_origin;
    /* First, shift the window so that as much of the      */
    /* block as possible fits in the margins.                */
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    pblock);
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    &hot2_region);
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    &hot1_region);
    /* Now shift it in case any of the block is still      */
    /* outside the viewport.                                */
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    pblock);
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    &hot2_region);
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    &hot1_region);

    /* Check to see if the origin has changed.              */
if ((data_origin.col == pwin->data_origin.col) &&
    (data_origin.row == pwin->data_origin.row) &&
    !pwin->internals.dirty)
{
    return(pwin);
}
pwin->data_origin = data_origin;
    /* Update real screen if possible and requested.        */
return(wnnupblk(pwin, 0, 0, wndata_h(pwin) - 1,
    wndata_w(pwin) - 1, option));
}

/**
 * Name      shift_win - Shift a window's data area to ensure that
 *            a block is showing.
 * Synopsis  shift_win (pwin, pdata_origin,
 *            bound_top, bound_left, bound_right,
 *            bound_bottom, pblock);

```

```

*          BWINDOW *pwin  The window whose data area is to be
>                          shifted.
*          LOC *pdata_origin Location of viewport data area
*                          within window data area, modified
*                          if necessary.
*          int bound_top   The bounding rectangle in which the
*          int bound_left  block should show.
*          int bound_bottom
*          int bound_right
*          REGION *pblock  The bounding rectangle of the desired
*                          block.
**/
void shift_win(pwin, pdata_origin, bound_top, bound_left,
              bound_bottom, bound_right, pblock)
BWINDOW      *pwin;
LOC          *pdata_origin;
int          bound_top;
int          bound_left;
int          bound_bottom;
int          bound_right;
const REGION *pblock;
{
    if (pblock->lr.row > bound_bottom)
        pdata_origin->row += pblock->lr.row - bound_bottom;
    if (pblock->lr.col > bound_right)
        pdata_origin->col += pblock->lr.col - bound_right;
    if (pblock->ul.row < bound_top)
        pdata_origin->row -= bound_top - pblock->ul.row;
    if (pblock->ul.col < bound_left)
        pdata_origin->col -= bound_left - pblock->ul.col;

    /* Now force the corner coordinates to be in bounds.*/
    utbound(pdata_origin->row, 0, wdata_h(pwin) - wview_h(pwin));
    utbound(pdata_origin->col, 0, wdata_w(pwin) - wview_w(pwin));
}

```

## WNSHOBK 在窗口视区内显示一长方形区域的数据。

presult = wns Hobk (pwin, pblock, phot1, phot2, option);  
 \*presult 指向其块被显示的窗口, 如果失败则指向NIL。  
 \*pwin 指向数据区被移动的窗口。  
 \*pblock 要显示的数据区的区域的角。  
 \*phot1, \*phot2 两“热点”, u在块内。如果整个块不能被显示, 设法显示这些点。其一的行或列的值-1可能被\*pblock的值替换。

int option 如下值之一:  
           WN\_UPDATE 立即更新窗口的屏幕图象。  
           WN\_NO\_UPDATE 不立即更新窗口的屏幕图象。

该函数 设置出现在视区的(0,0)的数据区的坐标。因此由pblock指定的区域尽可能地显示。首先设法将块放在窗口内, 然后如果区域太小的话设法放在视区内。如果块不能放在视区内, 指定的热点放进指定的块; 如果视区太小不能容纳两者的话, phot1比phot2优先。

返回 presult 返回窗口, 如果失败则为NIL。

*pwin	修改几个域
b_wncrr	可能值如下:
	(不改变)      成功
WN_BAD_WIN	*pwin无效
WN_NOT_SHOWN	pwin不被显示。
WN_ILL_DIM	pblock的尺寸无效, 或其中一个热点无效。

源程序(WNSHOBK.C):

```
#include <bwindow.h>

/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(pwin) - 1)
#define VISIBLE_TOP (data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(pwin) - 1)

/* Relative coordinates of area inside window's margins. */
#define MARGIN_LEFT (VISIBLE_LEFT + pwin->cur_left_marg)
#define MARGIN_RIGHT (VISIBLE_RIGHT - pwin->cur_right_marg)
#define MARGIN_TOP (VISIBLE_TOP + pwin->cur_top_marg)
#define MARGIN_BOTTOM (VISIBLE_BOTTOM - pwin->cur_bottom_marg)

static void shift_win(BWINDOW *, LOC *,
                     int, int, int, int, const REGION *);

BWINDOW *wnshobk(pwin, pblock, phot1, phot2, option)
BWINDOW *pwin;
const REGION *pblock;
const LOC *phot1;
const LOC *phot2;
int option;
{
    REGION hot1_region;
    REGION hot2_region;
    LOC data_origin;

    wnvalidw (pwin) /* Validate window structure. */
    /* Now set up REGION structures to keep track of
    /* the two hot spots.
    hot1_region.ul.row = (phot1->row == -1) ? pblock->ul.row
                        : phot1->row;
    hot1_region.ul.col = (phot1->col == -1) ? pblock->ul.col
                        : phot1->col;
    hot1_region.lr.row = (phot1->row == -1) ? pblock->ul.row
                        : phot1->row;
    hot1_region.lr.col = (phot1->col == -1) ? pblock->ul.col
                        : phot1->col;
    hot2_region.ul.row = (phot2->row == -1) ? pblock->ul.row
                        : phot2->row;
    hot2_region.ul.col = (phot2->col == -1) ? pblock->ul.col
                        : phot2->col;
    hot2_region.lr.row = (phot2->row == -1) ? pblock->ul.row
                        : phot2->row;
    hot2_region.lr.col = (phot2->col == -1) ? pblock->ul.col
                        : phot2->col;
```

```

        /* Confirm that the block area coordinates and the      */
        /* "hot spots" make sense.                                */
if ((pblock->ul.row > pblock->lr.row) ||
    (pblock->ul.col > pblock->lr.col) ||
    (pblock->ul.row < 0) ||
    (pblock->lr.row > (wndata_h(pwin) - 1)) ||
    (pblock->ul.col < 0) ||
    (pblock->lr.col > (wndata_w(pwin) - 1)) ||
    utrange(hot1_region.ul.row, pblock->ul.row, pblock->lr.row) ||
    utrange(hot1_region.ul.col, pblock->ul.col, pblock->lr.col) ||
    utrange(hot2_region.ul.row, pblock->ul.row, pblock->lr.row) ||
    utrange(hot2_region.ul.col, pblock->ul.col, pblock->lr.col))
{
    wnreterr(WN_ILL_DIM);
}
data_origin = pwin->data_origin;
    /* First, shift the window so that as much of the          */
    /* block as possible fits in the margins.                    */
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    pblock);
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    &hot2_region);
shift_win(pwin, &data_origin,
    MARGIN_TOP, MARGIN_LEFT, MARGIN_BOTTOM, MARGIN_RIGHT,
    &hot1_region);
    /* Now shift it in case any of the block is still          */
    /* outside the viewport.                                     */
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    pblock);
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    &hot2_region);
shift_win(pwin, &data_origin,
    VISIBLE_TOP, VISIBLE_LEFT, VISIBLE_BOTTOM, VISIBLE_RIGHT,
    &hot1_region);
    /* Check to see if the origin has changed.                  */
if ((data_origin.col == pwin->data_origin.col) &&
    (data_origin.row == pwin->data_origin.row) &&
    !pwin->internals.dirty)
{
    return(pwin);
}
pwin->data_origin = data_origin;
    /* Update real screen if possible and requested.            */
return(wnnupblk(pwin, 0, 0, wndata_h(pwin) - 1,
    wndata_w(pwin) - 1, option));
}

```

```

/**
 * Name      shift_win - Shift a window's data area to ensure that
 *            a block is showing.
 * Synopsis  shift_win (pwin, pdata_origin,
 *            bound_top, bound_left, bound_right,
 *            bound_bottom, pblock);
 *
 *            BWINDOW *pwin  The window whose data area is to be
 *            shifted.
 *
 *            LOC *pdata_origin Location of viewport data area
 *            within window data area, modified
 *            if necessary.
 *
 *            int bound_top    The bounding rectangle in which the
 *            int bound_left   block should show.
 *            int bound_bottom
 *            int bound_right
 *
 *            REGION *pblock  The bounding rectangle of the desired
 *            block.
 */
void shift_win(pwin, pdata_origin, bound_top, bound_left,
              bound_bottom, bound_right, pblock)
BWINDOW *pwin;
LOC *pdata_origin;
int bound_top;
int bound_left;
int bound_bottom;
int bound_right;
const REGION *pblock;
{
    if (pblock->lr.row > bound_bottom)
        pdata_origin->row += pblock->lr.row - bound_bottom;
    if (pblock->lr.col > bound_right)
        pdata_origin->col += pblock->lr.col - bound_right;
    if (pblock->ul.row < bound_top)
        pdata_origin->row -= bound_top - pblock->ul.row;
    if (pblock->ul.col < bound_left)
        pdata_origin->col -= bound_left - pblock->ul.col;

    /* Now force the corner coordinates to be in bounds.*/
    utbound(pdata_origin->row, 0, wdata_h(pwin) - wnview_h(pwin));
    utbound(pdata_origin->col, 0, wdata_w(pwin) - wnview_w(pwin));
}

```

## WNUNHIDE 重新显示隐藏的窗口。

presult = wnunhide (pwin);

\*presult 指向刚重新显示的BWINDOW结构，如果失败则指向NIL。

\*pwin 指向重新的BWINDOW结构。

该重新以前显示过但被隐藏的窗口。窗口位置和边界和以前一样。

如果窗口是当前显示和可见的则没有影响。

窗口的光标是不活动的。如果窗口不是当前的则窗口不被激活。

如果窗口覆盖活动窗口的一部分数据区,那么窗口的光标被挂起,屏幕的光标不可见。

如果pwin不指向一有效的窗口结构,如果窗口是不可删除的,如果窗口不与显示设备和显示页相联,或者如果由于屏幕尺寸修改(比改变了模式)而导致窗口在屏幕上放不下,则产生错误。

在大多数错误的事件中,函数使窗口不显示,因此可能用WINDSPY重新显示。

返回	presult	指向刚隐藏的BIWINDOW结构,如果失败则指向NIL。
	*pwin	修改了几个域。
	b_pactnode[]	有活动光标窗口节点。
	b_wncrr	值可能如下:
		(不改变)      成功
	WN_BAD_WIN	*pwin无效
	WN_NOT_SHOWN	不与设备与显示页相联。
	WN_NO_MEMORY	内存不够
	WN_BAD_NODE	内部错误
	WN_BAD_DEV	内部错误
	WN_BAD_PAGE	内部错误
	WN_COVERED	内部错误
	WN_ILL_DIM	内部错误
	WN_NULL_PTR	内部错误

源程序(WNUNHIDE.C):

```
#include <bwindow.h>
```

```
#define restore_video() \
    (scpage (old_npage), \
     scchgdev (old_dev), \
     (void) scpage (old_page))
```

```
BWINDOW *wnunhide (pwin)
BWINDOW *pwin;
{
    int mode, columns, act_page;
    int old_dev, old_page, old_npage;
        /* Validate window data structures. */
    wnvalidw (pwin)
        /* Note current page on former device; note former */
        /* device. */
    old_page = b_curpage;
    old_dev = scmode (&mode, &columns, &act_page);
        /* Validate and select device and page. */
    if (wnseldev (&pwin->where_shown, &pwin->view_size, &old_npage))
        wnreterr (WN_NOT_SHOWN);
        /* Quit if not hidden -- not an error. */
    if (!pwin->options.hidden)
    {
        restore_video ();
        return (pwin);
    }
        /* Remove existing node. */
    if (wnpgrem (pwin) == NIL)
    {
```

```

    pwin->where_shown.dev = ABSENT;
    restore_video ();
    return (NIL);
}

    /* Add this window to linked list for display page. */
if ((pwin->pnode == wnpadd (pwin,
    pwin->where_shown.dev,
    pwin->where_shown.page)) == NIL)
{
    pwin->where_shown.dev = ABSENT;
    restore_video ();
    return (NIL);
}

    /* If it is a removable window, and we do not */
    /* already have a buffer for previous contents, */
    /* allocate a buffer; read previous screen contents.*/
if (pwin->options.removable)
{
    if (pwin->prev.pdata == NIL)
        if ((pwin->prev.pdata =
            (CELL *) calloc ((unsigned int)
                ((wnbord_h(pwin)) *
                (wnbord_w(pwin))),
                sizeof (CELL))) == NIL)
            wnreterr (WN_NO_MEMORY);

    if (wngetimg (&pwin->prev, &pwin->where_prev) == NIL)
    {
        wnpgrm(pwin);      /* Clean up after failure. */
        pwin->where_shown.dev = ABSENT;
        restore_video ();
        return (NIL);
    }
}

pwin->options.hidden          = 0; /* Displayed, not hidden. */
pwin->internals.frozen        = 0; /* Not covered. */
pwin->internals.dirty         = 1; /* Need to write data. */
pwin->internals.any_data_covered = 0; /* Not covered. */
pwin->internals.cur_frozen    = 1; /* Inactive cursor. */
    /* Write the data. */
if (wnupblk(pwin, 0, 0,
    wndata_h(pwin) - 1, wndata_w(pwin) - 1,
    WN_UPDATE) == NIL)
{
    wnpgrm (pwin);
    pwin->where_shown.dev = ABSENT;
    restore_video ();
    return (NIL);
}

```



```

        /* Write the border. */
wnputhor (&pwin->view_size, &pwin->bord, &pwin->where_shown);
        /* Write the sensor items. */
wnputsen(pwin);
        /* If other windows are shown on this page, then
        /* mark them if covered. */
if (pwin->pnode->below != NIL)
{
    if (wncover (pwin->pnode->below,
                &pwin->where_prev.corner,
                &pwin->prev.dim) == NIL)
    {
        wnpgrem (pwin);
        pwin->where_shown.dev = ABSENT;
        restore_video ();
        return (NIL);
    }
}
restore_video();
return (pwin);
}

```

## WNUPDATE 将挂起的输出写入窗口

#include <bwindow.h>

BWINDOW \*wnupdate(BWINDOW \*pwindow);

pwindow 待更新BWINDOW结构的地址。

(返回) 更新后的BWINDOW结构的地址。若失败，返回NIL。

如果原来有等待，WNUPDATE 将任何挂起的输出请求写入指定窗口。如果窗口光标是活动的并且光标跟踪有效，WNUPDATE还更新窗口(如果窗口数据区的某一部分被另一个窗口遮盖，则不做任何操作。)

这个函数并不指定一个窗口为立即窗口”，用WNSTOPT可以指定一个窗口是“延迟的”或“立即的”。

如果pwindow未指向一个合法的窗口结构或窗口当前未显示，则出现一个错误。

源程序(WNUPDATE.C):

#include <conio.h>

#include <bwindow.h>

BWINDOW \*wnupdate(pwin)

BWINDOW \*pwin;

```

{
    int old_dev, old_page, old_npage;
    int mode, columns, act_page;
    BWINDOW *presult;
    unsigned save_delayed_flag;
        /* Validate window data structures. */
wnvalidw(pwin)
        /* Quit if invisible, if already up-to-date, or if
        /* frozen. */
if (pwin->options.hidden ||

```

```

        (!pwin->internals.dirty) ||
        pwin->internals.frozen      ||
        pwin->internals.any_data_covered)
    {
        return(pwin);
    }

    /* Note former device, current page on former device*/
    old_dev  = scmode(&mode, &columns, &act_page);
    old_page = b_curpage;

    /* Validate and select device and page. */
    if (wnseldev(&pwin->where_shown, &pwin->view_size, &old_npage))
        wnreterr(WN_NOT_SHOWN);
    save_delayed_flag      = pwin->options.delayed;
    pwin->options.delayed = 0;
    presult = wnputimg(pwin);
    pwin->options.delayed = save_delayed_flag;

    /* Restore current page on new device; restore old */
    /* device; restore current page on old device. */
    sepage(old_npage);
    sechgdev(old_dev);
    sepage(old_page);
    return(presult);
}

```

## WNVALEVO -- 验证窗口事件。

presult = wnvalev0(pevent\_list, signature);

\*presult           指向有效的WN\_EVENT\_LIST结构，如果失败则为NIL。

\*pnode            指向检查的WN\_EVENT\_LIST结构。

signature         调用函数要求的标签字。

函数从不面上检查窗口事件结构的有效性并返回结果。

函数由mwvalevn()宏调用，后者用来提供标签字。相同的值由wnnritev()提供给WNINITE0。通过使用这些宏，所有的窗口提供了比较的标签值。用此方法可检测BWINDOW.H的不兼容版本。

返回            presult           指向有效的WN\_EVENT\_LIST结构，如果失败则指向NIL。

源程序(WNVALEVO.C):

```

#include <bwindow.h>
WN_EVENT_LIST *wnvalev0(pnode, signature)
WN_EVENT_LIST *pnode;
unsigned signature;
{
    if (pnode == NIL ||
        /* Make sure it's not NIL. */
        /* Check signature. */
        signature != pnode->signature)
        return (NIL);

    return (pnode);
}

```

## WNVALNO0 -- 验证窗口节点。

presult = wnvalno0(pnode, signature);

\*presult       指向有效的WIN\_NODE结构, 如果失败则指向NIL。

\*pnode         指向检查的WIND\_NODE结构。

signature       调用函数要求的标签字。

该函数表面上检查一窗口节点结构的有效长返回结果。

函数由wnvalnod()宏调用, 后者用来提供标签字。相同的值由wncreate()提供给WNcreat0。通过使用这些宏, 所有的窗口提供了比较的标签值。用此方法可检测BWINDOW.H的不兼容版本。

返回           presult       指向有效的WIN\_NODE结构, 如果失败则指向NIL。

源程序(WNVALNO0.C):

```
#include <bwindow.h>
```

```
WIN_NODE *wnvalno0(pnode,signature)
```

```
WIN_NODE *pnode;
```

```
unsigned signature;
```

```
{
```

```
    if (pnode == NIL ||                   /* Make sure it's not NIL.       */
```

```
                                        /* Check signature.           */
```

```
        signature != pnode->signature)
```

```
        return (NIL);
```

```
    return (pnode);
```

```
}
```

## WNVALWI0 - 验证BIWINDOW结构。

```
presult = wnvalwi0(pwindow,signature);
```

\*presult       指向有效的WIN\_NODE结构, 如果失败则指向NIL。

\*pnode         指向检查的WIND\_NODE结构。

signature       调用函数要求的标签字。

该函数表面上检查一窗口节点结构的有效长返回结果。

函数由wnvalwen()宏调用, 后者用来提供标签字。相同的值由wncreate()提供给WNcreat0。通过使用这些宏, 所有的窗口提供了比较的标签值。用此方法可检测BWINDOW.H的不兼容版本。

返回           presult       指向有效的WIN\_NODE结构, 如果失败则指向NIL。

源程序(WNVALWI0.C):

```
#include <bwindow.h>
```

```
BWINDOW *wnvalwi0(pwindow,signature)
```

```
BWINDOW *pwindow;
```

```
unsigned signature;
```

```
{
```

```
    if (pwindow == NIL || signature != pwindow->signature)
```

```
        return (NIL);
```

```
    return pwindow;
```

```
}
```

## WNVDISP       在视口中显示一个虚拟窗口

```
#include <bwindow.h>
```

```
BWINDOW *wnvdisp(BWINDOW *pwin,
```

```
                  const WHERE *pwhere,
```

```
                  int view_ht,int view_wid,
```

int org\_row, int org\_col;  
const BORDER \*pbord);

pwin 待显示的BWINDOW 结构的地址。

pwhere WHERE结构的地址, 该结构指明设备、显示页和窗口显示位置的坐标。

view\_ht, view\_wid 视口的行数和列数。

org\_row, org\_col 出现在视口左上角的窗口数据区的行和列(相对于(0,0))。

pbord BORDER结构的地址, 该结构指示放置在视口周围的边界和标题。

(返回) 显示后的BWINDOW 结构的地址。若失败, 返回NIL。

WNDISP在视口中显示一个虚拟窗口并加上一个边界(边界可以是“无边界”类型或包含上/下标题。)视口不能比窗口数据区大, 但可以小于它。该窗口成为当前窗口(用于I/O), 窗口光标被激活(虽然光标可能是不可见的), 该页上其它窗口的光标被关闭。如果程序与NW\_???OBJ文件链接, 则Turbo C 字符窗口被设置为与视口相匹配。

要说明视口位置及边界和标题, 可以象在WNDSPRAY 中说明的那样建立pwhere和\*pbord结构。(WNDSPRAY与WNVDISP相同, 但WNDSPRAY使用与窗口同尺寸的视口)。如果视口数据区与屏幕边缘5 接(即无处容纳边界), 该函数就不画出边界。

org\_row或org\_col将被调整使得视口中显示的区域不会延伸到窗口数据区之外。

如果pwin未指向一个合法的窗口结构或视口超出窗口数据区及屏幕尺寸, 这时 出现一个错误; 如果窗口的显示位置不可能, 也会出现一个错误。例如请求了未知的设备。

源程序(WNVDISP.C):

#include <bwindow.h>

```

/* Internal function (see below). */
static REGION *bordim(REGION *, DIM *, const BORDER *, const WHERE *);
BWINDOW *wnvdisp (pwin, pwhere, view_height, view_width,
                   origin_row, origin_col, pborder)
BWINDOW *pwin;
const WHERE *pwhere;
int view_height;
int view_width;
int origin_row;
int origin_col;
const BORDER *pborder;
{
    REGION occupied; /* Region occupied by window and */
                    /* its border. */
    int old_npage;
    wnvalidw (pwin) /* Validate window structure. */

    /* Confirm that window is not currently shown. */
    if ((pwin->where_shown.dev == SC_MONO) ||
        (pwin->where_shown.dev == SC_COLOR))
    {
        wnreterr (WN_ALREADY_SHOWN);
    }

    /* Confirm that data area dimensions are not */
    /* smaller than the viewport, and that the initial */
    /* data area coordinates make sense. */
    if ((wndata_w(pwin) < 0) || (wndata_h(pwin) < 0) ||

```

```

    utrange(view_width, 0, wdata_w(pwin)) ||
    utrange(view_height, 0, wdata_h(pwin)) ||
    utrange(origin_col, 0, wdata_w(pwin) - view_width) ||
    utrange(origin_row, 0, wdata_h(pwin) - view_height))
{
    wnretcrr(WN_ILL_DIM);
}

    /* Set the dimensions of the viewport and the */
    /* data area origin coordinates. */
pwin->view_size.w = view_width;
pwin->view_size.h = view_height;
pwin->data_origin.row = origin_row;
pwin->data_origin.col = origin_col;
    /* Validate device, page, and dimensions and select */
    /* device and page. */
if (wnseldev (pwhere, &pwin->view_size, &old_npage))
    wnretcrr (WN_BAD_DEV);
    /* Obtain overall dimensions including the border. */
bordim (&occupied, &pwin->view_size, pborder, pwhere);
utcopy (&pwin->where_prev, pwhere, WHERE);
utcopy (&pwin->where_prev.corner, &occupied.ul, LOC);
pwin->prev.dim.h = REGION_H(occupied);
pwin->prev.dim.w = REGION_W(occupied);

utcopy (&pwin->where_shown, pwhere, WHERE); /* Record where */
utcopy (&pwin->bord, pborder, BORDER); /* shown and border.*/
/* Display the window by */
/* 1) adding this window to linked list for this display page*/
/* 2) marking it hidden; */
/* 3) un-hiding it. */
if (wnpgadd (pwin, pwhere->dev, pwhere->page) == NIL)
{
    pwin->where_shown.dev = ABSENT;
    return (NIL);
}
pwin->options.hidden = 1;
if (wnunhide (pwin) == NIL)
{
    if (pwin->where_shown.dev != ABSENT)
    {
        /* Clean up after error. */
        wnpgrm (pwin);
        pwin->where_shown.dev = ABSENT;
    }
    return (NIL);
}

    /* Select this window for I/O and cursor. */
wnselect (pwin);
wncursor (pwin);
return (pwin);
}

```

```

/**
 *
 * Name      bordim -- Calculate actual dimensions required by a
 *            border around a rectangular region
 *            on the current display device.
 * Synopsis  presult = bordim (preion, pdim, pborder, pwhere);
 *            REGION    *presult  Pointer to structure describing
 *                                location and dimensions of the
 *                                region to be occupied by the
 *                                border.
 *            REGION    *preion   Pointer to structure describing
 *                                location and dimensions of the
 *                                region to be occupied by the
 *                                border.
 *            DIM        *pdim    Pointer to structure describing
 *                                dimensions of rectangular region
 *                                to be surrounded by border.
 *            const BORDER *pborder Pointer to structure describing
 *                                type of border.
 *            const WHERE *pwhere Pointer to WHERE structure
 *                                denoting location of rectangular
 *                                region to be surrounded by border.
 *
 * Description This function calculates the actual dimensions required
 *             by a border around a rectangular region on the current
 *             display device.  It takes into account the way WNPUBOR
 *             works at the edges of the screen.
 *
 *             This function assumes that the rectangular region fits
 *             on the current screen.
 * Returns    *presult, *preion  Pointer to structure describing
 *                                location and dimensions of the
 *                                region to be occupied by the
 *                                border.
 */

```

```

static REGION *bordim (preion, pdim, pborder, pwhere)
REGION    *preion;
DIM        *pdim;
const BORDER *pborder;
const WHERE *pwhere;
{
    int    bottom_row, right_column;
    int    mode, columns, act_page;

    bottom_row = pwhere->corner.row + pdim->h;
    right_column = pwhere->corner.col + pdim->w;

    if ((pborder->type == 0)      || (pwhere->corner.row <= 0) ||

```

```

    (pwhere->corner.col <= 0) || (bottom_row >= scrrows()) ||
    (semode (&mode, &columns, &act_page), (right_column >= columns)))
{
    utcopy (&preion->ul, &pwhere->corner, LOC);
    preion->lr.row = bottom_row - 1;
    preion->lr.col = right_column - 1;
}

else
{
    preion->ul.row = pwhere->corner.row - 1;
    preion->ul.col = pwhere->corner.col - 1;
    preion->lr.row = bottom_row;
    preion->lr.col = right_column;
}

return (preion);
}

```

**WNWRAP** 以TTY方式向当前窗口写入一个字符串，常有整字换行

```
#include <bwindow.h>
```

```
void wnwrap( int num_spaces,
             const char *pbuffer,
             int fore,
             int back,
             int option);
```

**num\_spaces** 写入的字符数。如果指定了CHARS\_ONLY (见下面的option), 则num\_spaces为零表示缓冲区以NUL('\0')字符结尾。

**pbuffer** 待写入的数据的地址。

**fore,back** 新的前景和背景属性。值-1指定窗口的缺省前景或背景属性。

**option** 下面两个值之一:

符号	值	意义
----	---	----

CHARS_ONLY	0	缓冲区仅包含字符。如果指定了num_spaces为零, 则字符串以NUL('\0')字符结尾。fore和back的值用于属性。
------------	---	---

CHAR_ATTR	2	缓冲区包含(char,attr)9对。fore和back的值被忽略。
-----------	---	------------------------------------

WNWRAP向当前窗口写字符而不使字在窗口行之间断开。窗口滚动和光标移动与WNWRTTY相同。

一个“字”定义为一个由空白字符或缓冲区开头及结尾包围的非空字符序列。空白”字符指的是空格(' '), tab('\t')、响铃('\a')、退格('\b')、换行('\n')和回车('\r')。

仅当一个字比窗口宽度还要长时才会窗口行之间断开。如果一个字太长, 不断开便无法装入当前行时, 它就被写到窗口的下一行。必要时窗口滚动。如果一个字比窗口宽度还要长, 无法换转到下一行, 这时它被折断。

下列字符作为特殊字符处理:

Tab字符('\t') 作为一个空格对待。

DEL字符('\177') 作为空格写入屏幕。然而它是在这种意义上作为非空格字符看待的: 非空格符连同DEL字符在整字换行时不断开。写入屏幕的字符值是0x20(' ').

MUL('\0') 是一个可以打印非空字符(显示一个空格), 不必一定是一个字符串的结尾。仅当num\_spaces为0并且指定了CHARS\_ONLY时它才被看作字符串的结尾, 否则与DEL同等对待, 但NUL写往屏幕的字符值是0。

换行('\n')、回车('\r')、退格('\b')和响铃('\a')与WNWRTTY的处理方式相同。

多个空格逐字写入，直至当前行的末尾，它的属性象通常那样由fore和back指出。行首不写入空格。当为避免一个字的断裂而加入空格时，这些空格使用窗口的原始属性。

如果换行在行末出或行末溢出，则窗口发生滚动，新空行使用窗口的原始属性。

不论option为何值，当fore或back为-1时使用窗口的缺省前景或缺省背景属性。

除非窗口被指定为“延迟的”，否则输出结束时WNWRAP更新窗口。如果快速连续几次调用WNWRAP(滚动窗口)，数据可能看起来在每次调用后向上跳。为了避免这个现象，可以组合调用WNWRAP或使窗口输出延迟。

如果出现一个错误(即不存在指定为“当前窗口”的窗口)，错误代码将保存在全局变量b\_wncerr中。用WNPRINTF可以显示一个格式化的字符串。

源程序(WMWRAP.C):

```
#include <string.h>
#include <bwindow.h>
#define H (wndata_h(b_pcurwin))
#define W (wndata_w(b_pcurwin))
#define AREA (H * W)
int wnwrbuf (row, col, num_spaces, buffer, fore, back, option)
int row, col, num_spaces;
const char *buffer;
int fore, back, option;
{
    int want_attr, i, offset, last, last_row;
    int attr;
    CELL *pdata;

    /* Validate current window's data structure. */
    if (wnvalwin (b_pcurwin) == NIL)
        wnreterz (WN_BAD_WIN);

    /* Validate row, column. */
    if (utrange (row, 0, H - 1) || utrange (col, 0, W - 1))
        wnreterz (WN_ILL_DIM);

    /* Be sure there's valid data. */
    if (b_pcurwin->img.pdata == NIL)
        wnreterz (WN_NULL_PTR);

    attr = wndoattr (b_pcurwin->attr, fore, back);
    want_attr = ((option & CHAR_ATTR) != 0);
    if ((!want_attr) && num_spaces == 0)
        num_spaces = (int) strlen (buffer);

    /* If there are no characters to write, exit gracefully. */
    if (num_spaces == 0)
        return(num_spaces);

    /* Update memory copy of the window's data area. */
    offset = (row * W) + col;

    /* Pointer to first cell. */
    pdata = b_pcurwin->img.pdata + offset;

    /* Don't go beyond end of window. */
    utuplim (num_spaces, (AREA - offset))

    /* "last" is offset of last char to write. */
    last = offset + num_spaces;
```



```

utuplim (last, AREA - 1);
    /* Last row written to window. */
last_row = last / W;
for (i = 0; i < num_spaces; i++)
{
    /* Copy to data in memory. */
    pdata[i].ch = *buffer++;
    pdata[i].attr = (want_attr ? *buffer++ : (char) attr);
}

    /* Update the physical screen if possible. */
if (!(option & NO_MOVE_CUR))
{
    if (option & CUR_AFTER)
    {
        /* Leave cursor after last char. */
        wncurtrk(b_pcurwin, last_row, last % W);
    }
    else /* Leave cursor at beginning of string. */
    {
        wncurtrk(b_pcurwin, row, col);
    }
}

    /* Update only a part of a line if we can get away
    /* with it. */
if (row == last_row)
    wnnupblk (b_pcurwin, row, col, row, col + num_spaces - 1,
        WN_UPDATE);
else /* Otherwise update all lines changed. */
    wnnupblk (b_pcurwin, row, 0, last_row, W - 1, WN_UPDATE);
return (num_spaces);
}

```

## WNWRBUF 向当前视口中的一片连续位置写入字符

#include <bwindow.h>

```

int wnwrbuf( int row, int col,
             int num_spaces,
             const char *pbuffer,
             int fore, int back,
             int option);

```

row, col 相对于窗口数据区左上角(0,0) 的开始写入的行和列。

num\_spaces 写入的字符数。如果指定了CHARS\_ONLY(见下面的option), 则num\_spaces为零表示缓冲区以NUL("\0")结尾。

pbuffer 待写入的数据。

fore,back 前景和背景属性(-1使用窗口的缺省前景或背景属性)

option option值告诉WNWRBUF缓冲区是如何建立的, 写完缓冲区后光标放置在什么地方。请使用下列位值:

符号	值	意义
CUR_BEG	0	光标放在(row, col)。
CUR_AFTER	1	光标放在字符串尾的后面。
CHARS_ONLY	0	缓冲区仅包含字符, 属性为 fore 和back。

CHAR\_ATTR 2 缓冲区包含(char, attr) 9 对。  
 MOVE\_CUR 0 根据CUR\_BEG/CUR\_AFTER放置光标。  
 NO\_MOVE\_CUR 4 保持光标位置不变。

(返回) 实际写入的字符数。

WNWRBUF向当前窗口写入字符，写入时可带有或不带有相应的显示属性。它可以将光标放置在写入位置的开头或末尾。

WNWRBUF可以将正文换转到下一行，它不滚动窗口。

如果num\_spaces参数不为零，它指示屏幕上物理空间的数目。(如果声明了CHAR\_ATTR则缓冲区使用空间的数目将是这个物理空间数目的二倍。)如果num\_spaces超过当前位置与行尾之间的字符数，则写入过程将继续到下一行。如此下去，直到窗口的末端。写入操作在窗口末端停止，于是实际写入的字符数有可能比缓冲区包含的字符数要少。实际数目作为函数值返回。

如果写入区域包含窗口的右下角，则当指定MOVE\_CUR和CUR\_AFTER时，光标放置在右下角。

仅当num\_spaces为零并且指定了CHARS\_ONLY时，这个函数才把NUL ('\0')认为是缓冲区的末端。

其它特殊字符(象回车、换行、响铃、退格等)不予承认，缓冲区中所有字符在窗口中原样显示。

如果没有一个窗口被指定为当前窗口或row、col超出窗口的尺寸，这时出现一个错误。在出现错误的情况下错误代码将保存在b\_wncerr中，零作为函数值返回。

源程序(WNWRBUF.C):

```
#include <string.h>
#include <bwindow.h>
#define H (wndata_h(b_pcurwin))
#define W (wndata_w(b_pcurwin))
#define AREA (H * W)
int wnwrbuf (row, col, num_spaces, buffer, fore, back, option)
int row, col, num_spaces;
const char *buffer;
int fore, back, option;
{
    int want_attr, i, offset, last, last_row;
    int attr;
    ILL *pdata;

    /* Validate current window's data structure. */
    if (wnvalwin (b_pcurwin) == NIL)
        wnreterz (WN_BAD_WIN);

    /* Validate row, column. */
    if (utrange (row, 0, H - 1) || utrange (col, 0, W - 1))
        wnreterz (WN_ILL_DIM);

    /* Be sure there's valid data. */
    if (b_pcurwin->img.pdata == NIL)
        wnreterz (WN_NULL_PTR);
    attr = wndoattr (b_pcurwin->attr, fore, back);
    want_attr = ((option & CHAR_ATTR) != 0);

    if (!want_attr) && num_spaces == 0)
        num_spaces = (int) strlen (buffer);

    /* If there are no characters to write, exit gracefully. */
}
```

```

if (num_spaces == 0)
    return(num_spaces);
    /* Update memory copy of the window's data area. */
offset = (row * W) + col;

    /* Pointer to first cell. */
pdata = b_pcurwin->img.pdata + offset;
    /* Don't go beyond end of window. */
utuplim (num_spaces, (AREA - offset))

    /* "last" is offset of last char to write. */
last = offset + num_spaces;
utuplim (last, AREA - 1);
    /* Last row written to window. */
last_row = last / W;

for (i = 0; i < num_spaces; i++)
{
    /* Copy to data in memory. */
    pdata[i].ch = *buffer++;
    pdata[i].attr = (want_attr ? *buffer++ : (char) attr);
}

    /* Update the physical screen if possible. */
if (!(option & NO_MOVE_CUR))
{
    if (option & CUR_AFTER)
    {
        /* Leave cursor after last char. */
        wncurtrk(b_pcurwin, last_row, last % W);
    }
    else /* Leave cursor at beginning of string. */
    {
        wncurtrk(b_pcurwin, row, col);
    }
}

    /* Update only a part of a line if we can get away
    /* with it. */
if (row == last_row)
    wnnapblk (b_pcurwin, row, col, row, col + num_spaces - 1,
        WN_UPDATE);
else /* Otherwise update all lines changed. */
    wnnupblk (b_pcurwin, row, 0, last_row, W - 1, WN_UPDATE);

return (num_spaces);
}

```

## WNWRRECT 写入窗口的一个矩形区域

```

#include <bwindow.h>
BWINDOW *wnwrrect(BWINDOW *pwin,
    int u_row, int u_col,

```

```

        int l_row, int l_col,
        const char *pbuffer,
        int fore, int back,
        int option);
pwin      写入的窗口的地址。
u_row, u_col 相对于窗口数据区左上角(0,0)的矩形区域的顶行左列。
l_row, l_col 区域的底行右列。
pbuffer    待写入的数据。
fore,back   前景和背景属性(-1 使用窗口缺省的前景或背景属性)
option      如果缓冲区包含(char, attr) 9 对, 则为CHAR_ATTR(2); 如果缓冲区仅包含字符而fore
和back 用于属性, 则为CHARS_ONLY(0)。
(返回)      BWINDOW结构的地址。若失败, 返回NIL。
(返回)BWINDOW结构的地址。若失败, 返回NIL。

```

WNWRRECT 用缓冲区中的字符填充窗口的一个矩形区域, 填充时可带有或不带有相应的显示属性。数据以一行接一行的方式写入, 光标保持不动。所有字符, 包括NUL('\0'), 均被显示。必要时裁减矩形的尺寸使适合于窗口的数据区。

源程序(WNWRRECT.C):

```

#include <bvideo.h>
#include <bwindow.h>
#include <bmouse.h>

/* Screen coordinates of window's viewport. */
#define VIEW_LEFT (pwin->where_shown.corner.col)
#define VIEW_RIGHT \
    (pwin->where_shown.corner.col + wnview_w(pwin) - 1)
#define VIEW_TOP (pwin->where_shown.corner.row)
#define VIEW_BOTTOM \
    (pwin->where_shown.corner.row + wnview_h(pwin) - 1)
/* Absolute screen coordinates of window's visible data area. */
#define DATA_LEFT (VIEW_LEFT + pwin->data_origin.col)
#define DATA_RIGHT (DATA_LEFT + wndata_w(pwin) - 1)
#define DATA_TOP (VIEW_TOP + pwin->data_origin.row)
#define DATA_BOTTOM (DATA_TOP + wndata_h(pwin) - 1)
/* Relative coordinates of window's visible data area. */
#define VISIBLE_LEFT (pwin->data_origin.col)
#define VISIBLE_RIGHT (VISIBLE_LEFT + wnview_w(pwin) - 1)
#define VISIBLE_TOP (pwin->data_origin.row)
#define VISIBLE_BOTTOM (VISIBLE_TOP + wnview_h(pwin) - 1)
BWINDOW *wnwrrect (pwin, r1, c1, r2, c2, pbuffer, fore, back, option)
BWINDOW *pwin;
int r1, c1, r2, c2;
const char *pbuffer;
int fore, back;
int option;
{
    int row, column;
    unsigned char mask, attr;
    CELL *pto;
    wnvalidw (pwin) /* Validate window structure. */
    /* Force all coordinates to reasonable values. */
}

```

```

utbound (r1, 0, wndata_h(pwin) - 1);
utbound (r2, r1, wndata_h(pwin) - 1);
utbound (c1, 0, wndata_w(pwin) - 1);
utbound (c2, c1, wndata_w(pwin) - 1);
        /* Prepare to write out each line-block of          */
        /* information.                                       */
pto = pwin->img.pdata + (r1 * wndata_w(pwin));
if (option & CHAR_ATTR)
{
        /* Now write out the data.                            */
        for (row = r1; row <= r2; row++, pto += wndata_w(pwin))
            for (column = c1; column <= c2; column++)
            {
                pto[column].ch = *pbuffer++;
                pto[column].attr = *pbuffer++;
            }
}
else
{
        /* Set up attribute and mask.                          */
        mask = 0;
        if (fore == -1)
        {
            mask = 0x0f;
            attr = pwin->attr & 0x0f;
        }
        else
            attr = fore & 0x0f;

        if (back == -1)
        {
            mask |= 0xf0;
            attr |= pwin->attr & 0xf0;
        }
        else
            attr |= (back << 4) & 0xf0;

        for (row = r1; row <= r2; row++, pto += wndata_w(pwin))
            for (column = c1; column <= c2; column++)
            {
                pto[column].ch = *pbuffer++;
                pto[column].attr &= mask;
                pto[column].attr |= attr;
            }
}

return(wnnupblk(pwin, r1, c1, r2, c2, option));
}

```

**WNWRSTR**      以TTY 方式向当前窗口写入一个字符串

```
#include <bwindow.h>
```

```
void wnwrstr( const char *pstring,
              int fore, in back);
```

pstring 写入的字符串，以NUL('0')结尾。

fore, back 新前景和背景属性。值-1指定窗口缺省的前景或背景属性。

WNWRSTR 是一个宏，以电传方式向窗口写入一个字符串。它将回车、换行、退格和响铃看作命令而不看作可打印字符。窗口光标位于显示字符串之后。

如果换行('\n')出现在窗口最后一行或最后一行溢出，则窗口发生滚动，窗口的原始属性用于新空行。

下列字符作为特殊字符处理：

Tab字符('\t') 按原样显示。

NUL ('0') 指示字符串的末尾，不显示。

回车('\r') 将光标移到窗口的最左列。

换行('\n') 执行一个回车和换行。必要时窗口滚动。

退格('\b') (非破坏性地)将光标左移一列，如果光标已经位于窗口的最左列，这时不做任何操作。

响铃字符('\a') 在当前页是活动页的情况下发出鸣响。

WNPRINTF可以显示一个格式化的字符串，WNWRAP显示一个字符串而不使字在窗口行间断开。

源程序(BWINDOW.H)：

该宏定义在附录C的BWINDOW.H头文件。

## WNWRSTRN 以TTY方式向窗口写入一个字符串，带有任选项

```
#include <bwindow.h>
```

```
void wnwrstrn( BWINDOW *pwin,
               const char *pstring,
               int num_spaces,
               int fore, int back,
               int option);
```

pwin 窗口地址。

pstring 包含写入数据的缓冲区，以NUL\_spaces为零表示缓冲区以NUL('0')字符结尾。

fore,back 新的前景和背景属性。值-1表示指定窗口的缺省前景或背景属性。

option 四个任选的位值：

符号	值	意义
CHARS_ONLY	0x00	缓冲区仅包含字符。如果num_spaces为零，则字符串以NUL('0')字符结尾。
CHAR_ATTR	0x02	缓冲区包含(char,attr)9对。
WN_UPDATE	0x00	除非窗口是“延迟的”，否则更新该窗口。
WN_NO_UPDATE	0x04	不更新窗口。
WN_LF_CRLF	0x00	将换行作为回车/换行对待。
WN_LF_LF	0x08	将换行('\n')作为换行对待。
WN_SCROLL	0x00	必要时滚动窗口。
WN_NO_SCROLL	0x10	如果到达窗口的末端，停止写入，光标放在左下角。
WN_NO_TRACK	0x00	不跟踪光标。
WN_CHAR_TRACK	0x20	写入每个字符之后在视口中移动窗口。

WNWRSTRN以电传方式向窗口写入一个字符串，它将回车、换行、退格作为命令而不作为打印字符对待。所有输出和滚动均限定在窗口中，窗口光标保持在显示的字符串之后。

WN\_SCROLL和WN\_SCROLL的任选位决定当新行('\n')出现在窗口的末行或行溢出时出现的现象。如果指定了WN\_NO\_SCROLL, 光标移到左下角, 窗口不再被修改; 否则窗口上滚, 新空行使用窗口的原始属性。

下列字符作为特殊字符处理:

Tab字符('\t') 逐个显示。

NUL ('\0') 仅当在指定了CHARS\_ONLY并且num\_spaces为零的情况才作为特殊字符看待。

在这种情况下, NUL 表示字符串的结尾, 不显示。

回车('\r') 将光标称到窗口的最左列。

换行('\n') 执行一个回车和换行, 必要时滚动窗口。

退格('\b') 将光标左移一列(非破坏性地)。如果光标已经位于窗口的最左列, 这时不做任何操作。

响铃字符('\a') 在当前而是活动页的情况下响铃。

如果出现错误(也就是说不存在当前窗口), 则错误代码保存在b\_wncrr中。

源程序(WNWRSTRN.C):

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <bscreens.h>
#include <bwindow.h>
#define H (wndata_h(pwin))
#define W (wndata_w(pwin))
#define AREA (H * W)
#define OFFSET ((row * W) + col)
#define BEL '\a'
#define BS '\b'
#define CR '\r'
#define NL '\n'
#define NUL '\0'
#define TRUE 1
#define FALSE 0
void wnwrstrn (pwin, pbuffer, buflen, fore, back, option)
BWINDOW *pwin;
const char *pbuffer;
int      buflen;
int      fore, back;
int      option;
{
    char    c;
    int     attr;
    int     row, col, srow, scol;
    int     end_row, end_col;
    int     i;
    int     locked = FALSE;
    int     wrote_data = FALSE;
    int     scrolled = FALSE;
    CELL    *pdata;

    /* Validate window data structure. */
    if (wnvalwin (pwin) == NIL)
        wnretrv (WN_BAD_WIN)
```

```

        /* Be sure there's valid data. */
if (pwin->img.pdata == NIL)
    wnreterv (WN_NULL_PTR)
        /* Figure out length of string if CHARS_ONLY and */
        /* the length is not given to us. */
if ((buflen == 0) && ((option & CHAR_ATTR) == 0))
    buflen = strlen (pbuffer);
attr = wnoattr (pwin->attr, fore, back);
row = pwin->cur_loc.row;
col = pwin->cur_loc.col;
srow = H;
scol = W;
end_row = end_col = 0;
        /* Pointer to first data cell. */
pdata = pwin->img.pdata;
for (i = OFFSET;
    (buflen != 0) && (! locked);
    buflen--, (option & CHAR_ATTR) ? pbuffer += 2 : pbuffer++)
{
    /* Copy to data in memory. */
    switch (c = *pbuffer)
    {
        case BEL:
            settywrt ('\a', 0);
            break;

        case BS:
            if (col != 0)
            {
                col--;
                i--;
            }
            break;

        case NL:
            row++;
            if ( !(option & WN_LF_LF))
                col = 0;
            i = OFFSET;
            break;

        case CR:
            col = 0;
            i = OFFSET;
            break;

        default:
            pdata[i].ch = c;
            pdata[i].attr = (option & CHAR_ATTR)
                ? *(pbuffer + 1)
                : (char) attr;
    }
}

```



```

        utuplim(srow,row)
        utuplim(scol,col)
        utlowlim(end_row,row)
        utlowlim(end_col,col)
        wrote_data = TRUE;
        col++;
        i++;
    }

    while ((row >= H) || (col >= W))
    {
        if (col >= W)
        {
            col = 0;
            row++;
        }

        if (row >= H)
        {
            if (!(option & WN_NO_SCROLL))
            {
                wnsrblk (pwin, 0, 0, H - 1, W - 1,
                        utlonyb (pwin->attr), uthinyb (pwin->attr),
                        WNSCR_UP, 1, WN_NO_UPDATE);
                if (!(option & WN_LF_LF))
                {
                    col = 0;
                    scrolled = TRUE;
                }
            }
            else
            {
                locked = TRUE;
                row--;
            }
        }
        i = OFFSET;
    }

    /* Track the progress of the window's cursor if */
    /* requested to do so. */
    if (!(option & WN_NO_TRACK) && pwin->options.cur_track)
        wncurtrk(pwin, row, col);
}

/* Move window cursor to next character position. */
pwin->cur_loc.row = row;
pwin->cur_loc.col = (locked ? 0 : col);

/* Update the physical screen if possible and */
/* requested. */
if (scrolled & !(option & WN_NO_UPDATE))
    wnneedup(pwin);
else if (wrote_data)

```

```
wnnupblk (pwin, srow, scol, end_row, end_col, option);
}
```

## WNWRTTY 以TTY方式向当前窗口写入一个字符

```
#include <bwindow.h>
void wnwrtty( char ch,
              int fore, int back);
```

ch 写入的字符。

fore, back 新的前景和背景属性。-1指定使用窗口缺省的前景或背景属性。

WNWRTTY以电传方式向窗口写入一个字符，它将回车、换行、退格和响铃作为命令而不是作为可打印字符看待。所有输出和滚动均限定在该窗口，窗口光标保持在显示字符之后。

如果换行出现在窗口的末行或未行溢出，则窗口发生滚动，窗口的原始属性于新空行。

下列字符作为特殊字符处理：

Tab字符('\t') 逐个显示。

NUL('\0') 作为空格打印。

换行('\n') 将光标下移一行。如果光标已经位于窗口的底行，这时窗口滚动。

回车('\r') 将光标移到窗口的最左列。

退格('\b') 将光标左移一列(非破坏性地)；如果光标已经位于窗口的最左列，这时不做任何操作。响铃字符('\a')在当前页是活动页的情况下鸣叫。

如果出现错误(即不存在活动窗口)，则错误代码保存在b\_wncerr中。

源程序(BWINDOW.H)：

该宏定义在附录C的BWINDOW.HG中。

## WNWRTTYX - 写一字符到一TTY方式的窗口。

```
wnwrttyx (pwin, ch, fore, back, option);
```

\*pwin 要写入字符的窗口。

ch 要写的字符。

fore 如果使用缺省的前景属性则为-1；0到15之间的值指定新的前景属性。

back 如果使用缺省的背景属性则为-1；0到15之间的值指定新的背景属性。

option 与下列值相与的值：

WN\_UPDATE

输出是将更新。

WN\_NO\_UPDATE

通过时不更新。

WN\_SCROLL

当达到最后行窗口滚行。

WN\_NO\_SCROLL

不滚行。在窗口缓冲区后写的字符丢失。

WN\_LF\_LF

把CR作为CR，把LF作为LF。

WN\_LF\_CRLF

把CR作为CR，把LF作为CRLF。

CHARS\_ONLY

要求缓冲区只含字符。

CHAR\_ATTR

要求(字符，属性)对而不只仅仅是字符。

该函数以电传方式和窗口输出一字符。把回车、换行、退格和响铃作为命令而非作为可打印字符。

所有的输出和滚动限制于窗口内。

显示字符后窗口的光标可见, 该可见光标除非被通过WNCURSOR选择为具有活动光标才会移动。

如果换行出现在窗口的最后一行或最后行满, 窗口滚动。窗口的native属性用作新空行。

如果屏幕是文本模式, 前景或后景属性使用缺省属性时指定fore或back为-1。如果屏幕用图形模式, 指定fore为-1将选择第一种颜色。

下列字符作为特殊字符处理:

Tab字符('\t') 逐个显示。

NUL('\0') 作为空格打印。

换行('\n') 将光标下移一行。如果光标已经位于窗口的底行, 这时窗口滚动。

回车('\r') 将光标移到窗口的最左列。

退格('\b') 将光标左移一列(非破坏性地)1如果光标已经位于窗口的最左列, 这时不做任何操作。响铃字符('\a')在当前页是活动页的情况下鸣叫。

如果出现错误(即不存在活动窗口), 则错误代码保存在b\_wnerr中。

返回	b_wnerr	可能值如下:
		(不改变) 成功
		WN_BAD_WIN *pwin无效
		WN_ILL_DIM 内部错误。

源程序(WNWRITTYX.C):

```
#include <bwindow.h>
void wnwrityx (pwin, ch, fore, back, option)
BWINDOW *pwin;
char ch;
int fore, back, option;
{
    char s [1];

    s [0] = ch;
    wnwrstrn (pwin, s, 1, fore, back, option);
}
```

## WNZAPEVN 删除WNREAD认可的窗口事件表

#include <bwindow.h>

void wnzapevn(WN\_EVENT\_LIST \*\*ppfirst);

ppfirst 指针的地址, 该指针指向用户响应表的第一项。

WNZAPEVN 删除WNREAD认可的整个响应表, 将\*ppfirst置为NIL。

用WNREMEVN可以从响应表中删除一项。

WNZAPEVN不修改全局窗口错误代码b\_wnerr。

源程序(WNZAPEVN.C):

#include <butil.h>

#include <bwindow.h>

void wnzapevn(pplist\_head)

WN\_EVENT\_LIST \*\*pplist\_head;

```
{
    WN_EVENT_LIST *pcurrent_item, *pnext_item;
    if (*pplist_head == NIL)
    {
        return;
    }
}
```

```

    }
    pcurrent_item = *pplist_head;
    do
    {
        if (wnvalevn(pcurrent_item) == NIL)
            return;
        pnext_item = pcurrent_item->pnext;
        pcurrent_item->pnext = NIL;
        pcurrent_item->pprev = NIL;
        pcurrent_item->signature = BEVENT_DEAD;
        if (pcurrent_item->win_event.pdata != NIL)
            free(pcurrent_item->win_event.pdata);
        free(pcurrent_item);
        pcurrent_item = pnext_item;
    } while ((pcurrent_item != NIL) && (pcurrent_item != *pplist_head));
    *pplist_head = NIL;
}

```

### WNZAPSEN - 释放窗口的sensor列表。

wnzapsen(pwin);

\*pwin 指向释放sensor的窗口。

函数接受指向窗口的指针，释放该窗口的sensor的链表。

返回 无

源程序(WNZAPSEN.C):

```

#include <bwindow.h>
void wnzapsen(pwin)
BWINDOW *pwin;
{
    WN_SENSOR *pcurrent, *pnext;
    pcurrent = pwin->psensors;
    while (pcurrent != NIL)
    {
        pnext = pcurrent->pnext;
        if (pcurrent->pimage != NIL)
            free(pcurrent->pimage);
        pcurrent->pnext = NIL;
        pcurrent->pprev = NIL;
        free(pcurrent);
        pcurrent = pnext;
    }
    pwin->psensors = NIL;
}

```

## 附录 A 使用的汇编头文件

一些函数用汇编编写。使用了COMP\_T2S.MAC、COMP\_T2M.MAC、COMP\_T2C.MAC、COMP\_T2L.MAC和COMP\_T2H.MAC五个汇编头文件，设置C编译器和汇编语言模式。

还使用了BEGINASM.MAC头文件，定义编译依赖的符号和宏。

下面是头文件的清单。

### COMP\_T2S.MAC

---

```
: COMP_XXX.MAC -- Include file to set C compiler and model for
:                  assembly language functions.
```

```
: The following definitions specify the particular C compiler
: used:
```

```
: TC100   Turbo C v1.00 or later.
: MSC500  Microsoft C v5.00 or later.
```

```
: MSC500 EQU 0
```

```
: TC100   EQU 1
```

```
: BCOMPACT equ 0
```

```
: BIUGE   equ 0
```

```
: BLARGE  equ 0
```

```
: BMEDIUM equ 0
```

```
: BSMALL  equ 1
```

```
: BTINY   equ 0
```

### COMP\_T2M.MAC

---

```
: COMP_XXX.MAC -- Include file to set C compiler and model for
:                  assembly language functions.
```

```
: The following definitions specify the particular C compiler
: used:
```

```
: TC100   Turbo C v1.00 or later.
: MSC500  Microsoft C v5.00 or later
```

```
: MSC500 EQU 0
```

```
: TC100   EQU 1
```

```

BCOMPACT equ 0
BHUGE equ 0
BLARGE equ 0
BMEDIUM equ 1
BSMALL equ 0
BTINY equ 0

```

## COMP\_T2C.MAC

---

```

;
; COMP_xxx.MAC -- Include file to set C compiler and model for
;                assembly language functions.
;
; The following definitions specify the particular C compiler
; used:
;
; TC100 Turbo C v1.00 or later.
; MSC500 Microsoft C v5.00 or later.
;
MSC500 EQU 0
TC100 EQU 1

```

```

BCOMPACT equ 1
BHUGE equ 0
BLARGE equ 0
BMEDIUM equ 0
BSMALL equ 0
BTINY equ 0

```

## COMP\_T2L.MAC

---

```

;
; COMP_xxx.MAC -- Include file to set C compiler and model for
;                assembly language functions.
;
; The following definitions specify the particular C compiler
; used:
;
; TC100 Turbo C v1.00 or later.
; MSC500 Microsoft C v5.00 or later.
;
MSC500 EQU 0
TC100 EQU 1

```

```

BCOMPACT equ 0
BHUGE equ 0
BLARGE equ 1
BMEDIUM equ 0
BSMALL equ 0

```

BTINY equ 0

## COMP\_T2H.MAC

---

```
;  
; COMP_xxx.MAC -- Include file to set C compiler and model for  
; assembly language functions.  
;
```

```
;  
; The following definitions specify the particular C compiler  
; used:
```

```
;  
; TC100 Turbo C v1.00 or later.  
; MSC500 Microsoft C v5.00 or later.  
;
```

MSC500 EQU 0

TC100 EQU 1

BCOMPACT equ 0

BHUGE equ 1

BLARGE equ 0

BMEDIUM equ 0

BSMALL equ 0

BTINY equ 0

## BEGINASM.MAC 编译依赖的符号和宏。

---

```
;  
; BEGINASM.MAC Compiler-dependent symbols and macros.
```

```
include compiler.mac ; Specifies the C compiler and  
; memory model.
```

ShortP equ 0 ; Short Pointer

LongP equ 1 ; Long Pointer

if BTINY

\_\_TINY\_\_ equ 1 ; For Turbo C's RULES.ASI

CodeT equ ShortP ; CodeT -- Type of code (long or short)

DataT equ ShortP ; DataT -- Type of data (long or short)

endif

if BSMALL

\_\_SMALL\_\_ equ 1 ; For Turbo C's RULES.ASI

CodeT equ ShortP ; CodeT -- Type of code (long or short)

DataT equ ShortP ; DataT -- Type of data (long or short)

endif

if BMEDIUM

\_\_MEDIUM\_\_ equ 1 ; For Turbo C's RULES.ASI

```

    CodeT equ LongP          ; CodeT -- Type of code (long or short)
    DataT equ ShortP         ; DataT -- Type of data (long or short)
endif

```

```

if BCOMPACT
    __COMPACT__ equ 1        ; For Turbo C's RULES.ASI
    CodeT equ ShortP         ; CodeT -- Type of code (long or short)
    DataT equ LongP          ; DataT -- Type of data (long or short)
endif

```

```

if BLARGE
    __LARGE__ equ 1; For Turbo C's RULES.ASI
    CodeT equ LongP; CodeT -- Type of code (long or short)
    DataT equ LongP; DataT -- Type of data (long or short)
endif

```

```

if BHUGE
    __HUGE__ equ 1 ; For Turbo C's RULES.ASI
    CodeT equ LongP; CodeT -- Type of code (long or short)
    DataT equ LongP; DataT -- Type of data (long or short)
endif

```

```

longData equ (DataT eq LongP)
longProg equ (CodeT eq LongP)

```

```

shortData equ (DataT eq ShortP)
shortProg equ (CodeT eq ShortP)

```

```

sizeChar      equ 1
sizeCharOnStk equ 2
sizeDouble    equ 8
sizeEnum      equ 2
sizeFloat     equ 4
sizeInt       equ 2
sizeLong      equ 4
sizeLongDouble equ 8
sizeShort     equ 2

```

```

if longData
    sizeDPointer equ 4
else
    sizeDPointer equ 2
endif

```

```

if longProg
    sizeCPointer equ 4
else
    sizeCPointer equ 2
endif

```



```

beginCseg macro sname          ;;Begin code macro.
    if shortProg
        _TEXT segment byte public 'code'
        assume cs:_TEXT
    else
        sname&_TEXT segment byte public 'code'
        assume cs:sname&_TEXT
    endif
endm

endCseg macro sname            ;;End code macro.
    if shortProg
        _TEXT ends
    else
        sname&_TEXT ends
    endif
endm

beginDseg macro sname          ;;Start a data segment
    if BHUGE
        sname&_DATA segment word public 'DATA'
    else
        _DATA segment word public 'DATA'
    endif
endm

endDseg macro sname            ;;End a data segment
    if BHUGE
        sname&_DATA ends
    else
        _DATA ends
    endif
endm

if MSC500
    if longProg                  ; Set up parameter offset.
        stkoff equ 6
    else
        stkoff equ 4
    endif

    makeName macro myname        ;;Create a symbol with underscore
        myname&@ equ <_&myname> if necessary.
    endm

    beginMod macro mname          ;;Begin module macro.
        name mname
    endm

    endMod macro mname            ;;End module macro.

```

```

        ; end module mname
    endm

beginProc macro pname                ;;Begin procedure macro.
    makeName pname
    ifdef @Version
%       public pname&@
    else
        public pname&@
    endif
    if longProg
        pname&@ proc far
    else
        pname&@ proc near
    endif
    endm

endProc macro pname                ;;End procedure macro.
    pname&@ endp
    endm

pubSym macro vname,definition        ;;Define public variable.
    makeName vname
    public vname&@
vname&@ definition
    endm

extSym macro vname,data_type        ;;Declare external variable.
    makeName vname
    extrn vname&@:data_type
    endm

extDPtr macro pname                ;;Declare external data pointer.
    makeName pname
    if longData
        extrn pname&@:dword
    else
        extrn pname&@:word
    endif
    endm

extCPtr macro pname                ;;Declare external code pointer.
    makeName pname
    if longProg
        extrn pname&@:dword
    else
        extrn pname&@:word
    endif
    endm

```

```

extProc macro pname                ;;Declare external function.
    makeName pname
    if longProg
        extrn pname&@:far
    else
        extrn pname&@:near
    endif
endm

endif

if TC100
    include rules.asi

    if longProg                    ; Set up parameter offset.
        stkoff equ aParam
    else
        stkoff equ nearParam
    endif

    beginMod macro mname           ;;Begin module macro.
        name mname
    endm

    endMod macro mname             ;;End module macro.
        ; end module mname
    endm

    beginProc macro pname          ;;Begin procedure macro.
        PubProc@ pname,NOT_PASCAL
    endm

    endProc macro pname           ;;End procedure macro.
        EndProc@ pname,NOT_PASCAL
    endm

    pubSym macro vname,definition ;;Define public variable.
        PubSym@ vname,<definition>,NOT_PASCAL
    endm

    extSym macro vname,data_type   ;;Declare external variable.
        ExtSym@ vname,data_type,NOT_PASCAL
    endm

    extDPtr macro pname            ;;Declare external data pointer.
        dPtrExt@ pname,NOT_PASCAL
    endm

    extCPtr macro pname           ;;Declare external code pointer.
        cPtrExt@ pname,NOT_PASCAL
    endm

```

```

    extProc macro pname                ;;Declare external function.
        ExtProc@ pname,NOT_PASCAL
    endm
endif

beginProg macro zname                  ;;Generic begin program macro.
    beginMod zname
    beginCseg zname
    beginProc zname
    endm

endProg macro zname                   ;;Generic end' program macro.
    endProc zname
    endCseg zname
    endMod zname
    endm

popff macro                            ;; Simulate POPF instruction w/o bugs
    local    do_call, do_iret
    jmp      short do_call

do_iret:
    iret                    ;; Pop IP, CS, flags.

do_call:
    push     cs              ;; Push CS
    call     do_iret         ;; Push IP & jump.
    endm

```

## 附录 B 创建功能强大的函数库批命令

只要按第一章中的设置, 在命令行下执行BUILDLIB.BAT批命令就可创建新库。

```
:
:
: This batch file can be used to reconstruct the Turbo C TOOLS
: libraries. It assumes that all source files are in the
: current directory, that the necessary MAC files are in the
: subdirectory MAC, and the Turbo C assembly language helper file,
: rules.asi, is in the current directory. Moreover, it is assumed
: that the include files reside in a directory \turbo\include.
: If this is not your configuration you should alter the batch
: file accordingly.
:
: Furthermore, the Turbo C compiler (TCC), the Turbo Assembler (TASM),
: and the Turbo Librarian (TLIB) must be in directories accessible
: via the PATH environment variable.
:
: BUILDLIB is invoked with the command
:
:      buildlib model
:
: where model is either s, m, c, l, or h (in lower case). The
: corresponding small, medium, compact, large, or huge memory model
: library is constructed.
:
set saveprompt=%prompt%
prompt $h

if exist rules.asi goto check1
pause The Turbo C file rules.asi must be in the current directory.
goto end

:check1
if exist mac\beginasm.mac goto check2
pause The file beginasm.mac must be present in the subdirectory MAC.
goto end

:check2
if a%1==a goto fail3

if %1==s goto check4
if %1==m goto check4
if %1==c goto check4
if %1==l goto check4
if %1==h goto check4
:fail3
```

```

pause Either s(mall), m(edium), c(ompact), l(arge), or h(uge) must be specified.
goto end

:check4
if exist mac\comp_t2%1.mac goto check5
pause The file comp_t2%1.mac must be present in the subdirectory MAC.
goto end

:check5
copy mac\beginasm.mac
copy mac\comp_t2%1.mac compiler.mac
erase *.obj
if exist tct.lib erase tct.lib

tcc -c -w -O -m%1 -I\turbo\include ed*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include fl*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include hl*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include is*.c
if errorlevel 1 goto comperror
tasm /mx /w+ is*.asm
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include iv*.c
if errorlevel 1 goto comperror
tasm /mx /w+ ivctrl
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include kb*.c
if errorlevel 1 goto comperror
tasm /mx /w+ kb*.asm
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include mm*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include mn*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include mo*.c
if errorlevel 1 goto comperror
tasm /mx /w+ mocatch
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include pr*.c

```

```

if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include sc*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include st*.c
if errorlevel 1 goto comperror

tcc -c -w -O -m%1 -I\turbo\include ut*.c
if errorlevel 1 goto comperror
tasm /mx /w+ ut*.asm
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include vi*.c
if errorlevel 1 goto comperror
tasm /mx /w+ vidirec0
if errorlevel 1 goto asmerror

tcc -c -w -O -m%1 -I\turbo\include wn*.c
if errorlevel 1 goto comperror

echo Compilation complete

tlib tct.lib /O @libresp
if errorlevel 1 goto liberror
if not exist tct.lib goto liberror
if exist tct_t2%1.lib erase tct_t2%1.lib
rename tct.lib tct_t2%1.lib
pause Library has been constructed
goto end

:compererror
pause Error in compilation
goto end

:asmerror
pause Error during assembly
goto end

:liberror
pause Error in library construction

:end
set prompt=%saveprompt%

```

### 创建库文件的LIB程序的响应文件(LIBRESP)

```

+edbase.obj      +edbuffer.obj      +edchgkey.obj      +edinitky.obj      +edreduce.obj      &
+edremkey.obj    +edretinf.obj      +edretkey.obj      +edsetcur.obj      +edwrap.obj        &
+edwrrect.obj    +edzapkey.obj                      &
&

```

+fldlock.obj	+flush.obj	+fgetda.obj	+flock.obj	+flnorm.obj	&
+flprompt.obj	+flputda.obj	+flremvol.obj	+flretvol.obj	+flsetvol.obj	&
&					
+hlclose.obj	+hldisp.obj	+hlfrindx.obj	+hllookup.obj	+hlopen.obj	&
+hlpas2c.obj	+hlread.obj			&	
&					
+iscall.obj	+iscurprc.obj	+isdispat.obj	+isinstal.obj	+isprep.obj	&
+isremove.obj	+isreserv.obj	+issense.obj		&	
&					
+ivctrl.obj	+ivdetect.obj	+ivdisabl.obj	+ivinstal.obj	+ivsense.obj	&
+ivvecs.obj				&	
&					
+kbequip.obj	+kbextend.obj	+kbflush.obj	+kbgetkey.obj	+kbkeflsh.obj	&
+kbplace.obj	+kbpoll.obj	+kbquery.obj	+kbqueue.obj	+kbcady.obj	&
+kbseanf.obj	+kbset.obj	+kbstuff.obj	+kbwait.obj		&
&					
+mmctrl.obj	+mmfirst.obj	+mmsize.obj		&	
&					
+mnatr.obj	+mncreat0.obj	+mnndefkey.obj	+mnndisabl.obj	+mndlitms.obj	&
+mndlkeys.obj	+mndstroy.obj	+mnfindsl.obj	+mnhil0.obj	+mnhilite.obj	&
+mnitem.obj	+mnitmkey.obj	+mnkey.obj	+mnlitem.obj	+mnlitkey.obj	&
+mnmchitm.obj	+mnmchkey.obj	+mnmouse.obj	+mnmstyle.obj	+mnread.obj	&
+mnvalmn0.obj	+mnvdisp.obj			&	
&					
+moavoid.obj	+mobutton.obj	+mocatch.obj	+mocheck.obj	+moeurmov.obj	&
+moequip.obj	+mogate.obj	+mogetmov.obj	+mograph.obj	+mohandlr.obj	&
+mohard.obj	+mohide.obj	+moinst.obj	+mojump.obj	+molitpen.obj	&
+mopreclk.obj	+morange.obj	+mosoft.obj	+mospeed.obj	+mostat.obj	&
+movars.obj				&	
&					
+prcancel.obj	+prerror.obj	+prgetq.obj	+prinstld.obj	+prspool.obj	&
&					
+scapage.obj	+scattrib.obj	+scblink.obj	+scborder.obj	+sabox.obj	&
+schgdev.obj	+scclmsg.obj	+sccurset.obj	+sccurst.obj	+scequip.obj	&
+scgetvid.obj	+scmode.obj	+scmode4.obj	+scnewdev.obj	+scpage.obj	&
+scpages.obj	+scpal1.obj	+sepalett.obj	+scpcdr.obj	+sepgcur.obj	&
+scread.obj	+screstpg.obj	+scrows.obj	+scsavepg.obj	+scsetvid.obj	&
+scattywin.obj	+scattywrt.obj	+scwrap.obj	+scwrite.obj		&
&					
+stpcvt.obj	+stpexpan.obj	+stpjust.obj	+stptabfy.obj	+stpxlate.obj	&
+stschind.obj				&	
&					
+utamove.obj	+utansi.obj	+uterit.obj	+utctlbrk.obj	+utgetclk.obj	&
+utintflg.obj	+utmodel.obj	+utmvmem.obj	+utnulkhk.obj	+utsafecpy.obj	&
+utsleep.obj	+utspkr.obj	+utsqzscn.obj	+uttim2tk.obj	+uttk2tim.obj	&
+utunsqz.obj				&	
&					
+viatrect.obj	+vidirec0.obj	+vihoriz.obj	+viptr.obj	+virdsect.obj	&
+viscroll.obj	+viwrsect.obj			&	
&					



+wnatrbk.obj	+wnatrstr.obj	+wnchgevn.obj	+wncover.obj	+wncreat0.obj	&
+wncurmov.obj	+wncurpos.obj	+wncurset.obj	+wncursor.obj	+wncurtrk.obj	&
+wndstroy.obj	+wnerror.obj	+wnforget.obj	+wngetimg.obj	+wngetopt.obj	&
+wnhide.obj	+wnhoriz.obj	+wninitev.obj	+wnnatvwn.obj	+wnneedup.obj	&
+wnnupblk.obj	+wnorigin.obj	+wnovrlap.obj	+wnpgadd.obj	+wnpgrem.obj	&
+wnpimblk.obj	+wnpoll.obj	+wnprintf.obj	+wnputhor.obj	+wnputsen.obj	&
+wnquery.obj	+wnrdbuf.obj	+wnread.obj	+wnredraw.obj	+wnremevn.obj	&
+wnremove.obj	+wnresprv.obj	+wnretevn.obj	+wnretinf.obj	+wnrevupd.obj	&
+wnscrblk.obj	+wnscrbr.obj	+wnscroll.obj	+wnseldev.obj	+wnselect.obj	&
+wnsetbuf.obj	+wnsetcur.obj	+wnsetopt.obj	+wnshoblk.obj	+wnunhide.obj	&
+wnupdate.obj	+wnvalev0.obj	+wnvalno0.obj	+wnvalwi0.obj	+wnvdisp.obj	&
+wnwrap.obj	+wnwrbuf.obj	+wnwrrect.obj	+wnwrstm.obj	+wnwrttyx.obj	&
+wnzapevn.obj	+wnzapsen.obj				

## 附录 C 函数包使用的头文件

本书中的函数除了使用Turbo C 2.0中的标准头文件外, 还定义了如下头文件:

```

BEDIT.H      BFILES.H      BGENWIN.H    BHELP.H
BINTERV.H    VINTRUPT.H    BKKEYBRD.H  HBKEYS.H
BLAISE.H      BMEM.H        BMENU.H      BMOUSE.H
BNATVWN.H    BPRINT.H      BSCREENS.H  B$STRINGS.H
BUTIL.H       BVIDEO.H      BWINDOW.H

```

为了方便参考，把程序清单罗列如下。

**BEDIT.H** 域编辑函数的头文件。

```
#ifndef DEF_BEDIT                                /* Prevent second reading of */
#define DEF_BEDIT 1                             /* these definitions.      */

#include <bkeybrd.h>

#include <bcreens.h>

#include <bwindow.h>

/*****

/* Definitions of data types.

*****/

typedef enum                                     /* ED_ACTION:              */
{
    ED_NULL,                                    /* values for the EDIT_KEY */
    ED_ABORT,                                  /* action field.           */
    ED_BEEP,                                   /* No action.              */
    ED_ATTR,                                   /* Quit with no edit.      */
    ED_ASCII,                                  /* Sound the system speaker. */
    ED_LEFT,                                   /* Alter attributes.       */
    ED_RIGHT,                                  /* Display the character.   */
    ED_UP,                                     /* Move left.              */
    ED_DOWN,                                   /* Move right.             */
    ED_FRONT,                                 /* Move up one line.       */
    ED_END,                                    /* Move down one line.     */
    ED_TEXT_END,                              /* Move to front of buffer. */
    ED_WRAP,                                  /* Move to end of buffer.  */
    ED_REDUCE,                                /* Move to last non-whitesp. */
    ED_NEXT_WORD,                             /* Word wrap the buffer.   */
    ED_PREV_WORD,                            /* Reduce whitespace.      */
    ED_NEXT_WORD,                             /* Move to next word.      */
    ED_PREV_WORD,                            /* Move to previous word.  */

```

```

ED_BEGIN_WORD,          /* Move to start of word.      */
ED_END_WORD,            /* Move to end of word.      */
ED_NEXT_WHITE,          /* Move to next/previous      */
ED_PREV_WHITE,           /* whitespace character.      */
ED_NEXT_NONWHITE,        /* Move to next/previous non-*/
ED_PREV_NONWHITE,        /* whitespace character.      */
ED_DELETE,              /* Delete current character.  */
ED_RUBOUT,              /* Move left and delete char.*/
ED_DLELEFT,            /* Delete char to left.      */
ED_DEL_WORD,            /* Delete a word.            */
ED_CLEAR,               /* Clear the entire buffer.   */
ED_CLEAREOL,            /* Clear to end of line.     */
ED_CLEAREOB,            /* Clear to end of buffer.   */
ED_INSERT,              /* Insert mode.              */
ED_REPLACE,             /* Replace (not insert) mode.*/
ED_INSTOGGLE,           /* Toggle insert mode.       */
ED_UNDO,                /* Recover previous buffer.   */
ED_TRANSMIT,            /* Finish edit.              */
ED_INVALID_ACTION       /* Invalid edit action.      */
} ED_ACTION;

```

typedef struct

```

{
    int          num_actions; /* ED_ACTION_LIST structure: */
    ED_ACTION    *pactions;  /* number of action codes, */
                                /* array of action codes. */
} ED_ACTION_LIST;

```

typedef struct

```

{
    ED_ACTION_LIST edit_actions; /* ED_KEY structure: */
    KEY_SEQUENCE key_sequence; /* edit actions, key */
    int attribute; /* sequence and attribute */
                                /* for edit key. */
} ED_KEY;

```

typedef struct ed\_list

```

{
    ED_KEY edit_key; /* ED_LIST structure: */
    struct ed_list *pNext; /* ED_KEY & pointers to */
    struct ed_list *pprev; /* next and last items in */
                                /* edit key list. */
} ED_LIST;

```

typedef struct

```

{
    /* ED_BUFFER structure: */
    /* edit buffer passed to */
}

```

```

/* edbuffer(). */
char      *pbuffer; /* Edit buffer. */
DIM        dimensions; /* Dimensions of field. */
int        attribute; /* Display attributes. */
int        buffer_size; /* Maximum length of buffer. */
int        cursor_pos; /* Edit position in buffer. */
int        data_end; /* End of data in buffer. */
unsigned int control_flags; /* Flags for edbuffer/edbase. */
} ED_BUFFER;

```

```

typedef struct /* ED_CONTROL structure: */
{
/* position & info for an */
/* edit buffer. */
LOC        ul_corner; /* Upper left corner. */
DIM        dimensions; /* Dimensions of field. */
int        attribute; /* Display attributes. */
CUR_TYPE   replace_cursor; /* Cursor size for replace */
CUR_TYPE   insert_cursor; /* and insert modes. */
PKEY_CONTROL control_function; /* Key control function. */
unsigned int control_flags; /* edfield() control flags. */
} ED_CONTROL;

```

```

typedef struct
{
int        action_code; /* Single action code. */
KEY_SEQUENCE key_sequence; /* Key sequence for edit key. */
} ED_DEF_KEY;

```

```

typedef int cdecl (*PED_RET_FUNC)(void *, CUR_INFO *, const LOC *,
int, int);
typedef int cdecl (*PED_SET_FUNC)(void *, int, CUR_TYPE *, int, int);
typedef void cdecl (*PED_WRITE_FUNC)(void *, int, int, int, int,
const char *, int, int, int);

```

```

/*****
/* User Macros. */
*****/

```

```

#define edfield(pinitstr,pretstr,target_size,pfield_control,pfinal) \
    (edbase(NIL, (pinitstr), (pretstr), (target_size), \
    (pfield_control), (pfinal), edretinf, edsetcur, \
    edwrrrect))

```

```

/* Edit mode constants for control_flags member of */
/* the ED_CONTROL structure. */
#define ED_LEFTJUST 0x8000 /* Left justify upon exit. */
#define ED_CENTERJUST 0x4000 /* Center justify upon exit. */
#define ED_RIGHTJUST 0x2000 /* Right justify upon exit. */
#define ED_ZERO_FILL 0x1000 /* Zero fill. */
#define ED_BEEP_END 0x0800 /* Beep when at end of field. */
#define ED_AUTOSKIP 0x0400 /* Transmit at end of field. */
#define ED_INSERT_MODE 0x0200 /* Start in INSERT mode. */
#define ED_WORD_WRAP 0x0100 /* Word wrap buffer. */

/*****
/* Global variables */
/*****
extern ED_LIST *b_pkey_root; /* Edit key list head. */
extern int b_pkeys_init; /* Non-zero if edit key list */
/* has been initialized. */
extern ED_DEF_KEY b_defkeys[]; /* Default edit key records. */

/*****
/* Error codes.
/*****
#define ED_NO_ERROR WN_NO_ERROR
#define ED_NO_MEMORY WN_NO_MEMORY
#define ED_ILL_DIM WN_ILL_DIM
#define ED_NO_KEY 300
#define ED_KEY_EXISTS 301
#define ED_USER_ABORT 302

/*****
/* User function declarations.
/*****
int cdecl edinitky(void); /* Initialize edit key list. */
/*
ED_KEY *cdecl edretkey(const /* Return an edit key record from */
KEY_SEQUENCE *); /* the edit key list. */
/*
int cdecl edchgkey( /* Change an edit key record in */
const /* the edit list. */
KEY_SEQUENCE *, /* the edit list. */
const /*
ED_ACTION_LIST *); /*
/*

```

```

int      cdecl edremkey(const      /* Delete an edit key record in      */
                        KFY_SEQUENCE *)/* the edit list.                  */
                        /*                                              */
void      cdecl edzapkey(void);      /* Release the entire edit list.    */
                        /*                                              */
ED_ACTION cdecl edbuffer(          /* Edit a buffer.                  */
                        ED_BUFFER *, /*                                  */
                        const ED_KEY *); /*                                  */
                        /*                                              */

/*****
/* Internal function declarations.
*****/

int      cdecl edretinf(void *, CUR_INFO *, const LOC *, int, int);
int      cdecl edsetcur(void *, int, CUR_TYPE *, int, int);
void      cdecl edwrrrect(void *, int, int, int, int, const char *,
                        int, int, int);
int      cdecl edbase(void *, const char *, char *, int,
                        const ED_CONTROL *, KEY_SEQUENCE *, PED_RET_FUNC,
                        PED_SET_FUNC, PED_WRITE_FUNC);
void      cdecl edwrap(ED_BUFFER *);
void      cdecl edreduce(ED_BUFFER *);

#endif                      /* Ends "#ifndef DEF_BEDIT"      */

```

## BFILES.H 文件和目录函数的头文件。

---

```

#ifndef DEF_BFILES          /* Prevent redefinition.      */
#include <butil.h>
#define FL_DOS_INT 0x21     /* General DOS gate interrupt. */
#define AT_GENERAL0        /* File attributes            */
#define AT_RDONLY 1
#define AT_HIDDEN 2
#define AT_SYSTEM 4
#define AT_VOLUME 8
#define AT_DIR 16
#define AT_ARCHIVE 32
#define FL_LOCK 0           /* Constants for FLDOLOCK and */
#define FL_UNLOCK 1        /* FLLOCK.                    */
#define MAX_FLEN 67         /* Maximum length of filenames */
                        /* returned by FLNORM, etc.    */

/* Function declarations.

```

```

int cdecl flock (int, int,          /* Lock or unlock a section of */
                unsigned long,      /* an open file.                */
                unsigned long);     /*                               */
/*                               */
int cdecl fiprompt (const char *,   /* Accept a text response from */
                  char *,int);      /* standard input.              */
/*                               */
int cdecl fdlock (int,              /* Intelligently lock a section */
                unsigned long,      /* of a file.                   */
                unsigned long,      /*                               */
                int, int);          /*                               */
/*                               */
int cdecl fflush (int);             /* Flush buffer associated with */
/* a handle.                      */
/*                               */
int cdecl fnorm (const char *,      /* Normalize a path name (put   */
                char *, int *);     /* into standard form)          */
/*                               */
void cdecl fiputdt (void far *);    /* Set the Disk Transfer Area   */
/*                               */
void far *cdecl figetdt (void);     /* Return the DTA location      */
/*                               */
int cdecl firtvol (int, char *,     /* Return volume label          */
                unsigned *,         /*                               */
                unsigned *);        /*                               */
/*                               */
int cdecl fsetvol(int,              /* Create or alter volume label */
                const char *);      /*                               */
/*                               */
int cdecl fremvol (int);            /* Remove volume label          */
/*                               */
#define DEF_BFILES 1               /* Prevent second reading of    */
/* these definitions.              */
/* Ends "#ifndef DEF_BFILES"      */
#endif

```

## BGENWIN.H 禁止提供编译器本地文本窗口支持的头文件。

---

```

#ifndef DEF_BGENWIN                /* Prevent redefinition.        */
#define DEF_BGENWIN 1
/* The symbol B_NATIVE_WINDOWS is used at compile */
/* time to determine whether Blaise C TOOLS routines */
/* will support the compiler's native text window.  */
/* If it is defined as 1, the native text window is */

```

```

        /* supported; a value of 0 means it is not. */
#define B_NATIVE_WINDOWS 0 /* No native text window support. */
/* #define B_NATIVE_WINDOWS 1 */ /* Native text window support. */
#endif /* Ends "#ifndef DEF_BGENWIN". */

```

## BHELP.H 帮助函数的头文件。

---

```

#ifndef DEF_BHELP /* Prevent redefinition. */
#define DEF_BHELP 1
#include <bwindow.h>
#define HL_ID_SIZE 13
#define HL_SIGN_SIZE 7

/*****
/* Definitions of data types.
*****/

/* The types HL_WINDOW and HL_RECORD_DATA MUST
/* match each other field for field from view_top
/* to highlight_back, inclusive.

typedef struct
{
    int view_top; /* Top edge of viewport. */
    int view_left; /* Left edge of viewport. */
    int view_bottom; /* Bottom edge of viewport. */
    int view_right; /* Right edge of viewport. */
    int origin_row; /* Data area coordinates initially */
    int origin_col; /* appearing in viewport's origin.*/
    int data_fore; /* Data area foreground color. */
    int data_back; /* Data area background color. */
    int border_type; /* Viewport border type. */
    int border_fore; /* Border foreground color. */
    int border_back; /* Border background color. */
    int title_type; /* Window title. */
    int title_fore; /* Title Foreground. */
    int title_back; /* Title Background. */
    int data_rows; /* # rows in image. */
    int data_columns; /* # columns in image. */
    int xref_fore; /* Foreground/background attributes */
    int xref_back; /* for cross reference items. */
    int highlight_fore; /* Foreground/background attributes */
    int highlight_back; /* for highlight bar. */
    char *pwindow_title; /* Text of window title. */
} HL_WINDOW;

```



```

/**
 *
 * Database structure: the actual format of the help database is as
 * follows:
 *
 * +-----+ \
 * |         | |
 * | HL_FILE_HEADER | > Required file header record.
 * |         | |
 * +-----+ /
 *
 * +-----+ \
 * |         | |
 * | HL_RECORD_DATA||
 * |         | |
 * +-----+ |
 * +-----+ |
 * |         | | 0 or more help window records with
 * | Screen Image | > compressed screen images and optional
 * |         | | window titles.
 * +-----+ |
 * ..... |
 * :      : |
 * : Window Title : |
 * :      : |
 * : ..... /
 *
 * :
 * :
 *
 * +-----+ \
 * |         | |
 * | HL_INDEX_DATA |
 * |         | |
 * +-----+ | Index records with optional description
 * ..... > strings. Each help window record has
 * :      : | exactly one index record associated with it.
 * : Description : |
 * : String      : |
 * :      : |
 * : ..... /
 *
 *

```

```

*
*
*
**/

typedef struct
{
    char        file_sign[20];        /* Identifying text.          */
    char        text_terminator;      /* Control Z (#26).          */
    char        reserved;              /* Reserved.                  */
    unsigned     version;              /* database version number.   */
    unsigned long index_offset;        /* Position of the index.     */
    unsigned long file_size;          /* Size of the file.         */
} HL_FILE_HEADER;

#define HL_WINDOW_RECORD
#define HL_INDEX_RECORD 1
#define HL_FILE_SIGN      "Blaise Computing Inc"
#define HL_VERSION        0x0100
typedef struct
{
    unsigned type;                    /* Type of database record.   */
    unsigned length;                  /* Length of following record.*/
} HL_RECORD_ID;

typedef struct
{
    HL_RECORD_ID record_sign; /* Record header.          */
    char id_string[HL_ID_SIZE]; /* ID of help record.      */
    char    reserved;          /* Reserved.                */
    int     view_top;           /* Top edge of viewport.    */
    int     view_left;          /* Left edge of viewport.   */
    int     view_bottom;        /* Bottom edge of viewport. */
    int     view_right;         /* Right edge of viewport.  */
    int     origin_row;         /* Data area coordinates initially */
    int     origin_col;         /* appearing in viewport's origin.*/
    int     data_fore;          /* Data area foreground color. */
    int     data_back;          /* Data area background color. */
    int     border_type;        /* Viewport border type.     */
    int     border_fore;        /* Border foreground color.   */
    int     border_back;        /* Border background color.   */
    int     title_type;         /* Window title.             */
    int     title_fore;         /* Title Foreground.         */

```

```

int      title_back;          /* Title Background.          */
int      data_rows;           /* # rows in image.          */
int      data_columns;        /* # columns in image.       */
int      xref_fore;           /* Foreground/background attributes */
int      xref_back;           /* for cross reference items.  */
int      highlight_fore;      /* Foreground/background attributes */
int      highlight_back;      /* for highlight bar.         */
unsigned title_length;        /* Window title length.      */
unsigned image_length;        /* Screen image length.       */
} HL_RECORD_DATA;

```

typedef struct

```

{
    HL_RECORD_DATA record_sign; /* Record header.            */
    char      id_string[HL_ID_SIZE]; /* ID of help record.        */
    char      reserved;           /* Reserved.                  */
    unsigned long help_record;     /* File offset of help record. */
    unsigned long xref_list;       /* Cross reference list offset. */
    unsigned    description_len;   /* Length of description string. */
    unsigned long next_index;      /* Offset of next index record. */
} HL_INDEX_DATA;

```

typedef struct help\_index\_node

```

{
    HL_INDEX_DATA index_data;
    char      *pdescription;
    struct help_index_node *pleft;
    struct help_index_node *pright;
} HL_INDEX_NODE;

```

/\* User Macros. \*/

/\* Option codes for HLREAD and HLLOOKUP. \*/

```

#define HL_CHARS_ONLY    CHARS_ONLY
#define HL_CHAR_ATTR     CHAR_ATTR
#define HL_COMPRESSED    0x40
#define HL_REMOVE_WIN    0x10
#define HL_DESTROY_WIN   0x30
#define HL_KBIGNORE      (WN_KBIGNORE)
#define HL_USE_MOUSE     (WN_USE_MOUSE)
#define HL_NO_MOUSE      (WN_NO_MOUSE)
#define HL_TEXT_TYPE     (HL_CHARS_ONLY | HL_CHAR_ATTR | HL_COMPRESSED)

```

```

/*****
/* Global variables.
*/
*****/

extern HL_WINDOW b_def_help_win;
extern char b_help_path[];
extern HL_INDEX_NODE *b_phelp_index;

/*****
/* Error codes.
*/
*****/

#define HL_NO_ERROR (WN_NO_ERROR)
#define HL_NO_MEMORY (WN_NO_MEMORY)
#define HL_ILL_DIM (WN_ILL_DIM)
#define HL_NO_EXIT (WN_NO_EXIT)
#define HL_BAD_OPT (WN_BAD_OPT)
#define HL_NO_PATH 200
#define HL_NO_HANDLES 201
#define HL_BAD_DATABASE 202
#define HL_NO_INDEX 203
#define HL_NOT_FOUND 204
#define HL_VER_MISMATCH 205
#define HL_SYSTEM 206

/*****
/* User function declarations.
*/
*****/

int cdecl hlopen(const /* Open a help database and */
char *); /* construct its index in memory. */
/*
*/
int cdecl hlclose(void); /* Close the help database. */
/*
*/
WN_EVENT *cdecl hhread( /* Display a help window and accept */
BWINDOW *, /* user responses. */
const /*
HL_WINDOW */ /*
const char *, /*
int, /*
WN_EVENT *, /*
int); /*
/*
*/
int cdecl hllookup( /* Look up the help text and window */
const char *, /* information for a help record. */

```

```

        const char *, /*
        unsigned long,/*
        HL_WINDOW, /*
        char **,int *,/*
        int); /*
        /*
WN_EVENT *cdecl hldisp ( /* Look up help info, display it in */
        const char *, /* a help window and accept user */
        const char *, /* responses. */
        unsigned long,/*
        WN_EVENT *, /*
        int); /*
        /*
        /*

/*****
/* Internal function declarations and macros. */
/*****

void cdecl hlfrindx(HL_INDEX_NODE *);
int cdecl hlpas2c(const char *, char *, int);
#define hlerror(error) wnerror(error)
#define hlreterr(error) wnreterr(error)
#endif /* Ends "#ifndef DEF_BHELP" */

```

## BINTERV.H 插入函数的头文件。

```

#ifndef DEF_BINTERV /* Prevent redefinition. */
typedef struct /* IV_KEY: Specifies one hot key. */
{ /*
    char ch; /* Character value. */
    char keycode; /* Key code (pseudo-scan code). */
    int action; /* Action to be taken when this key is
                /* detected (see possibilities below). */
} IV_KEY;

/* Key actions */
/*
#define IV_KY_NONE 0 /* No hot key pressed. */
#define IV_KY_SERVICE 1 /* Invoke intervention function. */
#define IV_KY_SLEEP 2 /* Put intervention function to sleep. */
#define IV_KY_WAKE 3 /* Awaken intervention function. */
typedef struct /* IV_TIME: Specifies time of day or
{ /* time interval for intervention code.*/
    /*
    long ticks; /* Clock ticks since midnight or

```

```

/* between invocations. */
int action; /* Type of time event (moment or */
/* interval). */

} IV_TIME;

/* Time actions */
/* */

#define IV_TM_NONE 0 /* No time event occurred. */
#define IV_TM_INTERVAL 1 /* Regular interval between invocations.*/
#define IV_TM_MOMENT 2 /* Time of day. */
typedef struct /* IV_EVENT: Event that triggered the */
{ /* intervention function. */
/* */
IV_KEY key; /* Hot key found, if any. */
/* */
int time_action; /* Type of timer event (moment or */
/* interval), if any. */
int time_index; /* Index of timer event in table. */
long time; /* Current time of day, measured in */
/* clock ticks since midnight. */
} IV_EVENT;

typedef struct /* IV_VECTORS: set of interrupt */
{ /* vectors used by intervention code. */
/* */
const void far *ptimer; /* INT 08h: Timer tick. */
const void far *pkeybd; /* INT 09h: Keystroke. */
const void far *pdisk; /* INT 13h: BIOS disk services. */
const void far *pdos; /* INT 21h: DOS functions. */
const void far *pidle; /* INT 28h: DOS idle. */
/* */
const void far *pcom1; /* INT 0ch: Comm port 1 hardware event.*/
const void far *pcom2; /* INT 0bh: Comm port 2 hardware event.*/
} IV_VECTORS;

typedef struct /* IV_CTRL: intervention control block */
{ /* */
unsigned int signature; /* Signature identifying this structure.*/
unsigned int sign2; /* One's complement of "signature". */
char ident[16]; /* Identifying name. */
unsigned int psp; /* PSP of this program. */
int enabled; /* Nonzero if enabled (this prevents the*/
/* filters from being installed twice).*/
void far *pgate; /* Gateway for invoking this program */

```

```

/* from outside. */
IV_VECTOR$prev_vec; /* Previous values of interrupt vectors.*/
/* */
char dos_need; /* Nonzero if intervention function */
/* uses DOS services. */
char dkey_need; /* Nonzero if intervention function */
/* uses DOS functions 1-12. */
/* */
IV_KEY far *pkeytab; /* Table of hot keys. */
int ksize; /* Size of hot key table. */
/* */
IV_KEY key_event; /* Found key, if any. */
/* */
/* Internal state of filters: */
/* */
char wait; /* Nonzero if INT 21h or INT 28h filter */
/* is cycling, waiting for the */
/* intervention function to be invoked.*/
char req; /* Nonzero if intervention function */
/* needs invocation. */
char awake; /* Zero if intervention function is */
/* asleep. */
/* Busy flags: */
char timer_busy; /* INT 08h filter. */
char kb_busy; /* INT 09h filter. */
char com2_busy; /* INT 0bh filter. */
char com1_busy; /* INT 0ch filter. */
char disk_busy; /* INT 13h filter. */
char idle_busy; /* INT 28h filter. */
/* */
char idle_safe; /* Nonzero if INT 0x28 is in progress */
/* and DOS functions are available */
/* (but not functions 1-12). */
/* */
void far *pschedblk; /* Entry point into scheduler's ISR */
/* control block. */
/* */
const volatile unsigned /* Address of DOS critical section flag */
char far *pdos_crit; /* (busy flag). */
/* */
const void far * /* Address of previous interrupt 0x16 */
prev_16h_vec; /* handler (if replaced). */
/* */

```

```

char    cbreak;          /* Nonzero if Ctrl-C or Ctrl-Break      */
                          /* detected during this invocation of      */
                          /* the intervention function.              */
                          /*                                          */
unsigned char no_call;    /* Nonzero if user wants to prevent      */
                          /* invocation of intervention function.*/
                          /*                                          */
unsigned filter_mask;     /* Bit map of installed filters.          */
                          /*                                          */
unsigned char kb_ext;     /* Nonzero if extended BIOS keyboard     */
                          /* services to be used.                  */
                          /*                                          */
char    _res0;           /* Reserved for future use.              */
                          /*                                          */
char    _reserved[8];    /* Reserved for future use.              */
} IV_CTRL;

#define IV_SIGNATURE (('2' < 8) | 'B') /* "B2" */
/* Option values for IVINSTAL. */
#define IV_DOS_NEED 1 /* Intervention function uses DOS. */
#define IV_NO_DOS_NEED0 /* Intervention function doesn't use */
                          /* DOS. */
#define IV_DKEY_NEED 2 /* Intervention function uses DOS */
                          /* functions 1-12. */
#define IV_NO_DKEY_NEED /* Intervention function doesn't use */
                          /* DOS functions 1-12. */
#define IV_FLOAT_NEED 4 /* Intervention function uses floating */
                          /* point interrupt vectors while other */
                          /* programs do so also. */
#define IV_NO_FLOAT_NEED 0 /*
#define IV_DAILY 8 /* "Moment" events to recur daily.
/* Option values for IVVECS.
#define IV_RETVEC 1 /* Return current values of interrupt */
                          /* vectors. */
#define IV_SETVEC 0 /* Set interrupt vectors.
/* Symbols for error codes
#define IV_NO_ERROR 0
#define IV_NOT_ENABLED 1
#define IV_NOT_INSTALLED 2
#define IV_NULL_POINTER 3
#define IV_BAD_OPT 4
#define IV_INSTALLED 5
#define IV_PART_COVERED 6
#define IV_NOT_FOUND 7

```



```

/* Symbols for intervention filters (used for filter_mask). */
#define IV_F_TIMER 0x0001 /* INT 08h: Timer tick. */
#define IV_F_KEYSTROKE 0x0002 /* INT 09h: Keystroke. */
#define IV_F_DISK 0x0004 /* INT 13h: BIOS disk services.*/
#define IV_F_DOS 0x0008 /* INT 21h: DOS functions. */
#define IV_F_IDLE 0x0010 /* INT 28h: DOS idle. */
#define IV_F_2COM 0x0020 /* INT 0ch: Comm port 1 */
#define IV_F_1COM 0x0040 /* INT 0bh: Comm port 2 */
#define IV_STD_FILTERS
    (IV_F_TIMER | IV_F_KEYSTROKE | IV_F_DISK | IV_F_DOS | IV_F_IDLE)
#define IV_COM_FILTERS (IV_F_2COM | IV_F_1COM)

/* Internal symbols */
#define IV_RESCHED 0x40000000L /* Mask for rescheduling */
/* "moment" events. */

/* Global variables. */
extern int b_ivusex; /* Whether to use extended */
/* keyboard services when */
/* testing for hot keys: */
/* KB_USE_EXTEND or */
/* KB_USE_NORMAL. */
extern unsigned b_ivmask; /* Mask indicating which */
/* intervention filters to use. */

/* Function declarations */
/* Install intervention function*/
int cdecl ivinstal(void (*)(IV_EVENT *),
    const char *, /*
    char *,int, /*
    IV_KEY *,int, /*
    IV_TIME *,int, /*
    int); /*
/*
/* Detect presence of possibly */
/* obscured intervention */
/* function. */
/*
/*
/* Detect presence of removable */
/* intervention function. */
/*
/*
/* Disable intervention function*/
/*

```

```

int cdecl ivvecs(int, /* Set or return interrupt */
                  IV_VECTORS far *); /* vectors used by the */
/* intervention filters. */
/* */
IV_CTRL far * cdecl ivctrl(void); /* Return address of this */
/* program's intervention */
/* control block. */
#define DEF_BINTERV 1 /* Prevent second reading of */
/* these definitions. */
#endif /* Ends "#ifndef DEF_BINTERV" */

```

## BINTRUPT.H 中断服务函数的头文件。

```

#ifndef DEF_BINTRUPT /* Prevent redefinition. */
#define DEF_BINTRUPT 1
#include <stdlib.h> /* For size_t. */
#include <butil.h> /* For ALLREG. */
#include <bmem.h> /* For mmsize(). */

typedef struct /* ISRMSG: structure for */
{ /* passing messages between ISR */
/* and the dispatcher. */
/* */
int exit_style; /* Exit style: see possible */
/* values below. */
/* */
unsigned working_flags; /* Working value of flags */
/* register. This will be */
/* be propagated to an ISR's */
/* caller if the ISR returns */
/* without doing an IRET. */
} ISRMSG; /* */
/* Exit styles from an ISR: */
/* */
#define IEXIT_NORMAL 0 /* Normal: return to caller */
/* via IRET */
/* */
#define IEXIT_RETF 1 /* Special: return via far */
/* RETurn */

#define ISRCTRL struct isr_control /* ISRCTRL: ISR control block */
struct isr_control /* */
{ /* */
unsigned fcall_opcode; /* NOP + Far call opcode */

```

```

/* (0x9A90) */
void (far *isrdisp)(); /* Address of ISR dispatcher */
unsigned irect_opcode; /* IRET + RETF opcodes (0xcbcf) */
/* */
char far *isrstk; /* Beginning of space allocated */
/* for ISR stack(s). */
unsigned isrstksize; /* ISR stack size */
unsigned isrsp; /* ISR stack pointer value at */
/* start of current ISR call */
/* */
unsigned isrds; /* DS value required by ISR */
unsigned isres; /* ES value required by ISR */
void (far *isr)(ALLREG *, /* Address of ISR itself */
                ISRCTRL *, /* */
                ISRMSG *); /* */
unsigned isrsp; /* PSP of program containing ISR */
/* */
void far *prev_vec; /* Previous value of vector */
/* */
unsigned level; /* Number of calls in progress */
/* (0 if not in use) */
unsigned limit; /* Maximum number of nested */
/* calls */
unsigned signature; /* Signature identifying this */
/* structure */
unsigned sign2; /* One's complement of */
/* "signature" */
char ident[16]; /* Identifying name */
unsigned control; /* Bit fields to control */
/* dispatcher options */
unsigned status; /* Status info left by */
/* dispatcher */
char scratch[10]; /* Scratch space for use by */
/* dispatcher & related */
/* programs */

};
/* Offsets of various items in an ISRCTRL structure. */
#define ICB_ENTRY_OFFSET 0 /* Address to install in */
/* interrupt vector. */
#define ICB_PSP_OFFSET 24
#define ICB_PREV_OFFSET 26
#define ICB_SGN_OFFSET 34
#define ICB_S2_OFFSET 36

```

```

#define ICB_IDENT_OFFSET 38
    /* Control bits in ISRCTRL structure. */
#define ICB_NOCHECK 1 /* Don't adjust stack limit. */
#define ICB_SIGNATURE (('0' < 8) | 'B') /* "B0" */
typedef struct /* DEV_HEADER Device header */
{
    /* structure available to */
    /* DOS critical error (INT 0x24)*/
    /* handlers. */
    /* */
    void far *next_dev; /* Physical address of next */
    /* device header in chain */
    /* (0xffff:0xffff if end) */
    /* */
    unsigned attr; /* Attribute bits */
    /* */
    unsigned strategy; /* Offset of "strategy" routine */
    /* */
    unsigned service; /* Offset of service routine */
    /* */
    char name[8]; /* If character device, this is */
    /* its name; if block device, */
    /* name[0] is the number of */
    /* drives served by this driver.*/
} DEV_HEADER;
    /* Symbols for use with ISCURPRC */
#define IS_SETPROC1 /* Set currently executing process. */
#define IS_RETPROC0 /* Return currently executing process. */
    /* Error codes. */
#define IS_OK 0 /* No error detected. */
#define IS_NO_MEMORY 1 /* Insufficient memory. */
#define IS_RANGE 2 /* Value out of range. */
    /* Floating point interrupt vectors used by math library */
#define IS_1ST_FLOAT_VEC 0x34
#define IS_LAST_FLOAT_VEC 0x3e
#define IS_NUM_FLOAT_VECS (IS_LAST_FLOAT_VEC) - (IS_1ST_FLOAT_VEC) + 1)
    /* TURBO C TOOLS functions implemented as macros */
#define isgetvec(intype) ((void far *) getvect(intype))
#define isputvec(intype,ptr) (setvect((intype),(void interrupt (*)(ptr)))
#define isresext(excode) (keep((excode),mmsize()))
    /* Function declarations */
int cdecl iscall(void far *,ALLREG *);/* Simulate software interrupt
    /*
    /* Install a Turbo C TOOLS

```

```

int cdecl isinstal(int, /* interrupt service routine */
void (*)(ALLREG *,ISRCTRL *,ISRMSG *),
const char *, /* */
ISRCTRL *, /* */
char *, /* */
int, /* */
int); /* */
/* Prepare ISR control block */
void cdecl isprep(void (*)(ALLREG *,ISRCTRL *,ISRMSG *),
const char *, /* */
ISRCTRL *, /* */
char *, /* */
int, /* */
int); /* */
ISRCTRL far * cdecl issense( /* Detect presence of Turbo C */
void far *,const char *); /* TOOLS or C TOOLS PLUS or */
/* LIGHT TOOLS interrupt */
/* service routine */
/* */
int cdecl iremove(unsigned); /* Remove resident program */
/* */
unsigned cdecl iscurpre(int,unsigned); /* Return or set PSP of */
/* currently executing process. */
/* */
int cdecl isreserv(size_t,size_t, /* Allocate block; also reserve */
char **); /* space for malloc() from ISR. */
#endif /* Ends "#ifndef DEF_BINTRUPT" */

```

## BKEYBRD.H BIOS键盘函数的头文件。

```

#ifndef DEF_BKEYBRD /* Prevent redefinition. */
#define DEF_BKEYBRD 1 /* Prevent second reading of these */
/* definitions. */

#include <ctype.h>
#include <biofil.h>
#define KB_BIOS_INT 0x16 /* General BIOS keyboard gate. */
#define KB_HEAD 0 /* KBSTUFF, KBPLACE: Whether to */
/* place char(s) at HEAD or at */
#define KB_TAIL 1 /* TAIL of queue. */
/* KBPLACE error returns: */
#define KB_OK 0 /* no error, */

```

```

#define KB_FULL 1 /* buffer full, */
#define KB_PLACE 2 /* bad "where" specification. */
#define KB_DATASEG 0x40 /* Segment value of BIOS keyboard */
/* data. */
#define KB_HEADLOC 0x1a /* Location of queue head pointer. */
#define KB_TAILLOC 0x1c /* Location of queue tail pointer. */
/* */
#define KB_SHIFTLOC 0x17 /* Location of shift status register */
/* */
#define KB_BUFEND 0x3e /* Address of keyboard buffer end. */
#define KB_BUFSTART 0x1e /* Address of keyboard buffer start. */
/* */
/* Size of buffer in bytes. */
#define KB_BUF_SIZE (KB_BUFEND - KB_BUFSTART)
/* */
/* Size of buffer in keystrokes. */
/* */
#define KB_BUFAC_SIZE (((KB_BUFEND - KB_BUFSTART) >> 1) - 1)
/* Far pointer to buffer head word. */
#define KB_BUFHEADADDR (utfaru (KB_DATASEG, KB_HEADLOC))
/* */
/* Far pointer to buffer tail word. */
#define KB_BUFTAILADDR (utfaru (KB_DATASEG, KB_TAILLOC))
/* */
/* Far pointer to shift status byte. */
#define KB_SHIFTADDR (utfaru (KB_DATASEG, KB_SHIFTLOC))
#define KB_NO_EXTENDED /* Values for b_kbxten. */
#define KB_EXTENDED 1
#define KB_NO_ENHANCED /* Values for b_kbnhan. */
#define KB_ENHANCED 1
#define KB_USE_NORMAL /* Values for b_kbusex. */
#define KB_USE_EXTENDED
#define KB_ERROR (-1) /* Additional value returned by */
/* KBEXTEND. */
#define KB_NO_KEY_FOUND 0 /* Values for the key_found field */
#define KB_KEY_FOUND 1 /* of KB_DATA. */
/* Values to be placed in the control_action field */
/* of KB_DATA. */
#define KB_NO_REMOVE_KEY /* Do not remove key from buffer. */
#define KB_REMOVE_KEY 1 /* Remove key from head of buffer. */
#define KB_FLUSH 2 /* Flush the keyboard buffer. */
typedef struct kstatus /* KEYSTATUS structure: */
{ /* Maps keyboard status register */

```

```

/* onto unsigned integer. */
unsigned right_shift : 1; /* Right Shift key depressed. */
unsigned left_shift : 1; /* Left Shift key depressed. */
unsigned ctrl_shift : 1; /* Ctrl key depressed. */
unsigned alt_shift : 1; /* Alt key depressed. */
unsigned scroll_state : 1; /* Scroll Lock has been toggled. */
unsigned num_state : 1; /* Num Lock has been toggled. */
unsigned caps_state : 1; /* Caps Lock has been toggled. */
unsigned ins_state : 1; /* Insert state is active. */
/* */
unsigned filler : 3; /* Filler for word alignment. */
unsigned hold_state : 1; /* Suspend key has been toggled. */
unsigned scroll_shift : 1; /* Scroll Lock key depressed. */
unsigned num_shift : 1; /* Num Lock key depressed. */
unsigned caps_shift : 1; /* Caps Lock key depressed. */
unsigned ins_shift : 1; /* Insert key depressed. */
} KEYSTATUS;

typedef struct /* KEY_SEQUENCE structure: */
{ /* character & key codes for a key.*/
    unsigned char character_code;
    unsigned char key_code;
} KEY_SEQUENCE;

typedef struct /* KB_DATA structure: */
{ /* information for key control */
    /* function. */
    int key_found; /* Key was pressed if equal to */
    /* KB_KEY_FOUND. */
    KEY_SEQUENCE key_seq; /* Character and scan codes of key */
    /* (if key_found is KB_KEY_FOUND).*/
    void *pfunction_data; /* Pointer to user-supplied */
    /* information structure. */
    int control_action; /* Special action for key control */
    /* function to take. */
    int returned_action; /* Action to be taken on return */
    /* from key control function. */
    /* Either KB_REMOVE_KEY if the */
    /* caller should remove the */
    /* detected keystroke (if there */
    /* was one), or KB_NO_REMOVE_KEY */
    /* otherwise. */
} KB_DATA;

typedef void (*PKEY_CONTROL) /* PKEY_CONTROL: address of key */
(KB_DATA *); /* control function -- pointer */

```

```

/* to function returning void, */
/* accepting a pointer to KB_DATA.*/

/* Global variables */
extern unsigned char b_keycod [];
extern int b_kbxten; /* Presence of extended BIOS functions. */
extern int b_kbnhan; /* Presence of enhanced keyboard. */
extern int b_kbusex; /* Flag controlling whether to use
/* normal or extended keyboard functions*/
extern PKEY_CONTROL b_key_ctrl; /* Default key control procedure*/
/* for user input functions. */

/* Function declarations. */
/* Equipment options. */
int cdecl kbequip (void); /* Sense extended BIOS and
/* enhanced keyboard. */
int cdecl kbextend (int); /* Specify extended or
/* traditional services. */
/* Normal BIOS operations. */
int cdecl kbready (char *, int *); /* Check for a keystroke. */
char cdecl kbquery (char *, int, int *, /* Get a string from CON. */
int *); /* */
int cdecl kbgetkey (int *); /* Await & return char and
/* scan code via BIOS. */
int cdecl kbflush (void); /* Flush keyboard buffer. */
int cdecl kbqueue (int *); /* Total and # remaining in
/* keyboard queue. */
/* Key control functions. */
int cdecl kbpoll (PKEY_CONTROL, void *, /* Poll keyboard once for a */
KEY_SEQUENCE *, /* keystroke. */
int); /* */
KEY_SEQUENCE cdecl kbwait(PKEY_CONTROL, /* Await a keystroke using
void *); /* polling. */
void cdecl kbkeflush (PKEY_CONTROL, /* Call key control function*/
void *); /* to flush keyboard
/* buffer. */
/* Controlling keyboard. */
void cdecl kbset (const KEYSTATUS *); /* Set shift status. */
char *cdecl kbstuff (int, char *); /* Stuff string into queue. */
int cdecl kbplace (int, char, char); /* Place one key into queue.*/
#define kbstatus(pkeybd) (((int *) (pkeybd))) = utpeekw (KB_SHIFTADDR)
#define kbs_anof(c) (isascii ((int) c) \
? ((int) b_keycod [((int) (c))]) \
: (-1))
#define kbcharof(c) (isascii ((int) c) \

```



```

? ((int) (c))
: (-1))

```

```
endif
```

```
/* Ends "#ifndef DEF_BKEYBRD" */
```

## BKEYS.H 定义键码的头文件。

```
#ifndef DEF_BKEYS /* Prevent redefinition. */
```

```
#define DEF_BKEYS 1
```

```
#define KBNDEF (0xff)
```

```
/* The key names that follow are made of components: Example: */
```

```
/*
```

```
/* The character code for shift-up-arrow is: */
```

```
/* KB_C_S_UP */
```

```
/* ^ ^ ^ ^ */
```

```
/* | | | | */
```

```
/* | | | +----- Key name. */
```

```
/* | | | */
```

```
/* | | +----- S means shifted; can be (N for normal, S */
```

```
/* | | for shifted, C for control, or A for alt. */
```

```
/* | | */
```

```
/* | +----- C signifies character code. Can also be */
```

```
/* | S for scan code. */
```

```
/* | */
```

```
/* +----- This is a keyboard definition. */
```

```
/* */
```

```
#define KB_C_N_F1 0 /* Character code for normal F1 key */
```

```
#define KB_S_N_F1 59 /* Scan code for normal F1 key */
```

```
#define KB_C_S_F1 0 /* Character code for shift F1 key */
```

```
#define KB_S_S_F1 84 /* Scan code for shift F1 key */
```

```
#define KB_C_C_F1 0 /* Character code for ctrl F1 key */
```

```
#define KB_S_C_F1 94 /* Scan code for ctrl F1 key */
```

```
#define KB_C_A_F1 0 /* Character code for alt F1 key */
```

```
#define KB_S_A_F1 104 /* Scan code for alt F1 key */
```

```
#define KB_C_N_F2 0
```

```
#define KB_S_N_F2 60
```

```
#define KB_C_S_F2 0
```

```
#define KB_S_S_F2 85
```

```
#define KB_C_C_F2 0
```

```
#define KB_S_C_F2 95
```

```
#define KB_C_A_F2 0
```

```
#define KB_S_A_F2 105
```

#define KB\_C\_N\_F3 0  
#define KB\_S\_N\_F3 61  
#define KB\_C\_S\_F3 0  
#define KB\_S\_S\_F3 86  
#define KB\_C\_C\_F3 0  
#define KB\_S\_C\_F3 96  
#define KB\_C\_A\_F3 0  
#define KB\_S\_A\_F3 106

#define KB\_C\_N\_F4 0  
#define KB\_S\_N\_F4 62  
#define KB\_C\_S\_F4 0  
#define KB\_S\_S\_F4 87  
#define KB\_C\_C\_F4 0  
#define KB\_S\_C\_F4 97  
#define KB\_C\_A\_F4 0  
#define KB\_S\_A\_F4 107

#define KB\_C\_N\_F5 0  
#define KB\_S\_N\_F5 63  
#define KB\_C\_S\_F5 0  
#define KB\_S\_S\_F5 88  
#define KB\_C\_C\_F5 0  
#define KB\_S\_C\_F5 98  
#define KB\_C\_A\_F5 0  
#define KB\_S\_A\_F5 108

#define KB\_C\_N\_F6 0  
#define KB\_S\_N\_F6 64  
#define KB\_C\_S\_F6 0  
#define KB\_S\_S\_F6 89  
#define KB\_C\_C\_F6 0  
#define KB\_S\_C\_F6 99  
#define KB\_C\_A\_F6 0  
#define KB\_S\_A\_F6 109

#define KB\_C\_N\_F7 0  
#define KB\_S\_N\_F7 65  
#define KB\_C\_S\_F7 0  
#define KB\_S\_S\_F7 90  
#define KB\_C\_C\_F7 0  
#define KB\_S\_C\_F7 100  
#define KB\_C\_A\_F7 0

```
#define KB_S_A_F7 110
```

```
#define KB_C_N_F8 0
```

```
#define KB_S_N_F8 66
```

```
#define KB_C_S_F8 0
```

```
#define KB_S_S_F8 91
```

```
#define KB_C_C_F8 0
```

```
#define KB_S_C_F8 101
```

```
#define KB_C_A_F8 0
```

```
#define KB_S_A_F8 111
```

```
#define KB_C_N_F9 0
```

```
#define KB_S_N_F9 67
```

```
#define KB_C_S_F9 0
```

```
#define KB_S_S_F9 92
```

```
#define KB_C_C_F9 0
```

```
#define KB_S_C_F9 102
```

```
#define KB_C_A_F9 0
```

```
#define KB_S_A_F9 112
```

```
#define KB_C_N_F10 0
```

```
#define KB_S_N_F10 68
```

```
#define KB_C_S_F10 0
```

```
#define KB_S_S_F10 93
```

```
#define KB_C_C_F10 0
```

```
#define KB_S_C_F10 103
```

```
#define KB_C_A_F10 0
```

```
#define KB_S_A_F10 113
```

```
/* Note: The following 14 keys are located on the numeric keypad */
```

```
#define KB_C_N_HOME 0
```

```
#define KB_S_N_HOME 71
```

```
#define KB_C_S_HOME 55
```

```
#define KB_S_S_HOME 71
```

```
#define KB_C_C_HOME 0
```

```
#define KB_S_C_HOME 119
```

```
#define KB_C_A_HOME 0
```

```
#define KB_S_A_HOME ((unsigned char) 151)
```

```
#define KB_C_N_UP 0
```

```
#define KB_S_N_UP 72
```

```
#define KB_C_S_UP 56
```

```
#define KB_S_S_UP 72
```

```

#define KB_C_C_UP 0 /* Extended BIOS services only */
#define KB_S_C_UP ((unsigned char) 141) /* Ext. BIOS services only */
#define KB_C_A_UP KBNDEF
#define KB_S_A_UP KBNDEF

#define KB_C_N_PGUP 0
#define KB_S_N_PGUP 73
#define KB_C_S_PGUP 57
#define KB_S_S_PGUP 73
#define KB_C_C_PGUP 0
#define KB_S_C_PGUP ((unsigned char) 132)
#define KB_C_A_PGUP KBNDEF
#define KB_S_A_PGUP KBNDEF

#define KB_C_N_LEFT 0
#define KB_S_N_LEFT 75
#define KB_C_S_LEFT 52
#define KB_S_S_LEFT 75
#define KB_C_C_LEFT 0
#define KB_S_C_LEFT 115
#define KB_C_A_LEFT KBNDEF
#define KB_S_A_LEFT KBNDEF

#define KB_C_N_PAD5 0 /* Extended BIOS services only */
#define KB_S_N_PAD5 76 /* Extended BIOS services only */
#define KB_C_S_PAD5 53
#define KB_S_S_PAD5 76
#define KB_C_C_PAD5 0 /* Extended BIOS services only */
#define KB_S_C_PAD5 ((unsigned char) 143) /* Ext. BIOS only */
#define KB_C_A_PAD5 KBNDEF
#define KB_S_A_PAD5 KBNDEF

#define KB_C_N_RIGHT 0
#define KB_S_N_RIGHT 77
#define KB_C_S_RIGHT 54
#define KB_S_S_RIGHT 77
#define KB_C_C_RIGHT 0
#define KB_S_C_RIGHT 116
#define KB_C_A_RIGHT KBNDEF
#define KB_S_A_RIGHT KBNDEF

#define KB_C_N_END 0
#define KB_S_N_END 79

```

```

#define KB_C_S_END 49
#define KB_S_S_END 79
#define KB_C_C_END 0
#define KB_S_C_END 117
#define KB_C_A_END KBNDEF
#define KB_S_A_END KBNDEF

#define KB_C_N_DOWN 0
#define KB_S_N_DOWN 80
#define KB_C_S_DOWN 50
#define KB_S_S_DOWN 80
#define KB_C_C_DOWN 0          /* Extended BIOS services only */
#define KB_S_C_DOWN ((unsigned char) 145) /* Ext. BIOS only */
#define KB_C_A_DOWN KBNDEF
#define KB_S_A_DOWN KBNDEF

#define KB_C_N_PGDN 0
#define KB_S_N_PGDN 81
#define KB_C_S_PGDN 51
#define KB_S_S_PGDN 81
#define KB_C_C_PGDN 0
#define KB_S_C_PGDN 118
#define KB_C_A_PGDN KBNDEF
#define KB_S_A_PGDN KBNDEF

#define KB_C_N_INS 0
#define KB_S_N_INS 82
#define KB_C_S_INS 48
#define KB_S_S_INS 82
#define KB_C_C_INS 0          /* Extended BIOS services only */
#define KB_S_C_INS ((unsigned char) 146) /* Ext. BIOS only */
#define KB_C_A_INS KBNDEF
#define KB_S_A_INS KBNDEF

#define KB_C_N_DEL 0
#define KB_S_N_DEL 83
#define KB_C_S_DEL 46
#define KB_S_S_DEL 83
#define KB_C_C_DEL 0          /* Extended BIOS services only */
#define KB_S_C_DEL ((unsigned char) 147) /* Ext. BIOS services only */
#define KB_C_A_DEL KBNDEF
#define KB_S_A_DEL KBNDEF

```

```

#define KB_C_N_PADPLUS 43
#define KB_S_N_PADPLUS 78
#define KB_C_S_PADPLUS 43
#define KB_S_S_PADPLUS 78
#define KB_C_C_PADPLUS 0 /* Extended BIOS services only */
#define KB_S_C_PADPLUS ((unsigned char) 144) /* Ext. BIOS only */
#define KB_C_A_PADPLUS 0 /* Extended BIOS services only */
#define KB_S_A_PADPLUS 78 /* Extended BIOS services only */

#define KB_C_N_PADMINUS 45
#define KB_S_N_PADMINUS 74
#define KB_C_S_PADMINUS 45
#define KB_S_S_PADMINUS 74
#define KB_C_C_PADMINUS 0 /* Extended BIOS services only */
#define KB_S_C_PADMINUS ((unsigned char) 142) /* Ext. BIOS only */
#define KB_C_A_PADMINUS 0 /* Extended BIOS services only */
#define KB_S_A_PADMINUS 74 /* Extended BIOS services only */

#define KB_C_N_PRTSC 42 /* This is the PrtSc/* key; */
#define KB_S_N_PRTSC 55 /* do not confuse it with the */
#define KB_C_S_PRTSC KBNDEF /* * key on the numeric */
#define KB_S_S_PRTSC KBNDEF /* keypad of the Enhanced */
#define KB_C_C_PRTSC 0 /* Keyboard. */
#define KB_S_C_PRTSC 114
#define KB_C_A_PRTSC KBNDEF
#define KB_S_A_PRTSC KBNDEF

/* The following keys are located on the main keypad. */
#define KB_C_N_BACKSPACE 8
#define KB_S_N_BACKSPACE 14
#define KB_C_S_BACKSPACE 8
#define KB_S_S_BACKSPACE 14
#define KB_C_C_BACKSPACE 127
#define KB_S_C_BACKSPACE 14
#define KB_C_A_BACKSPACE 0 /* Extended BIOS services only */
#define KB_S_A_BACKSPACE 14 /* Extended BIOS services only */

#define KB_C_N_TAB 9
#define KB_S_N_TAB 15
#define KB_C_S_TAB 0
#define KB_S_S_TAB 15
#define KB_C_C_TAB 0 /* Extended BIOS services only */
#define KB_S_C_TAB ((unsigned char) 148) /* Ext. BIOS services only */

```

```

#define KB_C_A_TAB 0 /* Extended BIOS services only */
#define KB_S_A_TAB ((unsigned char) 165) /* Ext. BIOS services only */

#define KB_C_N_ENTER 13
#define KB_S_N_ENTER 28
#define KB_C_S_ENTER 13
#define KB_S_S_ENTER 28
#define KB_C_C_ENTER 10
#define KB_S_C_ENTER 28
#define KB_C_A_ENTER 0 /* Extended BIOS services only */
#define KB_S_A_ENTER ((unsigned char) 166) /* Ext. BIOS only */

#define KB_C_N_ESC 27
#define KB_S_N_ESC 1
#define KB_C_S_ESC 27
#define KB_S_S_ESC 1
#define KB_C_C_ESC 27
#define KB_S_C_ESC 1
#define KB_C_A_ESC 0
#define KB_S_A_ESC 1

#define KB_C_N_SPACE 32
#define KB_S_N_SPACE 57
#define KB_C_S_SPACE 32
#define KB_S_S_SPACE 57
#define KB_C_C_SPACE 32
#define KB_S_C_SPACE 57
#define KB_C_A_SPACE 32
#define KB_S_A_SPACE 57

#define KB_C_N_QUOTE 39
#define KB_S_N_QUOTE 40
#define KB_C_S_QUOTE 34
#define KB_S_S_QUOTE 40
#define KB_C_C_QUOTE KBNDEF
#define KB_S_C_QUOTE KBNDEF
#define KB_C_A_QUOTE 0 /* Extended BIOS services only */
#define KB_S_A_QUOTE 40 /* Extended BIOS services only */

#define KB_C_N_COMMA 44
#define KB_S_N_COMMA 51
#define KB_C_S_COMMA 60
#define KB_S_S_COMMA 51

```

```

#define KB_C_C_COMMA KBNDEF
#define KB_S_C_COMMA KBNDEF
#define KB_C_A_COMMA 0          /* Extended BIOS services only */
#define KB_S_A_COMMA 51        /* Extended BIOS services only */

#define KB_C_N_MINUS 45
#define KB_S_N_MINUS 12
#define KB_C_S_MINUS 95
#define KB_S_S_MINUS 12
#define KB_C_C_MINUS 31
#define KB_S_C_MINUS 12
#define KB_C_A_MINUS 0
#define KB_S_A_MINUS ((unsigned char) 130)

#define KB_C_N_PERIOD 46
#define KB_S_N_PERIOD 52
#define KB_C_S_PERIOD 62
#define KB_S_S_PERIOD 52
#define KB_C_C_PERIOD KBNDEF
#define KB_S_C_PERIOD KBNDEF
#define KB_C_A_PERIOD 0          /* Extended BIOS services only */
#define KB_S_A_PERIOD 52        /* Extended BIOS services only */

#define KB_C_N_SLASH 47
#define KB_S_N_SLASH 53
#define KB_C_S_SLASH 63
#define KB_S_S_SLASH 53
#define KB_C_C_SLASH KBNDEF
#define KB_S_C_SLASH KBNDEF
#define KB_C_A_SLASH 0          /* Extended BIOS services only */
#define KB_S_A_SLASH 53        /* Extended BIOS services only */

#define KB_C_N_0 48
#define KB_S_N_0 11
#define KB_C_S_0 41
#define KB_S_S_0 11
#define KB_C_C_0 KBNDEF
#define KB_S_C_0 KBNDEF
#define KB_C_A_0 0
#define KB_S_A_0 ((unsigned char) 129)

#define KB_C_N_1 49
#define KB_S_N_1 2

```



```
#define KB_C_S_1 33
#define KB_S_S_1 2
#define KB_C_C_1 KBNDEF
#define KB_S_C_1 KBNDEF
#define KB_C_A_1 0
#define KB_S_A_1 120
```

```
#define KB_C_N_2 50
#define KB_S_N_2 3
#define KB_C_S_2 64
#define KB_S_S_2 3
#define KB_C_C_2 0
#define KB_S_C_2 3
#define KB_C_A_2 0
#define KB_S_A_2 121
```

```
#define KB_C_N_3 51
#define KB_S_N_3 4
#define KB_C_S_3 35
#define KB_S_S_3 4
#define KB_C_C_3 KBNDEF
#define KB_S_C_3 KBNDEF
#define KB_C_A_3 0
#define KB_S_A_3 122
```

```
#define KB_C_N_4 52
#define KB_S_N_4 5
#define KB_C_S_4 36
#define KB_S_S_4 5
#define KB_C_C_4 KBNDEF
#define KB_S_C_4 KBNDEF
#define KB_C_A_4 0
#define KB_S_A_4 123
```

```
#define KB_C_N_5 53
#define KB_S_N_5 6
#define KB_C_S_5 37
#define KB_S_S_5 6
#define KB_C_C_5 KBNDEF
#define KB_S_C_5 KBNDEF
#define KB_C_A_5 0
#define KB_S_A_5 124
```

```
#define KB_C_N_6 54
#define KB_S_N_6 7
#define KB_C_S_6 94
#define KB_S_S_6 7
#define KB_C_C_6 30
#define KB_S_C_6 7
#define KB_C_A_6 0
#define KB_S_A_6 125
```

```
#define KB_C_N_7 55
#define KB_S_N_7 8
#define KB_C_S_7 38
#define KB_S_S_7 8
#define KB_C_C_7 KBNDEF
#define KB_S_C_7 KBNDEF
#define KB_C_A_7 0
#define KB_S_A_7 126
```

```
#define KB_C_N_8 56
#define KB_S_N_8 9
#define KB_C_S_8 42
#define KB_S_S_8 9
#define KB_C_C_8 KBNDEF
#define KB_S_C_8 KBNDEF
#define KB_C_A_8 0
#define KB_S_A_8 127
```

```
#define KB_C_N_9 57
#define KB_S_N_9 10
#define KB_C_S_9 40
#define KB_S_S_9 10
#define KB_C_C_9 KBNDEF
#define KB_S_C_9 KBNDEF
#define KB_C_A_9 0
#define KB_S_A_9 ((unsigned char) 128)
```

```
#define KB_C_N_SEMI 59
#define KB_S_N_SEMI 39
#define KB_C_S_SEMI 58
#define KB_S_S_SEMI 39
#define KB_C_C_SEMI KBNDEF
#define KB_S_C_SEMI KBNDEF
#define KB_C_A_SEMI 0
```

/\* Extended BIOS services only \*/

```

#define KB_S_A_SEMI 39          /* Extended BIOS services only */

#define KB_C_N_EQUALS 61
#define KB_S_N_EQUALS 13
#define KB_C_S_EQUALS 43
#define KB_S_S_EQUALS 13
#define KB_C_C_EQUALS KBNDEF
#define KB_S_C_EQUALS KBNDEF
#define KB_C_A_EQUALS 0
#define KB_S_A_EQUALS ((unsigned char) 131)

#define KB_C_N_A 97
#define KB_S_N_A 30
#define KB_C_S_A 65
#define KB_S_S_A 30
#define KB_C_C_A 1
#define KB_S_C_A 30
#define KB_C_A_A 0
#define KB_S_A_A 30

#define KB_C_N_B 98
#define KB_S_N_B 48
#define KB_C_S_B 66

```

```
#define KB_S_S_B 48
#define KB_C_C_B 2
#define KB_S_C_B 48
#define KB_C_A_B 0
#define KB_S_A_B 48
```

```
#define KB_C_N_C 99
#define KB_S_N_C 46
#define KB_C_S_C 67
#define KB_S_S_C 46
#define KB_C_C_C 3
#define KB_S_C_C 46
#define KB_C_A_C 0
#define KB_S_A_C 46
```

```
#define KB_C_N_D 100
#define KB_S_N_D 32
#define KB_C_S_D 68
#define KB_S_S_D 32
#define KB_C_C_D 4
#define KB_S_C_D 32
#define KB_C_A_D 0
#define KB_S_A_D 32
```

```
#define KB_C_N_E 101
#define KB_S_N_E 18
#define KB_C_S_E 69
#define KB_S_S_E 18
#define KB_C_C_E 5
#define KB_S_C_E 18
#define KB_C_A_E 0
#define KB_S_A_E 18
```

```
#define KB_C_N_F 102
#define KB_S_N_F 33
#define KB_C_S_F 70
#define KB_S_S_F 33
#define KB_C_C_F 6
#define KB_S_C_F 33
#define KB_C_A_F 0
#define KB_S_A_F 33
```

```
#define KB_C_N_G 103
```

#define KB\_S\_N\_G 34  
#define KB\_C\_S\_G 71  
#define KB\_S\_S\_G 34  
#define KB\_C\_C\_G 7  
#define KB\_S\_C\_G 34  
#define KB\_C\_A\_G 0  
#define KB\_S\_A\_G 34

#define KB\_C\_N\_H 104  
#define KB\_S\_N\_H 35  
#define KB\_C\_S\_H 72  
#define KB\_S\_S\_H 35  
#define KB\_C\_C\_H 8  
#define KB\_S\_C\_H 35  
#define KB\_C\_A\_H 0  
#define KB\_S\_A\_H 35

#define KB\_C\_N\_I 105  
#define KB\_S\_N\_I 23  
#define KB\_C\_S\_I 73  
#define KB\_S\_S\_I 23  
#define KB\_C\_C\_I 9  
#define KB\_S\_C\_I 23  
#define KB\_C\_A\_I 0  
#define KB\_S\_A\_I 23

#define KB\_C\_N\_J 106  
#define KB\_S\_N\_J 36  
#define KB\_C\_S\_J 74  
#define KB\_S\_S\_J 36  
#define KB\_C\_C\_J 10  
#define KB\_S\_C\_J 36  
#define KB\_C\_A\_J 0  
#define KB\_S\_A\_J 36

#define KB\_C\_N\_K 107  
#define KB\_S\_N\_K 37  
#define KB\_C\_S\_K 75  
#define KB\_S\_S\_K 37  
#define KB\_C\_C\_K 11  
#define KB\_S\_C\_K 37  
#define KB\_C\_A\_K 0  
#define KB\_S\_A\_K 37

```
#define KB_C_N_L 108
#define KB_S_N_L 38
#define KB_C_S_L 76
#define KB_S_S_L 38
#define KB_C_C_L 12
#define KB_S_C_L 38
#define KB_C_A_L 0
#define KB_S_A_L 38
```

```
#define KB_C_N_M 109
#define KB_S_N_M 50
#define KB_C_S_M 77
#define KB_S_S_M 50
#define KB_C_C_M 13
#define KB_S_C_M 50
#define KB_C_A_M 0
#define KB_S_A_M 50
```

```
#define KB_C_N_N 110
#define KB_S_N_N 49
#define KB_C_S_N 78
#define KB_S_S_N 49
#define KB_C_C_N 14
#define KB_S_C_N 49
#define KB_C_A_N 0
#define KB_S_A_N 49
```

```
#define KB_C_N_O 111
#define KB_S_N_O 24
#define KB_C_S_O 79
#define KB_S_S_O 24
#define KB_C_C_O 15
#define KB_S_C_O 24
#define KB_C_A_O 0
#define KB_S_A_O 24
```

```
#define KB_C_N_P 112
#define KB_S_N_P 25
#define KB_C_S_P 80
#define KB_S_S_P 25
#define KB_C_C_P 16
#define KB_S_C_P 25
```

```

#define KB_C_A_P 0
#define KB_S_A_P 25

#define KB_C_N_Q 113
#define KB_S_N_Q 16
#define KB_C_S_Q 81
#define KB_S_S_Q 16
#define KB_C_C_Q 17
#define KB_S_C_Q 16
#define KB_C_A_Q 0
#define KB_S_A_Q 16

#define KB_C_N_R 114
#define KB_S_N_R 19
#define KB_C_S_R 82
#define KB_S_S_R 19
#define KB_C_C_R 18
#define KB_S_C_R 19
#define KB_C_A_R 0
#define KB_S_A_R 19

#define KB_C_N_S 115
#define KB_S_N_S 31
#define KB_C_S_S 83
#define KB_S_S_S 31
#define KB_C_C_S 19
#define KB_S_C_S 31
#define KB_C_A_S 0
#define KB_S_A_S 31

#define KB_C_N_T 116
#define KB_S_N_T 20
#define KB_C_S_T 84
#define KB_S_S_T 20
#define KB_C_C_T 20
#define KB_S_C_T 20
#define KB_C_A_T 0
#define KB_S_A_T 20

#define KB_C_N_U 117
#define KB_S_N_U 22
#define KB_C_S_U 85
#define KB_S_S_U 22

```

```
#define KB_C_C_U 21
#define KB_S_C_U 22
#define KB_C_A_U 0
#define KB_S_A_U 22
```

```
#define KB_C_N_V 118
#define KB_S_N_V 47
#define KB_C_S_V 86
#define KB_S_S_V 47
#define KB_C_C_V 22
#define KB_S_C_V 47
#define KB_C_A_V 0
#define KB_S_A_V 47
```

```
#define KB_C_N_W 119
#define KB_S_N_W 17
#define KB_C_S_W 87
#define KB_S_S_W 17
#define KB_C_C_W 23
#define KB_S_C_W 17
#define KB_C_A_W 0
#define KB_S_A_W 17
```

```
#define KB_C_N_X 120
#define KB_S_N_X 45
#define KB_C_S_X 88
#define KB_S_S_X 45
#define KB_C_C_X 24
#define KB_S_C_X 45
#define KB_C_A_X 0
#define KB_S_A_X 45
```

```
#define KB_C_N_Y 121
#define KB_S_N_Y 21
#define KB_C_S_Y 89
#define KB_S_S_Y 21
#define KB_C_C_Y 25
#define KB_S_C_Y 21
#define KB_C_A_Y 0
#define KB_S_A_Y 21
```

```
#define KB_C_N_Z 122
#define KB_S_N_Z 44
```



```

#define KB_C_S_Z 90
#define KB_S_S_Z 44
#define KB_C_C_Z 26
#define KB_S_C_Z 44
#define KB_C_A_Z 0
#define KB_S_A_Z 44

#define KB_C_N_LBRACKET 91
#define KB_S_N_LBRACKET 26
#define KB_C_S_LBRACKET 123
#define KB_S_S_LBRACKET 26
#define KB_C_C_LBRACKET 27
#define KB_S_C_LBRACKET 26
#define KB_C_A_LBRACKET 0      /* Extended BIOS services only */
#define KB_S_A_LBRACKET 26     /* Extended BIOS services only */

#define KB_C_N_BACKSLASH 92
#define KB_S_N_BACKSLASH 43
#define KB_C_S_BACKSLASH 124
#define KB_S_S_BACKSLASH 43
#define KB_C_C_BACKSLASH 28
#define KB_S_C_BACKSLASH 43
#define KB_C_A_BACKSLASH 0     /* Extended BIOS services only */
#define KB_S_A_BACKSLASH 43    /* Extended BIOS services only */

#define KB_C_N_RBRACKET 93
#define KB_S_N_RBRACKET 27
#define KB_C_S_RBRACKET 125
#define KB_S_S_RBRACKET 27
#define KB_C_C_RBRACKET 29
#define KB_S_C_RBRACKET 27
#define KB_C_A_RBRACKET 0     /* Extended BIOS services only */
#define KB_S_A_RBRACKET 27    /* Extended BIOS services only */

#define KB_C_N_BACKQUOTE 96
#define KB_S_N_BACKQUOTE 41
#define KB_C_S_BACKQUOTE 126
#define KB_S_S_BACKQUOTE 41
#define KB_C_C_BACKQUOTE KBNDEF
#define KB_S_C_BACKQUOTE KBNDEF
#define KB_C_A_BACKQUOTE 0     /* Extended BIOS services only */
#define KB_S_A_BACKQUOTE 41    /* Extended BIOS services only */

```

```

/* The following keys are present only on the Enhanced Keyboard */
/* (101 or 102 keys). All of them require extended BIOS services */
/* to be distinguished from the older keys listed above. */
#define KB_C_N_F11 0
#define KB_S_N_F11 ((unsigned char) 133)
#define KB_C_S_F11 0
#define KB_S_S_F11 ((unsigned char) 135)
#define KB_C_C_F11 0
#define KB_S_C_F11 ((unsigned char) 137)
#define KB_C_A_F11 0
#define KB_S_A_F11 ((unsigned char) 139)

#define KB_C_N_F12 0
#define KB_S_N_F12 ((unsigned char) 134)
#define KB_C_S_F12 0
#define KB_S_S_F12 ((unsigned char) 136)
#define KB_C_C_F12 0
#define KB_S_C_F12 ((unsigned char) 138)
#define KB_C_A_F12 0
#define KB_S_A_F12 ((unsigned char) 140)

#define KB_C_N_PADSLASH 47
#define KB_S_N_PADSLASH ((unsigned char) 224)
#define KB_C_S_PADSLASH 47
#define KB_S_S_PADSLASH ((unsigned char) 224)
#define KB_C_C_PADSLASH 0
#define KB_S_C_PADSLASH ((unsigned char) 149)
#define KB_C_A_PADSLASH 0
#define KB_S_A_PADSLASH ((unsigned char) 164)

#define KB_C_N_PADENTER 13
#define KB_S_N_PADENTER ((unsigned char) 224)
#define KB_C_S_PADENTER 13
#define KB_S_S_PADENTER ((unsigned char) 224)
#define KB_C_C_PADENTER 10
#define KB_S_C_PADENTER ((unsigned char) 224)
#define KB_C_A_PADENTER 0
#define KB_S_A_PADENTER ((unsigned char) 166)

#define KB_C_N_PADASTERISK 42
#define KB_S_N_PADASTERISK 55
#define KB_C_S_PADASTERISK 42
#define KB_S_S_PADASTERISK 55

```

```
#define KB_C_C_PADASTERISK 0      /* Extended BIOS services only */
#define KB_S_C_PADASTERISK ((unsigned char) 150) /* Ext. BIOS only */
#define KB_C_A_PADASTERISK 0      /* Extended BIOS services only */
#define KB_S_A_PADASTERISK 55     /* Extended BIOS services only */
```

```
#define KB_C_N_NEW_HOME ((unsigned char) 224)
#define KB_S_N_NEW_HOME 71
#define KB_C_S_NEW_HOME KBNDEF
#define KB_S_S_NEW_HOME KBNDEF
#define KB_C_C_NEW_HOME ((unsigned char) 224)
#define KB_S_C_NEW_HOME 119
#define KB_C_A_NEW_HOME ((unsigned char) 224)
#define KB_S_A_NEW_HOME ((unsigned char) 151)
```

```
#define KB_C_N_NEW_UP ((unsigned char) 224)
#define KB_S_N_NEW_UP 72
#define KB_C_S_NEW_UP KBNDEF
#define KB_S_S_NEW_UP KBNDEF
#define KB_C_C_NEW_UP ((unsigned char) 224)
#define KB_S_C_NEW_UP ((unsigned char) 141)
#define KB_C_A_NEW_UP 0
#define KB_S_A_NEW_UP ((unsigned char) 152)
```

```
#define KB_C_N_NEW_PGUP ((unsigned char) 224)
#define KB_S_N_NEW_PGUP 73
#define KB_C_S_NEW_PGUP KBNDEF
#define KB_S_S_NEW_PGUP KBNDEF
#define KB_C_C_NEW_PGUP ((unsigned char) 224)
#define KB_S_C_NEW_PGUP ((unsigned char) 132)
#define KB_C_A_NEW_PGUP 0
#define KB_S_A_NEW_PGUP ((unsigned char) 153)
```

```
#define KB_C_N_NEW_LEFT ((unsigned char) 224)
#define KB_S_N_NEW_LEFT 75
#define KB_C_S_NEW_LEFT KBNDEF
#define KB_S_S_NEW_LEFT KBNDEF
#define KB_C_C_NEW_LEFT ((unsigned char) 224)
#define KB_S_C_NEW_LEFT 115
#define KB_C_A_NEW_LEFT 0
#define KB_S_A_NEW_LEFT ((unsigned char) 155)
```

```
#define KB_C_N_NEW_RIGHT ((unsigned char) 224)
#define KB_S_N_NEW_RIGHT 77
```

```

#define KB_C_S_NEW_RIGHT KBNDEF
#define KB_S_S_NEW_RIGHT KBNDEF
#define KB_C_C_NEW_RIGHT ((unsigned char) 224)
#define KB_S_C_NEW_RIGHT 116
#define KB_C_A_NEW_RIGHT 0
#define KB_S_A_NEW_RIGHT ((unsigned char) 157)

#define KB_C_N_NEW_END ((unsigned char) 224)
#define KB_S_N_NEW_END 79
#define KB_C_S_NEW_END KBNDEF
#define KB_S_S_NEW_END KBNDEF
#define KB_C_C_NEW_END ((unsigned char) 224)
#define KB_S_C_NEW_END 117
#define KB_C_A_NEW_END ((unsigned char) 224)
#define KB_S_A_NEW_END ((unsigned char) 159)

#define KB_C_N_NEW_DOWN ((unsigned char) 224)
#define KB_S_N_NEW_DOWN 80
#define KB_C_S_NEW_DOWN KBNDEF
#define KB_S_S_NEW_DOWN KBNDEF
#define KB_C_C_NEW_DOWN ((unsigned char) 224)
#define KB_S_C_NEW_DOWN ((unsigned char) 145)
#define KB_C_A_NEW_DOWN 0
#define KB_S_A_NEW_DOWN ((unsigned char) 160)

#define KB_C_N_NEW_PGDN ((unsigned char) 224)
#define KB_S_N_NEW_PGDN 81
#define KB_C_S_NEW_PGDN KBNDEF
#define KB_S_S_NEW_PGDN KBNDEF
#define KB_C_C_NEW_PGDN ((unsigned char) 224)
#define KB_S_C_NEW_PGDN 118
#define KB_C_A_NEW_PGDN 0
#define KB_S_A_NEW_PGDN ((unsigned char) 161)

#define KB_C_N_NEW_INS ((unsigned char) 224)
#define KB_S_N_NEW_INS 82
#define KB_C_S_NEW_INS KBNDEF
#define KB_S_S_NEW_INS KBNDEF
#define KB_C_C_NEW_INS ((unsigned char) 224)
#define KB_S_C_NEW_INS ((unsigned char) 146)
#define KB_C_A_NEW_INS 0
#define KB_S_A_NEW_INS ((unsigned char) 162)

```

```

#define KB_C_N_NEW_DEL ((unsigned char) 224)
#define KB_S_N_NEW_DEL 83
#define KB_C_S_NEW_DEL KBNDEF
#define KB_S_S_NEW_DEL KBNDEF
#define KB_C_C_NEW_DEL ((unsigned char) 224)
#define KB_S_C_NEW_DEL ((unsigned char) 147)
#define KB_C_A_NEW_DEL 0
#define KB_S_A_NEW_DEL ((unsigned char) 163)

/* The following key is present on the 102-key keyboard only.      */
/* It is next to the left shift key.                                */

#define KB_C_N_45 92
#define KB_S_N_45 86
#define KB_C_S_45 124
#define KB_S_S_45 86
#define KB_C_C_45 KBNDEF
#define KB_S_C_45 KBNDEF
#define KB_C_A_45 KBNDEF
#define KB_S_A_45 KBNDEF

#endif

```

## BLAISE.H 所有函数的头文件。

---

```

#include <bedit.h>          /* ED: Field editing functions.      */
                           /* Also includes BKEYBRD.H,          */
                           /* BSCREENS.H, BWINDOW.H.           */
                           /* BKEYBRD.H includes CTYPE.H and */
                           /* BUTIL.H; BUTIL.H includes       */
                           /* STRING.H and STDLIB.H.         */

#include <bfiles.h>          /* FL: File handling functions.      */
                           /* Also includes BUTIL.H.           */

#include <bhelp.h>           /* HL: Help system functions.        */
                           /* Also includes BWINDOW.H.         */

#include <binterv.h>         /* IV: Intervention code functions. */

#include <bintrpt.h>         /* IS: Interrupt service functions. */
                           /* Also includes BUTIL.H, STDLIB.H, */
                           /* BMEM.H.                         */

#include <bkeybrd.h>         /* KB: Direct keyboard functions.    */
                           /* Also includes CTYPE.H, BUTIL.H.  */

#include <bkeys.h>           /* Keyboard scan and character.      */
                           /* code definitions.                 */

```

```

#include <bmenu.h>          /* MN: Highlight bar menu      */
                           /* functions. Also includes   */
                           /* BKEYBRD.H, BWINDOW.H.     */
#include <bmouse.h>         /* MO: Mouse interface functions. */
                           /* Also includes BKEYBRD.H,   */
                           /* BINTRUPT.H.               */
#include <bprint.h>         /* PR: Printer control functions */
                           /* Also includes BIOS.H.     */
#include <bstrings.h>       /* ST: String functions.        */
#include <bvideo.h>         /* VI: Direct video functions   */
                           /* Also includes STRING.H,    */
                           /* BSCREENS.H.               */

```

## BMEM.H 内存管理函数的头文件。

---

```

#ifndef DEF_BMEM           /* Prevent redefinition.      */
typedef struct             /* MEMCTRLDOS memory control block */
{
    /* structure. */
    /*
    char    _dummy;        /* Dummy to force word alignment of
                           /* unsigned members, so no waste
                           /* space appears within structure.
                           /*
                           /* Actual 16-byte copy of control block */
                           /* starts here:
                           /*
    char    ident;         /* Identifying signature.
    unsigned owner_psp;    /* Segment address of owner process's
                           /* PSP.
    unsigned size;         /* Size of memory block in paragraphs.
    char    _reserved[11];
} MEMCTRL;
    /* Function declarations
unsigned cdecl mmsize(void);    /* Report program size.
    /*
int cdecl mmctrl(unsigned, MEMCTRL *, /* Fetch memory control block
    unsigned *);              /* and advance to next.
    /*
unsigned cdecl mmfirst(void);    /* Report first DOS memory
    /* block.
#define DEF_BMEM             /* Prevent second reading of
    /* these definitions.
#endif                       /* Ends "#ifndef DEF_BMEM"

```

## BMENU.H 菜单函数的头文件

---

```
#ifndef DEF_BMENU          /* Prevent redefinition.          */
#define DEF_BMENU          /* Prevent second reading of          */
                           /* these definitions.                */

#include <bkeybrd.h>
#include <bwindow.h>

/* Error constants specific to MENU operations.          */
#define MN_BAD_MENU 100    /* Menu failed test by MNVALMNU.    */
#define MN_NONE_SELECTED 1 /* No selectable item on menu.      */
#define MN_KEY_CONFLICT 102 /* A key not bound to a location    */
                           /* already exists with the given    */
                           /* character and key codes.         */
#define MN_NO_ITEM 103    /* No item exists at the given      */
                           /* coordinates.                     */
#define MN_BAD_KEY 104    /* Invalid keymap entry encountered.*/
#define MN_BAD_ITEM 105   /* Invalid item encountered.        */
#define MN_NO_EXIT 106    /* No enabled key has either the    */
                           /* transmit or abort bit set.       */
#define MN_READ_AB 110    /* User pressed a key with MN_ABORT */
                           /* action associated with it.       */
#define MN_UNKNOWN_AB 111 /* User pressed a key unknown to    */
                           /* the menu, and MN_UNKNOWN_TRANSMIT*/
                           /* was an option passed to a menu   */
                           /* reader function.                 */
#define MN_NO_MOUSE_EVENT 120 /* No matching mouse event found.   */
#define MN_MOUSE_CONFLICT 121 /* Cannot add an existing mouse    */
                           /* event.                           */
#define MN_BAD_MOUSE 122  /* Invalid mouse node.              */
#define MN_XMIT_NOBAR 130  /* Transmit without highlight bar.  */
#define MN_KEY_SIGN 0x0123 /* Arbitrary signature words.      */
#define MN_ITEM_SIGN 0x0320
#define MN_MOUSE_SIGN 0x4567
#define MN_DEAD_KEY 0xffef /* Arbitrary "deleted" signature   */
#define MN_DEAD_ITEM 0x0000 /* words for same.                  */
#define MN_DEAD_MOUSE 0x6745

/* Constants for use with MNKEY (the "howchange" argument). */
#define MN_ADD 0
#define MN_CHANGE
#define MN_DELETE

/* Constants for use with MNKEY (the "action" argument).    */
#define MN_NOMOVE 0x000
```

```

#define MN_UP          0x001
#define MN_DOWN        0x002
#define MN_RIGHT       0x003
#define MN_LEFT        0x004
#define MN_NEXT        0x005
#define MN_PREVIOUS    0x006
#define MN_FIRST       0x007
#define MN_LAST        0x008
#define MN_SELECT      0x009
#define MN_PGUP        0x00a
#define MN_PGDN        0x00b
#define MN_PGRIGHT     0x00c
#define MN_PGLEFT      0x00d
#define MN_TRANSMIT    0x010
#define MN_BEEP        0x020
#define MN_TEMP_DISABLE 0x040
#define MN_DISABLE     0x080
#define MN_ABORT       0x100
#define MN_KBIGNORE    0x200    /* Also duplicated below.    */
#define MN_HIDE_BAR    0x800    /* Also duplicated below.    */
#define MN_SHOW_BAR    0x1000   /* Also duplicated below.    */
#define MN_NOWRAP      0x0400
#define MN_MOVE_MASK   0x00f
#define MNMOVE(action) ((action) & MN_MOVE_MASK)
#define MN_INVALID_ACTION 0x4f3

/* Constants for use with adding/changing of items.    */
#define MN_PROTECT     0x0001
#define MN_NOPROTECT 0x0000
#define MN_CHAR_ATTR   0x0002
#define MN_CHARS_ONLY 0x0000

/* Constants for use with MNREAD (the "option" value). */
#define MN_REMOVE      0x01
#define MN_DESTROY     0x03
#define MN_KEEP_HIGHLIGHT 0x04
#define MN_ALL_TRANSMIT 0x08
#define MN_UNKNOWN_BEEP 0x10
#define MN_UNKNOWN_TRANSMIT 0x1f
#define MN_KEEP_DESCRIPTION 0x40
#define MN_HOLD_BEEP?   0x80

#define MN_KBIGNORE     0x200    /* Also duplicated above.    */

```



```

#define MN_KBDISCARD    0x400    /* Internal use only.          */
#define MN_HIDE_BAR     0x800    /* Also duplicated above.    */
#define MN_SHOW_BAR     0x1000   /* Also duplicated above.    */
#define MN_PREV_BAR     0x2000
#define MN_USE_DESCRIPTION 0x4000

/* Constants for use with MNATR, MNHILITE.          */
#define MN_UNHIGHLIGHT  0x00
#define MN_HIGHLIGHT    0x01

/* Menu mouse styles.                              */
#define MN_MOU_CLICK    1
#define MN_MOU_DRAG     2
#define MN_MOU_ALT_DRAG 3

/* Definitive signature for signature field in BMENU structure. */
#define BMENU_SIGN      0xf001
#define BMENU_DEAD      0

/* Mouse events in addition to those defined in BMOUSE.H.      */
#define MN_OUT_MENU     0x0100
#define MN_ITEMS        0x0200
#define MN_OFF_ITEMS    0x0400

/* User macros.                                           */

#define mncreate(height,width,textattr,hilattr,proattr,longattr) \
    (mncreat0((height),(width),(textattr),(hilattr), \
              (proattr),(longattr),(BMENU_SIGN)))
#define mndisplay(pmenu,pwhere,pbord) \
    (mnvdisp((pmenu),(pwhere), \
              wndata_h((pmenu)->pwin),wndata_w((pmenu)->pwin), \
              0,0,(pbord)))

#define mnread(pmenu,srow,scol,prow,pcol,pch,pscan,option) \
    (mnread((pmenu),(srow),(scol),(prow),(pcol),(pch),(pscan), \
            (option) | MN_USE_DESCRIPTION))

/* Internal Macros.                                       */
#define mnavaildm(pmenu) \
    {

```

```

        if (mnvalmnu (pmenu) == NIL)
            wnreterr (MN_BAD_MENU);
    }

#define mntrunc(pmenu, length, col)
    (((length + col) > pmenu->pwin->img.dim.w)
     ? (pmenu->pwin->img.dim.w - col)
     : (length))

#define mnvalmnu(pmenu) (mnvalmn0((pmenu),(BMENU_SIGN)))

/* Definitions of data types. */

/* BITEM -- data structure for an item in a menu. */
typedef struct bitem
{
    unsigned signature; /* Characteristic signature. */
    int row, col; /* Coordinates (relative to window) */
    /* of menu item text. */
    int lrow, lcol; /* Row and column at which to put */
    /* long description. */
    int attr; /* Attribute of this item. */
    int len; /* Length of ASCII text of menu */
    /* item. */
    int llen; /* Length of ASCII text of menu */
    /* description. */
    int option; /* Bitwise OR-ing of: */
    /* MN_PROTECT (protected item) */
    /* MN_NOPROTECT (unprotected item)*/
    char *plstring; /* Long description of item. Used */
    /* by "Lotus" menu functions. */
    char *pcharattr; /* Pointer to mixed character/ */
    /* attribute item string. */
    struct bitem *next; /* Pointer to next item */
} BITEM; /* specification in linked list. */

/* BKEYMAP -- data structure for a key associated */
/* with a menu. */
typedef struct bkeymap
{
    unsigned signature; /* Characteristic signature. */
    int ch; /* Character and scan code with */
    int scan; /* which the following action is to */

```

```

/* be associated with. */
/* */
int    row, col; /* The row and column this keycode */
/* is associated with. */
/* */
int    action; /* The action which should be taken */
/* when (ch,scan) is received by */
/* MNREAD: */
/* */
/* The bitwise OR-ing of the */
/* following: */
/* MN_TRANSMIT, MN_BEEP, */
/* MN_SELECT, MN_DISABLE, */
/* and MN_ABORT, */
/* */
/* OR-ed with only one of: */
/* */
/* MN_UP, MN_DOWN, MN_RIGHT, */
/* MN_LEFT, MN_NEXT, */
/* MN_PREVIOUS, MN_FIRST, */
/* MN_LAST, or MN_NOMOVE. */
/* */
struct bkeymap *next; /* Pointer to next key in keymap. */
} BKEYMAP;

```

```

/* BKEYTAB -- data structure for a default key table*/
/* entry. */
typedef struct bkeytab
{
    int    ch; /* Character and scan code with */
    int    scan; /* which the following action is to */
/* be associated with. */
/* */
    int    action; /* The action which should be taken */
/* when (ch,scan) is received by */
/* MNREAD: (see above). */
} BKEYTAB;

```

```

typedef struct bmnmouse_node /* BMNMOUSEMenu mouse event. */
{
    unsigned    signature;
    unsigned long event;
    unsigned long ignore;
}

```

```

    int          action;
    struct bmnmouse_node *pNext;
    struct bmnmouse_node *pprev;
} BMNMOUSE;

/* BMENU -- Data structure which governs behavior */
/*          and appearance of a menu.             */

typedef struct
{
    unsigned signature; /* Characteristic signature. */
                        /*                               */
    BWINDOW *pwin;      /* The window we use. */
                        /*                               */
    BITEM *pitems;      /* The items to display in the */
                        /* window.                  */
                        /*                               */
    BKEYMAP *pkeys;     /* The key definitions used. */
                        /*                               */
    int hilattr;        /* Highlight bar attributes. */
    int proattr;        /* Protected attributes. */
    int longattr;       /* Description attributes. */
    BMNMOUSE *pmouse;   /* List of mouse events. */
    BITEM *pbar_item;   /* Currently highlighted item, or */
                        /* NIL if none.                */
    BITEM *pbar_prev;   /* Previously highlighted item if */
                        /* pbar_item is NIL.           */

    struct          /* Internal items. */
    {
        unsigned desc_shown: 1; /* Lotus description shown. */
        unsigned _dummy :15; /* Pad to word boundary. */
    } internals;
} BMENU;

/* Most recent mouse menu event. */
extern unsigned long b_mnmoevent; /* Event, or OL if none. */
extern unsigned b_mnmovvert; /* Row and column where event took */
extern unsigned b_mnmohoriz; /* place, or 0xffff if none. */

/* Pointer to current default key map. */
extern BKEYTAB *b_mnkeytab;

/* Original default key map. */

```

```
extern BKEYTAB b_mndefkey [];
```

```

/* Function declarations -- User functions. */
BMENU *cdecl mnvdisp(BMENU *,/* Display a menu. Necessary */
    const WHERE *,int,int,int,int,/* before using MNREAD. */
    const BORDER *); /* */

int cdecl mndstroy (BMENU *); /* Release menu data structures, */
    /* undisplay if necessary. */

BMENU *cdecl mnitem (BMENU *,/* Add/Change/Delete item in menu. */
    int, int, int, /*
    const char *); /* */

BMENU *cdecl mnitemkey (BMENU *, /* Add item and key bindings for */
    int, int, int, /* that item to menu. */
    const char *, /*
    const char *, int); /* */

BMENU *cdecl mnkey (BMENU *, /* Add a key binding to a menu. */
    int, int, int, /*
    int, int, int);/* */

BMENU *cdecl mnitem (BMENU *, /* Add/Change/Delete item in */
    int, int, int, /* "Lotus"-style menu. */
    const char *, /*
    int, int, /*
    const char *); /* */

BMENU *cdecl mnitemkey (BMENU *, /* Add "Lotus" item and key */
    int, int, int, /* bindings for that item to menu. */
    const char *, /*
    int, int, /*
    const char *, /*
    const char *, int); /* */

BMENU *cdecl mnstyle(BMENU *,/* Install a mouse style. */
    int,unsigned); /* */

BMENU *cdecl mnmouse(BMENU *,/* Add, change, or delete a menu */
    unsigned long,unsigned long,/* mouse event. */
    int,int); /* */

```

```

BMENU *cdecl mnhighlight(BMENU *,/* Move or remove highlight bar */
    int,int,int); /* or item description. */
                /* */

int cdecl mnread (BMENU *, int, /* Request user input from a menu. */
    int, int *, /* */
    int *, int *, /* */
    int *, int); /* */

/* Function declarations -- Internal functions. */
BMENU *cdecl mnattr (BMENU *, /* Change attributes on an item in */
    const BITEM *, int); /* a menu. */
                /* */

BMENU *cdecl mncreat0 (int, /* Create menu data structures, */
    int, int, int, /* set up default key bindings and */
    int, int, unsigned); /* attributes. */
                /* */

BMENU *cdecl mndefkey (BMENU *);/* Set up default keys for a menu. */
                /* Internal function -- redefine if */
                /* you want different default key */
                /* bindings. */
                /* */

BMENU *cdecl mndisabl (BMENU *);/* Disable all keys which are bound */
                /* to locations for which no items */
                /* exist -- disable unbound keys. */
                /* */

int cdecl mndlitms (BMENU *); /* Delete all items from a menu */
                /* item list data structure. */
                /* */

int cdecl mndlkeys (BMENU *); /* Delete all keys from a menu key */
                /* list data structure. */
                /* */

BITEM *cdecl mnfindsl (BMENU *, /* Find the first selectable item */
    BITEM *, /* in a menu, given starting */
    int, int, /* coordinates and menu item. */
    int *); /* */
                /* */

BMENU *cdecl mnhighlight0(BMENU *,/* Move or remove highlight bar */
    BITEM *,int); /* or item description. */
                /* */

BITEM *cdecl mnmatchtm (BMENU *, /* Match a menu item to a (row,col) */
    BITEM *, /* specification. */
    int, int, /* */
    int, int *); /* */

```

```

/* */
const BKEYMAP *cdecl mnmmchkey ( /* Match a menu keymap entry to a */
    const BMENU *, /* (ch,scan) specification. */
    const BKEYMAP *, /*
        int, int, /*
        int *); /*
    /* */
BMENU *cdecl mnvalmn0 (BMENU *, /* Validate a menu data structure. */
    unsigned); /*
#endif /* Ends "#ifndef DEF_BMENU" */

```

## BMOUSE.H 鼠标函数的头文件

```

#ifndef DEF_BMOUSE /* Prevent second reading of */
#define DEF_BMOUSE 1 /* these definitions. */
#include <bkeybrd.h> /* For KEYSTATUS. */
#include <bintrupt.h> /* For ISRCTRL. */
/* General symbols. */
#define MO_LEFT 1 /* Left mouse button. */
#define MO_RIGHT 2 /* Right mouse button. */
#define MO_MIDDLE 4 /* Center mouse button, if any. */
/* (MO_LEFT, MO_RIGHT and MO_MIDDLE
/* must not conflict with MO_PRESS and */
/* MO_RELEASE.) */
#define MO_ALLBUTTONS (MO_LEFT | MO_RIGHT | MO_MIDDLE)
#define MO_INSTALL 0
#define MO_REMOVE 1
/* Error codes. */
#define MO_OK 0 /* Successful operation. */
#define MO_NOLINK (-3) /* Mouse functions not linked. */
#define MO_ABSENT (-2) /* Mouse driver not installed. */
#define MO_BAD_OPT 1 /* Unknown option. */
#define MO_RANGE 2 /* Value(s) out of range. */
#define MO_BAD_COMBO 3 /* Bad combination of options. */
#define MO_ALREADY 4 /* Can't do operation twice. */
#define MO_NOT_INSTALLED /* MOCATCH not installed. */
/* Symbol for b_mouse. */
#define MO_UNKNOWN(1) /* Mouse state not yet tested. */
/* Symbols for MORANGE. */
#define MO_HORIZ 0 /* Horizontal. */
#define MO_VERT 3 /* Vertical */
/* Symbols for MOHIDE. */
#define MO_HIDE 1

```

```

#define MO_SHOW      0
    /* Symbols for MOLTPEN. */
#define MO_PEN      1      /* Emulate light pen.   (Default */
                          /* condition when mouse is reset.) */
#define MO_NO_PEN    0      /* Don't emulate light pen. */
    /* Symbols for MOBUTTON and MOCHECK. */
#define MO_PRESS     8      /* Button events (must not conflict */
#define MO_RELEASEx10 /* with MO_LEFT, MO_RIGHT, and */
                          /* MO_MIDDLE). */

#define MO_CLEAR      0x0010
#define MO_NOCLEAR    0x0020
#define MO_CLICK      0x0040
#define MO_DCLICK     0x0080
#define MO_HOLD       0x0800
#define MO_RSHIFT     0x1000
#define MO_LSHIFT     0x2000
#define MO_CSHIFT     0x4000
#define MO_ASHIFT     0x8000

    /* Mouse event masks for MOHANDLR. */
#define MO_MOVE       0x0001      /* Mouse moved. */
#define MO_L_PRESS    0x0002      /* Left button depressed. */
#define MO_L_RELEASEx0004 /* Left button released. */
#define MO_R_PRESS    0x0008      /* Right button depressed. */
#define MO_R_RELEASEx0010 /* Right button released. */
#define MO_M_PRESS    0x0020      /* Middle button depressed. */
#define MO_M_RELEASEx0040 /* Middle button released. */

    /* Internal symbols. */
#define MO_HIST_LIMIT 5      /* Number of recent events recorded. */
                          /* Need at least 5 for double clicks. */
                          /* Must match MOCATCH.ASM. */
                          /* Mouse events handled by MOCATCH. */
#define MOCATCH_MASK(MO_L_PRESS | MO_L_RELEASE \
                    | MO_R_PRESS | MO_R_RELEASE \
                    | MO_M_PRESS | MO_M_RELEASE)

    /* Data types. */
typedef struct      /* MO_EVENT: Mouse event in history */
{
    /* array. This must match MOCATCH.ASM */
    int event;      /* MO_PRESS or MO_RELEASE. */
    long time;      /* BIOS clock ticks since midnight. */
    unsigned c_vert,c_horiz; /* Mouse cursor location in pixels. */
    int vert,horiz; /* Mouse location in mickeys. */
    KEYSTATUS kstat; /* Shift key status at time of event. */

```



```

    int        relevant;        /* Bits to prevent rereading same event */
                                /* (MO_CLICK, MO_DCLICK, MO_PRESS*/
                                /* MO_RELEASE). */
} MO_EVENT;

                                /* PMOHANDLER address of mouse */
                                /* interrupt handler function. */

typedef void cdecl (*PMOHANDLER)(const ALLREG *);

/* Global variables. */

extern int b_mouse;             /* Mouse presence: MO_UNKNOWN. */
                                /* 0 if absent, or number of buttons. */
extern long b_clicklimit;      /* Time limits for clicks and double */
extern long b_dclicklimit;     /* clicks (measured in clock ticks). */
                                /* History of recent mouse events */
                                /* maintained by MOCATCH. */
                                /* First subscript is for each button */
                                /* (0 = left, 1 = right, 2 = middle); */
                                /* second is for event number */
                                /* (0 is most recent). */

extern volatile MO_EVENT b_mohist[3][MO_HIST_LIMIT];

extern void (far *b_mocatch)(void); /* Pointer to MOCATCH if */
                                /* installed, or 0 if not. */
                                /* */

extern unsigned b_momask;       /* Current MOCATCH event mask*/
                                /* (0 or MOCATCH_MASK). */
                                /* */

extern ISRCTRL far *b_modispat; /* Address of dispatcher that */
                                /* invokes user interrupt */
                                /* handler, or 0 if no handler */
                                /* installed. */
                                /* */

extern unsigned b_mohanmask;    /* Call mask for user mouse */
                                /* interrupt handler (0 if */
                                /* none). */

/* Routines implemented as macros. */
#define moreset()               /* Reset mouse driver. */
    ((b_mouse=MO_UNKNOWN),moequip())
/* Function declarations. */
                                /* General mouse setup. */

int cdecl moequip(void);        /* Mouse driver presence test. */
                                /* */

int cdecl mogate(const DOSREG *, /* General gateway. */
                DOSREG *);      /* */
                                /* */

```

```

int far cdecl mopreclick(int);          /* Install MOCATCH (so MOCHECK
                                         /* can work).                */
                                         /*                                */

int cdecl mohandler(PMOHANDLER,         /* Install user's mouse          */
    unsigned,char *,int,int);           /* interrupt handler.            */
                                         /*                                */

int cdecl mospeed(unsigned,unsigned); /* Set mouse sensitivity (speed */
                                         /* of cursor motion).            */
                                         /*                                */

int cdecl mojump(unsigned);             /* Set speed-jump threshold.     */
                                         /*                                */

int cdecl molitpen(int);                /* Enable or disable light pen  */
                                         /* emulation.                    */
                                         /*                                */
                                         /* Mouse cursor appearance.      */

int far cdecl mohide(int);              /* Hide or show mouse cursor.    */
                                         /*                                */

int far cdecl moavoid(unsigned,         /* Hide mouse cursor when in    */
    unsigned,unsigned,unsigned); /* specified region.            */
                                         /*                                */

int cdecl mosoft(unsigned,unsigned);    /* Set software text cursor.     */
                                         /*                                */

int cdecl mohard(int,int);              /* Set hardware text cursor.     */
                                         /*                                */

int cdecl mograph(const unsigned *,     /* Set graphics-mode cursor.    */
    int,int);                          /*                                */
                                         /* Controlling mouse cursor movement.

int cdecl mocurmov(unsigned,unsigned); /* Move mouse cursor.            */
                                         /*                                */

int cdecl morange(int,unsigned,         /* Set range limit (horizontal  */
    unsigned);                        /* or vertical).                */
                                         /*                                */
                                         /* Querying mouse status.

int cdecl mostat(unsigned *,           /* Report mouse position and    */
    unsigned *,unsigned *);           /* button status.              */
                                         /*                                */

int cdecl mobutton(int,unsigned *,     /* Report mouse button press/   */
    unsigned *,unsigned *,unsigned *); /* release history.            */
                                         /*                                */

int cdecl mogetmov(int *,int *);        /* Report relative physical     */
                                         /* mouse motion since last      */
                                         /* inquiry.                    */
                                         /*                                */

int far cdecl mocheck(unsigned long,   /* Check for recent mouse event.*/

```

```

unsigned long,int,                /* */
unsigned long *,unsigned *,      /* */
unsigned *);                     /* */
/* Internal functions.           */
void far mocatch(void);           /* Record mouse interrupts. */
/* */
int cdecl moinst(void (far *) (void), /* Install interrupt handler */
                unsigned);          /* unconditionally. */
#endif                            /* Ends "#ifndef DEF_BMOUSE" */

```

## BNATVWN.H 支持编译器本地文本窗口的头文件。

```

#ifndef DEF_BGENWIN                /* Prevent redefinition. */
#define DEF_BGENWIN 1
/* The symbol B_NATIVE_WINDOWS is used at compile */
/* time to determine whether Blaise C TOOLS routines */
/* will support the compiler's native text window. */
/* If it is defined as 1, the native text window is */
/* supported; a value of 0 means it is not. */
/*#define B_NATIVE_WINDOWS 0*/ /* No native text window support. */
#define B_NATIVE_WINDOWS 1 /* Native text window support. */
#endif                            /* Ends "#ifndef DEF_BGENWIN". */

```

## BPRINT.H头文件

```

#ifndef DEF_BPRINT                /* prevent redefinition */
#define DEF_BPRINT 1
#include <bios.h>
/* Interrupt numbers for communicating with print spooler and BIOS. */
#define PR_BIOS_INT 0x17 /* BIOS printer control int for */
/* PRSTATUS and PRINIT. */
#define PR_DOS_INT 0x2F /* DOS printer control int. */
#define PR_DOSFUNC_INT 0x21 /* DOS general service gate. */
/* Error return codes. */
#define PR_OK 0 /* No PR error. */
#define PR_INSTAL 1 /* Printer not installed or DOS older */
/* than 3.00 */
#define PR_EXIST 2 /* File does not exist (in queue or file*/
/* system, depending on context. */
#define PR_PATH 3 /* Path does not exist. */
#define PR_HANDLES 4 /* Out of handles. */
#define PR_ACCESS 5 /* Access denied. */

```

```

#define PR_EMPTY 6      /* Print spooler queue empty. */
#define PR_RANGE 7      /* Item requested out of range of queue.*/
#define PR_FULL 8       /* Queue full. */
#define PR_BUSY 9       /* Spooler busy; can't get attention. */
#define PR_NAMELEN 12   /* Path name too long. */
#define PR_DRIVE 15     /* Invalid disk drive. */
/* Status return codes. */
#define PR_S_ACK 0x40    /* printer acknowledged. */
#define PR_S_IOERR 0x08  /* i/o errors with printer. */
#define PR_S_NOPAPER 0x20 /* out of paper. */
#define PR_S_NOTBUSY 0x80 /* printer not busy. */
#define PR_S_ONLINE 0x10 /* on line. */
#define PR_S_TIMEOUT 0x01 /* timeout. */

/* Defines which make PRSTATUS (i.e. BIOS) return codes into zero */
/* for no error. */
#define PR_BIOS_OK (PR_S_ONLINE | PR_S_NOTBUSY)
#define PR_MAKEOK(retcode) (retcode ^ PR_BIOS_OK)
/* Print a single character using BIOS. */
#define prechar(port,bval) (biosprint (0, (bval), (port)))
/* Initialize printer port using BIOS. */
#define print(port) (biosprint (1, 0, (port)))
/* Get status for a printer port. BIOS. */
#define prstatus(port) (biosprint (2, 0, (port)))
/* Definitions of functions for use with strong type checking */
int cdecl precancel (char *); /* Cancel file(s) from the */
/* PRINT.COM spooler queue. */
/* */
const char *cdecl prerror (int); /* Return error string */
/* corresponding to PR error*/
/* code. */
/* */
int cdecl prgetq (int, int *, char *); /* Get name of file in */
/* spooler queue, capacity */
/* and fullness of queue. */
/* */
int cdecl prinstld (void); /* Check to see whether */
/* PRINT.COM is installed. */
/* Always fails on dos 2.x. */
/* */
int cdecl prspool (const char *); /* Put a file in the DOS */
/* PRINT.COM spooler queue. */
#endif /* prevent redefinition */

```

## BSCREENS.H 屏幕函数的头文件

---

```
#ifndef DEF_BSCREENS          /* Prevent second reading of */
#define DEF_BSCREENS 1        /* these definitions. */
#include <butil.h>

#define SC_BIOS_INT 16        /* BIOS interrupt for screen */
                                /* functions. */
#define NORMAL 7              /* Common attributes for the */
#define REVERSE 112          /* monochrome adapter */
#define UNDERLINE
#define INTENSITY 8
#define MONOBLINK 8
#define SC_BLACK 0            /* (Byte) values representing */
#define SC_BLUE 1             /* color attributes. */
#define SC_GREEN 2
#define SC_CYAN 3
#define SC_RED 4
#define SC_MAGENTA 5
#define SC_BROWN 6
#define SC_WHITE 7

    /* Maximum dimensions of usual PC screens */
#define PC_ROWS 25
#define PC_COLS 80
#define LAST_ROW (PC_ROWS - 1)
#define LAST_COL (PC_COLS - 1)
#define PC_BIG_ROWS 30
#define MAX_DEVICES          /* Color and/or monochrome */
#define MAX_PAGES 8          /* Pages per video adapter */
                                /* (8 maximum on EGA) */

    /* Scrolling directions */
#define SCR_UP 0
#define SCR_DOWN 1
#define SCR_RIGHT 0
#define SCR_LEFT 1

    /* Options for SCPGCUR */
#define CUR_ADJUST 1          /* Adjust cursor size into normal */
                                /* limits if appropriate */
#define CUR_NO_ADJUST        /* Use requested cursor scan lines */
                                /* without any adjustment */

    /* Options for writing strings to screen */
#define CUR_BEG 0             /* Leave cursor at beginning of string. */
#define CUR_AFTER 1           /* Leave cursor after end of string. */
```

```

/*
#define CHARS_ONLY 0 /* Buffer contains characters only. */
#define CHAR_ATTR 2 /* Buffer contains (char,attr) pairs. */
/*
#define MOVE_CUR 0 /* Leave cursor according to */
/* CUR_BEG/CUR_AFTER bit. */
#define NO_MOVE_CUR 4 /* Preserve cursor location. */
/* Options for SCBLINK */
#define SC_BLINK 1 /* Attribute bit 7 == foreground blink. */
#define SC_INTENSE 0 /* Attribute bit 7 == background */
/* intensity. */
#define SC_INTENSITY 0 /* Synonym for SC_INTENSE. */
/* Error codes */
#define SC_NO_ERROR 0 /* Operation successful. */
#define SC_BAD_OPT 1 /* Video adapter does not support this */
/* operation. */
#define SC_RANGE 2 /* Value out of range or exceeds */
/* dimension. */
#define SC_NO_MEMORY /* Insufficient memory to perform */
/* operation. */

/* Symbols used by SCEQUIP to indicate the state of the various */
/* video adapters */
#define SC_DONT_KNOW /* We haven't checked yet */
#define SC_ABSENT (-2)
#define SC_MONO 0
#define SC_COLOR 1
#define SC_HIFUNC 2 /* PGC in high-function graphics mode */
/* (not CGA emulation mode) */

#define DONT_KNOW SC_DONT_KNOW
#define ABSENT SC_ABSENT
#define HIFUNC SC_HIFUNC

/* Global variables set by SCEQUIP indicating installed video */
/* hardware and switch settings */
extern int b_mdpa; /* Monochrome Display & Printer Adapter */
/* SC_DONT_KNOW, SC_ABSENT, or SC_MONO */
/* (SC_MONO if Hercules graphics adapter.) */
extern int b_ega; /* Color/Graphics Monitor Adapter */
/* SC_DONT_KNOW, SC_ABSENT, or SC_COLOR */
extern int b_ega; /* Enhanced Graphics Adapter: SC_DONT_KNOW, */
/* SC_ABSENT, SC_MONO or SC_COLOR */

```

```

extern int b_mega;    /* Multicolor Graphics Array: SC_DONT_KNOW,    */
                    /* SC_ABSENT, or SC_COLOR */
                    /* */
extern int b_vga;     /* Video Graphics Array */
                    /* SC_DONT_KNOW, SC_ABSENT, SC_MONO or SC_COLOR */
                    /* */
extern int b_herc;    /* Hercules monochrome graphics adapter */
                    /* SC_DONT_KNOW, SC_ABSENT, or SC_MONO */
                    /* */
extern int b_pgc;     /* Professional Graphics Controller: SC_ABSENT, */
                    /* SC_DONT_KNOW, SC_COLOR or SC_HIFUNC */
extern int b_mem_ega; /* Amount of memory installed on EGA in */
                    /* 1024-byte units: 64, 128, 256 */
                    /* */
extern unsigned b_sw_ega; /* Switch settings on EGA */
                    /* Low-order bit set means SW1 off, etc.*/
                    /* (Bit value 1 implies that */
                    /* corresponding switch is off.) */

/* Global variables for managing multiple devices & display */
/* pages */
extern int b_curpage; /* Current display page */
extern int b_device; /* Current adapter: SC_DONT_KNOW, */
                    /* SC_MONO, or SC_COLOR */
extern int b_know_hw; /* Flag stating whether we have */
                    /* yet run SCEQUIP: 0 if no, 1 */
                    /* if yes. */

typedef struct /* CUR_TYPE structure: */
{
    int high,low; /* high and low scan lines */
} CUR_TYPE; /* in a cursor. */

typedef struct /* CUR_INFO structure: */
{
    int off; /* State, location and size of */
    int row; /* a cursor. */
    int col;
    CUR_TYPE size;
} CUR_INFO;

/* Internal structures for preserving BIOS video variables */
typedef struct
{
    int known; /* 1 if this structure has */
               /* valid data, 0 if not. */
    int curpage; /* Copy of b_curpage. */

```

```

char pc_area[30];          /* Standard PC/XT area. */
char video_area[7];       /* Some video data. */
char far *save_ptr;       /* Pointer to EGA variables. */
char prtsc_status;        /* Print screen status byte. */
} VIDEO_STATE;
extern VIDEO_STATE b_adap_state[2]; /* One entry for SC_MONO, one */
/* for SC_COLOR. */
typedef struct             /* ADAP_STATE structure: */
{
    /* state of the video adapter.*/
    int mode;              /* Current video mode. */
    int act_page;          /* Active (displayed) page. */
    int cur_page;          /* Current page. */
    int rows, columns;     /* Dimensions of screen. */
    int curs_off;          /* Cursor on or off. */
    CUR_TYPE curs_size;    /* Cursor high & low scan lines.*/
} ADAP_STATE;
typedef struct             /* PAGE_STATE structure: */
{
    /* state of the current page. */
    int curs_row, curs_column; /* Location of cursor. */
    char *pimage;           /* Compressed screen image. */
    int image_length;       /* Length of compressed image. */
} PAGE_STATE;
/* Turbo C TOOLS functions implemented as macros */
#define scpage(page) (b_curpage == (page))
/* Function declarations */
char cdecl scequip(void); /* Sense hardware environment */
int cdecl scnewdev(int,int); /* Reset and select 25 or 43 */
int cdecl sechgdev(int); /* Select adapter */
int cdecl scmode(int *,int *,int*); /* Return the video mode info */
int cdecl scrows(void); /* Number of rows on screen */
int cdecl scpages(void); /* Number of pages in this mode */
int cdecl scapage(int); /* Display (activate) a page */
void cdecl scpcrlr(void); /* Clear the current page */

```



```

int cdecl seclrmmsg(int,int,int);      /* Clear a message          */
/*                                     */

int cdecl securset(int,int);           /* Set the cursor position  */
/*                                     */

int cdecl sepgcur(int,int,int,int);    /* Alter the cursor size on */
/* current page.                */
/*                                     */

int cdecl securst(int *,int *,int *, /* Return position & size of */
int *); /* cursor on current page. */
/*                                     */

char cdecl scread(int *,int *);        /* Read character and attribute */
/*                                     */

int cdecl scattrib(int,int,char, /* Set the display attributes */
unsigned); /*                                     */
/*                                     */

int cdecl scwrite(char,unsigned); /* Write copies of a character */
/*                                     */

int cdecl settywrt(char,int); /* Write character TTY-style */
/*                                     */

int cdecl sebox(int,int,int,int, /* Draw a box                */
int,char,int); /*                                     */
/*                                     */

void cdecl settywin(int,int,int,int, /* Write character TTY-style to */
char,int,int,int,int); /* rectangular region.          */
/*                                     */

void cdecl sewrap(int,int,int,int, /* Write to rectangular region */
int,const char *,int,int,int); /* with word wrap              */
/*                                     */

int cdecl seblink(int); /* Choose role of attribute bit */
/* 7: foreground blink or      */
/* background intensity.        */
/*                               */

int cdecl sepalett(const char *); /* Load EGA palette registers. */
/*                               */

int cdecl sepall(unsigned,unsigned); /* Load individual EGA palette */
/* register.                      */
/*                               */

int cdecl seborder(unsigned); /* Set screen border color.     */
/*                               */

int cdecl smode4(int,int); /* Set mode 4 palette and       */
/* background color.           */
/*                               */

void cdecl segetvid(ADAP_STATE *); /* Get the complete video state.*/

```

```

/* */
int cdecl  sresetvid(const      /* Set the complete video state.*/
                    ADAP_STATE *); /* */
/* */
int cdecl  ssavepg(PAGE_STATE *); /* Save the current page. */
/* */
int cdecl  sresetpg(const      /* Restore the current page. */
                    PAGE_STATE *); /* */
/* */
#endif /* Ends "#ifndef DEF_BSCREENS"*/

```

## BSTRINGS.H 字符串函数的文件。

---

```

#ifndef DEF_BSTRINGS /* Prevent redefinition. */
#define DEF_BSTRINGS /* Prevent second reading */
/* of these definitions. */

/* Direction codes for STPJUS */
#define JUST_LEFT (-1)
#define JUST_CENTER 0
#define JUST_RIGHT 1

/* Format codes for STPCVT */
#define RWHITE 1 /* Remove white space. */
#define RLWHITE 2 /* Remove leading white space. */
#define RTWHITE 4 /* Remove trailing white space. */
#define REDUCE 8 /* Reduce white space to 1 blank. */
#define NOQUOTE 16 /* Quoted substrings not altered. */
#define TOUP 32 /* Convert to upper case. */
#define TOLOW 64 /* Convert to lower case. */
#define RCONTROL 128 /* Remove all control characters. */

/* Function declarations */
char * cdecl stpevt(char *,int); /* Convert a string. */
/* */
char * cdecl stpexpan(char *, /* Expand TABs in a string. */
                    char *,int,int); /* */
/* */
char * cdecl stpjus(char *, /* Justify a string within */
                    const char *,char,int,int);/* a field. */
/* */
char * cdecl stptabfy(char *,int); /* Compress blanks to TABs. */
/* */
char * cdecl stpxlate(char *, /* Translate a string */
                    const char *,const char *);/* */
/* */

```

```

int      stschind(char,const char *);          /* Find a char in a string. */
#endif.                                     /* End "#ifndef DEF_BSTRING"*/

```

## BUTIL.H 实用函数的头文件

---

```

#ifndef DEF_BUTIL                          /* Prevent redefinition. */
#include <string.h>
#include <stdlib.h>
/* needs dos.h */                          /* For inp(), outp(), _osmajor, */
                                          /* _osminor, and _psp. */
/*****
/* Definitions of data types.
*****/

typedef struct                            /* HALFREGS:two halves of a 16-bit*/
{                                          /* general register. */
    unsigned char l;                      /* Lower 8 bits. */
    unsigned char h;                      /* Upper 8 bits. */
} HALFREGS;

typedef union                             /* DOUBLREGTwo ways to express */
{                                          /* a 16-bit general register: */
    HALFREGS hl;                          /* Two separated halves. */
    unsigned x;                           /* 16-bit unsigned quantity. */
} DOUBLREG;

typedef struct                            /* ALLREG: complete CPU state */
{                                          /* (This must match version */
                                          /* in ISCALINT.ASM.) */
    DOUBLREG ax,bx,cx,dx;                 /* General registers. */
    unsigned si,di;                       /* Index registers. */
    unsigned ds,es,ss,cs;                 /* Segment registers. */
    unsigned flags,bp,sp,ip;              /* Other registers. */
} ALLREG;

struct dreg                              /* DOSREG: registers for call */
{                                          /* to DOS. */
    unsigned ax,bx,cx,dx,si,di,ds,es;
};

#define DOSREGstruct dreg

typedef struct segads                     /* ADS: Physical address */
{                                          /* expressed as segment and offset*/
    unsigned r;                           /* Offset. */
    unsigned s;                           /* Segment. */
}

```

```

} ADS;

struct address
{
    /* Physical address.
    /* expressed as segment and offset*/
    unsigned r; /* Offset
    unsigned s; /* Segment
};

/*****
/* Flag bits for use with ALLREG structure.
/*****

#define CF_FLAG 0x0001 /* Carry flag.
#define PF_FLAG 0x0002 /* Parity flag (1 means even).
#define AF_FLAG 0x0010 /* Auxiliary carry flag.
#define ZF_FLAG 0x0040 /* Zero flag (1 means zero).
#define SF_FLAG 0x0080 /* Sign flag (1 means negative).
#define TF_FLAG 0x0100 /* Trap flag (1 means single step).
#define IF_FLAG 0x0200 /* Interrupt enable flag.
#define DF_FLAG 0x0400 /* Direction flag (0 means up).
#define OF_FLAG 0x0800 /* Overflow flag (1 means overflow).
#define DEF_FLAGS (IF_FLAG) /* Commonest value for flags:
    /* interrupts enabled,
    /* single-stepping off,
    /* direction up.

/*****
/* Handy system constants and addresses.
/*****

#define UTDOSINT 0x21 /* Standard DOS gate interrupt.
    /* Address of PC model code.
#define UTMODELADDR (uttfaru(0xffff, 0xe))
    /* Address of PC BIOS clock tick count.
#define UTCLOCKADDR (uttfaru(0x40, 0x6c))
    /* Address of PC BIOS clock rollover flag.
#define UTROLLADDR (uttfaru(0x40, 0x70))
    /* Possible values for IBM Model Code:
#define IBM_PC ((char) 0xff) /* PC.
#define IBM_XT ((char) 0xfe) /* PC-XT or Portable PC.
#define IBM_JR ((char) 0xfd) /* IBM PCjr.
#define IBM_AT ((char) 0xfc) /* IBM (or compat.) PC-AT.
#define IBM_X2 ((char) 0xfb) /* PC-XT with 256/640 motherboard.
#define IBM_CV ((char) 0xf9) /* PC Convertible.

```

```

#define IBM_30 ((char) 0xfa) /* PS/2 model 30. */
#define IBM_50 ((char) 0xfe) /* PS/2 models 50, 60. */
#define IBM_80 ((char) 0xf8) /* PS/2 model 80. */

/*****
/* Global variables maintained by UT functions and macros. */
*****/

/* Contains address of DOS critical section flag, */
/* after UTCRIT is called. */

extern unsigned char far *b_critad;
extern char b_pmodel; /* IBM PC Model Code. */
extern unsigned char b_submod; /* IBM PC submodel code, or 0 if */
/* unavailable. */
extern unsigned char b_biosrev; /* IBM PC BIOS revision level, or 0 */
/* if unavailable. */
extern int b_ansdis; /* Nonzero if UTANSI has disabled */
/* ANSI.SYS. */

/*****
/* Constant values for UTINTFLG, UTAMOVE, UTSAFCPY, UTCTLBRK, UTANSI*/
*****/

#define UT_INTON 1 /* Values to use with UTINTFLG. */
#define UT_INTOFF 0

/* Values to use with UTAMOVE. */
#define UT_MVFORWARD /* Increasing memory addresses. */
#define UT_MVBACKWARD 1 /* Decreasing memory addresses. */

/* Values to use with UTSAFCPY. */
/* */

#define UT_SRC_STRADDLE 1 /* Source buffer straddles segment */
/* boundary, target okay. */
#define UT_TGT_STRADDLE 2 /* Target buffer straddles segment */
/* boundary, source okay. */
/* Both source & target straddle. */

#define UT_2_STRADDLE ((UT_SRC_STRADDLE) + (UT_TGT_STRADDLE))
/* Values to use with UTCTLBRK. */

#define CBRK_SET 1 /* Set CTRL-BREAK state. */
#define CBRK_GET 0 /* Return CTRL-BREAK state. */
#define CBRK_ON 1 /* Check CTRL-BREAK at every */
/* DOS call. */
#define CBRK_OFF 0 /* Check CTRL-BREAK only when */
/* accessing standard input, */
/* output, print, or */
/* auxiliary device. */
/* Values to use with UTANSI. */

```

```

/*
/* As command:      As result:
#define UT_ANS_ABSENT (-2) /* (Invalid.)      Not detected. */
#define UT_ANS_ENABLE (-1) /* Enable.          Active.          */
#define UT_ANS_DISABLE 0 /* Disable.         Disabled.        */
#define UT_BAD_OPT 1 /* (Invalid.)       Bad option.     */
#define UT_ANS_DETECT 2 /* Sense presence.  (Impossible.) */
#define UT_ANS_HANDLES 3 /* (Invalid.)       No handles.     */
/*****
/* Constant values for UTSQZSCN, UTUNSQZ.
/*****
#define UT_CHAR_KEY 0xFF
#define UT_ATTR_KEY 0xFE
/*****
/* Macros to combine and extract words, bytes, and nybbles.
/*****
#define uthiword(a) (((a) >> 16) & 0xffffL) /* High word of long a. */
#define utlowword(a) ((a) & 0xffffL) /* Low word of long a. */
/* Combine high word a, low word b.
#define utwlong(a,b) (((0xffffL & (long)(a)) << 16) |
                    (0xffffL & (long)(b)))
#define uthibyte(a) (((a) >> 8) & 0x00ff) /* High byte of word a. */
#define utlobyte(a) ((a) & 0x00ff) /* Low byte of word a. */
/* Combine high byte a, low byte b.
#define utbyword(a,b) (((a) & 0x00ff) << 8) | ((b) & 0x00ff)
#define uthinyb(a) (((a) >> 4) & 0x000f) /* High nybble of byte a. */
#define utlonyb(a) ((a) & 0x000f) /* Low nybble of byte a. */
/* Combine high nybble a, low
/* nybble b.
#define utnybbyt(a,b) (((a) & 0x000f) << 4) | ((b) & 0x000f)
/*****
/* Macros for pointer manipulation.
/*****
/* Construct far void pointer.
#define uttofaru(seg,off) ((void far *)
                    (((unsigned long) (unsigned int)
                    (seg)) << 16L) |
                    ((unsigned long) (unsigned int) (off))))
/* Compute offset of memory pointed to.
#define utoff(p) ((unsigned int) (p))
/* Compute segment of memory pointed to.
#define utseg(p) ((unsigned int)
                    (((unsigned long) (const void far *) (p)) >> 16L))

```

```

        /* Make a far pointer to a type. */
#define uttofar(seg,off,type) ((type far *) uttofaru((seg),(off)))

        /* Normalize a pointer. */
#define utnorm(p,type) (uttofar (utseg (p) + (utoff (p) >> 4),
        (utoff (p) & 0x000f),
        type))

        /* Return 20-bit address in a pointer. */
#define utplong(p) (((unsigned long) utseg (p)) << 4L) +
        ((unsigned long) utoff (p)) &
        0xfffffL)

/*****
/* Macros for low-level memory manipulation.
*****/

        /* Return byte pointed to by pointer. */
#define utpeekb(p) (*((const unsigned char far *) (p)))

        /* Return word pointed to by pointer. */
#define utpeekw(p) (*((const unsigned int far *) (p)))

        /* Put byte in location pointed to. */
#define utpokeb(p,bval) (*((unsigned char far *) (p)) = (bval))

        /* Put word in location pointed to. */
#define utpokew(p,uval) (*((unsigned int far *) (p)) = (uval))

        /* Fetch N bytes from a far pointer. */
#define utpeekn(pf,pn,clen) (utmovmem ((const char far *) (pf),
        (char far *) (pn),
        (clen)))

        /* Put N bytes to a far pointer. */
#define utpoken(pn,pf,clen) (utmovmem ((const char far *) (pn),
        (pf),
        (clen)))

/*****
/* Macros for communication with ports.
*****/

        /* Return a byte from a port. */
#define utinp(port) (inp (port))

        /* Put a byte in a port. */
#define utoutp(port,bval) ((void) outp ((port), (int) (bval)))

/*****
/* Miscellaneous macros.
*****/

#define NIL ((void *) 0) /* Universal invalid data */
                        /* pointer. */
                        /* Far version of NIL. */

#define FARNIL ((void far *) 0)

```

```

#define utsign(a)      ((a)>0?1:((a)==0?0:-1)) /* Arithmetic sign. */
#define utalarm()      \      utsound(450,9)          /* Beep for 1/2 sec.*/
#define utskip         printf("\n")          /* CR/LF.          */
                        /* Allocate and zero a data object of a given type, */
                        /* return a pointer to it.                          */
#define utalloc(type) ((type *) calloc(1,sizeof(type)))
                        /* Copy a data object, and return a pointer to      */
                        /* destination.                                      */
#define utcopy(to,from,type) ((type *)      \
                        memmove ((to),        \
                                (from),      \
                                sizeof (type)))
                        /* Return DOS version numbers.                      */
#define utdosmajor ((unsigned char) uthibyte (utdosver))
#define utdosminor ((unsigned char) utlobyte (utdosver))
#define utdosver ((unsigned int) utbyword (_osmajor, _osminor))
                        /* Check for null pointer assignments.              */
#if (defined(BCHKNIL) || !(defined(BNOCHKNIL)))
#define utchknil() (utnulchk()
                        ? (fprintf (stderr,
                                "Null pointer assignment detected in <%s:%d>\n",
                                __FILE__, __LINE__),
                                exit (1),
                                (1))
                        : (0))
#else
#define utchknil() ((void) 0)
#endif

                        /* Return the segment address of the                */
                        /* program's PSP.                                    */
#define utpspseg ((unsigned) (_psp))
                        /* Return 0 if DOS is not ready, 1 if it is.*/
                        /* Always returns 0 is UTCRIT has not          */
                        /* executed yet.                                */
#define utdosrdy() ((b_critad != FARNIL)
                        ? (utpeekb (b_critad) == 0)
                        : (0))
                        /* Enable interrupts, and return old              */
                        /* interrupt flag value.                          */
#define utinton() (utintflg (UT_INTON))
                        /* Disable interrupts, and return old            */
                        /* interrupt flag value.                          */
#define utintoff() (utintflg (UT_INTOFF))

```



```

        /* Turn the speaker off. */
#define utspkoff() (utspkr (0))
        /* Turn the speaker on, with a specified
        /* frequency sound to be produced. */
#define utspkon(freq) (utspkr (freq))
        /* Make a sound of "freq" frequency with the*/
        /* speaker for a specified duration. */
#define utsound(freq, dura) (
        utspkon (freq),
        utsleep (dura),
        utspkoff ())
/*****
/* Macros to perform range checking and limiting. */
/*****
/* Beware: the following three macros (UTMAX, UTMIN, and UTABS) */
/* may have side effects because they evaluate their arguments more */
/* than once. */
        /* Maximum of two values. */
#define utmax(a,b) ((a) > (b) ? (a) : (b))
#ifndef max
#define max(a,b) utmax(a,b)
#endif
        /* Minimum of two values. */
#define utmin(a,b) ((a) <= (b) ? (a) : (b))
#ifndef min
#define min(a,b) utmin(a,b)
#endif
#define utabs(x) ((x)<0 ? -(x) : (x)) /* Absolute value. */
        /* Return 1 if a is outside the range
        /* defined by l and h, 0 if not. */
#define utrange(a,l,h) (((a) < (l)) || ((a) > (h)))
#define utoutrng(a,l,h) utrange(a,l,h)
        /* Confine a to the range defined by b and c*/
#define utbound(a,b,c)
{
    register int uttemp;

    if ((a) < (uttemp = (b)) || (a) > (uttemp = (c)))
        (a) = uttemp;
}
        /* Prevent a from exceeding b. */
#define utuplim(a,b)
{

```

```

        register int uttemp;

        if ((a) > (uttemp = (b)))
            (a) = uttemp;
    }

    /* Prevent a from being less than b. */
#define utlowlim(a,b)
    {
        register int uttemp;

        if ((a) < (uttemp = (b)))
            (a) = uttemp;
    }

/*****
/* Function declarations.
*****/

void cdecl utamove(const char far *,/* Copy memory from far pointer */
                  char far *,      /* source to far pointer target.*/
                  unsigned int,     /* Does not handle overlap. */
                  int);             /*
                                     */

int cdecl utctlbrk (int,int);       /* Set or return state of DOS
                                     /* CTRL-BREAK checking.
                                     */
                                     /*
                                     */
#ifdef utdosver                     /*
                                     */
int cdecl utdosver (void);          /* Return the DOS major, minor
                                     /* version number as 16-bit.
                                     */
                                     /*
                                     */
int cdecl utgetclk (long *);        /* Return the BIOS clock roll
                                     /* flag and tick count.
                                     */
                                     /*
                                     */
int cdecl utintflg (int);           /* Set interrupt state, return
                                     /* old state.
                                     */
                                     /*
                                     */
unsigned int cdecl utnulehk (void); /* Check for NIL-pointer
                                     /* assignments.
                                     */
                                     /*
                                     */
unsigned cdecl utsleep (unsigned);   /* Suspend processing for
                                     /* specified duration.
                                     */
                                     /*
                                     */
void cdecl utspkr (unsigned);        /* Speaker control.
                                     */

```

```

/* */
int cdecl utansi(int); /* Detect, disable, or */
/* re-enable ANSI.SYS. */
/* */
void cdecl utmovmem( /* Copy block of memory, using */
    const char far *, char far *, /* far pointer source and */
    unsigned int); /* target. */
/* */
int cdecl utsafcopy(const void far *, /* Copy block of memory if */
    void far *, /* source & target buffers */
    unsigned int); /* don't straddle segment */
/* boundaries. */
/* */
unsigned char far * cdecl /* Return the address of the */
    uterit (void); /* DOS critical section flag. */
/* Also puts the value in */
/* b_critad. */
/* */
char cdecl utmodel(void); /* Return machine model code. */
/* */
unsigned long cdecl uttim2tk /* Compute a tick value for a */
    (int, int, /* given time of day. */
    int, int); /* */
/* */
void cdecl uttk2tim(unsigned long, /* Compute a time value (in */
    int *, int *, /* hours, minutes, seconds, */
    int *, int *); /* hundredths) for a given */
/* tick value. */
/* */
int cdecl utsqzscn(const char far *, /* Compress a screen image. */
    char *, int, /* */
    int); /* */
/* */
int cdecl utunsqz(const char *, /* Uncompress a screen image. */
    char far *, int, /* */
    int); /* */
#define DEF_BUTIL 1 /* Prevent second reading of these */
/* definitions. */
#endif /* Ends "#ifndef DEF_BUTIL". */

```

## BVIDEO.H 直接存取视屏硬件函数的头文件

```

#ifndef DEF_BVIDEO /* Prevent redefinition. */

```

```

#include <string.h>                                /* For strlen(). */
#include <bscreens.h>
/* Global variables. */
extern int b_vifast;                               /* Nonzero if interference is
                                                    /* to be ignored -- use fastest*/
                                                    /* possible method. */

/* TURBO C TOOLS functions implemented as macros */
#define vidspmsg(row,col,fore,back,pmsg) \
    (viwrrect((row),(col), \
               (row),(col)+strlen(pmsg)-1, \
               (pmsg), \
               (fore),(back),CHARS_ONLY))
#define viwrrect(u_row,u_col,l_row,l_col,buffer,fore,back,option) \
    (viwrsect((u_row),(u_col),(l_row),(l_col),(buffer),0, \
               (fore),(back),(option)))
#define virdrect(u_row,u_col,l_row,l_col,buffer,option) \
    (virdsect((u_row),(u_col),(l_row),(l_col),(buffer),0,(option)))

/* Function declarations */
int cdecl virdsect(int,int,int,int,          /* Read rectangle from video */
                  char *,unsigned,int);      /* memory. */
int cdecl viwrsect(int,int,int,int,          /* Write rectangle to video */
                  const char *,unsigned,int,int,int); /* memory. */
int cdecl viatrect(int,int,int,int,          /* Change attributes in */
                  int,int);                 /* rectangular region. */
int cdecl viscroll(int,int,                 /* Vertically scroll area on */
                  int,int,int,int,int);      /* current display page. */
int cdecl vihoriz(int,int,                 /* Horizontally scroll area on */
                  int,int,int,int,int);      /* current display page. */
char far * cdecl viptr(int,int);            /* Convert screen location to
                                                    /* physical address. */

/* Declaration of internal function. */
int cdecl vidirec0(                          /* Perform direct access to
    const char far * const *,                /* video memory.
    char far * const *,                      /*
    int,int,int,int,int,unsigned);           /*
/* Aliases for older versions */
#define viads(row,col,pads) \
    (((* ((char far **) (pads)) = viptr (row, col)) != FARNIL)

```

```

        ? (pads)
        : (NIL))
#define vidirect(ppfrom,ppto,num_rows,row_length,scr_width,attr,option)\
        (vidirec0((ppfrom),(ppto),(num_rows),(row_length),\
        (scr_width),(attr),(option),0))
#define DEF_BVIDEO1          /* Prevent second reading of */
                             /* these definitions. */
#endif                      /* Ends "#ifndef DEF_BVIDEO" */

```

## BWINDOW.H 窗口函数头文件

```

#ifndef DEF_BWINDOW          /* Prevent second reading of */
#define DEF_BWINDOW          /* these definitions. */
#include <bscreens.h>         /* Turbo C TOOLS Screen functions. */
#include <bkeybrd.h>          /* Turbo C TOOLS Keyboard functions.*/
#define WN_EXT_ERROR 1       /* 1 if wnerror() is external, 0 if */
                             /* it's a macro. */
/*****/
/* Definitions of data types. */
/*****/
typedef struct               /* LOC structure: */
{
    int row,col;             /* row and column relative to a */
                             /* rectangular region. */
} LOC;
typedef struct               /* REGION structure: */
{
    LOC ul;                  /* rectangular region on a */
                             /* screen. */
    LOC lr;                  /* Upper left corner. */
                             /* Lower right corner. */
} REGION;
typedef struct               /* CELL structure: */
{
    char ch,attr;            /* character and attribute byte on */
                             /* a PC screen. */
} CELL;

typedef struct               /* DIM structure: */
{
    int h,w;                 /* height and width of a */
                             /* rectangular region. */
} DIM;
typedef struct               /* IMAGE structure: */
{
    /* a copy of a rectangular

```

```

DIM dim;
CELL *pdata;

} IMAGE;
typedef struct
{
    int dev;

    int page;

    LOC corner;

} WHERE;

typedef struct bjoint {
    unsigned char row;
    unsigned char col;
    char jtype;
    char _dummy;
    struct bjoint *next;
} BJOINT;

typedef struct {
    int type;

    /* portion of a screen's data. */
    /* Dimensions of the rectangle. */
    /* Pointer to array of cells */
    /* containing the data, row */
    /* by row. */

    /* WHERE structure: */
    /* device, page, and location */
    /* where a window may be shown. */
    /* Device where shown: COLOR, */
    /* MONO, or ABSENT if not shown. */
    /* Video page number: 0 to 7 */
    /* Upper left corner of region */
    /* where window data is shown */
    /* (does not include border). */

    /* Row and column of joint relative */
    /* to upper-left of border. (0,0). */
    /* IBM graphic char for joint. */
    /* Structure alignment. */
    /* Pointer to next joint in linked */
    /* list of joints. */

    /* Type of border: */
    /* 0 = no border. */
    /* 31 = Character "ch" border. */
    /* T R B L */
    /* o t o f */
    /* p t t */
    /* 1 = S S S S no title. */
    /* 2 = S S S D no title. */
    /* 3 = D S S S no title. */
    /* 4 = D S S D no title. */
    /* 5 = S D S S no title. */

```

```

/* 6 = S D S D no title. */
/* 7 = D D S S no title. */
/* 8 = D D S D no title. */
/* 9 = S S D S no title. */
/* 10 = S S D D no title. */
/* 11 = D S D S no title. */
/* 12 = D S D D no title. */
/* 13 = S D D S no title. */
/* 14 = S D D D no title. */
/* 15 = D D D S no title. */
/* 16 = D D D D no title. */
/*
/* OR the above with: */
/* 0x0020- top-left-title */
/* 0x0040- top-center-title */
/* 0x0080- top-right-title */
/* 0x0100- bottom-left-title */
/* 0x0200- bottom-center-title */
/* 0x0400- bottom-right-title */
/*
/* OR the above with: */
/* 0x0800- border has joints. */
/*
/* Attribute of border. */
/* Border character (if type==0). */
/* Structure alignment. */
/* Title on top border. */
/* Title on bottom border. */
/* Attributes of top title. */
/* Attributes of bottom title. */
/* Pointer to linked list of joints.*/

int attr;
char ch;
char _dummy;
char *ptitle;
char *pbtitle;
int tattr;
int battr;
BJOINT *pjoints;
} BORDER;
typedef enum
{
/* Values for the WN_SENSOR type */
/* field. */
/* Left scroll arrow. */
/* Right scroll arrow. */
/* Up scroll arrow. */
/* Down scroll arrow. */
/* Inside window's viewport. */
/* On window's border. */
/* Outside the window completely. */
WN_LEFT_ARROW,
WN_RIGHT_ARROW,
WN_UP_ARROW,
WN_DOWN_ARROW,
WN_IN_WINDOW,
WN_ON_BORDER,
WN_OUT_WINDOW
} WN_SENSOR_TYPE;

```

```

#define WN_LAST_SENSOR WN_OUT_WINDOW
#define WN_VIEWPORT 0
#define WN_DATA 1
#define WN_SCREEN 2
typedef struct wn_sensor /* WN_SENSOR structure: linked list */
{
    /* of window sensors. */
    WN_SENSOR_TYPE type; /* The type of sensor this is. */
    unsigned relative; /* Whether the sensor is relative */
    /* to the screen, viewport, or */
    /* data area. */
    REGION hot; /* The "hot" region for the sensor. */
    LOC corner; /* Displayed corner for the sensor. */
    DIM size; /* Displayed size of the sensor. */
    CELL *pimage; /* Sensor's image. */
    struct wn_sensor *pNext; /* Pointers to the next and */
    struct wn_sensor *pprev; /* previous sensors in the list. */
} WN_SENSOR;

typedef enum /* WN_ACTION; */
{
    /* values for the WN_EVENT action */
    /* field. */
    WN_NULL, /* No action. */
    WN_SCROLL_LEFT, /* Move left. */
    WN_SCROLL_RIGHT, /* Move right. */
    WN_SCROLL_UP, /* Move up. */
    WN_SCROLL_DOWN, /* Move down. */
    WN_PAGE_LEFT, /* Page left. */
    WN_PAGE_RIGHT, /* Page right. */
    WN_PAGE_UP, /* Page up. */
    WN_PAGE_DOWN, /* Page down. */
    WN_LEFT_EDGE, /* Go to left edge. */
    WN_RIGHT_EDGE, /* Go to right edge. */
    WN_TOP_EDGE, /* Go to top edge. */
    WN_BOTTOM_EDGE, /* Go to bottom edge. */
    WN_ABORT, /* Abort read. */
    WN_TRANSMIT, /* Finish read. */
    WN_INVALID_ACTION /* Invalid window action. */
} WN_ACTION;

#define WN_ABSENT_EVENT 0
#define WN_KB_EVENT 1
#define WN_MOUSE_EVENT

typedef struct /* MOUSE_EVENT structure: */

```



```

{
    unsigned long    event;      /* Mouse characteristics to pay */
    unsigned long    ignore;     /* attention to & ignore. */
    unsigned         location;    /* Location code. */
} MOUSE_EVENT;

```

```

typedef struct
{
    /* WN_EVENT structure: */
    unsigned         event_type; /* WN_KB_EVENT or WN_MOUSE_EVENT. */
    union            /* event union: */
    {
        /* The basic data of a keyboard or */
        KEY_SEQUENCE keystroke; /* mouse event. */
        MOUSE_EVENT mouse;      /* */
    } event;             /* */
    void             *pdata;    /* Event related data. */
    unsigned         data_size; /* Size of data pointed to by pdata.*/
} WN_EVENT;

```

```

typedef struct wn_event_list /* WN_EVENT_LIST structure: */
{
    /* WN_EVENT, identifying */
    unsigned         signature; /* signature, & pointers */
    WN_EVENT         win_event; /* to next and previous */
    struct wn_event_list *pNext; /* items in event list. */
    struct wn_event_list *pprev;
} WN_EVENT_LIST;

```

```

typedef struct /* BWINDOW structure: */
{
    /* everything needed to control */
    /* an individual window. */
    /* */
    unsigned signature; /* Identifying signature. */
    /* */
    LOC cur_loc; /* Location of window's own */
    /* cursor relative to window's */
    /* data area. */
    /* */
    CUR_TYPE cur_type; /* Cursor type. */
    /* */
    IMAGE img; /* Contents of data area. */
    /* */
    DIM view_size; /* Size of window's viewport. */
    /* */
    LOC data_origin; /* Coordinates of data area which */
}

```

```

/* appear at (0,0) in the viewport. */
/*
WHERE where_shown; /* Where window is currently */
/* shown. */
/*
IMAGE prev; /* Previous contents of screen */
/* (before window was shown), */
/* including data under border. */
/*
WHERE where_prev; /* Region occupied by window */
/* (including border). */
/*
struct bwin_node *pnode; /* Pointer to window node */
/* among all windows on a video */
/* display page. */
/*
BORDER bord; /* Border description. */
/*
WN_SENSOR *psensors; /* Window sensor list. */
/*
WN_EVENT_LIST *pevent_list; /* Window event list. */
/*
int attr; /* Default attributes for data */
/* area. */
unsigned cur_left_marg, /* Left, right, top, and bottom */
cur_right_marg, /* margins to maintain during */
cur_top_marg, /* cursor "auto-track" feature. */
cur_bottom_marg; /*
struct /* "options" substructure: */
{ /* items set by user request. */
/*
unsigned delayed : 1; /* Output postponed until next , */
/* update. */
/*
unsigned cur_off : 1; /* Cursor invisible. */
/*
unsigned removable : 1; /* Removable (hence previous */
/* contents of screen must be */
/* preserved). */
/*
unsigned hidden : 1; /* Invisible, yet attached to a */
/* location on a display page. */

```

```

        /* */
        unsigned cur_track : 1; /* Always keep cursor visible in */
        /* the viewport. */
        /* */
        unsigned _dummy :11; /* Pad to word boundary */
    } options;
    struct /* "internals" substructure: */
    { /* not directly set by user. */
        /* */
        unsigned frozen :1; /* Updates postponed involuntarily. */
        /* */
        unsigned dirty :1; /* Output request(s) are */
        /* awaiting update. */
        /* */
        unsigned /* Some portion of data area is */
        any_data_covered:1; /* covered by another window. */
        /* */
        unsigned cur_frozen :1; /* Not selected for visible */
        /* cursor. */
        /* */
        unsigned temp_hid :1; /* Temporarily hidden by */
        /* internal operations. */
        /* */
        unsigned _dummy :11; /* Pad to word boundary */
    } internals;
} BWINDOW;

typedef struct bwin_node /* WIN_NODE structure: */
{ /* node in the linked lists which */
    /* govern the hierarchy of */
    /* windows on each display page. */
    /* */
    unsigned signature; /* Identifying signature. */
    /* */
    struct bwin_node *above; /* Window above this one (or */
    /* NIL if this is top window). */
    /* */
    struct bwin_node *below; /* Window below this one (or */
    /* NIL if this is bottom window). */
    /* */
    BWINDOW *pwin; /* BWINDOW structure belonging */
    /* to this node. */
} WIN_NODE;

```

```

/*****
/* Internal Data Types. */
*****/
typedef struct
{
    WN_ACTION    action;        /* The action to be taken. */
    KEY_SEQUENCE key_sequence;   /* Key sequence to trigger it. */
} WN_KEY;

typedef struct
{
    WN_ACTION    action;        /* The action to be taken. */
    MOUSE_EVENT  mouse;        /* Mouse event to trigger it. */
} WN_MOUSE;

/*****
/* Internal Windows Macros. */
*****/
/* Signature to identify genuine BWINDOW structure. */
#define BWIN_SIGN 0xe928
#define BWIN_DEAD 0
/* Signature to identify genuine WIN_NODE structure. */
#define BNODE_SIGN 0xd928
#define BNODE_DEAD 0
/* Signature to identify genuine WN_EVENT_LIST structure. */
#define BEVENT_SIGN 0x65ab
#define BEVENT_DEAD 0
/* Height of a REGION */
#define REGION_H(region) ((region).lr.row - (region).ul.row + 1)
/* Width of a REGION */
#define REGION_W(region) ((region).lr.col - (region).ul.col + 1)
#define BBRD_HASJOINTS 0x800
#define BBRD_TYPE 0x01f
#define BBRD_TITLE 0x7e0
#define B_WNPRBF_SIZE 1024
#define B_NATIVE_MAX_ROWS 25 /* Max rows supported by compiler's */
/* native text windows. */
#define wnreterr(errnum) return ((wnerror (errnum)), NIL)
#define wnreterz(errnum) return ((wnerror (errnum)), 0)
#define wnretern(errnum) return (wnerror (errnum))
#define wnreterv(errnum)

```

```

{
    wnerror (errnum);
    return;
}
#define wnvalidn(pnode)
{
    if (wnvalnod (pnode) == NIL)
        wnreterr (WN_BAD_NODE);
}
#define wnvalidw(pwin)
{
    if (wnvalwin (pwin) == NIL)
        wnreterr (WN_BAD_WIN);
}
#define wnvalide(pevent)
{
    if (wnvalevn (pevent) == NIL)
        wnreterr (WN_BAD_EVENT);
}
#define wnvalnod(pnode)  (wnvalno0((pnode),(BNODE_SIGN)))
#define wnvalwin(pwin)   (wnvalwi0((pwin),(BWIN_SIGN)))
#define wnvalevn(pevent) (wnvalev0((pevent),(BEVENT_SIGN)))
    /* Construct a new attribute from a given attribute */
    /* and fore and back, either of which may be -1. */
#define wndoattr(attr, fore, back)
    utnybbyt ((back) == -1 ? uthinyb (attr) : (back),
        (fore) == -1 ? utlonyb (attr) : (fore))
    /* Update a window's image. */
#define wnputimg(pwin)
    wnnupblk((pwin), 0, 0, (wndata_h(pwin) - 1),
        (wndata_w(pwin) - 1), WN_UPDATE)
    /* Macros to find a window's dimensions. */
    /* Width & height of data area (no border). */
#define wndata_w(pwin)  ((pwin)->img.dim.w)
#define wndata_h(pwin)  ((pwin)->img.dim.h)
    /* Width & height of window's border. */
#define wnbord_w(pwin)  ((pwin)->prev.dim.w)
#define wnbord_h(pwin)  ((pwin)->prev.dim.h)
    /* Width & height of window's viewport (no border). */
#define wnview_w(pwin)  ((pwin)->view_size.w)
#define wnview_h(pwin)  ((pwin)->view_size.h)

/*****

```

```

/* Error codes. */
/*****/

#define WN_NO_ERROR      0
#define WN_NO_MEMORY     1
#define WN_ILL_DIM       2
#define WN_NULL_PTR      3
#define WN_BAD_WIN       4
#define WN_BAD_DEV       5
#define WN_BAD_PAGE      6
#define WN_BAD_NODE      7
#define WN_ALREADY_SHOWN 8
#define WN_NOT_SHOWN     9
#define WN_CANT_HIDE     10
#define WN_COVERED       11
#define WN_BAD_OPT       12
#define WN_HARD          13
#define WN_ILL_VALUE     14
#define WN_PR_OVERFLOW   15

#define WN_NO_EVENT      30
#define WN_EVENT_EXISTS  31
#define WN_BAD_EVENT     32
#define WN_NO_EXIT       33

/*****/
/* Constants (type of borders and title(s)) for WNPOTBOR/WNDSPRAY. */
/*****/

#define BBRD_NO_BORDER 0x000
#define BBRD_CHAR      0x01f

/* The letters in the part after the underscore in
/* the following definitions are symbolic names for */
/* line styles. A border style with double lines */
/* on the top and left, and single lines on the */
/* right and bottom would be BBRD_DSDD. The order */
/* of sides in the definition is clockwise from the */
/* top. */

#define BBRD_SSSS 0x001
#define BBRD_SSSD 0x002
#define BBRD_DSSS 0x003
#define BBRD_DSDD 0x004
#define BBRD_SDSS 0x005
#define BBRD_SDDSD 0x006
#define BBRD_DDSS 0x007

```

```

#define BBRD_DDSD      0x008
#define BBRD_SSDS      0x009
#define BBRD_SSDD      0x00a
#define BBRD_DSDD      0x00b
#define BBRD_DSDD      0x00c
#define BBRD_SDDS      0x00d
#define BBRD_SDDD      0x00e
#define BBRD_DDDS      0x00f
#define BBRD_DDDD      0x010

        /* The letters in the part after the underscore in      */
        /* the following definitions are symbolic names for */
        /* title styles.      A title style with a top centered */
        /* title would be BBRD_TCT; one with a bottom right */
        /* title would be BBRD_BRT, and so on.                  */

#define BBRD_NO_TITLE 0x000
#define BBRD_TLT      0x020
#define BBRD_TCT      0x040
#define BBRD_TRT      0x080
#define BBRD_BLT      0x100
#define BBRD_BCT      0x200
#define BBRD_BRT      0x400

/*****/
/* Constants (direction of scroll) for WNSCRBLK. */
/*****/
#define WNSCR_UP 0
#define WNSCR_DOWN
#define WNSCR_RIGHT 2
#define WNSCR_LEFT

/*****/
/* Constants (option OR values) for WNWRSTRN. */
/*****/
#define WN_NO_UPDATE 0x04
#define WN_UPDATE 0x00
#define WN_LF_LF 0x08
#define WN_LF_CRLF 0x00
#define WN_NO_SCROLL 0x10
#define WN_SCROLL 0x00
#define WN_NO_TRACK 0x20
#define WN_CHAR_TRACK 0x00

/*****/

```

```

/* Window control items & status items for WNGETOPT & WNSETOPT. */
/* Negative item codes require the window to be currently */
/* displayed (although it may be hidden). */
/*****/
#define WN_DEVICE      1 /* Where the window is */
#define WN_PAGE      (-2) /* displayed */
#define WN_ROW_CORNER  (-3) /*
#define WN_COL_CORNER  (-4) /*
                        /*
#define WN_HIDDEN     (-5) /* Whether it's hidden.
                        /*
#define WN_CUR_OFF    6 /* Cursor on/off state */
#define WN_CUR_HIGH   7 /* Cursor scan lines */
#define WN_CUR_LOW    8 /*

                        /* User-controllable options */
#define WN_DELAYED    9 /*
#define WN_REMOVABLE  10 /*

#define WN_FROZEN     11 /* Internal effects of
#define WN_DIRTY      12 /* displaying windows
#define WN_ANY_DATA_COVERED 13 /*

#define WN_ROWS       14 /* Dimensions of data area */
#define WN_COLS       15 /*
                        /*
#define WN_ROW_REL     16 /* Cursor location relative to
#define WN_COL_REL     17 /* data area
                        /*
#define WN_ROW_ABS     (-18) /* Absolute cursor location on
#define WN_COL_ABS     (-19) /* screen

#define WN_ROW_OVERALL (-27) /* Location of window with
#define WN_COL_OVERALL (-28) /* border.
#define WN_HT_OVERALL  (-29) /* Dimensions of window with
#define WN_WID_OVERALL (-30) /* existing border.
                        /*
#define WN_ATTR        20 /* Default attributes for data
                        /* area
                        /*
#define WN_BOR_TYPE    22 /* Border type
#define WN_BOR_CHAR    23 /* Border character
#define WN_BOR_ATTR    24 /* Border attribute

```



```

#define WN_BOR_TTATTR    32 /* Border top title attribute */
#define WN_BOR_BTATTR    33 /* Border bottom title attribute */
/* */
#define WN_IS_CURRENT    25 /* Whether this is the current */
/* window */
/* */
#define WN_ACTIVE_CUR    (-26) /* Whether this window (among */
/* all the windows displayed on */
/* its page) is the one whose */
/* cursor is active. */
/* */
#define WN_PREV_ALLOC    31 /* Whether space has been allocated */
/* for the text underneath this */
/* window (when displayed and not */
/* hidden). */
/* */
#define WN_CUR_TRACK     34 /* Whether the window's data area */
/* will always be positioned so */
/* so that the cursor is visible in */
/* the viewport. */
/* */
#define WN_CUR_L_MARG    35 /* The number of columns to leave */
/* as a margin on the left and */
/* right when performing cursor */
/* auto tracking. */
/* */
#define WN_CUR_T_MARG    37 /* The number of rows to leave as a */
/* margin on the top and bottom */
/* when performing cursor auto */
/* tracking. */
/* */
#define WN_HT_VIEW       (-39) /* The height and width of the */
#define WN_WID_VIEW      (-40) /* window's viewport. */
/* */

/*****/
/* Constants (option OR values) for WNSCRLBR. */
/*****/
#define WN_HORIZONTAL 0x01
#define WN_VERTICAL 0x02

/*****/
/* Constants (option OR values) for WNREAD and WNPOLL. */

```

```

/*****
#define WN_KNOWN_EVENTS 0x00
#define WN_UNKNOWN_TRANSMIT 0x01
#define WN_ALL_TRANSMIT 0x03
#define WN_KBIGNORE 0x04
#define WN_USE_MOUSE 0x00
#define WN_NO_MOUSE 0x08

/*****
/* User Macros. */
/*****
#define wncreate(height,width,attr) \
    (wncreat0((height),(width),(attr),(BWIN_SIGN),(BNODE_SIGN)))
#define wndisplay(pwin, pwhere, pbord) \
    (wnvdisp((pwin),(pwhere),wndata_h((pwin)),wndata_w((pwin)), \
    0,0,(pbord)))
    /* Write a character to the current window */
    /* TTY-style. */
#define wnwrtty(ch, fore, back) \
    (wnwrttyx (b_pcurwin, (ch), (fore), (back), \
    WN_UPDATE | WN_LF_LF))
    /* Write a string to the current window TTY-style */
    /* with scrolling and end-of-line character wrap. */
#define wnwstr(pbuffer, fore, back) \
    (wnwstrm (b_pcurwin, (pbuffer), 0, (fore), (back), \
    WN_UPDATE | CHARS_ONLY))
    /* Change attributes on current window (but don't */
    /* change default). */
#define wnattr(fore, back) \
    (wnatrbk(b_pcurwin, 0, 0, (wndata_h(b_pcurwin) - 1), \
    (wndata_w(b_pcurwin) - 1), (fore), (back), \
    WN_UPDATE))
    /* Set up and allow user input in a window oriented */
    /* edit field. */
#define wnfield(pwin,pinitstr,pretstr,target_size,pfield_control,pfinal) \
    (edbase((pwin), (pinitstr), (pretstr), (target_size), \
    (pfield_control), (pfinal), (PED_RET_FUNC) wnretinf, \
    (PED_SET_FUNC) wnsetcur, (PED_WRITE_FUNC) wnwrrect))

/*****
/* Global variables */
/*****
extern int b_wnerr; /* Most recent error */

```

```

/* (WN_NO_ERROR if none). */
extern BWINDOW *b_pcurwin; /* Window selected for I/O */
/* (NIL if none). */
extern long b_win_hold; /* Amount of time to wait before */
/* reporting MO_HOLD events. */
extern unsigned b_win_delay; /* Amount of time to wait between */
/* successive window polls. */
extern WN_KEY b_winkeys[]; /* Array of default window key */
/* assignments. */
extern WN_MOUSE b_winmouse[]; /* Array of default window mouse */
/* assignments. */

/*****
/* Global variables for internal use only */
*****/
extern char *b_wnprbf; /* Buffer allocated for WNPRINTF. */
extern unsigned int b_wnbfsz; /* Size of buffer allocated for */
/* WNPRINTF. */
/* List of windows displayed on */
/* each page (NIL if none). */
extern WIN_NODE *(b_wnlist[MAX_DEVICES][MAX_PAGES]);
/* The window (on each page) */
/* selected to have a visible */
/* cursor (NIL if none). */
extern WIN_NODE *(b_pactnode[MAX_DEVICES][MAX_PAGES]);

/*****
/* User function declarations by category. */
*****/
/* Window destruction. */
int cdecl wndstroy (BWINDOW *); /* Detach from screen & discard */
/* memory structures (does not */
/* alter screen). */

/* Controlling/retrieving window options & status. */
BWINDOW *cdecl wnsetopt ( /* Set option. */
BWINDOW *, /*
int, int); /*
/*
BWINDOW *cdecl wngetopt ( /* Retrieve option or status. */
BWINDOW *, /*
int,int *); /*

```

```

/* Error reporting. */
#if WN_EXT_ERROR /* External version: */
int cdecl wncerror(int); /* Set or clear error code and */
/* return it. */
#else /* Macro version: */
#define wncerror(rcode) \ /*
    (b_wncerr = (rcode)) /*
#endif /*

/* Displaying/removing windows. */
BWINDOW *cdecl wncremove ( /* Remove window from display page. */
    BWINDOW *); /*
/*
BWINDOW *cdecl wncupdate ( /* Satisfy pending output requests. */
    BWINDOW *); /*
/*
BWINDOW *cdecl wncrevupd (void); /* Read image of current window */
/* from the screen. */
/*
int cdecl wncdraw (int, /* Redisplay all windows on a */
    int); /* display page and restore cursor. */

/* Cursor action. */
BWINDOW *cdecl wncursor ( /* Select for visible cursor */
    BWINDOW *); /* (deselect other windows shown */
/* on this display page.) */
/*
BWINDOW *cdecl wncurmov (int, /* Move current window's cursor */
    int); /* relative to the window's data */
/* area. */
/*
BWINDOW *cdecl wncurpos (int *, /* Return current window's cursor */
    int *); /* position relative to the */
/* window's data area. */
/*
/* Miscellaneous. */
BWINDOW *cdecl wncselect ( /* Select window for I/O (deselect */
    BWINDOW *); /* all other windows) making this */
/* the "current" window. */
/*
unsigned cdecl wncsetbuf ( /* Make scratch buffer for WNCPRINTF. */
    unsigned); /*
/*

```

```

BWINDOW *cdecl wnscribr (      /* Make scrollbar sensors.      */
    BWINDOW *, int,/*
    int, unsigned, /*
    int);          /*
void    cdecl wnzapsen(      /* Zap a window's sensor list.  */
    BWINDOW *); /*
    /* Window output.      */
void    cdecl wnwrap (int,      /* Write a string TTY-style with */
    const char *, int,      /* word wrap.      */
    int, int); /*
    /*
void    cdecl wnwstrn (      /* Write a string of a given length */
    BWINDOW *, /* TTY-style with end-of-line */
    const char *, int, /* character wrap.      */
    int, int, int);/*
    /*
int     cdecl wnwrbuf (int,      /* Write a buffer to a given      */
    int, int,      /* location in the current window's */
    const char *, int, /* data area.      */
    int, int); /*
    /*
int     cdecl wnprintf (      /* Do a PRINTF to the current      */
    const char *, ...); /* window.      */
    /*
BWINDOW *cdecl wnwrect (      /* Write to a rectangular region of */
    BWINDOW *, int,/* a window.      */
    int, int, int, /*
    const char *, /*
    int, int, int);/*
    /*
    /* Window input.      */
int     cdecl wnrdbuf (int,      /* Read part of the current      */
    int, int,      /* window's data area into a      */
    char *, int); /* buffer.      */
    /*
char    cdecl wnquery (char *, /* Return input from operator.    */
    int,int *); /*
    /*
WN_EVENT *cdecl wnread (      /* Display a window and accept user */
    BWINDOW *, /* responses.      */
    const WHERE *,/*
    int, int, /*
    int, int, /*

```

```

        const          /*
        BORDER *, /*
        WN_EVENT *, /*
        int); /*
        /*
WN_EVENT *cdecl wnpoll( /* Poll the mouse/keyboard once for */
        WN_EVENT_LIST *,/* a user response belonging to a */
        unsigned *, /* specified set of events. */
        unsigned *, /*
        WN_EVENT *); /*
        /*
int cdecl wninitv ( /* Initialize window event list. */
        BWINDOW *); /*
        /*
int cdecl wncgevn ( /* Change a window event in the */
        WN_EVENT_LIST **,/* specified event list. */
        WN_EVENT *, /*
        WN_ACTION); /*
        /*
int cdecl wnremevn( /* Delete a window event in the */
        WN_EVENT_LIST **,/* specified event list. */
        const /*
        WN_EVENT *); /*
        /*
void cdecl wnzapevn( /* Release the entire event list */
        WN_EVENT_LIST **);/* specified. */
        /*
        /* Window scrolling. */
BWINDOW *cdecl wnsrblk ( /* Scroll a block-section of a */
        BWINDOW *, /* window a specified number of */
        int, int, int, /* rows/columns. */
        int, int, int, /*
        int, int, int);/*
        /*
BWINDOW *cdecl wnsroll ( /* Vertically scroll current window.*/
        int, int, /*
        int, int); /*
        /*
BWINDOW *cdecl wnhoriz ( /* Horizontally scroll current */
        int, int, /* window. */
        int, int); /*
        /*
        /* Attribute manipulation, */

```

```

BWINDOW *cdecl wnatrbk (      /* Change attributes on a block- */
    BWINDOW *, /* section of a window. */
    int, int, int, /*
    int, int, int, /*
    int);          /*
                    /*
BWINDOW *cdecl wnatrstr (      /* Change attributes on a stream- */
    BWINDOW *, /* section of a window. */
    int, int, int, /*
    int, int,      /*
    int);          /*
                    /*

/* Virtual windows. */

BWINDOW *cdecl wnvdisp (      /* Display virtual window with */
    BWINDOW *, /* border. */
    const WHERE *, /*
    int, int,      /*
    int, int,      /*
    const          /*
    BORDER *); /*
                /*
BWINDOW *cdecl wnorigin (      /* Set the coordinates of the data */
    BWINDOW *, int, /* area which will appear in the */
    int, int); /* viewport's origin. */
                /*
BWINDOW *cdecl wnsoblk (      /* Shift the window's data area to */
    BWINDOW *, /* show a specified block. */
    const REGION *, /*
    const LOC *, /*
    const LOC *, /*
    int);          /*
                /*

/*****
/* Internal functions by category */
/*****

/* Window creation. */

BWINDOW *cdecl wncreat0(int,int,int,unsigned,unsigned);
/* Device & page switching. */
int cdecl wnseldev(const WHERE *,const DIM *,int *);
/* Error detection. */

BWINDOW *cdecl wnvalwi0(BWINDOW *,unsigned);

```

```

WIN_NODE *cdecl wnvalno0(WIN_NODE *,unsigned);
WN_EVENT_LIST *cdecl wnvalno0(WN_EVENT_LIST *,unsigned);
/* Restoring previous screen contents. */
BWINDOW *cdecl wnresprv(BWINDOW *);
/* Building IMAGE structures. */
BWINDOW *cdecl wnimbld(BWINDOW *,int,int,int,int);
IMAGE *cdecl wnimgtme(IMAGE *,const WHERE *);
/* Manipulating chains of WIN_NODES. */
WIN_NODE *cdecl wnpgadd(BWINDOW *,int,int);
BWINDOW *cdecl wnpgrem(BWINDOW *);
/* Building borders. */
int cdecl wnputhor(const DIM *,const BORDER *,const WHERE *);
/* Building sensor lists. */
int cdecl wnputsen(BWINDOW *);
/* Handling window events. */
WN_EVENT *cdecl wnretevn(WN_EVENT_LIST *, const WN_EVENT *);
/* Handling inter-window effects: displaying,
/* overlapping, removing.
BWINDOW *cdecl wnneedup(BWINDOW *);
BWINDOW *cdecl wnnupblk(BWINDOW *,int,int,int,int,int);
WIN_NODE *cdecl wncover(WIN_NODE *,const LOC *,const DIM *);
int cdecl wnoverlap(const LOC *,const DIM *,const LOC *,const DIM *);
BWINDOW *cdecl wnforget(BWINDOW *);
BWINDOW *cdecl wnhide(BWINDOW *);
BWINDOW *cdecl wnunhide(BWINDOW *);
/* Window output.
void cdecl wnwrttyx(BWINDOW *,char,int,int,int);
/* Cursor handling.
BWINDOW *cdecl wncurtrk(BWINDOW *, int, int);
void cdecl wncurset(BWINDOW *);
/* Native text window handling.
void cdecl wncmove(BWINDOW *, int, int);
void cdecl wnsetwin(int,int,int,int);
int cdecl wnchkdm(BWINDOW *);
void cdecl wngetatr(BWINDOW *);
void cdecl wnsetatr(int);
/* Window edit fields.
int cdecl wnretinf(BWINDOW *, CUR_INFO *, const LOC *, int, int);
int cdecl wnsetcur(BWINDOW *, int, CUR_TYPE *, int, int);
#endif /* Ends "#ifndef DEF_BWINDOW"

```