

内 容 简 介

本书以通俗易懂的语言、列举大量的实例揭示了 悦垣垣月 的各种用法。全书共分为 5 章,分别介绍了系统编程、界面设计、菜单工具栏和状态栏、鼠标光标和键盘、文件操作和驱动器、图形图像与多媒体、数据共享、操作注册表、线程与动态链接库、网络与通信、数据库编程、 数据库等内容。本书所展示的小例子,短小精干,恰到好处,点出 悦垣垣月 的应用精髓。

本书可作为 悦垣垣月 编程人员的参考用书或使用手册,也可供计算机爱好者使用。

图书在版编目 (CIP) 数据

奇思异想编程序援悦垣垣月 葛一楠等编著

北京 国防工业出版社, 1999

ISBN 7-115-03655-5

I ① 奇 ② 葛 ③ 悦语言—程序设计

IV ① 计算机

中国版本图书馆 CIP 数据核字 (1999) 第 10000 号

(北京市海淀区紫竹院南路 10 号)

(邮政编码 100088)

北京奥隆印刷厂印刷

新华书店经售

*

开本 787mm×1092mm 1/32 印张 10.4 插页 4 总字数

1999 年 1 月第 1 版 1999 年 1 月北京第 1 次印刷

印数 1—5000 册 定价 18.00 元

(本书如有印装错误,我社负责调换)

丛书编委会名单

李智慧摇摇蒋明礼摇摇李明东摇摇李光琳
向孟光摇摇吴家碕摇摇汪令江摇摇周启海
白晓毅摇摇马义玲摇摇董毅摇摇刘奇
史济民摇摇杨硕摇摇葛楠摇摇方宏
杨晓龙摇摇周学文摇摇卿川摇摇朱敏
陈杰华摇摇李春摇摇冉蜀阳摇摇陈浩
黎明摇摇向重伦摇摇张钟澍

前 言

C++Builder 自问世以来,由于它强大的 C++编译核心和简洁明快的可视化设计方式,深受越来越多的程序员的喜欢。C++Builder 发展到今天,也已经拥有了极其广泛的使用者,并且不断地还有新手在学习这个方便好用的开发工具,但很多情况下,大家和笔者当初一样,在熟悉了基本的应用之后,想再进一步的提高就感到非常困难。因此,迫切希望得到一本既不需要太多的基本应用介绍,而又具有非常丰富的开发技巧和经验的书。现在,经过实践的积累,笔者提供长期以来在编程上的一些心得体会,综合而成《奇思异想编程——C++Builder 篇》,以满足大家编程学习的愿望。《奇思异想编程——C++Builder 篇》最大的特点就是“奇思异想”的集成。它没有过多关于基本概念、基本知识、基本应用的阐述,而是提供一个个精采的编程示例。

本书的每一章都包含有丰富的编程经验和技巧,所以,读者在阅读本书时得到的最大好处是:不用再为修改一个个零散的实例代码而伤透脑筋,只需将它们完整地移植,或者将整个程序工程直接添加到读者的工程中,略作修改,就将起到事半功倍的效果。所有应用程序的源代码都包含在光盘中,这样就避免了可能的输入错误,从而保证程序的一次编译通过。

不过,因为 C++Builder 的功能实在太强大了,我们在较短的时间内没有将所有方面的编程技巧全部包含进来,我们所考虑的是:这不是一本书就能够完全包含的,我们也会继续努力,将这些经验尽可能地奉献给读者。

本书由葛一楠、方宏编写,在编写过程中也得到许多专家学者的帮助和支持,在此我们表示谢意。

另外,由于笔者的水平有限,在许多问题的考虑和实现上,体现的仅仅是个人的思想,可能读者会觉得有更好的方法来解决同样的问题。特别是对于书中存在的不足和错误,欢迎广大读者批评指正。

作 者

2003 年 5 月

目 录

第一章 系统编程	1
1.1 如何获取 Windows 版本信息	1
1.2 如何获取程序的命令行参数	2
1.3 如何获取内存状态信息	3
1.4 如何检测声卡配置	5
1.5 如何检测显示器信息	6
1.6 如何为 Windows 的任务栏布告区设置图标	7
1.7 如何隐藏/显示 Windows 任务栏	10
1.8 如何隐藏/显示桌面上的图标	12
1.9 如何隐藏应用程序的任务栏图标	13
1.10 如何获取系统的度量信息和相关配置信息	14
1.11 如何存取系统的颜色信息	16
1.12 如何获取 Windows 及其系统路径	17
1.13 如何获取计算机名称	18
1.14 如何关闭 Windows	20
1.15 如何获取系统参数信息	21
1.16 如何隐藏应用程序	22
1.17 如何为程序在启动菜单中创建快捷方式	24
1.18 如何在 Windows 启动时运行某个应用程序	26
1.19 如何同时只运行应用程序的一个实例	28
1.20 如何使用 API 函数获取 CPU 信息	30
1.21 如何利用 CPUID 汇编指令获取 CPU 信息	31
第二章 界面设计	34
2.1 如何在 RichEdit 控件中存取文件	34
2.2 如何移动一个没有标题栏的窗体	35
2.3 如何制作动态字幕	38
2.4 如何制作一个不可改动的窗体	39
2.5 如何使窗体始终处于最上层	41
2.6 如何在窗体的标题栏上显示日期和时间	42
2.7 如何使创建的窗体始终处于最小化状态	43
2.8 如何使创建的窗体始终处于最大化状态	44

2.9	如何使创建的窗体利用帮助文件.....	45
2.10	如何制作一个 Splash 窗口.....	46
2.11	如何制作可移动的控件.....	48
2.12	如何限定所设计窗体的大小.....	49
2.13	如何在窗体中添加闪烁的文字.....	50
2.14	如何制作闪烁的窗体.....	51
2.15	如何制作一个椭圆形窗体.....	53
2.16	如何确定一个窗口是否为 Top Level 窗口.....	54
2.17	如何自定义控件 Memo 的边界.....	55
2.18	如何制作带背景的窗体.....	57
2.19	如何使窗体的大小不因屏幕分辨率的改变而改变.....	58
2.20	如何使窗体的背景色呈渐变状态.....	60
2.21	如何创建一个透明的窗体.....	62
2.22	如何制作应用程序的封面.....	63
2.23	如何将窗体从属于主窗体.....	65
2.24	如何显示旋转字体.....	70
2.25	如何制作一个半透明的窗体.....	71
2.26	如何询问用户是否真的想要关闭窗口.....	74
2.27	如何确保控件在任何情况下均居中.....	75
2.28	如何利用“容器”控制成组控件位置.....	76
2.29	如何在不同分辨率下保持窗体位置及大小.....	78
2.30	如何实现窗体上的图像呈拉幕式打开.....	78
2.31	如何实现窗体看上去像呈栅栏式闪动.....	80
2.32	如何让窗体图像呈翻页一样打开.....	81
2.33	如何改变提示的字体及颜色.....	82
2.34	如何实现窗体的自动隐藏.....	83
2.35	如何在运行期间从头创建一个窗体.....	84
2.36	如何为窗体设置最大/最小窗口.....	86
2.37	如何在 MDI 父窗体上绘制一个颜色渐变的背景.....	87
2.38	如何在程序开始运行时显示一个醒目的屏幕.....	90
第三章	菜单.....	93
3.1	如何获取窗口标题栏中的文字.....	93
3.2	如何动态管理菜单.....	94
3.3	如何实现模态显示.....	97
3.4	如何在状态栏中添加进度条.....	99
3.5	如何制作可四处拖动的工具栏.....	100
3.6	如何将菜单项移到菜单栏的最右边.....	102
3.7	如何在运行时移动控件.....	103
3.8	如何在系统菜单中添加自定义选项.....	104

3.9	如何制作帮助系统.....	105
3.10	如何拖放工具条.....	108
3.11	如何在菜单中放入图像.....	109
3.12	如何使用右键弹出菜单.....	110
第四章	鼠标和键盘.....	112
4.1	如何检测 Shift、Alt 和 Ctrl 键是否按下.....	112
4.2	如何屏蔽系统功能键.....	113
4.3	如何模拟按下键盘上的某个键.....	115
4.4	如何限制鼠标移动的范围.....	116
4.5	如何自定义鼠标形状.....	117
4.6	如何设置光标闪烁的速度.....	118
4.7	如何实现在一个应用程序中的拖曳操作.....	120
4.8	如何实现在不同应用程序间的拖曳操作.....	123
4.9	如何为窗体创建一个动画光标.....	124
4.10	如何使用 Enter 键控制焦点切换.....	125
4.11	如何检测鼠标位置.....	126
4.12	如何显示编辑框中的密码.....	127
4.13	如何调整字体的大小和阴影.....	128
第五章	文件目录和驱动器.....	130
5.1	如何在指定的路径中查找指定的文件.....	130
5.2	如何在 StringGrid 控件中显示文件的数据.....	131
5.3	如何将缓冲区中的数据写入到指定的文件中.....	133
5.4	如何复制文件并显示源文件的大小.....	134
5.5	如何获取文件的长度.....	136
5.6	如何按照指定的有效位转换数字.....	137
5.7	如何获取驱动器类型信息.....	138
5.8	如何将 Edit 控件中的信息保存到.ini 文件中.....	140
5.9	如何获取文件的日期信息.....	142
5.10	如何检测磁盘或光盘是否有变化.....	144
5.11	如何检测驱动器容量.....	146
5.12	如何复制整个目录.....	148
5.13	如何将文件删除到回收站.....	150
5.14	如何检测驱动器是否就绪.....	151
5.15	如何获取应用程序的文件名.....	152
5.16	如何操作临时文件.....	153
5.17	如何通过修改 boot.ini 文件来修复启动菜单.....	154
5.18	如何使用全局函数 Printer 实现打印.....	156
5.19	如何获取默认打印机的信息.....	158

5.20	如何获取打印机队列的状态信息.....	160
5.21	如何在 Win.ini 中保存信息.....	162
5.22	如何在删除、移动或复制文件时显示进程.....	164
5.23	如何监视文件的变化.....	165
5.24	如何在整个硬盘中搜索一个文件.....	167
5.25	如何实现组合框的自动搜索.....	168
5.26	如何改变“打开”对话框中“打开”按钮的标题.....	170
5.27	如何实现 ListView 的列标头点击排序功能.....	171
5.28	如何实现搜索文件的功能.....	173
第六章	图形图像与多媒体.....	176
6.1	如何使用 ScanLine 加快图像像素的访问速度.....	176
6.2	如何制作马赛克效果.....	177
6.3	如何在 DBGrid 控件的单元格中绘制图形.....	179
6.4	如何使 DBGrid 控件以不同的颜色突出显示一些重要数据.....	180
6.5	如何实现超大图像的显示.....	181
6.6	如何跟踪鼠标以产生橡皮条效果.....	183
6.7	如何在 Windows“开始”按钮上绘图.....	185
6.8	如何实现“中心扩散”效果.....	187
6.9	如何实现“百叶窗帘”效果.....	188
6.10	如何将位图旋转 90°.....	190
6.11	如何将位图左右旋转.....	192
6.12	如何利用 C++Builder 实现发送图像文件的功能.....	193
6.13	如何获取应用程序的图标.....	196
6.14	如何在 C++Builder 中显示透明位图.....	197
6.15	如何在 ListView 控件中绘底图.....	199
6.16	如何复制窗体并保存为位图.....	200
6.17	如何复制图像的一部分.....	202
6.18	如何通过 OpenGL 实现全屏幕.....	203
6.19	如何在 C++Builder 中快速显示不规则窗体.....	206
6.20	如何实现“Print Screen Sys Rq”键的功能.....	208
6.21	如何使用画线函数画线.....	210
6.22	如何在 C++ Builder 中自动关闭屏幕保护程序.....	214
6.23	如何实现图像的打印.....	215
6.24	如何将 BMP 文件转换为 JPEG 文件.....	217
6.25	如何实现拉动特技效果.....	218
6.26	如何实现从中心到四周扩散的效果.....	220
6.27	如何控制和播放 CD.....	222
6.28	如何对图像进行柔化处理.....	223
6.29	如何对图像进行锐化处理.....	226

6.30	如何对图像进行浮雕处理.....	228
6.31	如何实现文本的旋转.....	230
6.32	如何改变文本的宽度.....	232
6.33	如何改变文本的高度.....	234
6.34	如何复制当前窗体.....	235
第七章	数据共享.....	237
7.1	如何通过剪贴板实现对图形的复制与粘贴.....	237
7.2	如何通过剪贴板对控件进行操作.....	238
7.3	如何操作定制格式的数据.....	239
7.4	如何利用 DDE 实现在两个应用程序之间数据的交换.....	241
7.5	如何实现创建程序组和程序项的功能.....	245
7.6	如何利用内存映射文件.....	247
7.7	如何利用内存映射文件在多个应用程序间共享消息.....	249
第八章	操作注册表.....	253
8.1	如何使用 Tregistry 类来操作文件.....	253
8.2	如何使用 Tregistry 类来实现“ Windows 自动登录 ”.....	256
8.3	如何为 Windows XP 添加五笔字型输入法.....	260
8.4	如何为计算机增加启动日志.....	261
8.5	如何利用 Windows 注册表存储信息.....	264
8.6	如何获取用户注册信息.....	266
8.7	如何在 C++ Builder 环境中实现在菜单中显示历史文件列表.....	267
8.8	如何使应用程序在系统启动时运行.....	269
第九章	线程与动态链接库.....	271
9.1	如何防止一个没有窗体的 Windows 程序的重复运行.....	271
9.2	如何使用函数 Synchronize 实现线程的同步.....	272
9.3	如何设置线程的优先级.....	274
9.4	如何使用线程来比较三种排序方法的快慢.....	276
9.5	如何创建一个独立的执行线程.....	278
9.6	如何从后台线程中访问屏幕.....	279
9.7	如何在后台运行查询.....	280
9.8	如何使一个线程等待一个事件发生.....	282
9.9	如何动态调用 DLL.....	283
9.10	如何利用线程显示图像.....	285
9.11	如何通过匿名管道实现进程间的通信.....	287
第十章	网络与通信.....	291
10.1	如何获取拨号上网的 IP 地址.....	291
10.2	如何实现超级链接效果.....	292
10.3	如何获取本机的 IP 地址.....	294

10.4	如何实现映射网络驱动器.....	297
10.5	如何获取网络适配器的信息.....	298
10.6	如何实现语音拨号.....	301
10.7	如何在程序中控制 IE 窗口.....	302
10.8	如何判断是否安装了网络协议.....	304
10.9	如何获取工作组名称及个数.....	306
10.10	如何获取本机 MAC 地址.....	308
10.11	如何监测 Internet 连接类型.....	311
10.12	如何通过命名管道实现网络间的通信.....	313
10.13	如何通过邮槽实现广播消息.....	317
10.14	如何发送和接收短消息.....	320
10.15	如何实现发送和接收用户数据包.....	321
10.16	如何实现发送广播消息.....	324
10.17	如何实现点对点聊天.....	326
10.18	如何查询用户信息.....	328
10.19	如何使用控件检测主机是否提供相应的服务.....	329
10.20	如何检测主机是否提供某一服务.....	331
10.21	如何发送和接收文件数据流.....	332
10.22	如何对文件进行编码和解码.....	334
10.23	如何实现阅读和张贴新闻.....	336
10.24	如何通过 Internet 发送电子邮件.....	338
10.25	如何实现 POP3 电子邮件的接收.....	340
10.26	如何实现在线接收电子邮件.....	342
第十一章	数据库编程.....	347
11.1	如何修改指定字段值.....	347
11.2	如何浏览只读数据.....	348
11.3	如何显示和编辑图形图像数据.....	349
11.4	如何将数据库中字段的值转换成字符串.....	350
11.5	如何在数据库表中添加计算字段.....	351
11.6	如何使用 GotoKey 查询记录.....	353
11.7	如何使用 FindKey 查询记录.....	354
11.8	如何使用模糊查询查询记录.....	355
11.9	如何使用 Locate 查询记录.....	356
11.10	如何获取数据库别名的列表.....	357
11.11	如何获取数据库别名的参数信息.....	358
11.12	如何获取 BDE 数据库别名和所有 DataBase 控件的名称.....	359
11.13	如何获取 Session 控件可以使用的 BDE 驱动器的名称.....	360
11.14	如何获取一个指定的 BDE 驱动器的信息.....	362
11.15	如何获取与数据库控件相连的所有表格的名称.....	363

11.16	如何将 BMP 放入 dBASE 和 Paradox 的 BLOB 字段中	364
11.17	如何将文本文件转换成 Paradox 格式的数据库	365
11.18	如何动态选择数据库和数据表	366
11.19	如何使用 Lookup 查询数据库	367
11.20	如何通过 SetRange 方法查找固定范围的数据	368
11.21	如何通过 TQuery 方法查找固定范围的数据	369
11.22	如何通过 Filter 方法查找固定范围的数据	370
11.23	如何使用数据库的异常处理	371
11.24	如何压缩 Paradox 数据表	372
11.25	如何在运行时创建一个 BDE 别名	374
11.26	如何在相关数据库中使用查找控件	375
11.27	如何显示被删除的数据记录	376
11.28	如何加快记录指针的移动速度	378
11.29	如何获取数据库的操作状态	379
11.30	如何获取数据库记录信息	381
11.31	如何使用计算字段显示记录位置	382
11.32	如何实现记录指针位置的存储与返回	383
11.33	如何修改大量的数据	384
11.34	如何筛选数据	385
11.35	如何融合筛选功能和查找功能	386
11.36	如何不使用 data-ware 控件编辑数据库	388
11.37	如何在运行期间控制数据表的布局	389
11.38	如何在运行期间创建一个 BDE 别名	390
11.39	如何实现数据库的缓冲更新	391
11.40	如何实现自动 Login 数据库	393
11.41	如何通过 IBX 获取数据库信息	394
第十二章	SQL 数据库	396
12.1	如何使用 TQuery 实现参数化查询	396
12.2	如何在参数化查询中使用 Format 函数	397
12.3	如何实现统计图表与数据库的结合	399
12.4	如何利用 TQuery 和 TStoreProc 实现存储过程	400

第一章 系统编程

1.1 如何获取 Windows 版本信息



问题 :有些应用程序对 Windows 的版本有一定的要求,这就需要知道 Windows 的版本号,如何获取版本信息呢。

解决方法 :为了解决这个问题,可以利用 Windows API 函数 `GetVersionEx` 获取 Windows 版本信息。

函数 `GetVersionEx` 的原型如下:

```
BOOL GetVersionEx(  
    LPOSVERSIONINFO lpVersionInformation    //指向版本信息结构的指针  
);
```

版本信息的结构定义如下:

```
typedef struct _OSVERSIONINFO{  
    DWORD dwOSVersionInfoSize; //版本信息结构的字节大小  
    DWORD dwMajorVersion;      //主版本号  
    DWORD dwMinorVersion;      //次级版本号  
    DWORD dwBuildNumber;       //构件数  
    DWORD dwPlatformId;        //平台标识  
    TCHAR szCSDVersion[ 128 ];  
} OSVERSIONINFO;
```

使用函数 `GetVersionEx` 时,首先应将版本信息结构中的 `dwOSVersionInfoSize` 设置为结构的字节数,可以通过 `sizeof(OSVERSIONINFO)` 来获取版本信息结构的字节数。

版本信息结构中的 `dwMajorVersion` 标识了 Windows 的主版本号。例如,Windows 98 为 4,Windows NT4.0 也为 4; `dwMinorVersion` 标识了 Windows 的次级版本号。例如,Windows 98 为 10,Windows NT4.0 为 0; `dwBuildNumber` 标识了 Windows 的构件数。对于 Win9x 的版本,其中还包括了主版本号和次级版本号的信息。

版本信息结构中的 `dwPlatformId` 标识了 Windows 的平台标识,具体数值及含义如下:

※ <code>VER_PLATFORM_WIN32s</code>	Win32 on Windows 3.1
※ <code>VER_PLATFORM_WIN32_WINDOWS</code>	Win32 on Windows 95
※ <code>VER_PLATFORM_WIN32_NT</code>	Win32 on Windows NT

版本信息结构中的 `szCSDVersion` 包含了一些附加信息。例如,Windows NT 中的补丁信息等。





实例：获取 Windows 版本信息。

具体实现过程如下：

在窗体 Form1 上添加一个 Button 控件和一个 Edit 控件。

最后运行结果界面如图 1.1 所示。

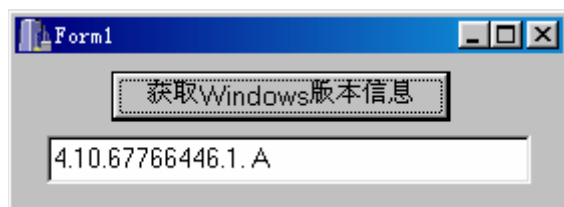


图 1.1

添加 Button 控件的 OnClick 事件的处理过程如下：

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    OSVERSIONINFO OSVI;
    OSVI.dwOSVersionInfoSize=sizeof(OSVERSIONINFO);           ①
    GetVersionEx(&OSVI);                                       ②
    Edit1->Text =IntToStr(OSVI.dwMajorVersion)+"."+
    IntToStr(OSVI.dwMinorVersion) + "."+IntToStr(OSVI.dwBuildNumber)+ "."
    +IntToStr(OSVI.dwPlatformId)+ "."+OSVI.szCSDVersion;        ③
}
```



程序说明：①句设置版本信息结构的大小；②句获取 Windows 版本信息；③句显示信息。



提示：1. 调用函数 GetVersionEx 时，可直接使用 OSVERSIONINFO 结构的一个变量作为参数。

2. 实际应用程序的开发过程中，应对获取的版本信息做进一步的处理，以使最后显示的信息更加明了。

3. 由于函数 GetVersion 的诸多局限性，建议读者使用函数 GetVersionEx 获取 Windows 版本信息。

1.2 如何获取程序的命令行参数



问题：在 DOS 系统流行的年代，人们对命令参数记得非常熟悉，但现在大多数用户使用的是 Windows 系统，难免对 DOS 下的命令参数比较陌生。有时用户仍需要知道命令行参数，如何在 C++Builder 中获取程序的命令行参数呢。





解决方法：可以用下面两种不同的方法来解决这个问题：

1. 调用 VCL ParamStr 函数，ParamStr 需要一个整数参数并且返回一个 AnsiString 对象。若参数为 0，ParamStr 将返回可执行文件的全称路径；若参数为 1，将返回程序名及第一个命令行参数；若参数为 2，将返回第二个参数，依次类推。

2. 调用 Windows API 函数 GetCommandLine。GetCommandLine 不需要参数，并且返回 char*，包含全部的命令行参数。

下面使用这两种方法来实现。



实例：获取程序的命令行参数。

具体实现过程如下：

在窗体 Form1 上添加六个 Label 控件。

最后运行结果界面如图 1.2 所示（上下两行分别对应解决方法中的两种方法）。

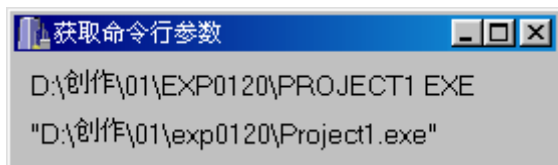


图 1.2

在窗口的构造函数中添加以下代码：

```
-----
_fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
    Label1->Caption = ParamStr(0);           ①
    Label2->Caption = ParamStr(1);
    Label3->Caption = ParamStr(2);
    Label4->Caption = ParamStr(3);
    Label5->Caption = ParamStr(4);
    Label6->Caption = AnsiString(GetCommandLine()); ②
}
-----
```



程序说明：① 句调用函数 ParamStr；② 句调用函数 GetCommandLine。



提示：函数 ParamStr 对目录中的空格能智能判断。为证实这点，把生成的 EXE 文件复制到 Program Files 目录下再运行，将会看到 ParamStr(0)返回全路径，并包含空格。

1.3 如何获取内存状态信息



问题：在编写的程序中常常要和内存打交道，那么如何在程序中确定系统中内存的运行状态呢。



解决方法：通过 Windows API 函数 GlobalMemoryStatus 可以获取内存信息。
函数原型如下：

```
VOID GlobalMemoryStatus(  
    LPMEMORYSTATUS lpBuffer    // 指向内存状态信息结构变量的指针  
);
```

其中内存状态信息结构变量的定义如下：

```
typedef struct _MEMORYSTATUS {    // mst  
    DWORD dwLength;                // sizeof(MEMORYSTATUS)  
    DWORD dwMemoryLoad;            // 正在使用的内存占总内存的百分比  
    DWORD dwTotalPhys;             // 物理内存的字节数  
    DWORD dwAvailPhys;             // 未使用的物理内存的字节数  
    DWORD dwTotalPageFile;         // 交换文件的字节数  
    DWORD dwAvailPageFile;         // 未使用的交换文件的字节数  
    DWORD dwTotalVirtual;          // 虚拟内存的字节数  
    DWORD dwAvailVirtual;          // 未使用的虚拟内存的字节数  
} MEMORYSTATUS, *LMEMORYSTATUS;
```

实例：获取内存状态信息。

具体实现过程如下：

在窗体 Form1 上添加一个 Button 控件和一个 Memo 控件。

最后运行结果界面如图 1.3 所示。



图 1.3

添加 Button 控件的 OnClick 事件的处理过程如下：

```
void __fastcall TForm1::Button1Click(TObject *Sender)  
{  
    MEMORYSTATUS MemInfo;  
    MemInfo.dwLength=sizeof(MEMORYSTATUS);  
    GlobalMemoryStatus(&MemInfo);
```

①



```

Memo1->Lines->Add(IntToStr(MemInfo.dwMemoryLoad)+"%的内存正在使用");
Memo1->Lines->Add("物理内存共有(Mb):"+String(int(MemInfo.dwTotalPhys /1024/1024))); ②
Memo1->Lines->Add("未使用的物理内存共有(Mb):"+String(int(MemInfo.dwAvailPhys / 1024 /
1024)));
Memo1->Lines->Add("交换文件的大小为(Mb):"+String(int(MemInfo.dwTotalPageFile/1024/
1024)));
Memo1->Lines->Add("未使用的交换文件的大小为(Mb):"+String(int(MemInfo.dwAvailPageFile
/1024/1024)));
Memo1->Lines->Add("虚拟内存空间大小为(Mb):"+String(int(MemInfo.dwTotalVirtual/1024/1024)));
Memo1->Lines->Add("未使用的虚拟空间的大小为(Mb):"+String(int(MemInfo.dwAvailVirtual/
1024/1024)));
}

```



程序说明：① 句获取内存信息；② 句显示物理内存信息。



提示：② 句也可改为(需在窗体上添加 Edit 控件)：

```
Edit1->Text="物理内存共有(Mb):"+IntToStr(MemInfo.dwTotalPhys /1024/1024));
```

1.4 如何检测声卡配置



问题：在编制多媒体程序时，常常会使用声音文件。但当这些程序在未配置声卡的机器上运行时，应该给出必要的警告。这就需要检测声卡的配置，如何检测呢。

解决方法：对于声卡的检测，可以通过调用 Windows API 函数 waveOutGetNumDevs 和 midiOutGetNumDevs 检测波形设备和 MIDI 设备，再利用 waveOutGetDevCaps 和 midiOutGetDevCaps 获得声音设备的细节资料。



实例：检测声卡配置。

具体实现过程如下：

在窗体 Form1 上添加一个 Button 控件和一个 Memo 控件。

最后运行结果界面如图 1.4 所示。



图 1.4

添加 Button 控件的 OnClick 事件的处理过程如下：



```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int wavedevice, mididevice;
    WAVEOUTCAPS wavecap;
    MIDIOUTCAPS midicap;
    wavedevice=(int)waveOutGetNumDevs();    ①
    mididevice=(int)midiOutGetNumDevs();    ②
    if (wavedevice==0)
        Memo1->Lines->Add ("没有发现波形设备");
    else
    {
        waveOutGetDevCaps(0,&wavecap,sizeof(WAVEOUTCAPS));
        Memo1->Lines->Add("当前波形设备:"+String(wavecap.szPname));
    }
    if (mididevice==0)
        Memo1->Lines->Add ("没有发现 MIDI 设备");
    else
    {
        midiOutGetDevCaps(0,&midicap,sizeof(MIDIOUTCAPS));
        Memo1->Lines->Add ("当前 MIDI 设备:"+String(midicap.szPname));
    }
}
```



程序说明：① 句波形设备信息；② 句 MIDI 设备信息。



提示：在 Unit.h 头文件中包含 mmSystem 单元的头文件：
#include <mmSystem.hpp> //声卡设备的 API 函数所在单元

1.5 如何检测显示器信息



问题：在编写和图形图像有关的程序时常常需要检测显示器的分辨率和色深，如何检测这些信息呢。

解决方法：分辨率的获取方法很简单，可直接调用 Screen 对象的属性就行了。而要求色深则要利用 Windows API 函数 GetDeviceCaps 获得每像素的比特数和色彩的页面数，然后计算 2 的“每像素的比特数”次幂即得“色彩的梯度数”，再计算“色彩的梯度数”的“色彩的页面数”次幂即得色深。



实例：检测显示器信息。

具体实现过程如下：

在窗体 Form1 上添加一个 Button 控件和一个 Memo 控件。

最后运行结果界面如图 1.5 所示。



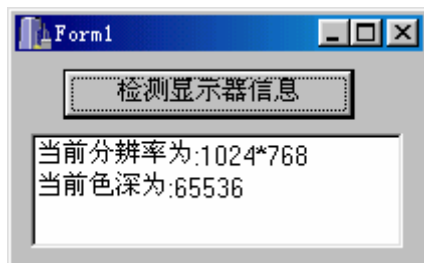


图 1.5

添加 Button 控件的 OnClick 事件的处理过程如下：

```
void _fastcall TForm1::Button1Click(TObject *Sender)
{
    int tcs;
    long int bpp,cp,tc;
    Memo1->Lines->Add ("当前分辨率为:"+String(Screen->Width)+"*"+String(Screen->Height)); ①
    bpp=GetDeviceCaps(Form1->Canvas->Handle,BITSPIXEL);
    tcs=pow(2,bpp); ②
    cp=GetDeviceCaps(Form1->Canvas->Handle,PLANES);
    tc=pow(tcs,cp); ③
    Memo1->Lines->Add("当前色深为:"+String(tc));
}
```



程序说明：① 句显示分辨率；② 句计算色彩的梯度数；③ 句计算色深。



提示：由于此实例程序用到了幂运算，所以要包含数学运算的头文件：

```
#include <Math.h>
```

1.6 如何为 Windows 的任务栏布告区设置图标



问题：许多应用程序，例如输入法管理器、杀毒软件等均在任务栏布告区中放置一个有自己特色的图标，使用户知道有一个后台程序正在运行，同时也提供了一种修改系统设置的快捷方法。如何在 C++Builder 中实现此功能呢。



解决方法：调用 Windows API 函数 Shell_NotifyIcon 给系统发送消息来增加、修改或者删除任务栏区域的图标，函数原型如下：

```
WINSHELLAPI BOOL WINAPI Shell_NotifyIcon(
    DWORD dwMessage,          //消息标识符
    NOTIFYICONDATA pnid       // 结构指针
);
```



实例：为 Windows 的任务栏布告区设置图标。