内摇容摇简摇介

本书以通俗易懂的语言、列举大量的实例揭示了 悦垣垣月遇瞬的各种用法。全书共分为 员圆章，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、杂匝蕴数据库等内容。本书所展示的小例子，短小精干，恰到好处，点出 悦垣垣月遇瞬的应用精髓。

本书可作为 悦垣垣月遇瞬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。

摇图书在版编目（悦陨孕）数据

摇奇思异想编程序援悦垣垣 月遇瞬篇 辕葛一楠等编著援—北京：国防工业出版社，圆园园缘
摇陨杂月晕 苑鄄员员愿鄄园猿怨苑缘鄄苑

摇Ⅰ援奇摇援援援摇Ⅱ援葛摇援援援摇Ⅲ援悦语言—程序设计
Ⅳ援栽孕猿员员

摇中国版本图书馆 悦陨孕数据核字（圆园园缘）第 员园远愿缘员号

（北京市海淀区紫竹院南路 圆猿号）
（邮政编码摇员园园园源源）
北京奥隆印刷厂印刷
新华书店经售

*

开本 苑愿苑伊员园怨圆摇员辕员远摇印张 圆苑¼摇缘愿怨千字
圆园园缘年 员月第 员版摇摇圆园园缘年 员月北京第 员次印刷
印数：员—缘园园园册摇摇定价：猿缘郾园园元

（本书如有印装错误，我社负责调换）

丛书编委会名单

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《奇思异想编程序——C++Builder 篇》，以满足大家编程学习的愿望。《奇思异想编程序——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　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：

※ VER_PLATFORM_WIN32s　　Win32 on Windows 3.1

※ VER_PLATFORM_WIN32_WINDOWS　　Win32 on Windows 95

※ VER_PLATFORM_WIN32_NT　　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。

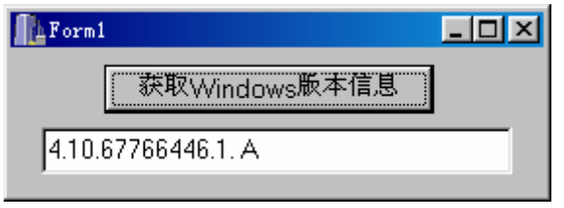

图 1.1

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 ParaStr **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）。**

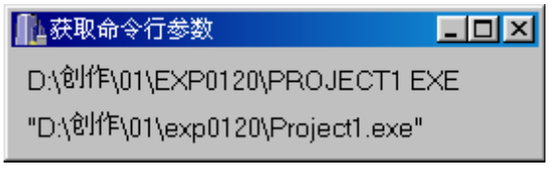

图 1.2

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, *LPMEMORYSTATUS;
```

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。**

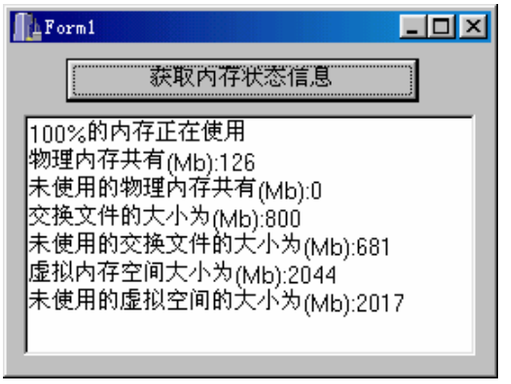

图 1.3

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):"+IntToStrt(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。

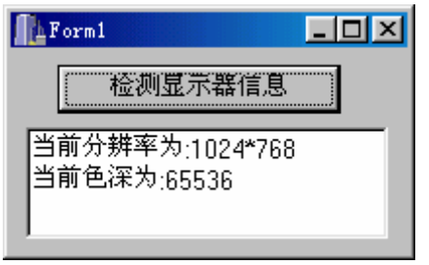

图 1.5

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
      PNOTIFYICONDATA pnid              //  结构指针
    );
```

实例：为 Windows 的任务栏布告区设置图标。