

3

TP 312C
Z68C

C++ Builder 与 Windows API 经典范例

郑 明 郑世伟 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

WinAPI 函数实例在 Windows 操作平台上具有跨程序语言的共性,也是除 C++ Builder 之外程序员必须参考的程序。本书内含 132 个 WinAPI 函数及 206 个完整、可执行的实例。在实例中,针对其中使用最频繁的 WinAPI 函数提出可行的实例,同时提供窗口程序员简易快捷的参考资料,让读者翻阅时可即时提醒思考方向,减少回忆函数用法的时间。

实例中还有利用多个不同函数协调完成一项功能的范例,可引导程序员使用函数用法。另外,还包含具有完整注释的鼠标和键盘的 Hook 连接文件与可执行文件程序代码。适合在 Windows 操作系统中,编写窗口程序而需使用 WinAPI 函数的程序员和其他相关技术人员使用或参考。

本书繁体字版书名为《C++ Builder & Windows API 范例辞典》,由文魁资讯股份有限公司出版,版权属郑明、郑世伟所有。本书简体字中文版由文魁资讯股份有限公司授权清华大学出版社独家出版。未经本书原版出版者和本书出版者书面许可,任何单位和个人不得以任何形式或任何手段复制或传播本书的部分或全部内容。

北京市版权局著作权合同登记号:图字 01-2002-2549 号

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

书 名: C++ Builder 与 Windows API 经典范例

作 者: 郑 明 郑世伟 编著

责任编辑: 马 丽

出 版 者: 清华大学出版社(北京清华大学学研大厦,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印 刷 者: 北京牛山世兴印刷厂

发 行 者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 31.25 字数: 750 千字

版 次: 2002 年 8 月第 1 版 2002 年 8 月第 1 次印刷

书 号: ISBN 7-900643-29-X

印 数: 0001~5000

定 价: 49.00 元(含 1 张光盘)

目 录

第 1 章 文件函数	1	1.17 GetLogicalDrives 函数	49
1.1 AreFileApisANSI 函数, SetFileApisToANSI 函数, SetFileApisToOEM 函数	1	1.18 GetLogicalDriveStrings 函数	51
1.2 CopyFile 函数	3	1.19 GetShortPathName 函数	53
1.3 CreateDirectory 函数, CreateDirectoryEx 函数	5	1.19.1 范例一	53
1.4 CreateFile 函数	7	1.19.2 范例二	56
1.4.1 范例一	7	1.20 GetTempFileName 函数	58
1.4.2 范例二	9	1.21 GetTempPath 函数	60
1.4.3 范例三	11	1.22 GetVolumeInformation 函数	62
1.5 DeleteFile 函数	14	1.22.1 范例一	62
1.5.1 范例一	14	1.22.2 范例二	64
1.5.2 范例二	15	1.23 LockFile 函数	66
1.6 FindFirstFile 函数, FindNextFile 函数, FindClose 函数	17	1.24 MoveFile 函数	68
1.6.1 范例一	18	1.25 ReadFile 函数	69
1.6.2 范例二	20	1.26 RemoveDirectory 函数	73
1.7 FindCloseChangeNotification 函数, FindFirstChangeNotification 函数, FindNextChangeNotification 函数	22	1.27 SearchPath 函数	75
1.8 GetBinaryType 函数	26	1.28 SetCurrentDirectory 函数	77
1.9 GetCurrentDirectory 函数	28	1.29 SetFileAttributes 函数	79
1.10 GetDiskFreeSpace 函数	29	1.30 SetFilePointer 函数	81
1.10.1 范例一	29	1.31 SetVolumeLabel 函数	84
1.10.2 范例二	31	1.32 WriteFile 函数	86
1.11 GetDriveType 函数	33	1.32.1 范例一	86
1.12 GetFileAttributes 函数	35	1.32.2 范例二	89
1.13 GetFileInformationByHandle 函数	38	第 2 章 位图函数	92
1.14 GetFileSize 函数	41	2.1 BitBlt 函数	92
1.14.1 范例一	41	2.1.1 范例一	92
1.14.2 范例二	43	2.1.2 范例二	95
1.15 GetFileType 函数	45	2.1.3 范例三	99
1.16 GetFullPathName 函数	47	2.1.4 范例四	101
		2.1.5 范例五	103
		2.1.6 范例六	105
		2.2 CreateBitmap 函数	108
		2.2.1 范例一	108
		2.2.2 范例二	111
		2.3 CreateBitmapIndirect 函数	115

2.4	CreateCompatibleBitmap 函数	118	3.3.2	范例二	193
2.4.1	范例一	118	3.4	CascadeWindows 函数	195
2.4.2	范例二	120	3.5	ChildWindowFromPoint 函数	199
2.5	CreateDIBSection 函数	124	3.6	ChildWindowFromPointEx 函数	201
2.5.1	范例一	124	3.7	CloseWindow 函数	204
2.5.2	范例二	127	3.8	CreateWindow 函数	205
2.5.3	范例三	129	3.8.1	范例一	205
2.6	ExtFloodFill 函数	132	3.8.2	范例二	208
2.7	FloodFill 函数	134	3.8.3	范例三	210
2.8	GetDIBColorTable 函数	136	3.8.4	范例四	212
2.8.1	范例一	136	3.8.5	范例五	216
2.8.2	范例二	138	3.9	BeginDeferWindowPos 函数, DeferWindowPos 函数, EndDeferWindowPos 函数	218
2.9	GetDIBits 函数	140	3.10	DestroyWindow 函数	220
2.10	GetPixel 函数	145	3.10.1	范例一	220
2.10.1	范例一	145	3.10.2	范例二	222
2.10.2	范例二	147	3.11	EnableWindow 函数	225
2.11	GetStretchBltMode 函数	149	3.11.1	范例一	225
2.12	LoadBitmap 函数	151	3.11.2	范例二	227
2.13	PatBlt 函数	153	3.11.3	范例三	231
2.14	SetBitmapBits 函数	157	3.12	EnumChildProc 函数	232
2.15	SetDIBColorTable 函数	159	3.12.1	范例一	232
2.16	SetDIBits 函数	162	3.12.2	范例二	235
2.17	SetDIBitsToDevice 函数	165	3.13	EnumChildWindows 函数	237
2.18	SetPixel 函数	167	3.13.1	范例一	237
2.18.1	范例一	167	3.13.2	范例二	240
2.18.2	范例二	169	3.14	EnumWindows 函数	242
2.19	SetStretchBltMode 函数	171	3.15	EnumWindowsProc 函数	244
2.20	SetDIBits 函数	172	3.16	FindWindow 函数	246
2.20.1	范例一	173	3.16.1	范例一	247
2.20.2	范例二	175	3.16.2	范例二	248
2.20.3	范例三	177	3.17	FindWindowEx 函数	251
2.20.4	范例四	179	3.17.1	范例一	251
2.20.5	范例五	181	3.17.2	范例二	253
2.21	StretchDIBits 函数	183	3.18	GetClientRect 函数	255
3.1	AdjustWindowRect 函数	187	3.18.1	范例一	255
3.2	ArrangeIconicWindows 函数	189	3.18.2	范例二	258
3.3	BringWindowToTop 函数	191	3.18.3	范例三	259
3.3.1	范例一	191			

3.18.4 范例四.....	262	3.33 IsWindow 函数.....	329
3.18.5 范例五.....	264	3.33.1 范例一.....	329
3.19 GetDesktopWindow 函数.....	267	3.33.2 范例二.....	331
3.19.1 范例一.....	267	3.34 IsWindowUnicode 函数.....	334
3.19.2 范例二.....	270	3.35 IsWindowVisible 函数.....	336
3.19.3 范例三.....	272	3.35.1 范例一.....	336
3.19.4 范例四.....	273	3.35.2 范例二.....	338
3.20 GetForegroundWindow 函数.....	276	3.36 IsZoomed 函数.....	340
3.20.1 范例一.....	276	3.37 MoveWindow 函数.....	341
3.20.2 范例二.....	278	3.37.1 范例一.....	342
3.21 GetLastActivePopup 函数.....	279	3.37.2 范例二.....	343
3.22 GetNextWindow 函数.....	282	3.37.3 范例三.....	345
3.22.1 范例一.....	282	3.37.4 范例四.....	346
3.22.2 范例二.....	284	3.38 OpenIcon 函数.....	352
3.23 GetParent 函数.....	287	3.39 SetForegroundWindow 函数.....	354
3.23.1 范例一.....	287	3.39.1 范例一.....	354
3.23.2 范例二.....	289	3.39.2 范例二.....	357
3.24 GetTopWindow 函数.....	292	3.40 SetParent 函数.....	359
3.25 GetWindow 函数.....	294	3.41 SetWindowLong 函数.....	361
3.25.1 范例一.....	294	3.41.1 范例一.....	362
3.25.2 范例二.....	296	3.41.2 范例二.....	364
3.25.3 范例三.....	298	3.41.3 范例三.....	365
3.26 GetWindowPlacement 函数.....	300	3.41.4 范例四.....	367
3.27 GetWindowRect 函数.....	302	3.42 SetWindowPlacement 函数.....	369
3.27.1 范例一.....	302	3.43 SetWindowPos 函数.....	371
3.27.2 范例二.....	305	3.44 SetWindowText 函数.....	373
3.27.3 范例三.....	308	3.44.1 范例一.....	373
3.27.4 范例四.....	310	3.44.2 范例二.....	374
3.27.5 范例五.....	313	3.45 ShowOwnedPopups 函数.....	376
3.27.6 范例六.....	314	3.46 ShowWindow 函数.....	378
3.28 GetWindowText 函数.....	316	3.46.1 范例一.....	378
3.28.1 范例一.....	316	3.46.2 范例二.....	379
3.28.2 范例二.....	318	3.47 ShowWindowAsync 函数.....	381
3.29 GetWindowTextLength 函数.....	319	3.48 TileWindows 函数.....	382
3.30 GetWindowThreadProcessId 函数.....	321	3.49 WindowFromPoint 函数.....	385
3.30.1 范例一.....	321	3.49.1 范例一.....	385
3.30.2 范例二.....	323	3.49.2 范例二.....	386
3.31 IsChild 函数.....	325	3.50 WinMain 函数.....	388
3.32 IsIconic 函数.....	327		

第 4 章 键盘输入函数	392
4.1 ActivateKeyboardLayout 函数	392
4.2 GetActiveWindow 函数	394
4.3 GetAsyncKeyState 函数	396
4.3.1 范例一	396
4.3.2 范例二	398
4.4 GetFocus 函数	400
4.5 GetKeyboardLayout 函数	402
4.6 GetKeyboardLayoutList 函数	404
4.7 GetKeyboardLayoutName 函数	405
4.8 GetKeyboardState 函数	407
4.9 GetKeyNameText 函数	409
4.10 GetKeyState 函数	411
4.10.1 范例一	411
4.10.2 范例二	413
4.11 IsWindowEnabled 函数	414
4.12 keybd_event 函数	415
4.13 LoadKeyboardLayout 函数	418
4.14 MapVirtualKey 函数	420
4.14.1 范例一	421
4.14.2 范例二	423
4.15 MapVirtualKeyEx 函数	425
4.16 RegisterHotKey 函数	427
4.17 SetKeyboardState 函数	429
4.18 UnregisterHotKey 函数	431
4.19 VkKeyScan 函数	434
4.19.1 范例一	434
4.19.2 范例二	435
第 5 章 鼠标输入函数	438
5.1 DragDetect 函数	438
5.1.1 范例一	438
5.1.2 范例二	440
5.2 GetCapture 函数	442
5.2.1 范例一	442
5.2.2 范例二	444
5.3 GetDoubleClickTime 函数	447
5.4 mouse_event 函数	449
5.5 ReleaseCapture 函数	451
5.5.1 范例一	451
5.5.2 范例二	453
5.6 SetCapture 函数	455
5.7 SetDoubleClickTime 函数	458
5.7.1 范例一	458
5.7.2 范例二	460
5.8 SwapMouseButton 函数	461
附录: Hook 实例	464
A.1 Hook_WH_KEYBOARD	464
A.2 Hook_WH_MOUSE	469

第 1 章 文件函数

1.1 AreFileApisANSI 函数, SetFileApisToANSI 函数, SetFileApisToOEM 函数

以下 3 个函数, 性质相似且相互依存出现, 本节以相同范例演示。

- **AreFileApisANSI**
函数语法格式: `BOOL AreFileApisANSI (VOID)`
功能简易说明: 判断 Win32 函数是否为 OEM/ANSI 字符集
- **SetFileApisToANSI**
函数语法格式: `VOID SetFileApisToANSI (VOID)`
功能简易说明: 设置文件 API 为 ANSI 字符集
- **SetFileApisToOEM**
函数语法格式: `VOID SetFileApisToOEM (VOID)`
功能简易说明: 设置文件 API 为 OEM 字符集

1. 范例目的

单击 Button1 按钮, 判断现在程序是否为 ANSI 字符集或 OEM 字符集, 并可转换 OEM 字符集或 ANSI 字符集。本范例参见附书光盘“配盘范例\Ch1_File Functions\ch01-01-1”文件夹。

2. 程序代码

```
Unit1.h
#01 //-----
#02 #ifndef Unit1H
#03 #define Unit1H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>
#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 #include <ComCtrls.hpp>
#10 #include <ToolWin.hpp>
#11 //-----
#12 class TForm1 : public TForm
#13 {
#14     __published:      // IDE-managed Components
#15         TToolBar *ToolBar1;
#16         TButton *Button1;
#17         TMemo *Memo1;
#18         TStaticText *StaticText1;
#19         TButton *Button2;
```

```

#20     TButton *Button3;p
#21     void __fastcall Button1Click(TObject *Sender);
#22     void __fastcall Button2Click(TObject *Sender);
#23     void __fastcall Button3Click(TObject *Sender);
#24 private:    // User declarations
#25 public:     // User declarations
#26     __fastcall TForm1(TComponent* Owner);
#27 };
#28 //-----
#29 extern PACKAGE TForm1 *Form1;
#30 //-----
#31 #endif

```

Unit1.cpp

```

#01 //-----
#02 #include <vcl.h>
#03 #pragma hdrstop
#04
#05 #include "Unit1.h"
#06 //-----
#07 #pragma package(smart_init)
#08 #pragma resource "*.dfm"
#09 TForm1 *Form1;
#10 //-----
#11 __fastcall TForm1::TForm1(TComponent* Owner)
#12     : TForm(Owner)
#13 {
#14     //默认按钮状态
#15     Button1->Enabled = true;
#16     Button2->Enabled = false;
#17     Button3->Enabled = false;
#18 }
#19 //-----
#20 void __fastcall TForm1::Button1Click(TObject *Sender)
#21 {
#22     //删除 Memo1 中所有文字
#23     Memo1->Text = "";
#24     //判断现在程序是否为 ANSI 字符集或 OEM 字符集
#25     //这项判断功能对 8 位输入输出设备极为有用
#26     if(AreFileApisANSI())
#27     {
#28         Memo1->Lines->Add("现在程序是 ANSI 字符集!");
#29         Memo1->Lines->Add("为 WIN32 函数使用。");
#30         Memo1->Lines->Add("若单击 Button2 按钮, 则转成 OEM 字符集, ");
#31         Memo1->Lines->Add("可以直接由 8 位输入输出设备使用。");
#32         Button2->Enabled = true;
#33         Button3->Enabled = false;
#34     }
#35     else
#36     {
#37         Memo1->Lines->Add("程序是 OEM 字符集!");
#38         Memo1->Lines->Add("可以直接由 8 位输入输出设备使用。");
#39         Memo1->Lines->Add("若单击 Button3 按钮, 则转成 ANSI 字符集, ");
#40         Memo1->Lines->Add("为 WIN32 函数使用。");
#41         Button2->Enabled = false;

```



```

#42  Button3->Enabled = true;
#43  }
#44  Memol->Lines->Add("以上都会影响现有函数返回的结果。");
#45  Button1->Enabled = false;
#46  }
#47  //-----
#48  void __fastcall TForm1::Button2Click(TObject *Sender)
#49  {
#50  //由 ANSI 变成 OEM
#51  SetFileApisToOEM();
#52  Memol->Text = "由 ANSI 字符集转成 OEM 字符集";
#53  //重设按钮状态
#54  Button1->Enabled = true;
#55  Button2->Enabled = false;
#56  Button3->Enabled = false;
#57  }
#58  //-----
#59  void __fastcall TForm1::Button3Click(TObject *Sender)
#60  {
#61  //由 OEM 变成 ANSI
#62  SetFileApisToANSI();
#63  Memol->Text = "由 OEM 字符集转成 ANSI 字符集";
#64  //重设按钮状态
#65  Button1->Enabled = true;
#66  Button2->Enabled = false;
#67  Button3->Enabled = false;
#68  }
#69  //-----

```

3. 运行结果

程序执行前画面如图 1.1 所示，程序执行后画面如图 1.2 所示。

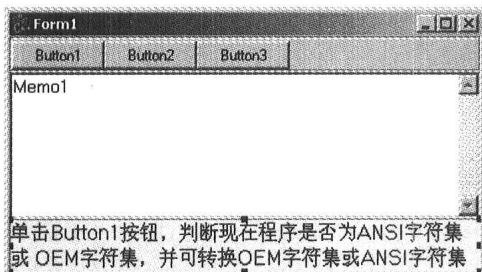


图 1.1

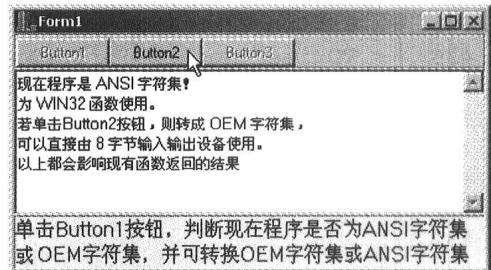


图 1.2

1.2 CopyFile 函数

函数语法格式：BOOL CopyFile(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName, BOOL bFailIfExists)

功能简易说明：复制指定文件

1. 范例目的

单击按钮，将上方选取的文件复制到下方目录中。本范例参见附书光盘“配盘范例 Ch1_File Functions\ch01-02-1”文件夹。

2. 程序代码

Unit1.h

```
#01 //-----
#02 #ifndef Unit1H
#03 #define Unit1H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>
#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 #include <ComCtrls.hpp>
#10 #include <FileCtrl.hpp>
#11 //-----
#12 class TForm1 : public TForm
#13 {
#14     __published: // IDE-managed Components
#15         TButton *Button1;
#16         TStaticText *StaticText1;
#17         TDirectoryListBox *DirectoryListBox1;
#18         TDriveComboBox *DriveComboBox1;
#19         TFileListBox *FileListBox1;
#20         TEdit *FileEdit;
#21         TFilterComboBox *FilterComboBox1;
#22         TDirectoryListBox *DirectoryListBox2;
#23         TFileListBox *FileListBox2;
#24         TEdit *Edit1;
#25         TFilterComboBox *FilterComboBox2;
#26         TDriveComboBox *DriveComboBox2;
#27         void __fastcall Button1Click(TObject *Sender);
#28
#29     private: // User declarations
#30     public: // User declarations
#31         __fastcall TForm1(TComponent* Owner);
#32 };
#33 //-----
#34 extern PACKAGE TForm1 *Form1;
#35 //-----
#36 #endif
```

Unit1.cpp

```
#01 //-----
#02 #include <vcl.h>
#03 #pragma hdrstop
#04
#05 #include "Unit1.h"
#06 //-----
#07 #pragma package(smart_init)
#08 #pragma resource "*.dfm"
#09 TForm1 *Form1;
```

```

#10 //-----
#11 __fastcall TForm1::TForm1(TComponent* Owner)
#12     : TForm(Owner)
#13 {
#14 }
#15 //-----
#16
#17 void __fastcall TForm1::Button1Click(TObject *Sender)
#18 {
#19     //汇集来源路径
#20     AnsiString target = DirectoryListBox2->Directory+"\\"+FileEdit->Text;
#21     //将指定文件复制到目的目录下
#22     if (::CopyFile(FileListBox1->FileName.c_str(), //目的路径
#23                 target.c_str(), // 源路径文件
#24                 true) == 0)
#25     {
#26         ShowMessage(" 指定目录已有相同文件, \n 所以无法复制文件!");
#27     }
#28     //将目的区域更新
#29     FileListBox2->Update();
#30 }
#31 //-----

```

3. 运行结果

程序执行前画面如图 1.3 所示, 程序执行后画面如图 1.4 所示。

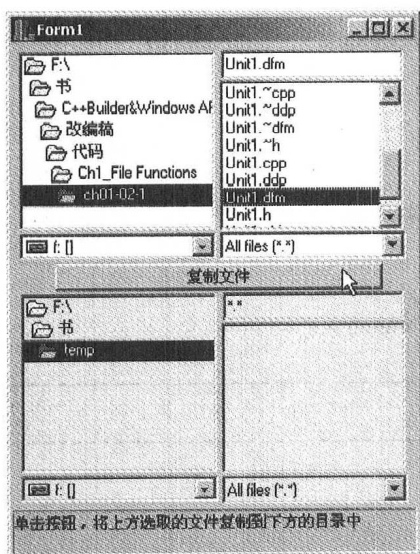


图 1.3

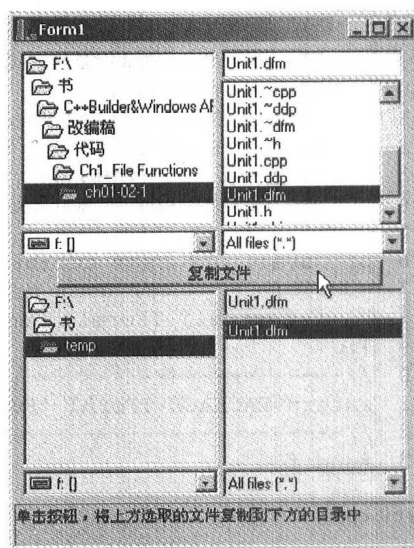


图 1.4

1.3 CreateDirectory 函数, CreateDirectoryEx 函数

函数语法格式: `BOOL CreateDirectory(LPCTSTR lpPathName, LPSECURITY_ATTRIBUTES lpSecurityAttributes)`

功能简易说明: 建立一个新目录

1. 范例目的

单击按钮，在指定目录下增加一个新目录。本范例参见附书光盘“配盘范例\Ch1_File Functions\ch01-03-1”文件夹。

2. 程序代码

Unit1.h

```
#01 //-----
#02 #ifndef Unit1H
#03 #define Unit1H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>
#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 #include <FileCtrl.hpp>
#10 #include <Grids.hpp>
#11 #include <Outline.hpp>
#12 #include <ExtCtrls.hpp>
#13 //-----
#14 class TForm1 : public TForm
#15 {
#16     __published: // IDE-managed Components
#17         TStaticText *StaticText1;
#18         TDirectoryListBox *DirectoryListBox1;
#19         TDriveComboBox *DriveComboBox1;
#20         TBevel *Bevel1;
#21         TStaticText *StaticText2;
#22         TEdit *Edit1;
#23         TButton *Button1;
#24         void __fastcall Button1Click(TObject *Sender);
#25
#26     private: // User declarations
#27     public: // User declarations
#28         __fastcall TForm1(TComponent* Owner);
#29 };
#30 //-----
#31 extern PACKAGE TForm1 *Form1;
#32 //-----
#33 #endif
```

Unit1.cpp

```
#01 //-----
#02 #include <vcl.h>
#03 #pragma hdrstop
#04
#05 #include "Unit1.h"
#06 //-----
#07 #pragma package(smart_init)
#08 #pragma resource "*.dfm"
#09 TForm1 *Form1;
#10 //-----
#11 __fastcall TForm1::TForm1(TComponent* Owner)
#12     : TForm(Owner)
#13 {
```

```

#14 }
#15 //-----
#16 void __fastcall TForm1::Button1Click(TObject *Sender)
#17 {
#18 //若已经输入新路径名称
#19 if(Edit1->Text != "")
#20 {
#21     AnsiString dir = DirectoryListBox1->Directory;
#22     //分辨是否在根目录下
#23     if(dir.Length()>4)
#24         dir = dir + "\\\" + Edit1->Text;
#25     else
#26         dir = dir + Edit1->Text;
#27     //在程序中显示新目录的完整路径
#28     StaticText1->Caption = "新建目录路径: " + dir;
#29     //若新建目录失败
#30     if(!::CreateDirectory(dir.c_str(), NULL))
#31         ShowMessage("新建目录失败!");
#32     //若新建目录成功,则更新显示
#33     DirectoryListBox1->Update();
#34 }
#35 else
#36     ShowMessage("空白无效! 请输入有效目录名称。");
#37 }
#38 //-----

```

3. 运行结果

程序执行前画面如图 1.5 所示, 程序执行后画面如图 1.6 所示。



图 1.5



图 1.6

1.4 CreateFile 函数

函数语法格式: HANDLE CreateFile(LPCTSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode, LPSECURITY_ATTRIBUTES lpSecurityAttributes, DWORD dwCreationDistribution, DWORD dwFlagsAndAttributes, HANDLE hTemplateFile)

功能简易说明: 打开、建立一个新文件或新对象

1.4.1 范例一

1. 范例目的

单击按钮, 在指定目录下增加一个新文本文件(aaa01.txt)。本范例参见附书光盘“配盘范例\Ch1_File Functions\ch01-04-1”文件夹。

2. 程序代码

Unit1.h

```
#01 //-----
#02 #ifndef Unit1H
#03 #define Unit1H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>
#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 //-----
#10 class TForm1 : public TForm
#11 {
#12     __published: // IDE-managed Components
#13         TStaticText *StaticText1;
#14         TButton *Button1;
#15         void __fastcall Button1Click(TObject *Sender);
#16 private: // User declarations
#17 public: // User declarations
#18     __fastcall TForm1(TComponent* Owner);
#19 };
#20 //-----
#21 extern PACKAGE TForm1 *Form1;
#22 //-----
#23 #endif
```

Unit1.cpp

```
#01 //-----
#02 #include <vcl.h>
#03 #pragma hdrstop
#04
#05 #include "Unit1.h"
#06 //-----
#07 #pragma package(smart_init)
#08 #pragma resource "*.dfm"
#09 TForm1 *Form1;
#10 //-----
#11 __fastcall TForm1::TForm1(TComponent* Owner)
#12     : TForm(Owner)
#13 {
#14 }
#15 //-----
#16
#17 void __fastcall TForm1::Button1Click(TObject *Sender)
#18 {
#19
#20     AnsiString FileName="C:\\aaa01.txt";
#21     char TheText[MAX_PATH]="这是 CreateFile 函数测试文字内容。";
#22     HANDLE TheFile;
#23     DWORD BytesWritten;
#24
#25     TheFile=CreateFile(FileName.c_str(),
#26                       GENERIC_WRITE,FILE_SHARE_READ,
#27                       NULL,
#28                       CREATE_ALWAYS,
```

```

#29             FILE_ATTRIBUTE_NORMAL,
#30             NULL);
#31
#32 if(TheFile==INVALID_HANDLE_VALUE)
#33 {
#34     ShowMessage("无法建立新文件!");
#35     return;
#36 }
#37
#38 if(!WriteFile(TheFile,TheText,sizeof(TheText),&BytesWritten,NULL) ||
#39     sizeof(TheText)!=BytesWritten)
#40 {
#41     ShowMessage("文件无法写入新数据!");
#42 }
#43
#44 // 关闭一个打开的对象 handle
#45 ::CloseHandle(TheFile);
#46 }
#47 //-----
#48

```

3. 运行结果

程序执行前画面如图 1.7 所示，程序执行后画面如图 1.8 和图 1.9 所示。

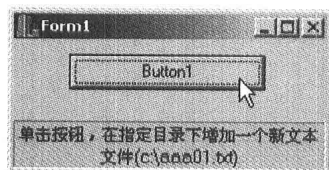


图 1.7

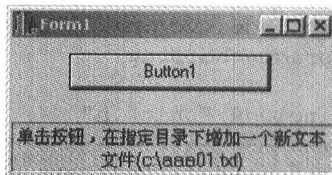


图 1.8

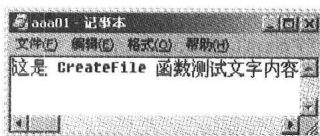


图 1.9

1.4.2 范例二

1. 范例目的

单击按钮，取得所有 COM 端口的设备状态。本范例参见附书光盘“配盘范例\Ch1_File Functions\ch01-04-2”文件夹。

2. 程序代码

Unit2.h

```

#01 //-----
#02 #ifndef Unit2H
#03 #define Unit2H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>

```

```

#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 #include <ComCtrls.hpp>
#10 #include <ToolWin.hpp>
#11 //-----
#12 class TForm1 : public TForm
#13 {
#14     __published: // IDE-managed Components
#15         TToolBar *ToolBar1;
#16         TButton *Button1;
#17         TStaticText *StaticText1;
#18         TListBox *ListBox1;
#19         void __fastcall Button1Click(TObject *Sender);
#20 private: // User declarations
#21 public: // User declarations
#22     __fastcall TForm1(TComponent* Owner);
#23 };
#24 //-----
#25 extern PACKAGE TForm1 *Form1;
#26 //-----
#27 #endif

```

Unit2.cpp

```

#01 //-----
#02 #include <vcl.h>
#03 #pragma hdrstop
#04
#05 #include "Unit2.h"
#06 //-----
#07 #pragma package(smart_init)
#08 #pragma resource "*.dfm"
#09 TForm1 *Form1;
#10 //-----
#11 __fastcall TForm1::TForm1(TComponent* Owner)
#12     : TForm(Owner)
#13 {
#14 }
#15 //-----
#16
#17 void __fastcall TForm1::Button1Click(TObject *Sender)
#18 {
#19     //删除所有项目
#20     ListBox1->Clear();
#21     //默认检测 COM 端口的返回值
#22     HANDLE h = INVALID_HANDLE_VALUE;
#23     //逐一进行 COM 端口检测
#24     for(int i=1;i<=32;i++)
#25     {
#26         AnsiString comname = "\\.\COM" + String(i);
#27         //打开 COM 端口, 并返回检测值
#28         h = ::CreateFile( comname.c_str(),
#29             GENERIC_READ | GENERIC_WRITE,
#30             0,
#31             0,
#32             OPEN_EXISTING,
#33             FILE_ATTRIBUTE_NORMAL,
#34             0);
#35         //若无法打开指定 COM 端口

```



```

#36     if (h == INVALID_HANDLE_VALUE)
#37         {
#38         //取得错误信息
#39         DWORD err = ::GetLastError();
#40         //若 err 为 2 , 即此 COM 端口尚未安装
#41         if (err!=2)
#42             {
#43             ListBox1->Items->Add("COM" + AnsiString(i) +
#44             " 尚未使用, 错误信息为 (" +
#45             AnsiString((int)err) + ")");
#46             }
#47         }
#48     else
#49         {
#50         ListBox1->Items->Add("COM" + AnsiString(i) + " 正在使用中!");
#51         }
#52     //释放 handle
#53     ::CloseHandle(h);
#54     }
#55
#56 }
#57 //-----

```

3. 运行结果

程序执行前画面如图 1.10 所示, 程序执行后画面如图 1.11 所示。

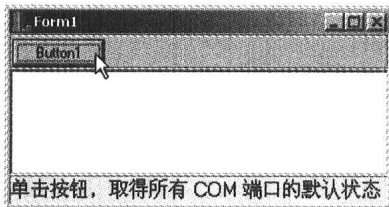


图 1.10

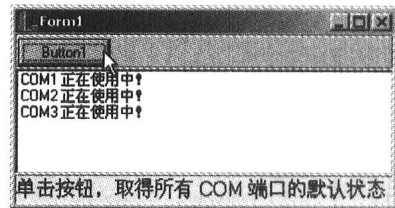


图 1.11

1.4.3 范例三

1. 范例目的

单击按钮, 由上次打开“设置”文件时间重新设置系统时间。本范例参见附书光盘“配置范例\Ch1_File Functions\ch01-04-3”文件夹。

2. 程序代码

Unit3.h

```

#01 //-----
#02 #ifndef Unit3H
#03 #define Unit3H
#04 //-----
#05 #include <Classes.hpp>
#06 #include <Controls.hpp>
#07 #include <StdCtrls.hpp>
#08 #include <Forms.hpp>
#09 #include <ComCtrls.hpp>
#10 #include <ToolWin.hpp>

```