Visual C++ 编程经验四则

湖南师范大学计算机部 徐大宏

摘要：本文总结讨论了笔者使用 Visual C++ 开发软件中的几则经验：对话框中 Esc 键的屏蔽；应用程序热键的实现；将文件删除到回收站；使用可变参数表的函数。

关键词：对话框 钩子函数 热键 可变参数表

对话框中 Esc 键的屏蔽方法

无论是使用VC6或是VC5编写基于对话框的应用程序；或是在程序中使用到对话框，都不可避免地遇到同一个问题：当这些程序处于激活状态时，敲 Esc 键都可以直接关闭该程序，且不向应用程序发送 WM_CLOSE 消息。可以说这一缺陷给我们的编程，以及用户对软件的操作都带来一定的影响。笔者在实际开发过程中，利用“钩子函数”巧妙地回避了这一些缺陷，在程序中屏蔽了对话框对 Esc 键的响应。具体作法如下：

1. 在相应代码段处（可为对话框类的 CPP 文件中）定义一个钩子句柄全局变量，及声明一个全局钩子函数。

```
HHOOK  hHook=NULL;
extern "C" LRESULT WINAPI HookProc(int nCode,WPARAM wParam,LPARAM lParam);
```

2. 重载对话框类的 OnCreate 函数，并在其中加入如下代码：

```
hHook = SetWindowsHookEx(WH_KEYBOARD, HookProc,NULL,0);
```

代码含义为：安装一个监视键盘击键消息的钩子函数至系统钩子函数链中。

SetWindowsHookEx函数的第二个参数为用户自定义的钩子函数。

3. 编写如下格式的钩子函数

```
extern "C" LRESULT WINAPI HookProc(int nCode,WPARAM wParam,LPARAM lParam)
{  if ( wParam == VK_ESCAPE )
return TRUE;
else
return CallNextHookEx(hHook,nCode,wParam, lParam);     }
```

其中 VK_ESCAPE 为 Esc 键的虚拟键代码。

代码含义为：当系统检测到 Esc 键的击键消息时，并不将此消息传递至钩子函数链中的下一个钩子函数做进一步处理，而是直接中断此消息的传递，也即截断了它的后续处理，从而达到屏蔽效果。这一细节是整个方法的关键。在正常情况下，自定义钩子函数 HookProc 通过调用 CallNextHookEx函数，能够将所检测到的消息传递到系统钩子函数链中的下一个钩子函数，以待进一步处理。

4. 最后，重载对话框类的 OnDestroy 函数，并加入如下代码：

```
bool bUnHook;
if ( hHook )
{  bUnHook = (bool)UnhookWindowsHookEx (hHook);
```

```
if ( bUnHook )
  hHook = NULL;      }
```

卸载应用程序中所安装的钩子函数。

应用程序热键的实现

在一个应用程序内部的菜单可以设置热键，如在菜单中一般用Alt+F进入“文件”之类的子菜单。另外在桌面上设置的快捷方式里的快捷键，让我们无论任何时候按下所设置的快捷键后，就会启动相应的应用程序。在多个正在运行的应用程序中，如何利用一个按键动作就迅速地回到所需要的应用程序呢？这就需要利用热键(HOTKEY)的技术来实现。

1. 热键的设置

windowsAPI中有一个函数RegisterHotKey用于设置热键，它的调用方式如下:

```
BOOL RegisterHotKey(
  HWNDhWnd,    // 响应该热键的窗口句柄
  Intid, // 该热键的唯一标示符
  UINTfsModifiers, // 该热键的辅助按键
  UINTvk ); // 该热键的键值
```

其中热键的唯一标示符，在Windows应用程序中规定取值范围为0x0000到0xBFFF之间，动态链接库中取值范围为0xC000到0xFFFF之间。为了保证其唯一性可以使用GlobalAddAtom函数来设置热键的唯一标示符。需要注意的是GlobalAddAtom返回的值是在0xC000到0xFFFF范围之间，为适应RegisterHotKey的调用要求，在应用程序中设置热键可用GlobalAddAtom的返回值减去0xC000。

热键的辅助按键包括MOD_CTRL、MOD_ALT、MOD_SHIFT，对于Windows兼容键盘还支持Windows键，即其键面上有Windows标志的那个键，其值为MOD_WIN。

2. 实现

(1) 在应用程序中手工操作(MFC ClassWizard没有列出WM_HOTKEY消息)，加入响应WM_HOTKEY消息的各项代码。

在头文件中加入 afx_msg LRESULT OnHotKey (WPARAM wParam,LPARAM lParam);

在实现文件的 BEGIN_MESSAGE_MAP(CTestartDlg, CDialog)与 END_MESSAGE_MAP()之间加入

```
ON_MESSAGE(WM_HOTKEY, OnHotKey)
```

在实现文件中加入函数体

```
LRESULT MyClassName::OnHotKey(WPARAM wParam,LPARAM lParam)
{ if(!((HIWORD(lParam) == 0x41 ) && ( LOWORD (lParam)==MOD_CONTROL | MOD_ALT)))
    return true;
  if ( IsIconic() )
    ShowWindow(SW_SHOWNORMAL);
  SetForegroundWindow();
  return true;  }
```

OnHotKey函数在收到热键消息后，将处于非激活状态的应用程序转为激活状态。

(2) 在应用程序的OnInitDialog或OnCreate等函数处加入如下代码段，以期实现向系统注册热键信息。

```
int hotkeyid;
// 减去 0x0C000 是为了保证取值范围的限制
hotkeyid = GlobalAddAtom("UserDefineHotKey") - 0x0C000;
// 热键为 Ctrl+Alt+A
RegisterHotKey(m_hWnd, hotkeyid, MOD_CONTROL | MOD_ALT,0x41);
```

将文件删除到回收站的实现

在应用程序中如要执行各种文件操作，如拷贝、移动、删除或文件更名等，都可以利用 Win32外壳中的SHFileOperation()函数。此函数仅用一个参数: SHFILEOPSTRUCT结构，通过这个结构中的几个有效域的设置，就可以定制文件操作的多种方式。如在应用程序中要实现将文件删除到回收站中的功能，则可使用以下代码段。

```
LPSHFILEOPSTRUCT lpFileOp;
lpFileOp->hwnd = NULL;
lpFileOp->wFunc = FO_DELETE;
 lpFileOp->pFrom = "c:\\commands\\mscdex.exe\0";
lpFileOp->fFlags = FOF_ALLOWUNDO;
SHFileOperation(lpFileOp);
```

其中wFunc表明操作的类别: 如删除 FO_DELETE; 拷贝 FO_COPY 等。fFlags 控制文件操作，如FOF_ALLOWUNDO 表明应用程序可记录撤消信息等。

要注意的是:

1. pFrom域所给定的文件必须是绝对路径名，否则wFunc域所设定的FOF_ALLOWUNDO值将不起作用，即应用程序不会记录操作中可能的撤消信息。

2. 给定的每个文件名都必须有一个NULL终结符。

3. 给定件多个文件时，每个文件名必须被#)字符分隔，而整个字符串必须用两个 #0 结束。

使用可变参数表函数的实现

使用可变的参数表，能使函数处理较多的数据以及不同的数据类型，增强了程序的实用性；同时，又可将多个相关的函数合写成为一个函数，增强了程序的简洁性。

要访问可变参数表中的参数，需解决两个关键问题：如何确定可变参数表的长度；如何确定可变参数表中所包含数据的类型，通常有以下一些方法:

1. 使用格式串表参数表示参数的长度和数据类型

如wsprintf(outstr, "j = %d %s", j, "This is test");此类函数中使用一套符号来表示可变参数表中的参数类型和个数。

2. 在可变参数表中使用结束标志值，表明参数长度

在参数表的末尾加入一个特殊值，以标志此表中参数的结束，这种方法的缺点是数据类型操作不很灵活，要么所有参数固定同一类型，要么所有参数的类型依约定顺序固定。

3. 在可变参数表使用一变量，用以表明参数长度

在参数表中固定一变量，用以表明参数的个，但存在与方法2类似的缺点。

4. 可利用Visual C++中提供的宏及运行库标准数据类型，实现可变参数表

在Windows中有宏：va_start va_arg va_end，及数据类型：va_list。这一方法实现比较简单，且操作灵活。

笔者以方法4为例，编程演示说明其实现过程。

数据类型va_list：用来说明一个指向可变参数表中数据项的指针。

void va_start(va_list ap, lastfix)用来设置一个表指针，它指向表中的第一个变量参数。参数ap表明va_start用哪一个指针来访问最后一个固定的参数。va_star()宏使ap只指向lastfix规定的参数以外的第一个参数。这个宏必须在va_arg() 或va_end()使用之间调用。

type va_arg(va_list ap, tpye)有两个目的：第一它返回参数指针ap指向的对象的值；第二它修改参数指针，把它指向表里的下一个数据项。va_arg()的参数中规定的type(类型)表明 va_arg()正在读的是什么类型的数据，如va_arg()所指向的是int或者double等。同时参数type也给va_arg()提供了正确修改参数指针的有关信息。

void va_end(va_list ap)完成一些清理、组织工作，这是被调用的函数能正确返回的必需步骤。否则可能发生错误。

简单地示例函数如下:

```
int average( int varlist, ... )
{ int count, sum, i;
  va_list marker;
  // 初始化操作，用 marker 指向参数表
  va_start( marker, varlist );
  sum = varlist;
  count = 1;
  //依次取出参数表中的每一个参数，直到遇上结束标志
  while( (i = va_arg( marker, int))!= -1 ) {
  sum += i;
  count++;          }
// 撤消指针
va_end( marker );
return( sum ? (sum / count) : 0 );                }
```

当然，在这些技术基础上可编写功能强大的使用可变参数表的函数。

结 语

本文所描述的方法与技巧，笔者均在Visual C++6.0环境，Windows 9x / NT系统中调试运行。且都已用于实际应用中，并取得很好的效果。■

参考文献

1 Kruglinski, D.J著，潘爱民，王国印译《Visual C++技术内幕》清华大学出版社，1999

2 Bennett.D.著，徐军等译《Visual C++5开发人员指南》机械工业出版社，1998

3《计算机世界》日报网络版 www.computerworld.com.cn