注册表安全编程实验

1、注册表基础

1.1 注册表历史背景

什么是注册表？我们可以把注册表理解成一种数据库，里面保存着各种数据，如：系统的配置信息，桌面环境，系统软件，IE 浏览器等等软件的信息。在微软以前发布的操作系统中(如 windows 3.X)，把这些信息保存在 System.ini，Win.ini 等文件中，随着数据量越来越大，越来越复杂，这样 ini 文件难以胜任，从 windows 95 开始引入注册表来保存这些复杂的，海量的数据，总之我们权且把它当着一个数据库吧。

1.2 注册表作用

注册表的用处一下也说不全，我作为一个程序员，站在程序员的角度来讲讲吧。对我来讲，注册表最直接的用处，就是用来保存一些配置信息，如开发的桌面软件，一定会有些配置数据需要保存，那么我们可以像 Windows 早期版本那个，保存在 ini 文件中，不过 ini 文件容易被人修改。如果保存在注册表中，稍稍保密些，一般使用者也不懂得怎么改。注册表另外一个常用功能是，修改其他软件的配置信息，如浏览器的个性设置，电脑桌面的主题设置，光标配置等，不过这些对开发人员来讲，好像意义不是很大，修改别人的软件配置干什么，是吧？如果你是黑客那就另当别论了。

注册表还保存着硬件的一些驱动相关信息，如 COM 口的数据，串口通信编程时，你是不是会为判断 PC 机是否有串口驱动而烦恼呀，不知道 PC 机上的串口逻辑号是多少很烦心呢？其实这些数据在注册表里都有，到注册表里一查就清

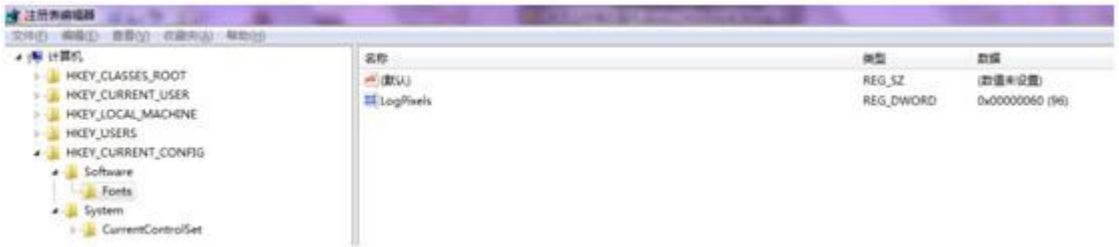
楚了，关于如何判断 PC 机串口是否正常，PC 现在有串口逻辑号是多少的相关技术，在本人博文《串口通信编程--多线程异步方式》里有详细的介绍，还有代码参考。

用注册表保存配置信息，那么我们得先大体认识一下注册表的组成与架构，然后再介绍一下如何创建，查询，修改注册表。

1.3 **注册表结构**

打开注册表：可能通过控制面板打开，也有很多其他途径能打开，本人常用方法是在【开始】菜单中选择【运行】命令，然后在框中输入命令 regedit，如图：

然后点【确定】按钮，将会打开注册表，如图：



从图中可看出，注册表是层叠式构架的，左边的树形列表，第一级有 5 个根键，点开第一级接着有第二级，第三级等等，这叫项，也就是说，根键下面的是项，而项下面如果还有子级那叫子项，直到最后一级如图中路径”HKEY_CURRENT_CONFIG\SOFTWARE\FONTS”，其中 Fonts 是 Software 的

子项，Fonts 是最后一级，它没有子项了，再 Fonts 下面就是值项了，在右边一栏，值项由名称，类型，数据组成，值项就是最终保存具体数据的地方。

所以说，注册表由：键----项----子项----值项构成，而值项由：名称，类型，数据组成，其中名称就不用说了，数据也不用说了，说说类型吧，类型是指数据的类型，数据的类型有很多种，从其他资料中切个图吧，更直观：

值项类型	数据类型	功　能
REG_BINARY	二进制值	二进制数据
REG_DWORD	四字节	四字节数值数据
REG_DWORD_BIG_ENDIAN	四字节	逆序存放的四字节数值数据
REG_DWORD_LITTLE_ENDIAN	四字节	四字节数值数据
REG_QWORD	八字节	八字节数值数据
REG_QWORD_LITTLE_ENDIAN	八字节	与 REG_QWORD 相同
REG_LINK	字符串	文件路径
REG_NONE	未知	用于无需分类的数据
REG_UNKNOWN	未知	用于无法确定类型的数据
REG_SZ	字符串	文本字符串
REG_EXPAND_SZ	字符串	带变量的文本字符串
REG_MULTI_SZ	多字符串	以 null 分隔的字符串集合
REG_FILE_NAME	字符串	文件名
REG_FILE_TIME	字符串	文件时间
REG_FULL_RESOURCE_DESCRIPTOR	字符串	硬件资源列表
REG_RESOURCE_LIST	字符串	设备使用的资源列表
REG_RESOURCE_REQUIREMENTS_LIST	字符串	驱动程序要求的资源列表

看到没，乱糟糟的一堆，总之对一般程序员来说，我们别管那么多，把它们分成两在类：数值型与字符型，数值型分整型与二进制型，其中整形有 4 字节 8 字节等，还分高位在前还是低位在前呢，反正自己看了，一看就懂，我也就不多说了。字符串也更不多说了，自己看一下就懂了。

总结一下，注册表与根键、项(子项)、值项组成，根键与项是用来区别层次关系的，不具体存储数据，项下面有子项的同时还可以有值项。值项是具体存

储数据的，形式是：名称、类型、值。先把这种关系捋顺了，下面看起来就容易了。

1.4 各根键主要存储的信息

Ø HKEY_CLASSES_ROOT

定义了系统中所有文件类型标志和基本操作标志

Ø HKEY_CURRENT_USER

当前用户的配置信息，包括环境变量、桌面设置、网络连接，软件运行信息等

Ø HKEY_LOCAL_MACHINE

本机相关的系统信息，包括硬件信息，驱动信息，内存数据，总线数据等等。

Ø HKEY_USER

所有用户的信息。

Ø KKEY_CURRENT_CONFIG

本地计算机启动时配置的相关信息，如环境信息，桌面主题，背景色之类。

1.5 手动操作注册表

打开注册表编辑器：【开始】菜单选择运行命令，输入 regedit，【确定】，打开注册表编辑器，你可以在上面创建，修改，删除，还有查找等等操作。都是可视化操作，自己动手操作一下就懂，不赘述。

2、注册表编程

本章介绍注册表的创建、修改、查询等知识。通过低层的 API 实现。

2.1 相关函数分类

关于注册表的函数可大致分为：项管理类，值项管理类，杖举类，实用类，安全类。项管理与值项管理不言而喻是对项与值项的增删改查操作的。杖举是对

项与值项数据进行分析查找。实用类是远程操作的，操作远程计算机的注册表。安全类主要是对注册表的安全信息操作。常用的是项管理，值项管理与杖举这三类。后面我们也介绍这三类。其他的不深究，如果想了解的，可以联系我，我有相关电子档书籍。

2.2 项管理函数

很多书籍中分项，还有子项，弄得一头雾水，其实子项是相对而言的，有父项才有子项之说，所以说子项也是项，子项也可能是某项的父项，这个概念要搞清楚。

项管理函数是对项(子项)的创建、打开、关闭、删除的几个操作，分 4 个函数，下面一一介绍。

1）项创建函数

RegCreateKeyEx 与 RegCreateKey，后者请不要再用，win32 以后版本都用前者。此函数的功能是在指定项下建一子项。

```
LONG WINAPI RegCreateKeyEx(
 __in       HKEY hKey,
 __in       LPCTSTR lpSubKey,
 __reserved DWORD Reserved,
 __in_opt   LPTSTR lpClass,
 __in       DWORD dwOptions,
 __in       REGSAM samDesired,
 __in_opt   LPSECURITY_ATTRIBUTES lpSecurityAttributes,
 __out      PHKEY phkResult,
 __out_opt  LPDWORD lpdwDisposition
);
```

hKey 父项：打开的句柄，表明在此项下建立子项，也可以是五个(大多数 windows 版本是五个，也有六个的)根键之一，就是以下常量：

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
```

HKEY_USERS
HKEY_CURRENT_CONFIG

lpSubKey 子项名称：你要创建的子项，接在父项下，如：”REGTEST\\GPS 设备网络通道\\”.

Reserved:必须为 0

lpClass：建议为 NULL

dwOptions：标志创建的注册表保存在文件中还是内存中

一般取值 REG_OPTION_NON_VOLATILE 更多信息请查 MSDN

samDesired：访问权限，有读、写、查询等等，取 KEY_ALL_ACCESS 全部权限即可

lpSecurityAttributes：安全属性，不知道怎么设置，取 NULL 即可

phkResult：创建成功后，返回的新项的句柄

lpdwDisposition：返回状态描述，有两种状态，项原来存在 | 项是新建的(REG_CREATED_NEW_KEY | REG_OPENED_EXISTING_KEY)

返回值：成功返回 ERROR_SUCCESS

以下是伪代码：

```
HKEY hKey;
DWORD dw;
if(ERROR_SUCCESS == RegCreateKeyEx(HKEY_CURRENT_CONFIG,
                    " REGTEST\\GPS 设备网络通道\\", 0,
                    NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
                    NULL, &hKey, &dw))
    AfxMessageBox (_T(“创建成功”));
```

效果切图：

2）项打开函数

RegOpenKeyEx 打开一个注册表项,语法定义如下：

```
LONG WINAPI RegOpenKeyEx(
 __in       HKEY hKey,  //可以是根键，也可以是打开的项
 __in_opt   LPCTSTR lpSubKey,//需要打开的子项名称
 __reserved DWORD ulOptions,//同上
 __in       REGSAM samDesired,//访问权限,同上
 __out      PHKEY phkResult//打开成功后,返回项的句柄
);
```

以下库伪代码:

```
HKEY hKey;
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey))
        AfxMessageBox(_T("打开成功"));
```

3）项关闭函数

RegCloseKey 这个就太简单了,只有一个参数，就是关闭打开的项

```
LONG WINAPI RegCloseKey(
 __in HKEY hKey//传入一个打开的项的句柄
);
```

伪代码:

```
HKEY hKey;
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)){
        if (ERROR_SUCCESS == RegCloseKey(hKey))
                AfxMessageBox(_T("关闭成功"));
```

}

4）项删除函数

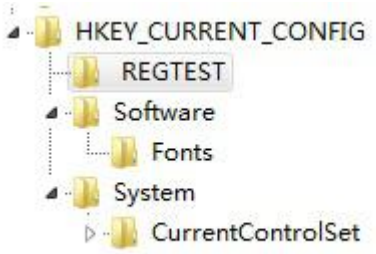
RegDeleteKey 这个函数也简单,形参只须传入根健与子项路径即可:

```
LONG WINAPI RegDeleteKey(
 __in HKEY hKey,
 __in LPCTSTR lpSubKey
);
```

伪代码:

```
HKEY hKey;
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)){
        if (ERROR_SUCCESS == RegCloseKey(hKey))
                AfxMessageBox(_T("关闭成功"));
}
```

效果切图(已删除):



2.3 值项管理函数

值项管理函数是对注册表中的值项进行设置(增加)、查询、删除操作，一共介绍三个函数，分别是设置值、查询值、删除值。

1）值设置函数

以下为函数原型:

```
LONG WINAPI RegSetValueEx(
 __in    HKEY hKey, //已打开注册表项的句柄
 __in_opt  LPCTSTR lpValueName,//值项名称
```

__reserved DWORD Reserved,//保留，为 0

__in DWORD dwType, //值项类型，前面介绍过，有整形，字符串等

__in const BYTE *lpData,//存放值项的值的指针，全部转换成 PBYTE

__in DWORD cbData //值项大小(字节)

);

以下是伪代码：

```
HKEY hKey;
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)){
        DWORD nType = REG_SZ;
        char szName[] = {"我叫小王"};
        RegSetValueEx(hKey, "Name", 0, nType, (const PBYTE)szName,
strlen(szName));
        nType = REG_DWORD;
        DWORD nYears = 60;
        RegSetValueEx(hKey, "Years", 0, nType, (const PBYTE)&nYears, 4);
}
```

效果切图：

从图中可看出，增加了两个值项，一个是字符串型，一个是整形。

2）值查询函数

查询打开的项的值项，函数是 RegQueryValueEx，以下为函数原型：

```
LONG WINAPI RegQueryValueEx(
 __in        HKEY hKey,//打开项的句柄
```

__in_opt LPCTSTR lpValueName,//值项名

__reserved LPDWORD lpReserved,//保留，为 0

__out_opt LPDWORD lpType, //返回值项的值类型，如果不关心值项类型，可谓 NULL

__out_opt LPBYTE lpData, //装值的 BUFF

__inout_opt LPDWORD lpcbData //BUFF 的长度，同时返回取得数据的长度

);

以下为伪代码：

```
HKEY hKey;
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)){
        DWORD nType = 0, nLen = 100;
        char szName[100];
        RegQueryValueEx(hKey, "Name", 0, &nType, (const PBYTE)szName,
&nLen);
        nType = 0;
        DWORD nYears = 0; nLen = 4;
        RegQueryValueEx(hKey, "Years", 0, &nType, (const PBYTE)&nYears,
&nLen);
        int i = 0;
}
```

3）值项删除函数

删除打开的项的值项，首先得打开值项所属的项，然后删除值，函数原型如下：

```
LONG WINAPI RegDeleteValue(
 __in     HKEY hKey, //打开的项的句柄
 __in_opt LPCTSTR lpValueName//值项名字
);
```

以下为伪代码：

```
HKEY hKey;
```

```
if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)){
        RegDeleteValue(hKey, "Name");
        RegDeleteValue(hKey, "Years");
}
```

效果切图：

2.4 **注册表遍历**

对注册表进行查找，项、子项、值项进行分析、提取数据等操作。一般有五个函数，下面一一介绍。

1）**枚举子项**

该函数功能是枚举注册表中指定项的子项，提取子项信息，如子项名，修改时间，子项类型等，一次取一个，函数原型如下：

```
LONG WINAPI RegEnumKeyEx(
 __in HKEY hKey, //已打开的项句柄，必须有 KEY_ENUMERATE_SUB_KEYS 权限，
                 //同时也可以是几大根键值：
                 //HKEY_CLASSES_ROOT
                 //HKEY_CURRENT_CONFIG
                 //HKEY_CURRENT_USER
                 //HKEY_LOCAL_MACHINE
                 //HKEY_USERS
```

__in DWORD dwIndex, //子项序号，第一次调用可为 0，如果序号溢出将返回 259 号错误

__out LPTSTR lpName, //返回子项名称的 buff

__inout LPDWORD lpcName,//lpName 的容量，函数返回后，返回 lpname 内容字节

__reserved LPDWORD lpReserved,//保留，为 NULL

__inout LPTSTR lpClass,//返回子项类型，可以为 NULL

__inout_opt LPDWORD lpcClass,//lpClass 容量，可以为 NULL，同时返回装载的字节数

__out_opt PFILETIME lpftLastWriteTime//返回子项最后修改时间，可以为 NULL
);

以下为伪代码：

```
char szKeyName[100]; memset(szKeyName, 0, 100);
char szClass[100]; memset(szClass, 0, 100);
HKEY hKey; long nErr = 0;
DWORD nNameLen = 100, nClassLen = 100;
FILETIME timeFile;
if (ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG, "
REGTEST",
                                  0, KEY_ALL_ACCESS/*KEY_ENUMERATE_SUB_KEYS*/,
                                  &hKey)) {
                if (ERROR_SUCCESS == (nErr =
RegEnumKeyEx(HKEY_CURRENT_CONFIG/*hKey*/,
                                   0, szKeyName, &nNameLen, NULL, szClass, &nClassLen,
                              &timeFile)))
                                  AfxMessageBox(szKeyName);
}
```

2）枚举值项

此函数是提取指定值项(通过序号指定)的信息，如：值，值项类型，值项名称等。

LONG WINAPI RegEnumValue(

__in HKEY hKey, //已打开的项句柄，必须有 KEY_QUERY_VALUE 权限，

//同时也可以是几大根键值：

//HKEY_CLASSES_ROOT
//HKEY_CURRENT_CONFIG
//HKEY_CURRENT_USER
//HKEY_LOCAL_MACHINE
//HKEY_USERS

__in DWORD dwIndex, //值项序号，第一次调用可为 0，如果序号溢出将返回 259 号错误

__out LPTSTR lpValueName,//装值项名的 buff

__inout LPDWORD lpcchValueName,//前一个参数的容量，同时返回其长度

__reserved LPDWORD lpReserved,//保留，为 NULL

__out_opt LPDWORD lpType,//返回值项类型，有 REG_SZ,REG_DWORD 等类型

__out_opt LPBYTE lpData,//装值项值的 buff

__inout_opt LPDWORD lpcbData//lpData 的容量，同时返回 lpData 接收的字节数

);

以下是伪代码：

```
char szValueName[100]; memset(szValueName, 0, 100);
BYTE pValue[100]; memset(pValue, 0, 100);
HKEY hKey; long nErr = 0;
DWORD nNameLen = 100, nValueLen = 100, nType = 0;;
FILETIME timeFile;
```

```
if (ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_CONFIG,
        " REGTEST\\GPS 设备网络通道\\", 0, KEY_ALL_ACCESS, &hKey)) {
    if (ERROR_SUCCESS == (nErr = RegEnumValue(hKey, 1, szValueName,
&nNameLen,
                        NULL, &nType, pValue, &nValueLen))){
        CString szResult;
        if (REG_SZ == nType)
            szResult.Format(_T("%s 的值是：%s"), szValueName,
(char*)pValue);
        else if (REG_DWORD == nType)
            szResult.Format(_T("%s 的值是：%d"), szValueName,
*((DWORD*)pValue));
        AfxMessageBox(szResult);
    }
}
```

3）项批量查询(提取)

查询指定项下面的子项数据与其下值项数据，如：子项的数量，值项的数量，子项中名称最长的长度，值项中名称最长的长度。其函数原型如下：

```
LONG WINAPI RegQueryInfoKey(
 __in        HKEY hKey, //已打开的项句柄，必须有 KEY_QUERY_VALUE 权限，
                        //同时也可以是几大根键值：
                        //HKEY_CLASSES_ROOT
                        //HKEY_CURRENT_CONFIG
                        //HKEY_CURRENT_USER
                        //HKEY_LOCAL_MACHINE
                        //HKEY_USERS
 __out_opt   LPTSTR lpClass, //返回项的类型
 __inout_opt LPDWORD lpcClass,//lpClass 容量，同时返回其长度
 __reserved  LPDWORD lpReserved,//保留，必须为 NULL
 __out_opt   LPDWORD lpcSubKeys,//返回其下子项的数量
 __out_opt   LPDWORD lpcMaxSubKeyLen,//返回子项中名称最长的项的长度
```

__out_opt LPDWORD lpcMaxClassLen,//返回子项中类型名最长的长度

__out_opt LPDWORD lpcValues,//返回指定项的值项的数量

__out_opt LPDWORD lpcMaxValueNameLen,//返回指定项的值项中名称最长的长度

__out_opt LPDWORD lpcMaxValueLen,//返回指定项的值项中值的最长长度

__out_opt LPDWORD lpcbSecurityDescriptor,//安全描述，可以为 NULL

__out_opt PFILETIME lpftLastWriteTime//指定项，或其下的所有项中最晚修改时间

);

以下为伪代码：

```
char szClassName[100]; memset(szClassName, 0, 100);
DWORD nClassLen = 0, nSubKeys = 0, nMaxSubKey = 0, nMaxClass = 0;
DWORD nValues = 0, nMaxValueNameLen = 0, nMaxValueLen = 0;
FILETIME timeFile;
HKEY hKey; long nError = 0;
if (ERROR_SUCCESS == RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\360desktop\\",
                        0, KEY_ALL_ACCESS, &hKey)) {
      if (ERROR_SUCCESS == (nError = RegQueryInfoKey(hKey, szClassName,
&nClassLen,
         NULL, &nSubKeys, &nMaxSubKey, &nMaxClass, &nValues,
         &nMaxValueNameLen, &nMaxValueLen, NULL, &timeFile ))){
            CString szResult;
            szResult.Format(_T("szClassName:%s, nClassLen:%d,
nSubKeys:%d, \
                        nMaxSubKey:%d, nMaxClass:%d,\
                              nValues:%d, nMaxValueNameLen:%d,
nMaxValueLen:%d "),
                              szClassName, nClassLen, nSubKeys,
nMaxSubKey,
                        nMaxClass, nValues, nMaxValueNameLen,
                        nMaxValueLen);
            AfxMessageBox(szResult);
      }
}
```

4）值项批量查询(提取)

RegQueryMultipleValue 此方法是批量查询、提取指定项下的值项数据，操作繁琐，不是很方便，不建议新手用，新手可用 RegEnumValue 或 RegQueryValueEx 一个一个提取。

RegQueryMultipleValue 函数原型如下：

```
LONG WINAPI RegQueryMultipleValues(
 __in        HKEY hKey, //已打开的项句柄，必须有 KEY_QUERY_VALUE 权限，
                        //同时也可以是几大根键值：
                        //HKEY_CLASSES_ROOT
                        //HKEY_CURRENT_CONFIG
                        //HKEY_CURRENT_USER
                        //HKEY_LOCAL_MACHINE
                        //HKEY_USERS
 __out       PVALENT val_list,//数组指针，装值项信息用的，对这个参数下面详细介绍
 __in        DWORD num_vals,//val_list 数组元素个数
 __out_opt   LPTSTR lpValueBuf,//装值项的值的 buff，全部值项的值串行存放，配合
                        //val_list 中的信息提取值，所以说麻烦吧
 __inout_opt LPDWORD ldwTotsize//plValueBuf 容量，返回实际接收字节数
);
```

此函数，通过第四个参数 lpValueBuf 接收指定项下所有值项的值，把所有值项的值安串行方式装载，提取的时候，配合第二个参数 val_list 提取，VALENT 结构中存储了值项的值类型，还有值在 lpValueBuf 中的具体位置，还有值数据在 lpValueBuf 中占的字节数。下面介绍一下 VALENT 结构原型：

typedef struct value_ent {

LPTSTR ve_valuename; //值项名称，调用 RegQueryMultipleValue 前必须填充，

//RegQueryMultipleValue 函数是通过这个名称

来提取值项数据的

DWORD ve_valuelen;//取回的值项的值的字节数

DWORD_PTR ve_valueptr;//值项的值在 RegQueryMultipleValues 参数

lpValueBuf 中的位置

DWORD ve_type; //取回值项的值的类型

} VALENT, *PVALENT;

以下为伪代码：

```
char szClassName[100]; memset(szClassName, 0, 100);
VALENT pValent[3]; DWORD nValentCount = 3;
memset(pValent, 0, sizeof(pValent));

pValent[0].ve_valuename = _T("Version");//输入值项名称

pValent[1].ve_valuename = _T("uac0");
pValent[2].ve_valuename = _T("TestDword");

char szVale[1024]; DWORD nValueLen = 1024;
FILETIME timeFile;
HKEY hKey; long nError = 0;
if (ERROR_SUCCESS == RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\360desktop\\",
        0, KEY_ALL_ACCESS, &hKey)) {
        if (ERROR_SUCCESS == (nError = RegQueryMultipleValues(hKey,
pValent,
                nValentCount, szVale, &nValueLen))){
                        CString szResult;
                        for (int i = 0; i < nValentCount; ++i) {
                                CString szTemp;
                                if(REG_SZ == pValent[i].ve_type){

                                        szTemp.Format(_T("第%d 个参数[%s]的值

是:%s\r"), i,
```

```
                                        pValent[i].ve_valuename,
(char*)pValent[i].ve_valueptr);
                        }
                        else if (REG_DWORD == pValent[i].ve_type){
                                szTemp.Format(_T("第%d 个参数[%s]的值
是:%d\r"), i,
                                        pValent[i].ve_valuename,
*((DWORD*)pValent[i].ve_valueptr));
                        }
                        if(!szTemp.IsEmpty())
                                szResult += szTemp;
                }
                AfxMessageBox(szResult);
        }
}
```

注册表相关项的切图如下：

运行代码效果切图如下：