# FoxPro 应用程序编程接口及其库构造工具

张能立　（武汉汽车工业大学计算机工程系　430070）
丁俊英　（湖北省物资学校计算机教研室）

**摘要**：本文介绍了利用 VC 开发 FoxPro 动态连接库的一般方法和相关软件环境的设置，并以一实例介绍了实现的过程。

## 一、FoxPro 应用编程接口

FoxPro 2.5 是目前微机上性能最好的关系型数据管理系统，其良好的图形界面、窗口功能、多种资源可视化设计及举例相关查询技术等卓越的性能使人耳目一新，已逐渐成为微机上主流数据库产品。但是象读写 A/D 转换器、数字 I/O 和串行通讯口等应用，单靠 FoxPro 是不能完成的。自 FoxPro2.0 推出以来，由于具有 API（应用程序编程接口），所以可以利用 C 语言或汇编语言方便地生成 FoxPro 所要求的动态连接库，利用该动态连接库可以轻而易举地实现读写 A/D 转换器等功能。此外，利用 API 还可增加图形、加密和网络等功能，还可以方便地使用 Windows 应用程序和 .DLL 库。FoxPro 2.5 for DOS 的 API 库使用的是 .PLB 格式，而 FoxPro 2.5 for Windows 的 API 使用的是 .FLL（Fox Link Library）格式。事实上，.FLL 库文件是 Windows 动态连接库 DLL（Dynamic Link Library）的一种形式。

## 二、如何编写 FoxPro 动态连接库

一般 FoxPro 动态连接库（.PLB 或 .FLL）文件可用 C 语言或汇编语言编写，但还要利用 Microsoft 的库构造工具集 LCK（Library Construction Kit）才能达到目的。LCK 提供了所有生成 .FLL 或 .PLB 文件所必需的头文件和库文件。

所有用 C 编写的动态连接库原代码框架必须是如下形式：

```
#include <pro-ext.h>
void FAR function(ParamBlk FAR * parm)
{
  //用户编写的原程序代码
}
FoxInfo myFoxInfo[] =
{
    {"FUNC-NAME", (FPFI)function, ParmNumb, ParmType},
};
FoxTable -FoxTable =
{
    (FoxTable FAR * )0, sizeof(myFoxInfo) / sizeof(FoxInfo), myFoxInfo
};
```

下面对以上有关部分作一介绍：

1. pro-ext.h **文件**

这个文件由 LCK 工具集提供，它包含了对编写 FoxPro API 动态连接库函数所需用到的各种句柄、结构、相关函数的定义。

2. FoxInfo **结构**

这一结构定义了每一个提供给 FoxPro 的新函数。第一项 FUNC-NAME 给出的是 FoxPro 用来调用新函数的名称，名称必须用引号引起来，并用大写字母给出；第二项（FPFI）function 是 C 函数名，C 函数名可以与 FoxPro 用的名不同，但必须与实际用的 C 函数名相同，FPFI 是在 pro-ext.h 中定义指向函数的指针；第三项 ParmNumb 指定传递参数的个数；第四项 ParmType 指定传递参数的类型：C 字符；D 日期；I 整数；L 逻辑；R 变量传送；？允许各种类型。

3. FoxTable **结构**

该结构向 FoxPro 提供建立动态库所需要的内存，并定义函数地址。第一项是供 FoxPro 内部使用的指针，初始化为 0；第二项是库文件中函数的个数；第三项是函数定义表指针。

4. **函数原型说明** void FAR function（ParamBlk FAR * parm）

所有被 FoxPro 直接调用的 API 函数必须是 void 类型，这并不意味着不能得到函数的返回值，而是不能用常规的返回机制。LCK 提供了专门的函数来返回数字、字符或逻辑等值。ParmBlk 是一个参数块结构，该参数块保存了从 FoxPro 中传递来的参数信息，通过指针 parm 可以得到参数值。下面介绍 FoxPro 和 C 函数之间传递参数的方法。

## 三、FoxPro 和 C 函数之间参数传递机理

1. FoxPro **传递参数给 C 函数**

不论实参数目多少，FoxPro 向 C 函数只传一个参数（ *

parm),该参数是 ParamBlk 型指针。在 FoxPro 中可以通过传值(Call－by－value)或传地址(call－by－reference)的方式将参数传给 C 函数。要获得参数值,每一种方式都需要一个唯一的结构。传值方式是通过一个值结构来处理,传地址方式是通过一个指针结构来处理。当两种结构物理上重叠时,则建立一联合。ParamBlk 结构即是值结构和指针结构的联合。该结构在 pro-ext.h 中定义如下:typedef struct {

```
short int pCount; / * 传递参数个数 * /
Parameter P[1];
}ParamBlk;
typedef union{
Value val; / * 传值方式,value 为值结构 * /
Locator loc; / * 传址方式,Locator 为指针结构 * /
}Parameter;
```

Value 和 Locator 结构的第一个元素都用来指明被传递参数的类型。ParamBlk 结构的值结构定义如下:

```
typedef struct{
char ev-type;
char ev-padding;
short ev-width;
unsigned short ev-length;
long ev-long;
double ev-real;
MHANDLE ev-handle;
}Value;
```

其中第一个元素 ev-type 取以下的值之一:C, N, I, D, G, L, M,其余元素的定义取决于参数的类型,具体含义见表 1:

表 1

| 元素类型 | 字符 | 数值 | 整数 | 日期 | 逻辑 | Memo/Gen |
|---|---|---|---|---|---|---|
| ev-type | 'C' | 'N' | 'I' | 'D' | 'L' | 'M'/'G' |
| ev-width | | 宽度 | 宽度 | | | 文件通道 |
| ev-length | 字符串长度 | 小数位数 | | | 逻辑值 | |
| ev-long | | | 长整数 | | | 内存字段长度 |
| ev-real | | 双精度数 | | 日期值 | | 内存字段偏移量 |
| ev-handle | 指向字符串的内存句柄 | | | | | |

指针(Locator)结构用于按地址方式的传参,也用于传数组,备注字段和通用(general)字段。该结构定义如下:

```
typedef struct{
char l-type;
short l-where;
short l-NTI;
short l-offset;
short l-subs;
short l-sub1;
short l-sub2;
}Locatory;
```

第一个元素 l-type 必须为 'R',其它元素含义见表 2:

表 2

| 结构元素 | 用法 |
|---|---|
| l-type | 'R' |
| l-where | 数据库工作区号或－1 表示内存变量 |
| l-NTI | 变量名表偏移 |
| l-offset | 数据库索引 |
| l-subs | 0 为内存变量,或者为一下标数(若是数组) |
| l-sub1 | 第一下标(若是数组) |
| l-sub2 | 第二下标(若是二维数组) |

2.C 函数返回值给 FoxPro

C 函数不能用 return 或虚实结合的方式给 FoxPro 返回函数值。库构造工具集 LCK 提供一组专门的库函数实现 C 函数向 FoxPro 返回值。如:

void -RetInt (long ival, int width);该函数返回指定宽度的整数值给 FoxPro

void -RetFloat(double flt, int width, int dec);该函数返回指定宽度、小数值 的浮点型值给 FoxPro

void -RetChar(char FAR * string);该函数返回指定的字符串给 FoxPro

void -RetDateStr(char FAR * string);该函数返回指定的日期型值给 FoxPro

void -RetLogical(int);该函数返回指定的逻辑值给 FoxPro

## 四、项目文件中有关编译、连接选项的设置

虽然 LCK 提供了一个用于 Microsoft C/C++ 7.0 和 Windows 环境下的项目文件 winc 7.mak,并说明该文件同样适用于 Visual C++。其实并不然,应将该文件有关部分修改后才能用于 VC。假设 Visual C++ 和 LCK 分别安装在 D 盘 MSVC 子目录和 FOXLCK 子目录下,下面介绍修改 winc7.mak 的方法。

1.将子目录下 \ MSVC \ SAMPLES \ INSTVER \ dllentry.obj 拷贝到 \ MSVC \ LIB 子目录下,并将其改名为 libentey.obj。

2.将 winc7.mak 中路径设置改为:

C7DIR = D:\MSVC

FOXDIR = D:\FOXLCK

3.将 winc7.mak 中的连接(link)选项

LFLAGS = $(LFLAGS)/NOF 改为

LFLAGS = $(LFLAGS)/NOE

4.**将修改过** winc7.mak **另保存为** vc.mak。

建立 FoxPro 动态连接库(.FLL)的步骤如下:

(1)将写好 C 源代码放在 \FOXLCK 子目录下,

(2)path d:\msvc\bin。

(3)执行 nmake FLLNAME=源文件名(不要带扩展名.C) /F vc.mak

如果 C 源代码没有什么错误,则会生成好 FoxPro 动态连接库(.FLL)。

建立 FoxPro 动态连接库(.PLB)的步骤

(1)将写好 C 源代码放在 \FOXLCK 子目录下。

(2)path d:\msvc\bin。

(3)将 docc7.mak 项目文件中路径设置改为:

C7DIR = D:\MSVC

FOXDIR = D:\FOXLCK

(4)执行 nmake PLBNAME=源文件名(不要带扩展名.C) MODEL=<model> /F dosc7.mak 其中 model 是内存模式选择,可以为 S(小模式)、M(中模式)和 L(大模式),实际应用中建议使用 L(大模式)。如果 C 源代码没有什么错误,则会生成好 FoxPro 动态连接库(.PLB)。

## 五、FoxPro 动态连接库的编写及使用

```
/* REVERSE.C - 简单 API 例程
 * 功能:该函数能使字符串反转
 * 返回值:反转后的字符串
 * 示例:reverse('abcd')
 */
#include <pro-ext.h>
void FAR reverse(ParamBlk FAR *parm)
{
int i;
MHANDLE mh-out; /* 定义一内存句柄 */
char FAR * in-string; /* 定义一指向待输入字符串指针 */
char FAR * out-string; /* 定义一指向待输出字符串指针 */
if ((mh-out = -AllocHand(parm->p[0].val.ev-length + 1)) == 0) /* 分配内存 */
-Error(182); /* 分配内存失败 */
in-string = -HandToPtr(parm->p[0].val.ev-handle); /* 将内存句柄转换成地址指针 */
out-string = (char FAR *) -HandToPtr(mh-out) + parm->p[0].val.ev-length;
*(out-string--) = '\0'; /* -RetChar() 需要字符串终止字符 '\0' */
for (i = 0; i < parm->p[0].val.ev-length; i++)
*(out-string--) = *(in-string++); /* 字符串反转 */
-HLock(mh-out); /* 向 FoxPro 回传值期间须锁定相关内存句柄 */
-RetChar(out-string+1); /* 向 FoxPro 回传反转后的字符串 */
-FreeHand(mh-out); /* 释放内存句柄 */
}
FoxInfo myFoxInfo[] =
{
{"REVERSE", (FPFI) reverse, 1, "C"},
};
FoxTable -FoxTable =
{
(FoxTable FAR *)0, sizeof(myFoxInfo) / sizeof(FoxInfo), myFoxInfo
};
```

该程序按上述所介绍的方法会编译连接生成 reverse.fll 和 reverse.plb 动态连接库,在 FoxPro 中用命令 SET LIBRARY TO <动态连接库文件名> 打开动态连接库,用户就可以象使用内部函数那样使用动态连接库中的函数。例如,在 FoxPro 命令窗口中可以执行如下命令:

```
SET LIBRARY TO d:\foxlck\reverse
? reverse('abcd')
```

屏幕上会显示出结果 dcba,还可以使用下列命令补充新库

```
SET LIBRARY TO Libl ADDITIVE
```

这里必须用 ADDITIVE 关键字,否则,FoxPro 会冲掉原来的库。用不带参数的 SET LIBRARY TO 命令,可把所有的库卸下来,释放某一个库可用 RELEASE LIBRARY <库文件名> 即可。

以上程序在 486 兼容机(8M 内存、540M 硬盘)上用 Visual C++1.0 和 LCK 编译,在 FoxPro for Windows 下运行通过。

**参考文献:**

[1] [美] George F. Goley IV《FoxPro 应用程序开发方法与技巧》清华大学出版社

[2] [美] Pat Adams and Jordan Powell《FocPro2.5 高级开发指南》清华大学出版社