FoxPro 与 C 语言程序接口分析

盛裕平　（总后科学研究所）

一、值和寻址的数据结构

1.值结构定义

```
typedef stuct
        char                    ev_type;
        char                    ev_padding;
        short                   ev_width;
        unsigned short          ev_lengh;
        long                    ev_long;
        double                  ev_real;
        MHANDLE                 ev_handle;
}Value;
```

下表是值结构各字段的意义：

字段	FoxPro 数据类型					
^	字符型	数值型	整数型	日期型	逻辑型	备注型(4)
ev_type	'C'	'N'	'T'	'D'	'L'	'M'
ev_width		显示宽度	显示宽度			
ec_padding						FCHN(5)
ev_length	字符串长(1)	小数位数			逻辑值	
ev_long			长整数			备注字段的长度
ev_real	字符串的	双精度型		日期值(3)		备注字段
ev_handle	MHANDLE(2)	准确值				的偏移量

(1)字符串长度的真正表示；

(2) 一种数据类型；

(3)日期是按格林威治时间计算的浮点双精度数；

下表是寻址结构各字段的意义：

寻址字段	字 段 使 用
l_type	'R'
l_where	包含此字段的数据库号
l_NTI	名字表索引，FoxPro 内部使用
l_offset	数据库字段的序号，FoxPro 内部使用
l_subs	仅供内存变量使用，数组的维数
l_sub1	仅供内存变量使用，如果 l_subs 不是 0，表示第一个下标
l_sub2	仅供内存变量使用，如果 l_subs 是 2，表示第二个下标

(4)为了避免 memo 文件的损坏，不要在调用 AllocMemo 之前写入一个 memo 文件；

(5)FCHAN 数据类型是赋给每个被 FoxPro 打开或 API 用_FCreate( )、_FOpen( )打开文件的文件通道。

2.寻址结构的定义

```
typedef struct
{   char    l_type;
    short   l_where,l_nti,l_offset,l_subs,l_sub1,l_sub2;
}Locator;
```

二、参数的传递

1、值传递和寻址传递

如果主程序传递的是参数的拷贝，那么该传递是值传递；如果传递的是参数的地址，那么该传递是寻址传递。在缺省情况下，FoxPro 通过值传递到过程，通过寻址传递到函数。

2、参数块

参数块用于从 FoxPro 向应用程序接口传递参数，参数块结构也称作'ParamBlk'。

ParamBlk 结构包括参数个数和参数数组。参数是值结构和寻址结构的联合。寻址结构的第一个字节是'R'，值结构的第一个字节是：'C'、'N'、'I'、'D'、'L'、'M'等字符之一。

```
typedef union                   //参数结构
{       Value val;
        Locate loc;
}Parameter;
typedef struct                  //参数块结构
{       short       pCount;     //传递参数的个数
        Parameter p[1];         //pCount 个参数
}ParamBlk;
```

外部过程通过接收一个指向参数块的远指针来接收参数块，参数块包含了调入、调出 FoxPro 接口函数参数的所有信息。函数的定义必须采用以下格式：

void FAR <函数名>(ParamBlk FAR * parm)

每个参数的信息储存在值或寻址结构中，要获得参数的信息，首先要获得参数联合数组的元素。

三、FoxInfo 和 FoxTable 结构

FoxPro 和 C 语言之间通过 FoxInfo 结构通讯，FoxPro 通过 FoxInfo 结构决定函数名、参数个数和参

数数据类型。FoxTable 是保存 FoxInfo 结构信息的连接表。

1.FoxInfo 和 FoxTable 结构定义

```
//FPFI 是指向返回整数的函数的 32 位指针
typedef int (FAR *FPFI)( );
#/define INTERNAL   -1    //不能从 FoxPro 调用
#define   CALLONLOAD -2  //当 FLL 库装入后调用
#define CALLONUNLOAD -3  //当 FLL 库没装入时调用
  typedef struct
  {   char FAR     *funcName;      //函数名
      FPFI         function;       //实际调用函数
      short        parmCount;      //指定参数个数
      char FAR     *parmTypes;     //参数表描述
  }FoxInfo;
  typedef struct _FoxTable
  {       struct _FoxTable FAR     //库连接表
                  *nextLibrary;
          short     infoCount;     //该库中函数个数
          FoxInfo  FAR *infoPtr;   //函数列表
  }FoxTable;
```

2.FoxInfo 结构描述

FoxInfo 结构是 FoxPro 和应用接口程序之间通讯的工具，通讯内容包括函数名和参数描述。FoxInfo 一般的结构如下：

```
FoxInfo arrayname[]={
  {funcName1,   FPFI   function1,   parmCount1,
parmType1},
  ·····················,
  {funcNamen,   FPFI   functionn,   parmCountn,
parmTypen}
  };
```

arrayname 是 FoxInfo 数组变量；funcName 是 FoxPro 用户调用的函数名(全部大写，不超过 10 个字符)；FPFI function 是 C 语言程序的地址；parmCount 指定在 parmTypes 字符串中描述的参数个数或以下标志值：INTERNAL、CALLONLOAD、CALLONUNLOAD；parmTypes 描述每个参数的数据类型，parmTypesD 的有效值是：'C'、'N'、'I'、'D'、'L'、'R'和'?'，这些值之间用逗号相隔。

3.FoxTable 结构描述

FoxTable 结构是一个保存库中所有 FoxInfo 信息的连接表。

```
FoxTable FoxTable={nextLibrary, infoCount,
infoPtr};
```

nextLibrary 是 FoxPro 内部使用的指针，必须初始化为 0。infoCount 是库中定义的 FoxPro 外部过程的个数。infoPtr 是 FoxInfo 数组结构第一个元素的地址。

四、程序举例

下面是用 MacroSoft C 语言写的一段接口程序，该程序在 FoxProw 中显示一个窗口，这段程序没有参数传递。

```
#include <stdio.h>
#include <pro_ext.h>
void FAR hello( ParamBlk FAR *parm )
 {   WHANDLE wh;
     Point pt;
     wh =_WOpen( 10, 20, 14, 50,
        WMINIMIZE + MOVE + SHADOW +
CLOSE,
        DIALOG SCHEME,0,WO DOUBLEBOX );
     if( wh != 0 )
     {_WSetTitle( wh, "你好，FoxPro!" );
        _WSetFooter( wh, "MacroSoft C" );
        _WShow( wh );
        pt.h = 9;
        pt.v = 1;
        _WPosCursor( wh, pt );
        _WPutStr(wh,"欢迎进入应用程序接口！");
     }
     _RetInt( 0L, 1 );
 }
FoxInfo myFoxInfo[] =
{{ "HELLO", (FPFI) hello, 0, "?" }
};
FoxTable  FoxTable =
{ (FoxTable FAR *) 0,sizeof( myFoxInfo) / sizeof(
FoxInfo ),myFoxInfo};
```

这段程序编译后生成后缀为"FLL"的动态连接库文件"hello.fll"，在 FoxProw 中调用过程如下：

```
set library to hello.fll
=hello()
set library to
```