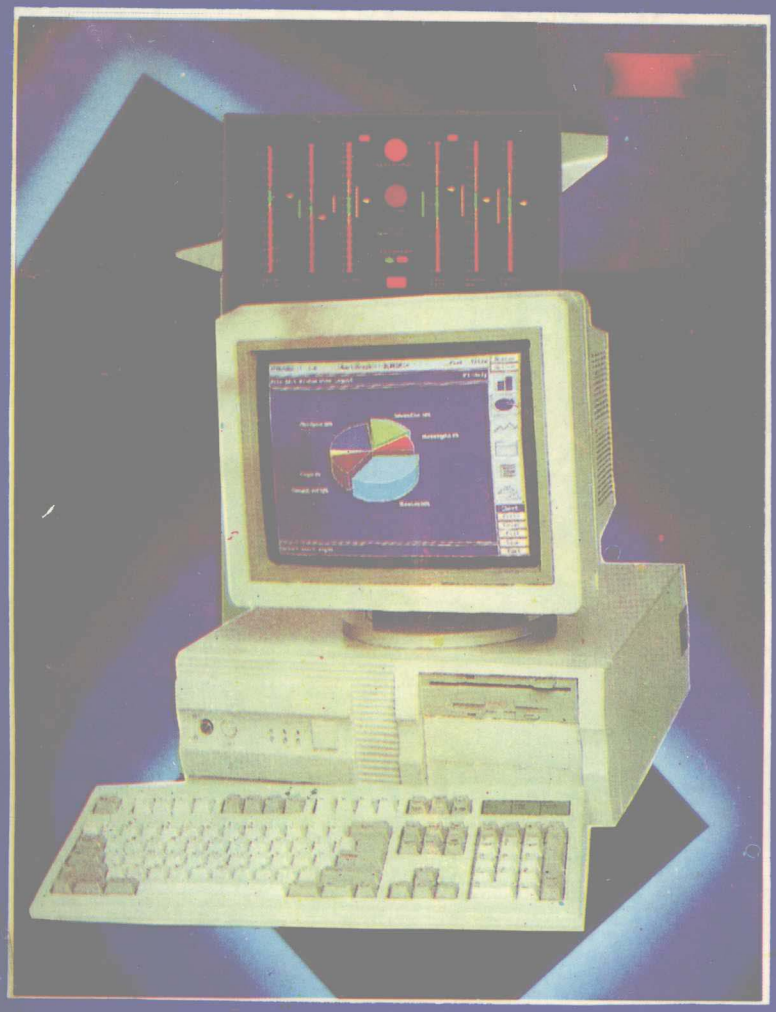
Foxpro 2.6
高级开发技巧

刘键　龙明　樊永良　刘茗　编著
张振礼　审校



陕西电子杂志社

FOXPRO 2.6
高级开发技巧

刘键　龙明　樊永良　刘茗　编著
张振礼　　审校

陕西电子杂志社

内 容 提 要

本书是一本介绍 FoxPro 编程经验和技巧的专著，重点介绍了利用 FoxPro 2.6 For DOS 开发信息管理系统中关键性问题及技术难点的解决方法。涉及到的主要内容有：网络工作站的环境服务，MIS 菜单设计，表驱动方式的程序设计方法，动态查询、录入、显示、打印通用代码的设计，字典功能的实现，动态生成报表结构，西文 FoxPro 环境下半个汉字的处理，文字编辑器，图文数据库，彩色直方图的显示与黑白打印。

所涉及到的内容在随书软盘上均有 FoxPro 源代码、C++和汇编语言的源代码及可执行文件，可供用户直接调用。对于熟悉和不熟悉 C++、汇编语言的用户均适用。

Foxpro 2.6 高级开发技巧

刘 健等编 张振礼 审校

* * * * *

陕西电子杂志社出版发行

西安电子科技大学印刷厂印刷

开本：787×1092 1/16 印张：17 字数：406.5 千字

1995 年 12 月第 1 版 1995 年 12 月第 1 次印刷

印数：1—5000 册

国内统一刊号：CN61—1224/TN

定价：25.00 元

前　言

随着计算机应用的推广普及，以数据库为基础的管理信息系统 MIS (Managment Information System)的需求迅速增长，在企业的激烈竞争中信息管理和分析已成为成败的关键因素。但是各类管理信息系统的开发工作量大，而且 MIS 软件的复杂程度也越来越高。采用传统的软件开发方法开发 MIS 软件，不但要求软件开发人员具有较强的技术能力，而且生产效率低，开发周期长，软件质量差，维护费用高，软件的生产能力远远满足不了社会的需求。这促使人们寻求新方法和新工具。

各类 MIS 自动生成器的问世，提高了管理信息系统的开发效率，缩短了开发周期，降低了开发成本。但是各行各业的管理信息系统又都有其自身的特殊情况和实际要求，仅靠 MIS 自动生成器是难以满足要求的，还必须借助于手工实现。并且，由于各类 MIS 生成器均不给用户提供开发工具的源代码，所以对高水平的 MIS 开发人员来说，当进行深入开发时，总是摆脱不了 MIS 开发工具的种种限制，开发工作总有点不得心应手的感觉。

鉴于此，本书力图以手工与 MIS 自动生成器两者之间给 MIS 开发者提供一些技巧性较强、通用性较好的程序模块及相应源代码，开发者完全可以将其融入自己开发的管理信息系统中去。这样即可以减少不必要的重复性劳动，又为开发者建立自己的一套开发工具奠定基础。

本书由刘键担任主编，龙明编写了第一章的大部分内容，樊永良编写了第六章的大部分内容，张振礼对全书进行了审校。

在本书的编写过程中得到了陕西电子杂志社全体同志特别是张忠智高级工程师的热情帮助和大力支持，在此表示衷心的感谢。

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第四章　报表结构的动态生成

第五章　西文 FOXPRO 环境下半个汉字的处理方法

第六章　FOXPRO 图文数据库

第一章

MIS 开发中如何使用网络工作站的环境服务

§1 引言

在局域网络环境中，利用网络本身提供的命名服务是非常重要的。在 MIS 开发过程中，我们必须要验证使用者身份，设定其工作权限，分配网络资源。在目前，一般的 MIS 管理软件都是利用网络系统提供的一些管理软件，设置有关用户的工作信息。MIS 软件则按照软件设计者预先设想，对诸系统资源进行运作处理。而一旦网络系统环境设置不正确，便造成 MIS 软件的运行失败。这种对网络用户和资源的管理方法，即给软件使用者带来极大的不便，也容易造成 MIS 所记录的软件运行环境和网络实际系统环境不统一的现象。

事实上，每个 NetWaer 文件服务器都有一个有关资源和能在网络上工作的客户方面的数据库，NetWaer 将这个具有特殊功能的数据库称为装订库。我们可以利用 NetWaer 提供的命名服务，直接运用装定库提供的有关数据，获取网络用户和有关资源的信息。显然，实现上述功能，仅仅依靠数据库本身所提供的标准函数是不够的。

本篇介绍一种如何利用 NetWaer 所提供的 Network C 接口，实现对 Novell 网络系统的增值开发，构造一个有组织的、安全的 MIS 管理系统的方法。

§2 相关知识

每个服务器都是利用 SYS:SYSTEM 目录下隐含文件来保存它自己的装订库的。装订库文件是 NET $ BIND. SYS 和 NET $ BVAL. SYS。其中，NET $ BIND. SYS 文件中包含有网络上任何有名字的物理或逻辑实体，如用户、用户组、文件服务器等，NET $ BVAL. SYS 保存的是特性值。应用程序可通过调用装订库函数，查询网络上有关被命名的网络目标信息，如文件服务器的用户信息等等。显然，除了读取装订库外，应用程序还可以创建装订库目标，增加或改变有关目标的特性（如应用权限等），通过 Network C 接口，这是可以做到的，出于本篇的目的和篇幅，我们不涉及这方面的内容，着重介绍装订库信息的获取及如何取得网络工作站环境信息的方法。

§3 例子程序——装订库信息查询程序

BINDSCAN. C 程序可以完成查看装订库的目标名、特性名及特性值等功能。并且，允

许使用通配符，可以完成整个装订库的查询。

通过 BORLAND C 3.1 或 TURBO C 2.0 编译。

有关命令如下：

bcc －ms －K －c －G －d －wnod bindscan.c

tlink /c /x c0s bindscan，bindscan.exe，nul.map，snit cs maths emu

其工作方式如下：

（1）运行参数的获取

BINDSCAN.C 程序可以带或不带参数运行。主函数 MAIN 通过对命令行扫描、解释，获取有关参数，如果不足则向用户索取。注意，－1 为本程序的通配符。

```
if( argc == 3 )
    doScanBinderyObject(strupr(argv[1]), atoi(argv[2]));
else{
    printf("Enter searchName, wildcards are okay: ");
    scanf("%s",searchName);
    printf("Enter searchType, enter -1 for WILD: ");
    scanf("%d",&searchType);
    doScanBinderyObject(strupr(searchName),searchType);
}
```

（2）装订库的扫描

一旦程序拥有了所需要的参数后，它将调用 doScanBinderyObject 函数。doScanBinderyObject 反复调用 NetWare 函数 ScanBinderyObject，直到没有指定扫描的信息目标，即其返回代码 0xFC（NO _ SUCH _ OBJECT）为止。

输送给 BINDSCAN 的参数中，searchName 是指定目标，凡是与这个参数匹配的目标，ScanBinderyObject 都能为其找到相应的装订库信息（目标名、特性及特性值）。当查询到有关特定的装订库目标信息时，ScanBinderyObject 将数据格式化，并输出其名称、类型、类别及读/写权限。

（3）目标名、ID 号和类型

ScanBinderyObject 将这三个值不经过任何转换步骤，直接输出打印。其中，目标名（objectName）是作为 ASCII 子串打印的，ID 号（objectID）和类型（objectType）都是以十六进制打印的。

其它由 ScanBinderyObject 返回的值，则需要做一些转换，再格式化处理。

（4）目标类别

ScanBinderyObject 将目标标志（objectFlag）参数转换为便于阅读的字符串，然后输出打印。

```
printf("%#08lx   %#02x      %-7s ",objectID,objectType,
        objectFlag == BF _ DYNAMIC ? "Dynamic" : "Static" );
```

（5）目标的读/写权限

在输出目标的读/写权限时，ScanBinderyObject 首先用宏指令，屏蔽无效位，然后利用 nibbleToString 将有关信息转变为用户可读的文字格式。

```
printf("%-12s%-10s\n\n",
        nibbleToString(readRights(objectSecurity)),
        nibbleToString(writeRights(objectSecurity)));
if( objectHasProperties )
    doScanProperty(objectName,objectType,"*");
else
    printf("***** No properties associated with %s *****\n",
        objectName);
```

注意两个宏指令 readRights 和 writeRights 的运算方式，其利用特征值屏蔽 objectSecurity 无效位，然后将最后结果以 BYTE 打印出来。

nibbleToString 函数将上述宏指令的返回值，转换为 nibble 参数，然后利用五个 case 语句，将读/写权限转换成字符串，如“Anyone”、“Logged”等等。

（6）特性检查

doScanBinderyObject 核查由 NetWare 系统函数 ScanBinderyObject 带回目标的特性参数，看看诸目标是否具有与之相关的特性。如果没有，该函数就通知相应用户。如果目标确实拥有该特性，doScanBinderyObject 将调用 doScanProperty 函数。

doScanProperty 反复调用一些其它函数，以找到相应目标的特征信息。当查找到有关信息后，其将转换为每种特性的英文名子，如目标的名称、类别、类型以及读/写权限等。如果某值是与被扫描特性相关，doScanProperty 函数将调用 dumpPropertyValue，返回相应的数值。

doScanBinderyObject 将特性名以 ASCII 字符串形式直接输出，没有进行任何转换，而其它值在输出前，都将需要一些转换。

（A）特性类别（Class）和类型（Type）

在输出特性的类别（static 静态或 dynamic 动态）和类型（Item 分项或集合 Set）之前，doScanProperty 中解释了 ScanProperty 的特性标志参数并将其转换为相应的字符串。

其中，类别的转换方式为

```
isDynamic(propertyFlags) ? "Dynamic" : "Static",
```

类型的转换方式为

```
isItem(propertyFlags) ? "Item" : "Set");
```

（B）特性的读/写权限

在输出目标特性的读/写权限时，由 doScanProperty 重复进行一次，将目标的特性数值，转换为字符串，转换算法同前。

（C）特性值

doSccanProperty 函数检查是否存在与被扫描的特性相关的值，然后将其转换为“YES”或“NO”输出。

```
propertyHasValue ? "Yes" : "No" );
```

（7）获取特性值

doScanProperty 调用 dumpPropertyValue 函数并把从 ScanProperty 中得到的目标名、目标类型及特性名等参数传递给它。

dumpPropertyValue 调用 NetWare 系统的函数 ReadPropertyValue 并把从此函数得到的信息分解为数段，用以指明是一个特性集合还是一个特性分项。

如果从 ReadPropertyValue 中返回的特性标志（propertyFlags）与 BF _ SET "与"后，结果为真，则 dumpPropertyValue 为属性集将得来的信息分成相应的段。换言之，它把信息分成 4 BYTE 长的目标 ID 号（一个段最多能有 32 个 ID 号），NetWare 规定，各段均为 128 个字节长。

```
if( propertyFlags & BF _ SET ){
    printf("\n");
    for( i=0; i < 32 && setValue[i]; ++i ){
        printf("%#10lx      %c", LongSwap(setValue[i]),
                (i +1) % 4 ? ' ' : '\n');
    }
}
```

如果从 ReadPropertyValue 中返回的特性标志（propertyFlags）与 BF _ SET "与"后，结果不为真，则 dumpPropertyValue 也仍然将得来的信息分成 128BYTE 长的各段。但是，因为分项特性并不受存在它们段中的信息或那些信息的存储方式的限制，故 dumpPropertyValue 仅输出一个十六进制数值，其输出格式是每 8bit 的特性值（propertyValue）打印成两个十六进制的数，用两个空格分开。

```
printf("--------\n");
for( i=0; i < 128; ++i ){
    printf("%02x %c", (WORD)propertyValue[i],
            (i + 1) % 16 ? ' ' : '\n');
}
```

（8）程序的退出

doScanBindery 函数完成扫描和输出其输入参数所指定的每一目标的装订库信息后，程序将输出一条信息，表明有关目标的信息处理结束。同样，在它完成扫描和输出与某一个特定目标相关的所有特性之后，也会输出一条信息，表示那个目标已没有其它特性存在了。

§4　如何使用网络工作站的环境服务

在 MIS 开发工作中，有时要对一些程序的运行环境进行辨别，限制一些程序的运行地点，或出于某种目的，辨别程序应用者的合法性，这时，就要利用网络工作站的环境服务了。

工作站环境服务是一种特殊的 API 服务，借助于它，我们可以获得网络中的工作站地址、连接 ID 号表、文件服务器名称、驱动器连接 ID 号表等等。本章利用 C 函数来说明对工作站的一些典型的环境操作。

由于本章涉及的知识比较容易理解，故我们主要以例子说明。

§4.1 工作站环境信息获取程序

HELLO.C 程序主要完成确定目标的登录时间、所连接的服务器名称、工作站的网络地址、物理地址及连接号等等。

其编译方式与 BINDSCAN.C 相同。

其工作方式如下：

(1) 运行参数的获取

HELLO.C 程序必须带参数运行。主函数 MAIN 通过对命令行扫描、解释，获取有关参数。例如，要查询在名为 NOVELL 服务器上运行的，以 GUEST 登录工作站的环境信息时，其命令行的形式为

C>HELLO NOVELL GUEST

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef LC60
   #include <dos.h>
#else
   #include <conio.h>
#endif

#include <ctype.h>
#include <nit.h>
#include <niterror.h>

int    MapLoginDrive( void );
int    ConnectionInformation( void );
int    MyInternetAddress( void );
int    MyStationAddress( void );
int    HowManyConnections( void );

char     serverName[48];
WORD     objType;
char     objName[48];
WORD     connectionNumber;

void main( int argc, char *argv[])
{
int      ccode, i;
```

此为试读，需要完整PDF请访问：www.ertongbook.com

```
char    ch;
char    objPassword[48];
WORD connectionID, oldConnectionID;

/*-------------------------------------------------------------------
        如果用户没有给予一个正确的运行参数,输出一个信息,提示
        用户,正确的形式参数的格式
-------------------------------------------------------------------*/

  if ( (argc < 3) || (argc > 3) )
    {
      printf("\nusage:  HELLO <ServerName> <UserName>\n");
      exit(-1);
    }
  ConvertToUpperCase( argv[1] );
  ConvertToUpperCase( argv[2] );
  strcpy( serverName, argv[1] );
  strcpy( objName, argv[2] );

/*-------------------------------------------------------------------
        获取用户登录服务器的连接号
-------------------------------------------------------------------*/
  ccode = AttachToFileServer( serverName, &connectionID );
  if ( (ccode != 0) && (ccode != 0xF8) )
    {
      printf(" Error %d attaching to file server\n", ccode );
      goto FINISHED;
    }

/*-------------------------------------------------------------------
        保存原始的 ID 并且设定欲联接的服务器 ID
-------------------------------------------------------------------*/
  oldConnectionID = GetPreferredConnectionID();
  SetPreferredConnectionID( connectionID );

/*-------------------------------------------------------------------
        输入口令,并进行登录
-------------------------------------------------------------------*/
  printf("Enter your password:  ");
  i = 0;
  do
    {
      ch = (char)( toupper( getch() ) );
```

```
        objPassword[i++] = ch;
        if ( ch == 13 )
          objPassword[i-1] = '\0';
    } while ( ch != 13 );

    ccode = LoginToFileServer( objName, 1, objPassword );

    if ( ccode )
      {
        printf(" Error %d logging into file server\n", ccode );
        goto FINISHED;
      }

/*-------------------------------------------------------------------
        登录成功后,将"F"盘符赋给已登录的文件服务器目录
-------------------------------------------------------------------*/
    ccode = MapLoginDrive();
    if ( ccode )
      {
        printf(" Error %d mapping login drive\n", ccode );
        goto FINISHED;
      }

/*-------------------------------------------------------------------
        获取已建立连接的文件服务器信息
-------------------------------------------------------------------*/
    ccode = ConnectionInformation();
    if ( ccode )
      {
        printf(" Error %d getting connection information.\n", ccode );
        goto FINISHED;
      }

/*-------------------------------------------------------------------
        得到网络内部的连接地址
-------------------------------------------------------------------*/
    ccode = MyInternetAddress();
    if ( ccode )
      {
        printf(" Error %d getting your internet address\n", ccode );
        goto FINISHED;
      }
```

```
    MyStationAddress();

/*_____
        得到用户和文件服务器联接号码
    _____*/
    ccode = HowManyConnections();
    if ( ccode )
      {
        printf(" Error %d getting number of connections\n", ccode );
        goto FINISHED;
      }

FINISHED:
  /*_____
          Set the connection ID back to the original
    _____*/
    SetPreferredConnectionID( oldConnectionID );
}

/*_____
      本函数返回登录目标的环境信息.
      如目标名称、目标类型、目标 ID、登录时间和目标的连接号码
    _____*/

int   ConnectionInformation(void)
{
int     ccode;
char    objName[48];
long    objID;
BYTE loginTime[7];
char    day[10], time[9], ttime[3], AmPm[3];
WORDyear;
int     PM;
char    month[9];

    connectionNumber = GetConnectionNumber();
    ccode = GetConnectionInformation( connectionNumber, objName, &objType,
                                      &objID, loginTime );
    if ( ccode )
        return( ccode );

      if ( loginTime[0] < 80 )
          year = 2000 + loginTime[0];
```

```
else
    year = 1900 + loginTime[0];

switch (loginTime[3])
 {
    case 0:
            strcpy( time, "12:" );
            PM = FALSE;
            break;
    case 1:
            strcpy( time, "1:" );
            PM = FALSE;
            break;
    case 2:
            strcpy( time, "2:" );
            PM = FALSE;
            break;
    case 3:
            strcpy( time, "3:" );
            PM = FALSE;
            break;
    case 4:
            strcpy( time, "4:" );
            PM = FALSE;
            break;
    case 5:
            strcpy( time, "5:" );
            PM = FALSE;
            break;
    case 6:
            strcpy( time, "6:" );
            PM = FALSE;
            break;
    case 7:
            strcpy( time, "7:" );
            PM = FALSE;
            break;
    case 8:
            strcpy( time, "8:" );
            PM = FALSE;
            break;
    case 9:
            strcpy( time, "9:" );
```

```
            PM = FALSE;
            break;
case 10:
            strcpy( time, "10:" );
            PM = FALSE;
            break;
case 11:
            strcpy( time, "11:" );
            PM = FALSE;
            break;
case 12:
            strcpy( time, "12:" );
            PM = TRUE;
            break;
case 13:
            strcpy( time, "1:" );
            PM = TRUE;
            break;
case 14:
            strcpy( time, "2:" );
            PM = TRUE;
            break;
case 15:
            strcpy( time, "3:" );
            PM = TRUE;
            break;
case 16:
            strcpy( time, "4:" );
            PM = TRUE;
            break;
case 17:
            strcpy( time, "5:" );
            PM = TRUE;
            break;
case 18:
            strcpy( time, "6:" );
            PM = TRUE;
            break;
case 19:
            strcpy( time, "7:" );
            PM = TRUE;
            break;
case 20:
```

```
                strcpy( time, "8:" );
                PM = TRUE;
                break;
        case 21:
                strcpy( time, "9:" );
                PM = TRUE;
                break;
        case 22:
                strcpy( time, "10:" );
                PM = TRUE;
                break;
        case 23:
                strcpy( time, "11:" );
                PM = TRUE;
                break;
    }

switch (loginTime[4])
    {
        case 0:
                strcat( time, "00" );
                break;
        case 1:
                strcat( time, "01" );
                break;
        case 2:
                strcat( time, "02" );
                break;
        case 3:
                strcat( time, "03" );
                break;
        case 4:
                strcat( time, "04" );
                break;
        case 5:
                strcat( time, "05" );
                break;
        case 6:
                strcat( time, "06" );
                break;
        case 7:
                strcat( time, "07" );
                break;
```

此为试读，需要完整PDF请访问：www.ertongbook.com