Você pode estender as capacidades nativas do Visual FoxPro aproveitando as vantagens das facilidades dos controles ActiveX (.arquivos OCX) e bibliotecas de vínculo dinâmico (DLLs). As bibliotecas externas permitem que você acesse não só as capacidades de outros programas, mas também do próprio Windows. Por exemplo, você pode utilizar uma biblioteca classe OCX para ler e atualizar diretamente o **Registro** do Windows, ou você pode abrir funções do nível do sistema vinculando a um dos DLLs no Windows.

Se a funcionalidade de que você precisa ainda não estiver disponível em uma biblioteca externa, você pode criar o seu próprio arquivo OCX no C++ utilizando um compilador de 32 bits, como o Microsoft Visual C++™ versão 4.0 ou posterior. Para detalhes, consulte o capítulo 28, Acessando a API do Visual FoxPro.

Este capítulo discute:

- Utilizando bibliotecas externas
- Acessando os controles ActiveX
- Acessando bibliotecas de vínculo dinâmico
- Acessando uma biblioteca do Visual FoxPro

Utilizando bibliotecas externas

Na maior parte das vezes, o Visual FoxPro fornece todas as ferramentas que você precisa para completar o seu aplicativo. No entanto, você pode achar ocasionalmente que um aplicativo requer uma funcionalidade adicional ainda não está disponível no Visual FoxPro. Nestes casos, você pode ir além do Visual FoxPro e aproveitar as capacidades das bibliotecas externas.

O Visual FoxPro permite que você tenha acesso a estes tipos de bibliotecas externas:

- **Controles ActiveX (arquivos .OCX)**. Os controles ActiveX são bibliotecas de classes que incluem objetos criados para realizar tarefas específicas. A maior parte dos controles ActiveX adicionam objetos novos ao Visual FoxPro — tudo, desde um novo tipo de caixa de texto a um calendário, calculadora ou outro objeto complexo. Alguns objetos ActiveX também incorporam facilidades adicionais, como acesso ao seu sistema de correio eletrônico ou às portas de comunicação do seu computador. Como regra, depois de incorporar um controle ActiveX ao Visual FoxPro, você pode utilizar estes objetos nas suas bibliotecas de classes, como faria com qualquer classe base do Visual.
- **Bibliotecas de vínculo dinâmico (arquivos .DLL)**. Um arquivo .DLL é uma biblioteca de funções que você pode chamar a partir dos programas do Visual FoxPro, como faria com qualquer função definida pelo usuário no Visual FoxPro. Muitos programas do Windows — e o próprio Windows — tornam sua funcionalidade disponível utilizando os arquivos .DLL. Por exemplo, você pode acessar as configurações de cores do sistema para o Windows vinculando a um arquivo de sistema .DLL e chamando funções nele.
- **Bibliotecas externas do Visual FoxPro (arquivos .FLL)**. Um arquivo .FLL é como um arquivo .DLL, mas utiliza um protocolo especial para compartilhar dados com o Visual FoxPro, e freqüentemente contém chamadas para funções internas do Visual FoxPro. Como resultado, os arquivos .FLL são específicos do Visual FoxPro, ao contrário dos arquivos .DLL, que podem ser chamados a partir de qualquer programa do Windows. Você pode chamar as funções em um .FLL como faria com qualquer função definida pelo usuário no Visual FoxPro.

Antes de utilizar qualquer biblioteca, você deve se familiarizar com as convenções utilizadas para acessar suas funções ou controles. Por exemplo, se você quiser incluir um controle ActiveX em um formulário, precisa saber quais propriedades, eventos e métodos poderá utilizar para gerenciar o controle. Da mesma forma, se quiser chamar uma função em um arquivo .DLL, precisará saber o nome da função, o número e tipos de dados do parâmetro que ela requer, e o tipo de dados de seu valor de retorno. Em geral, você pode obter este tipo de informação na documentação que acompanha a biblioteca, em um livro ou sistema de **Ajuda**. Para obter informações sobre os arquivos do sistema .DLL para o Windows, você pode consultar o *Software Development Kit (SDK)* apropriado à sua versão do Windows.

Acessando os controles ActiveX

Você pode utilizar qualquer controle ActiveX que estiver disponível em seu computador. Para utilizar um controle ActiveX, você o adiciona a um formulário, em seguida define suas propriedades, escreve manipuladores para seus eventos ou chama os seus métodos. Você pode adicionar um controle ActiveX a um formulário utilizando a **Barra de ferramentas controles de formulários** ou o controle OLE Container, ou utilizando um código. Para maiores detalhes sobre adição de um controle ActiveX ao **Criador de formulários**, consulte o capítulo 16, Adicionando OLE.

Você pode criar um controle ActiveX em código de um modo muito semelhante ao que utilizaria para criar qualquer controle do Visual FoxPro. No entanto, antes de criar o controle, você precisa determinar o nome da biblioteca de classes do controle, que está armazenado no **Registro** do Windows. Se você não tiver outra forma de determinar o nome da biblioteca de classes, utilize o **Criador de formulários** para criar o controle (conforme descrito na seção anterior) e, em seguida, obtenha a propriedade OLEClass do controle.

Para criar um controle ActiveX em código

1. Chame CREATEOBJECT( ) para criar um formulário.
2. Chame o método AddObject do formulário novo para adicionar o controle, especificando `olecontrol` como a classe. Você precisa passar o nome da biblioteca de classe do controle como o terceiro parâmetro do método AddObject.

Por exemplo, o programa seguinte cria um formulário novo e adiciona um controle delineado a ele:

```
oMeuForm = CREATEOBJECT("form")
oMeuForm.AddObject("oleOutline","olecontrol", ;
   "MSOutl.Outline")
```

Depois de criar o formulário e o controle, você pode exibir o formulário chamando seu método Show e exibir o controle configurando sua propriedade Visible como True:

```
oMeuForm.oleOutline.Visible = .T.
oMeuForm.Show
```

Alguns controles ActiveX não são basicamente criados para serem utilizados interativamente pelo usuário. Por exemplo, um controle de cronômetro não suporta métodos para interação com o usuário. Mesmo assim, você pode criar o controle em um formulário, pois o controle geralmente deixará disponível um componente padrão visível, como um ícone. Freqüentemente você não estará habilitado a alterar ou redimensionar o ícone.

Se você não quiser que seu aplicativo exiba o ícone para controles não interativos, você pode ocultar o controle definindo a propriedade Visible do seu controle de recipiente OLE como False, ou definir sua propriedade Left como um valor negativo (como –100), que o move para fora da parte visível da tela. Alternativamente, você pode colocar o controle em um formulário que nunca fica visível (isto é, para o qual o método Show nunca é chamado). Em todos os casos, você ainda pode chamar os métodos do controle como se o controle estivesse visível.

Acessando bibliotecas de vínculo dinâmico

Se a funcionalidade de que você precisa estiver disponível em um DLL, você pode vincular à biblioteca e chamar suas funções. Antes de chamar uma função DLL, você precisa determinar seu protocolo de chamada, inclusive o nome da função, o número e tipos de dados dos seus parâmetros, e o tipo de dados do seu valor de retorno.

No Visual FoxPro, você só pode utilizar os DLLs que tenham sido escritos para um ambiente de 32 bits. No entanto, se você precisa acessar um DLL de 16 bits, você pode chamá-lo utilizando funções disponíveis no FOXTOOLS.FLL. Para detalhes, consulte a **Ajuda** para Foxtools (FOXTOOLS.HLP).

Para chamar uma função DLL

1. Registre a função DLL utilizando o comando DECLARE. Os nomes de funções são sensíveis a maiúscula/minúscula.

Observação Se você especificar **WIN32API** como o nome da biblioteca, o Visual FoxPro procura a função de 32 bits Windows DLL em KERNEL32.DLL, GDI32.DLL, USER32.DLL, MPR.DLL e ADVAPI32.DLL.

2 Chame a função como faria com qualquer função do Visual FoxPro.

Por exemplo, o programa a seguir registra a função GetActiveWindow( ) do DLL USER do sistema do Windows, que exibe o identificador da janela principal do Visual FoxPro. GetActiveWindow( ) não toma parâmetro algum, mas retorna um inteiro único.

```
DECLARE INTEGER GetActiveWindow IN win32api
MESSAGEBOX(STR( GetActiveWindow() ) )
```

O DLL que contém a função que você está registrando deve estar disponível no diretório padrão, nos diretórios Windows ou System, ou no caminho do DOS.

Se a função que você quer chamar tem o mesmo nome de outra função já disponível no Visual FoxPro (uma função nativa ou uma função DLL declarada anteriormente), você pode atribuir um alias à função com o nome duplicado e, então, chamá-la utilizando o alias.

```
DECLARE INTEGER GetActiveWindow IN win32api AS GetWinHndl
MESSAGEBOX(STR( GetWinHndl() ) )
```

As funções DLL vinculadas permanecem disponíveis até que você saia do Visual FoxPro, portanto você só precisa declará-las uma vez por sessão. Se você não pretende chamar novamente as funções em um DLL, você pode utilizar o comando CLEAR DLLS para removê-las da memória e liberar recursos.

Observação A utilização de CLEAR DLLS limpa da memória todas as funções DLL declaradas.

Passando parâmetros para um DLL

Quando você registra uma função DLL, você precisa especificar o número e o tipo de dados dos seus parâmetros. Como padrão, os dados são passados por valor. Você pode forçar um parâmetro a ser passado por referência incluindo um sinal de arroba (@) na frente do parâmetro.

Em geral, as funções DLL seguem as convenções de tipos de dados utilizadas para C, que são diferentes das utilizadas no Visual FoxPro. Por exemplo, as funções DLL não suportam um tipo de dados de data ou de moeda. Se o tipo de dados que você está passando a uma função DLL for um tipo de dados não suportado pela função, você deve convertê-lo em um tipo apropriado antes de passá-lo. Por exemplo, você pode converter um dado em um formato numérico Julian utilizando comandos como os que se seguem:

```
cDate = sys(11, date())
nDate = val( cDate )
```

Algumas funções DLL requerem parâmetros mais complexos, como estruturas ou matrizes. Se a função requer um apontador para uma estrutura, você precisa determinar o layout da estrutura, em seguida emulá-la como uma seqüência no Visual FoxPro antes de passá-la ou recebê-la da função DLL. Por exemplo, a função GetSystemTime( ) do sistema Windows precisa de um apontador em uma estrutura, que consiste em oito palavras ou inteiros de 16 bits sem sinal que indique o ano, mês, dia e assim por diante. A estrutura é definida da seguinte forma:

```
typedef struct _SYSTEMTIME {
   WORD wYear ;
   WORD wMonth ;
   WORD wDayOfWeek ;
   WORD wDay ;
   WORD wHour ;
   WORD wMinute ;
   WORD wSecond ;
   WORD wMilliseconds ;
} SYSTEMTIME
```

Para passar dados entre o Visual FoxPro e a função GetSystemTime( ), você precisa criar um buffer de seqüência de 40 bytes (consistindo inicialmente de espaços) e em seguida passar o endereço

desta seqüência para a função, para que esta a preencha. Quando a seqüência retornar, você precisará dividi-la em incrementos de 2 bytes para extrair os campos individuais da estrutura. O fragmento seguinte ilustra como você poderia extrair três dos campos da estrutura:

```
DECLARE INTEGER GetSystemTime IN win32api STRING @
cBuff=SPACE(40)
=GetSystemTime(@cBuff)

tYear = ALLTRIM(STR(ASC(SUBSTR(cBuff,2)) *  ;
   256 + ASC(SUBSTR(cBuff,1))))
tMonth = ALLTRIM(STR(ASC(SUBSTR(cBuff,4)) * ;
   256 + ASC(SUBSTR(cBuff,3))))
tDOW = ALLTRIM(STR(ASC(SUBSTR(cBuff,6)) * ;
   256 + ASC(SUBSTR(cBuff,5))))
```

Para maiores informações, você pode examinar o formulário de exemplo SYSTIME.SCX em VFP\SAMPLES\SOLUTION\WINAPI. Para outros exemplos de como passar parâmetros para funções DLL, consulte o programa REGISTRY.PRG em VFP\SAMPLES\CLASSES.

Se os dados com os quais você está trabalhando no Visual FoxPro forem uma matriz, você deve fazer o loop pela matriz e concatená-la como uma seqüência única, que representa uma matriz estilo C antes de passá-la à função DLL. Se a função Windows espera valores de 16 bits ou 32 bits, você deve converter os valores para seus equivalentes hexadecimais antes de concatená-los em uma seqüência. Quando você passar a seqüência que contém os dados da matriz, o Visual FoxPro passará o endereço da variável da seqüência para o DLL, que poderá, então, manipulá-la como uma matriz. Como um exemplo disto, veja o formulário de exemplo SYSCOLOR.SCX em VFP\SAMPLES\SOLUTION\WINAPI.

Acessando uma biblioteca do Visual FoxPro

Como um DLL, uma biblioteca do Visual FoxPro (arquivo .FLL) contém funções que você pode chamar como faria com qualquer outra função. Como os arquivos .FLL são criados especificamente para serem chamados pelo Visual FoxPro, geralmente é mais fácil passar parâmetros para e das funções .FLL.

Para utilizar uma biblioteca do Visual FoxPro, você deve especificar o nome do arquivo .FLL e, em seguida, chamar a função normalmente. Ao contrário do registro das funções DLL, você não precisa registrar funções individuais dentro do arquivo FLL nem especificar informações sobre os parâmetros ou tipos de dados utilizados pela função.

Observação Se quiser utilizar uma biblioteca .FLL de uma versão anterior do Visual FoxPro, a biblioteca deve ser recompilada para trabalhar com o Visual FoxPro versão 5.0.

Para chamar uma função .FLL

1. Registre a biblioteca .FLL utilizando um comando SET LIBRARY.
2. Chame qualquer das funções da biblioteca como chamaria qualquer função.

Por exemplo, o programa a seguir chama uma função da biblioteca FOXTOOLS.FLL para determinar que tipo de unidade de disco é a unidade de disco C:.

```
SET LIBRARY TO "C:\VFP\FOXTOOLS.FLL"
? DriveType("C:")
```

Se você precisa registrar mais de um arquivo .FLL, inclua a palavra chave ADDITIVE no comando SET LIBRARY. Se não o fizer, o arquivo .FLL registrado anteriormente será limpado e substituído pelo arquivo registrado mais recentemente.

Se o nome de uma função entrar em conflito com o de outra função já disponível no Visual FoxPro, a função definida por último tem a precedência. Se o nome da função em uma biblioteca vinculada for o mesmo nome de uma função intrínseca do Visual FoxPro, a função intrínseca do Visual FoxPro terá precedência.

As funções em um arquivo .FLL permanecem disponíveis até que você saia do Visual FoxPro e,

portanto, é necessário somente registrá-las uma vez por sessão. Se você não pretende chamar novamente as funções em um arquivo .FLL, utilize RELEASE LIBRARY, RELEASE ALL, ou SET LIBRARY TO para removê-la da memória e liberar recursos.