文章编号:1004－5694(2001)01－0051－06

COM 技术及其程序设计*

华奇兵,许文波,李　琳,汪林林
(重庆邮电学院,重庆 400065)

摘　要:COM 技术是一种优秀的软件组件技术,主要说明 COM 技术体系结构、二进制标准、客户/服务器结构、基本接口等基本概念与基本原理,并详细介绍了用 C++ 进行 COM 程序设计的过程。

关键词:COM;体系结构;基本接口;C++;程序设计
中图分类号:TP311.11　　文献标识码:A

COM Technology and Its Programming

HUA Qi-bing, XU Wen-bo, LI　Lin, WANG Lin-lin
(*Institute of Computer Science & Technology, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)

Abstract: COM technology is an excellent software component technology. In this paper, the authors introduce the basic concepts and theory of COM, such as its architecture, binary standard, client/server model and basic interfaces. In addition, the authors give a detailed description of COM programming steps via C++.

Key words: component object model; architecture; basic interface; C++; programming

0　COM 概述

组件对象模型 COM (Component Object Model)是一种以组件为发布单元的对象模型。它作为一种跨平台的客户服务器系统开发技术,具有开放的体系结构。它是 ActiveX 和 OLE 2.0 的技术基础,COM 的体系结构图,如图 1 所示。

COM 的体系结构包括 COM 的核心、统一数据传输、持久存储和智能命名。其中,COM 核心包括服务控制管理员、接口代理、接口基和 COM 库。COM 核心定义了 COM 对象与其使用者(客户)如何通过二进制标准接口进行交互的规格说明。永久存储通过 IStorage 和 IStream 接口提供一个"文件内的文件系统",IStorage 接口提供构造"文件结构"

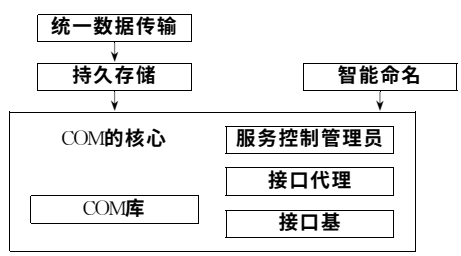

图1　COM体系结构

所需的函数,IStream 接口提供访问由 IStorage 函数指向的实际数据所需的函数,并且定义了复合文

* 收稿日期:2000－09－10

作者简介:华奇兵(1976-),男,重庆邮电学院计算机应用专业硕士研究生,主要研究方向为面向对象技术、分布计算技术及软件复用;许文波(1973-),男,重庆邮电学院硕士研究生,主要研究方向为网络管理与控制;李琳(1974-),女,重庆邮电学院硕士研究生,主要研究方向为面向对象技术、分布计算技术及软件应用;汪林林,男,教授,主要研究方向为分布式数据库与专家系统。

档的存储格式。智能命名通过 Moniker COM 对象实现 IMoniker 接口，使用户可以在以后重新连接一个指定的对象实例，并且使对象实例仍保持原来的状态；另外还提供保存它们名字和其他持久信息的机制。

1　COM 的二进制标准

在 COM 中，对象是展示一个或多个接口的功能单元，客户就是通过 COM 接口使用 COM 对象的服务，接口是一组语义相关的方法（或函数）的集合，一个对象“实现一个接口”是指对象实现了该接口中所有的成员函数，并把指向这些成员函数的指针提供给 COM。COM 对象接口用 IDL（Interface Definition Language）来描述。用 IDL 描述的 IPlane 接口的具体例子如下：

```
[object,uuid(××××××××-××××-××××-×××
×-××××××××××××)]
interface IPlane; IUnknown{
HRESULT Fly(void);
HRESULT Speed(void);
}
```

当接口用 IDL 定义之后，我们就可以用 MIDL 编译器对其进行编译，生成接口头文件（*.H）、接口代理、接口基所需要的源代码以及静态联编所需的类型库。客户通过接口头文件可以使用该接口。接口代理和接口基的源代码用来构造代理 DLL 和基 DLL.

在客户端，客户应用维护着一个接口指针，这个指针指向整个对象结构，如图 2 所示。

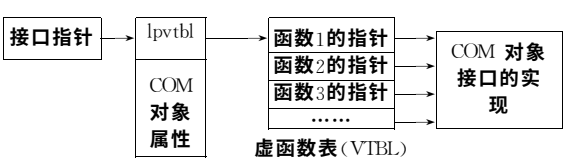

图2　COM接口的二进制标准

在对象结构中有一个指针 lpvtbl 指向一个由指向接口成员函数实现的指针所组成的数组（表）。这个数组称为“虚函数表”。在二进制水平上，VTBL 完全与程序设计语言和用来生成它的工具无关。由于 VTBL 语言无关性，VTBL 又称“二进制标准”，使得 COM 能够确保不同厂商、不同语言编写的应用之间的互操作性。

2　COM 客户/服务器结构

在 COM 中 COM 组件和 COM 组件用户之间的交互在某种意义上是基于客户/服务器模型。但 COM 客户服务器模型不仅仅是一种简单的客户/服务器模型，有时客户也可以反过来提供服务，或者服务方本身也需要其他对象的一些功能，在这些情况下，一个对象可能既是服务器也是客户。

2.1　COM 服务器

COM 服务器就是一个实现一个或多个 COM 类对象的组件。一个 COM 类对象是一个可以通过类工厂来创建的 COM 对象。每一个类对象都有一个与之相关联的类标识（CLSID），任何一个实现了 IUnknown 接口的部件都可以叫做类对象。

一个 COM 服务器往往是一个动态连接库（.DLL）或一个可执行文件（.EXE）。一个基于 DLL 的 COM 服务器叫做进程内服务器（in-proc），因为它在使用时和 COM 客户处于相同的地址空间，这样客户可以快速高效地直接调用 COM 对象。但是由于它们处于相同的地址空间，DLL 的崩溃就会破坏客户应用的地址空间，造成客户应用地址崩溃。基于 EXE 的 COM 服务器叫做进程外（out-proc）服务器，因为它拥有自己对立的进程空间。基于 EXE 的 COM 服务器和客户的地址空间是隔离的，这就使得它更加可靠。如果服务器崩溃，它不会破坏客户的地址空间。但是由于它处于一个分离的进程空间，所有的接口调用必须进行参数调整，这就使得它的性能受到很大影响。

2.2　COM 客户

一个 COM 客户（简称客户）就是一个使用 COM 服务器的应用。COM 的设计允许客户能够透明地与 COM 服务器进行交互而不管 COM 服务器是进程内服务器、本地服务器还是远程服务器。从 COM 客户方来看，所有的对象都是通过接口指针来调用的。一个指针必须是进程内的，实际上，任一对接口函数的调用总是先使用进程内的代码。如果对象在进程内，这种调用是直接的；如果对象在进程外，那么这种调用将首先调用一种称之为“proxy”的

对象，由这种对象产生对其它进程甚至是远程机器的组件的远程过程调用。

2.3 COM 库

COM 库实现了一些应用程序编程接口(API)，也定义了一些不同的接口。COM 库 API 函数向 COM 应用提供功能支持，下面列出几个常用的 API 函数。

• CoInitialize：这个函数用来初始化 COM 库。COM 库在被调用它的函数之前必须先初始化，所以这个函数必须在 COM 客户启动时被首先调用。

• CoUnInitialize：关闭 COM 库。调用这个函数以释放所有 COM 资源，关闭 RPC 连接，这个函数应该在 COM 应用退出时调用。

• CoCreateInstance：创建一个对象类的实例。这个函数内部封装其它 COM 中预定义的函数和接口方法。它依次包装了如下 API：COM API CoGetClassObject；IClassFactory 接口中的 CreateInstance 方法；IClassFactory 接口中的 Release 方法。

2.4 COM 基本接口

COM 本身预定义了一套通用接口，用来建立基于客户服务器结构的 COM 应用。在这些接口中，最基本的是 IUnknown 和 IClassFactory。IUnknown 接口是任何 COM 对象都必需的，其中的 QueryInterface 方法使客户能够访问对象的标识符，并且能够区分不同的接口。类厂组件的唯一的功能是创建其它组件，更精确地讲，某个特定的类厂将创建只同某个特定的 CLSID 相应的组件，一个类厂对象实现了 IClassFactory 接口。

3 用 C++ 实现 COM 编程

用一个 C++ 语言来实现一个进程内 COM 组件服务器及其客户，该组件对象只向外界暴露 IMath 与 IUnknown 两个接口，IMath 接口只有：加法运算 Add 和减法运算 Subtract，如图 3 所示。

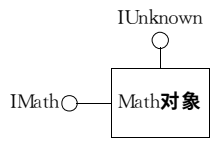

图3　Math对象的图示表示方法

这里，作者编译以下程序的环境是 WINNT4.0/SP6/VC6.

3.1 COM 服务器的创建

启动 Visual C++ 创建一个新的工程，选择 Win32 Dynamic-Link Library. 工程命名为“Server”。

(1) 创建 IMATH.H。我们需要两个新的 GUID：一个用做类 ID(CLSID)，另外两个用于组件的接口 ID(IID)。生成的方法是使用 VC6.0 提供的工具 GUIDGEN.exe，然后在 IMATH.H 中定义。

接着是以抽象类的形式声明组件的接口 IMath，完整的代码如下所示。

```
//
// imath.h
//
// {3B4227D3-5DB8-11d4-8DE2-0080C82F3814}
DEFINE_GUID(CLSID_Math,
0x3b4227d3, 0x5db8, 0x11d4, 0x8d, 0xe2, 0x00, 0x80,
0xc8, 0x2f, 0x38, 0x14);
// {3B4227D4-5DB8-11d4-8DE2-0080C82F3814}
DEFINE_GUID(IID_IMath,
0x3b4227d4, 0x5db8, 0x11d4, 0x8d, 0xe2, 0x00, 0x80,
0xc8, 0x2f, 0x38, 0x14);

class IMath:public IUnknown
{
public:
    STDMETHOD(Add)( long, long, long * ) PURE;
    STDMETHOD(Subtract)( long, long, long * ) PURE;
};
```

(2) 声明组件及类厂。下面我们声明组件类及该组件的类厂，创建一个新的文件名为 MATH.H，完整的代码如下：

```
// math.h
#include "imath.h"
// 该全局变量跟踪 DLL 中的所有的组件实例的数目
extern long g_lObjs;
//该全局变量用于跟踪对 IClassFactory::LockServer 的调用
extern long g_lLocks;

class Math : public Imath
{
protected:
    //Reference count
    long m_lRef;
public:
    Math();
```

```
    ~ Math();
public:
    //IUnknown
    STDMETHOD(QueryInterface( REFIID, void * * ));
    STDMETHOD_(ULONG, AddRef());
    STDMETHOD_(ULONG, Release());
    //IMath
    STDMETHOD(Add) ( long, long, long * );
    STDMETHOD(Subtract) ( long, long, long * );
};

class MathClassFactory : public IClassFactory
{
protected:
    long m_lRef;
public:
    MathClassFactory();
    ~ MathClassFactory();
    //IUnknown
    STDMETHOD( QueryInterface(REFIID, void * * ));
    STDMETHOD_(ULONG, AddRef());
    STDMETHOD_(ULONG, Release());
    //IClassFactory
    STDMETHOD(CreateInstance)(LPUNKNOWN, REFIID, void * * );
    STDMETHOD(LockServer)(BOOL);
};
```

(3) 实现组件及类厂。创建 MATH.CPP 文件用于组件及类厂的实现,完整代码如下:

```
// Math.cpp
#include <windows.h>
#include "math.h"
// Math class implementation
Math::Math()
{
    m_lRef=0;
    //Increment the global object count
    InterlockedIncrement(&g_lObjs);
}
// The destructor
Math::~ Math()
{
    // Decrement the global object count
    InterlockedDecrement( &g_lObjs );
}

STDMETHODIMP Math::QueryInterface( REFIID riid, void * * ppv )
{
    * ppv = 0;
    if(riid==IID_IUnknown )
        * ppv=(IMath * )this;
    else if(riid==IID_IMath )
        * ppv=(IMath * )this;
    if( * ppv){
        AddRef();return( S_OK );
    }
    return (E_NOINTERFACE);
}

STDMETHODIMP_(ULONG) Math::AddRef()
{
    return InterlockedIncrement( &m_lRef );
}
STDMETHODIMP_(ULONG) Math::Release()
{
    if ( InterlockedDecrement( &m_lRef ) == 0 ){
        delete this;return 0;}
    return m_lRef;
}
STDMETHODIMP Math::Add(long lOp1,long lOp2,long * pResult )
{
    * pResult=lOp1+lOp2; return S_OK;
}
STDMETHODIMP Math::Subtract(long lOp1,long lOp2,long * pResult )
{
    * pResult=lOp1-lOp2; return S_OK;
}
MathClassFactory::MathClassFactory()
{
    m_lRef=0;
}
MathClassFactory::~MathClassFactory()
{
}
STDMETHODIMP MathClassFactory::QueryInterface(REFIID riid,oid** ppv)
{
    * ppv=0;
    if(riid==IID_IUnknown||riid==IID_IClassFactory )
        * ppv=this;
    if( * ppv){
```

```
        AddRef();return S_OK;}
    return( E_NOINTERFACE );
}
STDMETHODIMP_(ULONG) MathClassFactory::AddRef()
{
    return InterlockedIncrement( &m_lRef );
}
STDMETHODIMP_(ULONG) MathClassFactory::Release()
{
    if(InterlockedDecrement(&m_lRef)==0){
        delete this;
        return 0;
    }
    return m_lRef;
}
STDMETHODIMP MathClassFactory::CreateInstance
(LPUNKNOWN pUnkOuter, REFIID riid, void * * ppvObj)
{
    Math * pMath;
    HRESULT hr;
    * ppvObj=0;
    pMath=new Math;
    if(pMath==0)
        return(E_OUTOFMEMORY);
    hr=pMath->QueryInterface(riid,ppvObj);
    if(FAILED(hr))
        delete pMath;
    return hr;
}
STDMETHODIMP MathClassFactory::LockServer ( BOOL
fLock )
{
    if(fLock)
        InterlockedIncrement( &g_lLocks );
    else
        InterlockedDecrement( &g_lLocks );
    return S_OK;
}
```

（4）定义并实现输出函数。由于要生成进程内服务器，所以应当输出被客户访问的函数，对于进程内服务器来说，需要输出 DllGetClassObject **和** DllCanUnloadNow **两个函数。输出定义在** SERVER.DEF **文件中，完整代码如下：**

```
; Server.def : Declares the module parameters for the DLL.

LIBRARY "SERVER"
DESCRIPTION 'SERVER Windows Dynamic Link Library'

EXPORTS
    ; Explicit exports can go here
    DllGetClassObject    @1
    DllCanUnloadNow    @2
```

为了实现输出函数，创建一个新文件，命名为 SERVER.CPP. **完整的代码如下：**

```
// server.cpp : Defines the initialization routines for the DLL.
#include <windows.h>
#include <initguid.h>
#include "math.h"
long g_lObjs = 0;
long g_lLocks = 0;
// This entry point provides COM a standard way of accessing
the housing's
// class factories.
STDAPI DllGetClassObject( REFCLSID rclsid, REFIID riid,
void * * ppv )
{
    HRESULT hr;
    MathClassFactory *pCF;
    // Make sure the CLSID is for our Math component
    if ( rclsid ! = CLSID_Math )
        return( E_FAIL );
    pCF = new MathClassFactory;
    if ( pCF == 0 )
        return( E_OUTOFMEMORY );
    hr = pCF->QueryInterface( riid, ppv );
    // Check for failure of QueryInterface
    if ( FAILED( hr ) )
        delete pCF;
    return hr;
}
STDAPI DllCanUnloadNow(void)
{
    if ( g_lObjs || g_lLocks )
        return( S_FALSE );
    else
        return( S_OK );
}
```

（5）编译程序，生成一个 DLL **文件。**

（6）注册服务器。建立 SERVER.REG，**注册方法如** regedit/s server.reg. SERVER.REG **内容如下：**

```
REGEDIT
```

HKEY_CLASSES_ROOT\QBHUA.Math.1 = QBHUA Math Component

HKEY_CLASSES_ROOT\QBHUA.Math.1\CLSID = {3b4227d3-5db8-11d4-8de2-0080c82f3814}

HKEY_CLASSES_ROOT\QBHUA.Math = QBHUA Math Component

HKEY_CLASSES_ROOT\QBHUA.Math\CurVer = QBHUA.Math.1

HKEY_CLASSES_ROOT\QBHUA.Math\CLSID = {3b4227d3-5db8-11d4-8de2-0080c82f3814}

HKEY_CLASSES_ROOT\CLSID\{3b4227d3-5db8-11d4-8de2-0080c82f3814} = QBHUA Math Component

HKEY_CLASSES_ROOT\CLSID\{3b4227d3-5db8-11d4-8de2-0080c82f3814}\ProgID= QBHUA.Math.1

HKEY_CLASSES_ROOT\CLSID\{3b4227d3-5db8-11d4-8de2-O080c82f3814}\VersionIndependentProgID = QBHUA.Math

HKEY_CLASSES_ROOT\CLSID\{3b4227d3-5db8-11d4-8de2-0080c82f3814}\InprocServer32 = c:\server\debug\server.dll

3.2 COM 客户的实现

服务器注册后,客户就可以编程使用。一般遵循以下几个步骤:

① **初始化** COM **库** CoInitialize(NULL);

② **创建** COM **组件实例;**

③ **检查第二步调用** CoCreateInstance **的返回值;**

④ **使用组件提供的服务;**

⑤ **调用** CoUninitialize()**释放** COM **所有资源。**

下面为一个客户程序的实现:

```
// Client.cpp
#include <windows.h>
#include <tchar.h>
#include <iostream.h>
#include <initguid.h>
#include "..\server\imath.h"
int main()
{
    cout << "initializing COM" << endl;
    if (FAILED( CoInitialize(NULL)))
    {
        cout <<"Unable to initialize COM" << endl;
        return -1;
    }
    // Get the class factory for the Math class
    IClassFactory * pCF;
    HRESULT hr = CoGetClassObject(CLSID_Math,
        CLSCTX_INPROC,
        NULL,
        IID_IClassFactory,
        (void * * )&pCF);
    if ( FAILED( hr ))
    {
    cout<<"Failed to GetClassObject server instance. HR=" << hr << endl;
        return -1;
    }
    IUnknown * pUnk;
    hr = pCF->CreateInstance( NULL, IID_IUnknown, (void * * ) &pUnk );
    // Release the class factory
    pCF->Release();
    if ( FAILED( hr ))
    {
        cout <<"Failed to create server instance. HR =" << hr << endl;
        return -1;
    }
    cout <<"Instance created" << endl;
    IMath * pMath = NULL;
    hr = pUnk->QueryInterface( IID_IMath, (void * * ) &pMath );
    pUnk->Release();
    if ( FAILED( hr ))
    {
        cout <<"QueryInterface() for IMath failed" << endl;
        CoUninitialize();
        return -1;
    }
    long result;
    pMath->Multiply( 100, 8, &result );
    cout <<"100 * 8 is " << result << endl;
    pMath->Subtract( 1000, 333, &result );
    cout <<"1000 - 333 is " << result << endl;
    cout << "Shuting down COM" << endl;
    CoUninitialize();
    return 0;

}
```

(下转 61 页)

(上接 56 页)

4 结束语

COM 技术作为一种优秀的软件模型,不仅提供程序与程序间通信的标准,而且 COM 可以使我们改变传统的程序设计方法。COM 结合了对象技术和组件技术两种特性,对象特性使得应用系统的设计和实现更加符合现实世界;组件特性使应用系统可以充分发挥组件的优势,以适应现代应用的需要。COM 是个开放的组件标准,它有很强的扩充和扩展能力,随着 COM/DCOM 技术与 MTS、MSMQ 的结合与发展,已经形成了新的组件技术标准——COM+,但 COM 仍然是其核心的技术之一。

参 考 文 献

[1] MICROSOFT CORPORATION. The Component Object Model Specification [S]. 1998.

[2] 潘爱民. COM 原理与应用[M]. 北京:清华大学出版社,1999.

[3] 杨秀章译,COM 技术内幕[M]. 北京:清华大学出版社,1997.