维普资讯 http://www.cqvip.com

基于 Linux 的 ActiveX 控件运行机制的设计与实现*)

石 磊[1,2] 杨维康[1] 杨孟辉[1] 阳 昕[1] 向建华[3]
(清华大学计算机科学与技术系 北京 100084)[1] (武警工程学院基础部 西安 710086)[2]
(北京交通大学计算机科学与技术系 北京 100044)[3]

摘 要 当前,ActiveX 控件在互联网和桌面上应用非常广泛,但是尚不能在 Linux 平台上无缝运行。为解决这个问题,本文提出了一种由两层功能层组成的中间件机制,这两层功能层分别是操作系统兼容层和 AtiveX 控件机制支撑层。在该机制中,操作系统兼容层消除了 Windows 和 Linux 两种操作系统的差别,ActiveX 控件机制支撑层中添加了支持 ActiveX 控件的 COM 机制。提出了以 XML 格式文件实现注册表的组件注册方法,提供了 Linux 上直接运行 ActiveX 控件的"控件-容器"机制的必要条件,最后实现了一个在 Linux 上直接运行轻量级 ActiveX 控件机制的原型系统。

关键词 ActiveX 控件,Linux,跨平台,COM

Design and Implementation of ActiveX Control Runtime Mechanism Based on Linux

SHI Lei[1,2] YANG Wei-Kang[1] YANG Meng-Hui[1] YANG Xin[1] XIANG Jian-Hua[3]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)[1]
(Department of Foundation, Engineering College of Armed Police Force, Xi'an 710086)[2]
(Department of Computer Science and Technology, Beijing Jiaotong University, Beijing 100044)[3]

Abstract ActiveX control is used widely in Internet and desktop fields nowadays, whereas it couldn't run on Linux directly. To solve the problem, this paper presented a middle-ware mechanism consisting of two functional layers, which were operating system compatible layer and ActiveX control mechanism supporting layer. The difference between Windows and Linux was eliminated in operating system compatible layer. The supporting and improvement of COM mechanism as ActiveX control's infrastructure was added, an implementation of registry using XML file was presented and the necessary supporting of "Control-Container" mechanism for Active control running was provided. A prototype of light weight ActiveX control mechanism running on Linux was implemented.

Keywords ActiveX control; Linux, Cross-platform, COM

1 引言

当前,ActiveX 控件因其具有可以被 C++、Dephi、Java、C# 等许多编程语言或应用程序(组件容器)调用,可以高效地实现软件组件的集成、提高软件的可重用性、便于软件动态维护和升级等优点,被广泛应用在桌面和网络应用中。尤其是 ActiveX 控件具有较高的安全性、代码保密性等特点,在网上银行、电子支付等领域有着不可取代的地位。但是由于 ActiveX 控件是在 Windows 平台上开发的,至今尚不能无缝地运行在 Linux 平台上。而 Linux 操作系统作为一种源码开放的自由软件操作系统在服务器和嵌入式设备上占有率稳步上升,在桌面领域也有相当多的用户。但是 Linux 对 ActiveX 控件的不兼容性无法满足广大 Linux 用户对于 ActiveX 控件应用的需求,也限制了 ActiveX 控件技术的发展和 Linux 操作系统的进一步推广。因此将包括 ActiveX 在内的 Windows 平台上的应用完整地迁移到 Linux 上,弥补 Linux 的不足,无论是对 Linux 的推广,还是对于无须再重新开发新的应用取代现有的系统、节约开发成本等方面都有着重要的现实意义。

目前对这种跨平台软件的二进制组件兼容的研究已经有一些研究成果。一类是"系统级"[1]虚拟机技术,主要的代表技术有 VMware 和 Xen 等。这类技术可以在裸机或操作系统上虚拟或半虚拟化若干个完整的计算机,并在其中运行相同或不同操作系统上的程序。它们的体系结构允许"客户"操作系统与"主"操作系统共存于一台计算机上,并且可以依赖"主"操作系统提供低层的设备支持。VMware 的虚拟机模式可以分为两个部分:一部分通过虚拟机监视器完全虚拟 CPU 的执行,另一部分利用"主"操作系统提供的设备驱动程序直接运行应用程序[2]。Xen 则是虚拟出了一个抽象硬件层,这个抽象层实际是个"部分 linux 内核"。其他的"客户"操作系统都运行在这个硬件抽象层上[3]。另一类是"进程级"[1]虚拟机技术,主要的代表有开源项目 WINE。WINE 利用 Windows 模拟器技术,通过在 Linux 上建立针对 Windows API 进行仿真的系统兼容层,即把 Windows 下的 DLL 格式映射为 Linux 下 ELF 格式的共享库文件,用一个 ELF 格式的 WINE Server 模拟 Windows 内核,进而运行 Windows 程序[5]。虚拟机技术在过去十几年中广泛地应用于各类科学实验和商业产品[6]。

以上几种解决方案中,"系统级"虚拟机有的对硬件配置的要求高,而且其在资源优化和性能上有较大缺陷,运行时对 CPU 及系统资源的占用率高,比如 Vmware;有的"客户"操作系统必须修改其内核,以适应硬件抽象层[4],例如 Xen,而且

*)国家 863 项目"面向普适计算的自适应软件集成环境"(2006AA01Z198);清华大学信息科学与技术国家实验室基础研究基金项目"面向普适计算的构件化基础软件平台"。石 磊 硕士生,主要研究方向:操作系统、组件技术;杨维康 博士,研究员,硕士生导师,研究方向:计算机体系结构;杨孟辉 博士后,研究方向:中间件技术;阳 昕 硕士生,研究方向:操作系统、组件技术。

维普资讯 http://www.cqvip.com

Xen还需要实现了VT技术的CPU等硬件的支持。“进程级”虚拟机中，WINE的系统兼容层技术是使用Linux本身的系统调用访问硬件，不存在以仿真的方式执行I/O操作，通过对Windows API仿真彻底摆脱了Windows系统，而且是源码开放的自由软件项目，有利于学习和开发。但是WINE对COM机制及ActiveX控件的支持还不够，导致ActiveX控件无法直接运行。本文基于WINE提供的平台兼容层，设计并实现了一种在Linux上运行ActiveX机制的轻量级中间件模型，以较小的开发成本实现了Linux上运行ActiveX控件应用的需求。

本文以下内容是这样安排的：第2节介绍了ActiveX控件的工作机理，第3节介绍了ActiveX控件在Linux上运行机制模型的设计，第4节实现了一个在Linux上直接运行轻量级ActiveX控件机制的原型，最后对全文做总结。

2 ActiveX控件的工作机制

ActiveX控件是ActiveX组件的成员之一，是一种特殊的软件组件对象，它的工作机制是基于COM(Component Object Model，组件对象模型)和OLE(Object Linking and Embedding)的“控件--容器”机制。在使用ActiveX控件的对话当中，客户应用程序及包含着链接或嵌入对象的应用程序称为容器，控件负责处理容器对特定数据的请求，也称为对象服务器。ActiveX控件机制的实现是和Windows系统紧密相关的，包括控件机制和容器机制，如图1所示[7]。控件和容器都通过支持特定的接口实现特定的功能。ActiveX控件是一个带有现场激活功能的进程内对象服务器，有自己的固有属性和事件，它通过支持附加接口实现包括自动化方法、与容器进行通信、获得环境属性、属性页、属性变化通知和激发事件等功能。ActiveX控件的容器实际是一个OLE容器，并且实现了相应的接口来支持ActiveX控件，成为ActiveX控件的容器。它需要实现保存控件的属性状态等信息、载入控件、获得控件属性变化的信息、与控件交互等功能。

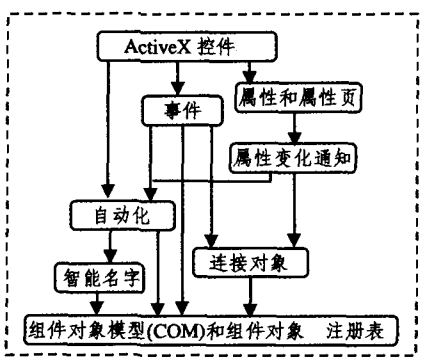

图1 ActiveX控制机制基本结构图

3 ActiveX控件在Linux上运行模型的设计

ActiveX控件无法直接在Linux平台上直接运行的原因在于操作系统的平台差异性和Linux平台上不支持软件组件。操作系统是应用程序的运行平台，提供系统调用和函数链接库作为程序的调用接口。Windows和Linux这两种操作系统在内核中的进程/线程调度机制、资源管理策略、图形支撑等运行时接口内部机制方面存在着较大的差异，两者几乎不直接支持对方的应用程序接口。显然，如果没有第三方组件库函数的支持，Linux系统本身也不支持COM机制以及COM机制基础之上的ActiveX技术及其实现机制。这些原因造成Windows上开发的包括ActiveX在内的二进制可执行文件无法直接在Linux上运行。根据以上关于ActiveX控件无法在Linux上运行原因的分析，本文提出了一个提供上述机制的方案，如图2所示。这个机制可分为两个大的层次：(1)操作系统之间的平台兼容层；(2)ActiveX控件机制支撑层。通过这两个层，分别解决消除操作系统的差异性和支持ActiveX机制这两个问题。

3.1 操作系统平台兼容层的结构

操作系统之间的平台兼容层的功能是消除两个系统运行时接口(包括系统调用和图形显示机制等)的差异、提供进程/线程管理调度、提供Windows消息机制在Linux上的实现等。这一层的实现主要是基于开源项目WINE及其提供的接口。WINE是一个Linux上的Windows仿真层，这层的主要结构如图2所示[5]。① Win32核心DLL：由WINE以Linux上共享库形式重写的Windows底层核心的动态链接库，如NTDLL. DLL、KENERL32. DLL、GDI32. DLL、USER32. DLL等。② WINE Server：管理WINE启动的各类进程，实现Windows进程/线程及其管理调度与Linux进程/轻量级进程及其管理调度的转换，实现Windows API在Linux上的转换和Windows消息机制等功能。③ PE文件加载器：加载并解析Windows上PE格式的可执行文件，提取其中的文件头部、节表、重定位等关键信息，分析其与其他动态链接库的依赖关系，修改PE文件的导入表，使导入函数转向WINE实现的Windows的核心DLL等。④ WINE驱动库：在Linux内核上建立了一个提供Windows上NTDLL. DLL和KERNEL32. DLL功能的代理层，以满足使用Windows驱动程序的需求，完成Windows驱动API与Linux驱动接口的粘合。

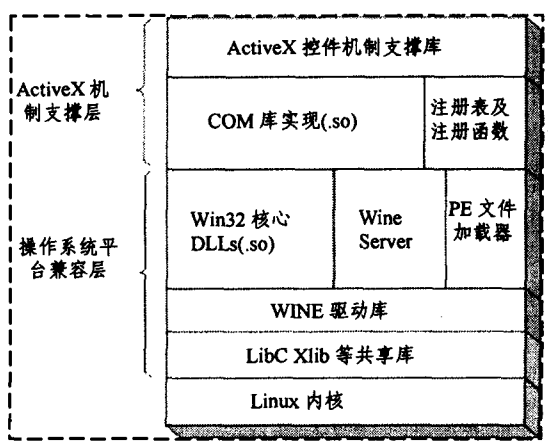

图2 系统结构图

3.2 ActiveX机制支撑层的结构

AcitveX控件以COM为基础，充分发挥了COM的优势，使得应用程序具有极强的可交互性，可以说COM组件技术是ActiveX控件技术的基础。所以，如果在Linux上运行ActiveX组件，必须首先在Linux上实现COM机制。COM机制在Linux上实现主要包括COM库函数、接口结构和注册表机制。

维普资讯 http://www.cqvip.com

同时，ActiveX 构件不仅是 COM 对象，而且它是在控件容器内运行的，有自己独特的“容器-控件”机制，如图 1 所示。主要有控件的属性、方法，对容器的属性变化通知机制、容器与控件属性变化的批准或忽略机制、对控件事件的响应机制以及通过接口实现自动化(Automation)、命名与解析机制等。经过以上的分析，将 ActiveX 机制支撑层分为 COM 机制层和 ActiveX 控件机制支撑库层两个层次。

4 ActiveX 控件在 Linux 运行机制的实现

4.1 COM 机制层的实现

COM 是微软提出的组件标准，它不仅定义了组件程序之间交互的标准，也提供了组件程序运行所需要的环境。一个组件程序可以包含一个或多个组件对象，COM 是以对象为基本单元的模型。当程序与程序进行通信时，通信的双方应该是组件对象(COM 对象)，而组件程序则是组件对象的代码载体。

COM 机制十分复杂，图 3 描述了一个简单的 COM 机制模型：组件在系统注册后，客户程序通过调用系统提供的 COM 库函数，通过“类工厂”(实际为“对象工厂”)创建 COM 对象(图中序号为程序执行的顺序)。

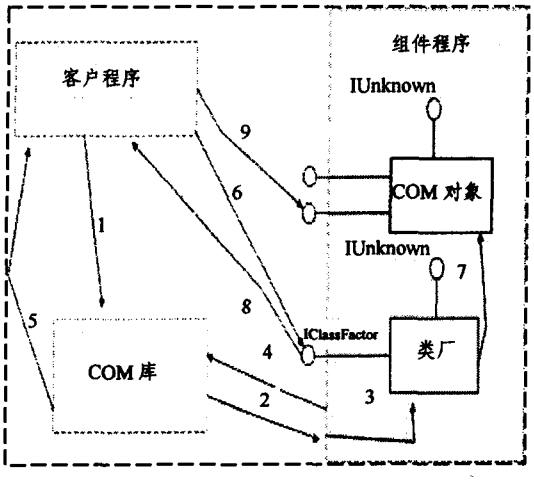

图 3 利用 COM 机制生成构件过程的示意图

虽然 COM 的实现与 Windows 操作系统平台密切相关，COM 的实现部分(COM 库函数)用到了 Windows 的一些系统特性，如动态连接库、注册表等，但实际上 COM 是一个与平台无关的模型[8]。

4.1.1 Linux 上实现 COM 的关键问题

在非 Windows 平台上实现 COM 机制要解决如下三个问题。第一，组件的存在形式，即以何种方式实现 Windows 上的动态链接库；第二，组件的注册，即如何实现 Windows 上的注册表机制；第三，客户与组件服务器之间的通信方式，包括完成运行时发现接口和组件生存期管理等功能。

(1) 组件的存在形式：因为组件和客户程序运行于不同的模块中，客户程序还要动态的加载和卸载组件，所以组件在 Windows 采用动态链接库(. dll)形式。在 Linux 上组件可以采用共享库(. so)形式，同时利用 Linux 提供的操作共享库函数，可以完成动态链接库的加载和卸载功能。

(2)注册表：Windows 注册表是一个树状的层次结构，根节点下包含了一些键(key)和一些值(value)，每个键又可以包含子键和值，按照分层原则，形成树状层次结构。Windows 注册表在根节点下包含了六个键节点，但是对于 COM 来说，只用到了其中之一：HKEY_CLASSES_ROOT，在 HKEY_CLASSES_ROOT 键下，最主要的是 CLSID 子键，该子键下，列出了当前机器上的所有组件信息，如图 4 所示。

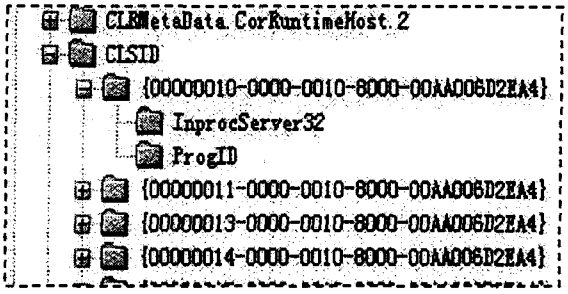

图 4 Window 注册表

Windows 上的注册表结构复杂，功能强大，在 Linux 上完全实现有很多困难，但考虑到 COM 只需要完成组件信息的注册的实际需要，可以用文件、数据库或目录服务来代替注册表。在本文中采用 XML 文件 registry. xml 来模拟注册表的实现，如下所示：

```
〈Component〉
    〈Class〉Class1〈/Class〉
    〈CLSID〉000300-…-0046〈/CLSID〉
     〈Interface〉Interface1(IN arg1, OUT arg2)
     〈/Interface〉
〈/Component〉
```

(3)客户与组件的通信方式：Windows 上 COM 中的对象通信方式组件和客户之间通过 Vtable 进行通信，而所有的接口都是 C++的纯虚基类。这套面向接口的机制，保证了客户能够在运行时发现接口和组件的引用计数、生存期管理机制。在本文中采用 C/C++的机制实现接口，下面是 IUnknown 接口的定义：

```
#ifndef_cplusplus //Definition in C
typedef struct IUnknown Vtbl IUnknown Vtbl;
struct IUnknown{
   IUnknownVtbl* lpVtbl;
};
struct IUnknownVtbl {
   HRESULT (*pQueryInterface)
   (IUnknown* this, REFIID riid, LPVOID* ppvObj);
   ULONG (*pAddRef)(IUnknown* this);
   ULONG (*pRelease)(IUnknow* this);
}
#else // Definition in C++
class IUnknown
{
  virtual HRESULT QueryInterface
  (IUnknown* this, REFIID riid, LPVOID* ppvObj);
  virtual ULONG AddRef(IUnknown* this );
  virtual ULONG Release(IUnknow* this);
}
#endif
```

4.1.2 COM 库的实现

COM 组件把它包含的 COM 类和接口的信息在系统的注册表中进行注册，这样客户程序无须知道服务端组件的名称和存储路径，只需要知道它的 CLSID 或 ProgID，就可以通过 COM 库函数间接地访问注册表。COM 库函数主要包括如下层次：①用来创建 COM 对象的底层 API，包括声明注册例程和对象创建例程；②组件程序的定位函数，它通过注册表来确定实现了该 COM 类的服务器地址；③透明的 RPC 实现函数，支持在本地或远程服务器中运行的对象；④控制进程空间中内存分配和释放的标准机制的 API。表 1 中列出了一些常用的 COM 库函数。

维普资讯 http://www.cqvip.com

表 1　COM 库主要函数

函数名	功　能
CoInitialize	COM 库初始化
CoUninitialize	COM 库功能服务终止
CoFreeUnusedLibraries	卸载进程中所有不需要的组件
CLSIDFromProgID	字符形式的对象标识转换为 CLSID
ProgIDFromCLSID	CLSID 结构转换为字符串形式的对象标识
CLSIDFromString	字符串形式的 CLSID 转换为 CLSID 结构
StringFromCLSID	CLSID 结构转换为字符串形式 CLSID
IIDFromString	字符串形式的 IID 转换为 IID 结构
StringFromIID	IID 结构转换为字符串形式的 IID
CoGetClassObject	获取对象的类厂
CoCreateInstance	创建 COM 对象
CoTaskMemAlloc	分配内存
CoTaskMemRealloc	重新分配内存
CoTaskMemFree	释放内存
RegSvr32	注册 COM 组件

这里以 DllGetObject 函数的伪码实现为例，说明 COM 库函数的实现(这里的 COM 构件只考虑到进程内调用的情况，暂不考虑进程外调用)。

```
HRESULT CoGetClassObject(
    REFCLSID rclsid, DWORD dwClsContext,
    void * pServerInfo,
    REFIID iid, LPVOID * ppv)
{
  if (GetCacheRegClassObj(rclsid,
  dwClsContext,&regClassObject)==S_OK){
  //首先查找该构件是否已存在构件缓存中，
  //如果有，则不必去注册表中查找
  hr = QueryInterface(regClassObject,
     iid, ppv);
  Release(regClassObject);
  return hr;
  }
  // 是否是进程内组件
  if (CLSCTX_INPROC_SERVER&dwClsContext){
    hr = OpenRegForCLSID(rclsid,
    wszInprocServer32, KEY_READ, &hkey);
    if (FAILED(hr)){
      if (hr == NOTARGETCLS)
        ERROR(COMPONENTNOREG);
    else
        ERROR(COMPONENTNOREGASINPROC);
    }
    else {
      //在注册表获得该组件
      hr=GetClassObject(hkey,rclsid,iid,ppv);
      RegCloseKey(hkey);
      return hr
    }
  }
  //创建该构件对象失败
  if (FAILED(hr)){
    ERROR(COMPONENTNOCREATE);
    return hr;
  }
}
```

4.2　ActiveX 机制支撑库层的实现

ActiveX 机制支撑库层的作用是通过实现 ActiveX 控件必须支持的接口来完成控件和容器所必须完成的功能。ActiveX 控件是特殊的 COM 对象，它不但要支持所有 COM 对象都必须支持的接口，比如 IUnknown，而且因为 ActiveX 控件必须在被称为“容器”的独立软件中运行(例如 IE、Word 等)，所以 ActiveX 控件和容器都必须支持一些特定的接口协议。这就意味着将 ActiveX 控件在 Linux 上的运行机制，也将会由控件机制和容器机制两部分组成。

4.2.1　ActiveX 控件机制的结构

ActiveX 控件和容器需要具备如下特征：①属性和方法：ActiveX 控件必须提供属性的名称、方法的名称和参数，通过这项机制容器可以存取和改变 ActiveX 控件的属性参数。②事件：ActiveX 控件由这项机制通知容器在 ActiveX 控件中发生的事件，比如属性参数的改变，用户按下鼠标左键等。③存储：容器由这项机制通知 ActiveX 控件存储和提取有关信息数据等。

为了提供这些特征，ActiveX 控件及容器应该满足自注册、自动化、连接对象、事件和属性变化通知等机制，这些机制的层次如图 5 所示。

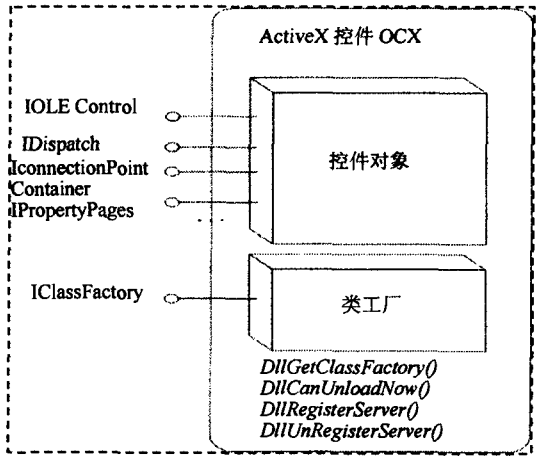

图 5　AnctiveX 控件的基本结构

4.2.2　ActiveX 控件机制的实现

ActiveX 控件是多种技术的集成，包含 COM 和 OLE 的许多技术。ActiveX 机制也是通过实现 ActiveX 继承的 COM 和 OLE 接口来实现的，一个基本的 ActiveX 控件机制包括：

(1)自注册。ActiveX 必须通过实现 DllRegisterServer 和 DllUnregisterServer 函数来支持自注册。ActiveX 控件必须注册嵌入式对象和自动化服务器的所有标准注册入口。注册函数算法如下：

```
HRESULT DllRegisterServer(void)
{
  // 注册构件
  hr = RegCoClass(coclass_list);
  if (SUCCEEDED(hr))
  // 注册接口
    hr = RegInterface(interface_list);
  return hr;
}
```

(2)自动化(Automation)。组件软件的脚本用于按需生成小程序对象，并让这些小程序对象知道怎样响应用户的交互作用。这种类型的嵌入，被称为自动化或自动控制。方法和属性是自动化对象的两个基本特征，方法是指自动化对象所提供的功能，属性是指自动化对象的数据特征。自动化对象的核心是 IDispatch 接口，每个自动化对象都要实现 IDispatch 接口。IDispatch 接口的定义如下：

```
class IDispatch : public Iunknown
{
  public:
  virtual HRESULT
  GetTypeInfoCount(UINT * pctinfo) = 0;
  virtual HRESULT
  GetTypeInfo(UNIT iTinfo,LCID lcid,
   ITypeInfo * * ppTInfo) = 0;
  virtual HRESULT
  GetIDsOfNames(REFIID riid,LPOLESTR
    * rgszNames, UINT cNames,LCID lcid,
    DISPID * rgDispId) = 0;
  virtual HRESULT
  Invoke(DISPID dispIdMember,
   REFIID riid,LCID lcid,WORD wFlags,
   DISPPARAMS * pDispParams,VARIANT *
   pVarResult,EXCEPINFO * pExcepInfo,
   UINT * puArgErr) = 0;
}
```

其中，前两个成员函数 GetTypeInfoCount 和 GetTypeInfo 用来处理组件对象的类型信息。组件对象的类型信息是指它与外界交互的必要信息，包括组件的 CLSID、它支持的

维普资讯 http://www.cqvip.com

接口 ID、接口的每个成员函数和返回值类型等。类型信息是客户程序与组件对象之间通信的基础，不管通过什么样的方式，客户程序必须在访问组件对象之前得到基本的类型信息，比如客户程序只有知道了 CLSID 才能创建对象。客户程序在获取类型信息之前会先调用 GetTypeInfoCount 函数判断对象是否提供类型信息。GetTypeInfo 函数返回对象的类型信息。lcid 参数指定类型信息的本地化标识，ppTInfo 输出参数指向一个类型信息对象的 ItypeInfo 接口指针，以后客户程序可以通过此类型信息对象得到该自动化接口的所有类型信息。成员函数 Invoke 是个关键函数，客户程序必须通过 Invoke 函数才能访问属性或方法，它是自动化对象的命令翻译器。这个函数的第一个参数 dispIdMember 给出了分发 ID，根据这个参数执行有关的属性访问函数或者方法执行函数。其中的 wFlags 参数指示是访问属性还是调用方法。

(3) 属性变化通知。对于 ActiveX 控件，通常它被用于控件应用的状态，所以属性变化通知机制非常有意义。比如，ActiveX 控件的颜色属性被用于控件应用的颜色，那么当 ActiveX 控件的颜色改变时，系统可以刷新显示以便反映出用户的最新选择。实现属性变化通知的接口是 IpropertyNotifySink。其定义如下：

```
class IpropertyNotifySink:public Iunknown
{
  HRESULT OnChanged(DISPID dispid)=0;
  HRESULT OnRequestEdit(DISPID dispid)=0;
};
OnChanged 函数的算法实现为：
void Dispatch::OnChanged(DISPID dispid)
{
//获得改属性的名称
String name = propertyName(dispid);
//获得当前控件这个属性的值
union propertyInfo=ControlPropertyInfo(name,
        Property(name));
//将新的属性值通知容器
SendProperty(propertyInfo);
}
```

4.2.3 ActiveX 控件容器机制的实现

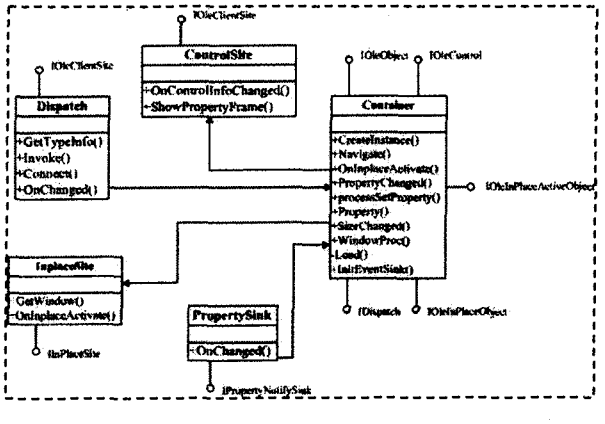

图 6 控件容器实现的主要对象及结构

ActiveX 的容器之间的通信涉及到许多方面，它们之间的通信不只是简单的客户-服务器的关系，因此容器为每个 ActiveX 控件提供了一个称为控件站点的对象，ActiveX 控件通过控件站点与容器进行交互。控件站点对象把容器的环境属性暴露给 ActiveX 控件，ActiveX 控件取得环境属性以便更好地集成到包容器程序中。根据 ActiveX 控件的特性及容器本身的要求，容器必须能够完成创建窗口(通过控件的类厂对象)、暴露环境属性给控件(通过自动化机制)、实地激活机制以及事件处理机制[9,10]。在实现中，分别用容器类实现创建窗口、暴露环境属性等功能，设置一个监听器(PropertySink)接收控件激发的事件，然后提交给容器处理，用控件站点(ControlSite)和实地站点(InplaceSite)共同完成 ActiveX 控件与容器之间的交互，用一个 Dispatch 类配合控件实现的 IDispatch 接口共同完成自动化的功能，如图 6 所示。

4.3 在实现的原型中运行 ActiveX 控件

本文已经开发了一个在 Linux 上运行简单 ActiveX 控件(日历时间显示程序)的原型系统。首先在 Fedora 4 上编译原型系统的所有代码，然后，将一个在 Windows 上编写的利用 ActiveX 控件制作的日历的时间显示程序，分别在 Windows XP 和 Fedora 4 上运行，得到如图 7 所示的结果。

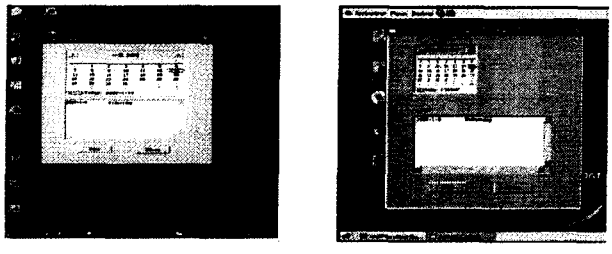

图 7 ActiveX 控件的应用程序在 Windows 和 Linux 上运行的示意图

结束语 ActiveX 控件在 Linux 上运行机制的模型，是利用虚拟机技术的一种中间件实现。从分析 ActiveX 控件和容器的结构、基本功能入手，提出了 ActiveX 控件在 Linux 平台运行需要提供的机制，这个机制的基础是 COM 机制，本文在 Linux 上实现了轻量级的 COM 机制，进而实现了注册表、必须的 OLE 机制的接口等，最后实现了 ActiveX 控件机制的简单原型。ActiveX 机制庞大复杂，涉及的技术也很多，全方位的实现有很大困难，工作量也很大，本文对 Linux 上直接运行 ActiveX 控件的探索，为 ActiveX 应用在 Linux 上运行提供了一个解决方案，对开发相关领域的系统有一定的参考价值。ActiveX 控件在互联网应用中地位十分重要，也有着很多与网络应用紧密相关的特性，这个课题未来的工作将向着适应网络应用的方向进一步探索。

参 考 文 献

1 Smith J E, Nair R. 虚拟机—系统与进程的通用平台(英文版)[M]. 北京：电子工业出版社，2006. 83～442
2 Sugerman J, Venkitachalam G, Lim Beng-Hong. Virtualizing I/O Devices on Vmware Workstation's Hosted Virtual Machine Monitor [EB/OL]. 2006-12-18. http://www. vmware. com/vmtn/resources/530
3 Menon A, et al. Diagnosing performance overheads in the xen virtual machine environment. In: Proc. eedings of the 1st ACM/USENIX International Conference 2005 [C]. New York: ACM Press, 2005. 13～23
4 Baraham P, et al. Xen and the Art of Virtualization. In: Proc. eeding of the Nineteenth ACM Symposium Operaing Systems Principles [C]. NY: ACM Press, 2003. 164～177
5 Wine Developer's Guide [EB/OL]. 2006-10 http://www. winehq. org/site/docs/winedev-guide
6 Bugnion E, et al. Disco: Running Commodity Operating Systems on Scalable Multiprocessors [J]. ACM Transaction on Computer Systems, 1997, 15 (4): 412～417
7 Brockschmidt K. Notes on Implementing an OLE Control Container [R]. MSDN: Microsoft Corporation , 1994
8 潘爱民. COM 原理与应用[M]. 北京：清华大学出版社，1999. 319～327
9 Water B. OLE2 从入门到精通[M]. 北京：电子工业出版社，1996:213～216
10 王映辉. ActiveX 控件的工作机理与实现研究[J]. 计算机科学，2003(3):165～166