基于 DSF 架构的 USB 设备虚拟①

陈林虎

(复旦大学 计算机科学技术学院，上海 200433)

摘 要： USB(Universal Serial Bus)已经成为电子产品对外接驳接口中最为流行的总线标准，针对 Window 操作系统下 USB 设备驱动预先开发的需求，介绍了如何使用 DSF(Device Simulation Framework:设备虚拟框架)虚拟在物理上并不存在的 USB 设备。开发该虚拟设备的主要过程包括：基于一种符合 COM 标准的语言编写描述虚拟设备功能的动态链接文件(COM 服务器端)，编写描述模拟设备行为的脚本文件(COM 客户端)。使用 DSF 架构的虚拟设备在操作系统层面以上和真实存在的物理设备没有区别，可以使用该虚拟设备进行驱动或者应用程序的开发与测试。

关键词： USB；设备模拟；DSF；驱动开发；WDM

USB Device Simulation Based on DSF

CHEN Lin-Hu

(School of Computer Science, Fudan University, Shanghai 200433, China)

Abstract: USB(Universal Serial Bus) is currently the most popular interface standard using on computer peripheral devices. In order to develop device driver for the USB device in advance, we introduce how it use DSF(Device Simulation Framework) to simulate USB devices that are not physically existed. The steps of developing a simulated device mainly include: build the dynamic link library for the device in order to describe the functional feature of the device using a COM complied language, then using a script file to describe the behavior of the simulated device. Above the operating system level, there is no difference between the simulated devices and the physically existed devices, and the simulated device can be used for developing and testing the drivers or the applications.

Keywords: USB; device simulation; DSF; driver development; WDM

1 引言

USB 总线标准自从 1996 年诞生以来，经过不断的改进，其传输带宽从诞生时 1.0 版本的 1.5Mbps，发展到了今天 3.0 版本的 4.8Gbps，与此同时，支持该总线标准的产品在当前市场上占有绝对主导地位，可以说，USB 总线催化了整个电子产业的发展与演化。

面对旺盛的市场需求以及这些市场需求所同时带来的激烈竞争，如果设备制造厂商要想在整个市场中保持一定的竞争力，就必须尽可能的缩短产品的开发周期，降低产品研发成本。因此，如果能做到驱动与产品的同步开发测试或者是驱动的超前开发测试，那么整个产品开发周期将显著缩短，产品将更加具有竞争力。然而，驱动的开发测试离不开实体硬件支持，当硬件仍在开发阶段的时候，采用虚拟硬件对驱动进行开发测试就成了必然的选择。本文通过 Windows 下的 DSF 架构介绍了一种实现 USB 虚拟设备的方法，虚拟出的 USB 设备在操作系统层面上与物理设备没有区别，可以代替物理设备进行驱动或者应用程序的开发与测试。

2 研究背景

一般情况下，操作系统若要访问安装在系统中的硬件，需要经过设备的驱动，而驱动才是真正跟硬件打交道的部分。因此，虚拟硬件设备常用的办法是从驱动端下手，让驱动欺骗操作系统使其认为系统中安装了相应的硬件，并且如果在特制的驱动中实现一些

① 收稿时间:2010-06-12;收到修改稿时间:2010-07-13

设备硬件功能的模拟，那么这个虚拟的设备也可完成相应的物理设备的性能。

然而，随着操作系统的和硬件技术的发展，设备驱动也逐渐复杂，由单一的驱动文件发展成了一个驱动体系[1]，为了简化设备驱动的开发并且提升通用性，操作系统厂家与设备标准的制定者协商，将同类设备驱动中公用的部分抽离出来，封装成单独驱动，而厂家自身只需要开发设备相关的部分驱动即可，这就构成了驱动的层级结构，例如 SCSI 设备与 USB 设备等等。例如，Windows 下的 USB 设备驱动层级结构如图 1 所示[2]：

图 1　USB 设备驱动的层级结构

如上图所示，最上层的设备驱动就是一般意义上的需要由 USB 设备厂商提供的驱动，而类驱动，USB 总线驱动，控制器驱动，则是由操作系统厂商提供，控制器小端口驱动由 USB 控制器厂商提供。类驱动根据设备的不同，可以分为 USB 音频驱动，USB 存储设备驱动，USB 人体输入驱动等等，在实现设备驱动时，可以调用对应类驱动提供的接口以简化开发。

设备驱动的分层架构确实给设备的驱动开发和使用带来了很大的便利，例如，采用 USB 接口的摄像头等设备，如果找不到相应的原厂驱动，安装特定的通用设备驱动也能正常工作，这在以前各个厂商各自为政，开发专属自己的设备驱动时是行不通的。

然而，驱动分层架构却给设备虚拟带来了一定的不便，在驱动分层以后，自行开发的驱动只能嵌入到操作系统业已提供的驱动层级架构中去，功能上受到限制，在驱动中添加虚拟硬件的想法变得不可行。除此之外，如果想从整个驱动体系上找寻突破点也是比较困难的(尤其是在 Windows 系统下)，因为 Windows 平台是一个成功的商业平台，微软为了维护自己商业上的利益，并未公开 Windows 操作系统实的很多技术细节，这其中就包括 USB 层级驱动中相应实现的源代码，这就封死了从整个驱动体系上虚拟 USB 设备的道路。

不过,幸运的是，微软也看到了在虚拟 USB 设备方面的需求，因此在保护自己商业利益的前提下给出了一种折中的解决方案，这就是 DSF 架构。该架构从 Windows Drive Development Kit 6.0 起向开发者提供，目前仅支持 USB 设备的虚拟，预计外来会扩大设备的支持范围。

3　DSF(Device Simulation Framework)USB设备虚拟简介

正如上一节所提到的，Windows 平台上的设备驱动已经构成了一个复杂的体系结构，如果想要虚拟物理设备，最好的方法是从整个驱动体系上入手，而 DSF 则正好是提供这个架构中的一些基础设施，并将相应的接口暴露给虚拟设备的开发者，使开发者不再需要了解整个驱动体系全部技术细节，而仅仅是关注虚拟设备功能的实现。DSF 的总体架构图如图 2 所示：

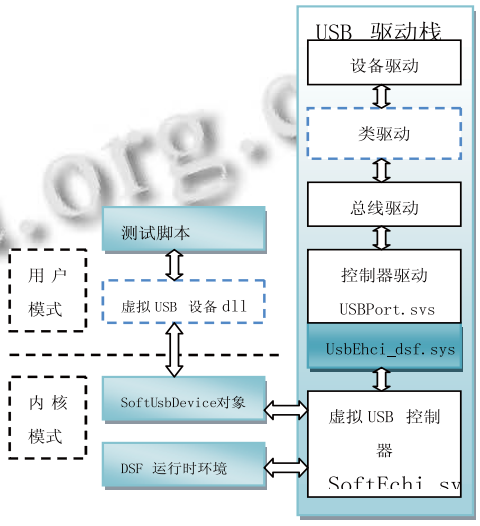

图 2　DSF 架构示意图

与图 1 的标准 USB 设备驱动结构相比，DSF 架构的变化主要在 USB 控制器驱动之下，也就是图 2 中 USB 驱动栈中深颜色矩形框以及之下的部分。在其之上的部分没有任何改变，仍旧是设备驱动，设备类驱动和控制器驱动的 USBPort.sys 部分，因此，对于 USB

设备驱动来说，虚拟的 USB 设备与物理的 USB 没有区别，这就保证了用虚拟 USB 设备开发测试相应设备驱动的正确性。

DSF 架构的核心是在 USB 控制器之下的部分，包括 UsbEhci_dfs.sys，这是一个 USB 控制器的小端口驱动，负责和其下层的模拟 USB 控制器(SoftEchi.sys)进行交互。除此之外，DSF 的运行时环境提供模拟 USB 控制器运行的必要基础设施。和开发者关系最为密切的是 SoftUsbDevice 对象，它是所有虚拟 USB 设备的抽象对象，提供虚拟 USB 设备的“实体”，开发者需要根据需要的功能结合其它提供接口的对象，并将其封装成一个提供 COM 服务的动态连接库(虚线框中的虚拟 USB 设备 dll)。最后，对于驱动的测试人员或者虚拟设备的使用人员来说，还需要一个调用生成的动态连接库的脚本文件，该脚本控制虚拟 USB 设备的物理行为，比如，设备的插入与拔出等。在 DSF 架构中，需要用户进行开发的部分全部处于用户模式下，这就隐藏了内核编程的复杂性，同时也免去了可能带来的一些安全性问题。

对于开发者来说，DSF 架构就像是在系统内安装了一个虚拟的 USB 总线控制器，而该控制器的下层并没有其他的物理总线设备，而与此不同的是，物理存在的 USB 总线控制器一般是是运行在系统的 PCI 总线之上。 自行开发的虚拟 USB 设备就“接驳”在这个虚拟的 USB 总线控制器上，从而构成了一套完整的虚拟的 USB 设备体系结构。

4 USB虚拟设备的实现

DSF 架构中，虚拟 USB 设备主要由三个部分实现，虚拟 USB 控制器，描述设备功能的 COM 服务器动态链接库，以及一个用以控制虚拟设备行为的脚本文件(COM 客户端)，来模仿设备的接入，拔出等一些行为。虚拟 USB 控制器微软已经提供了可用的版本，并且目前没有向开发者提供相关资料，其它两部分需要由用户自行开发。

4.1 USB 虚拟设备 COM 服务器动态链接库的实现

在 DSF 架构中，虚拟 USB 设备的实现主要通过以下几个类：ISoftUSBDevice，ISoftUSBEndPoint，IConnectionPoint 实现，这些类的对象被组合在一个 COM 实体类中，同时该 COM 类需要继承一些 DSF 所实现的类或者接口。整体的架构如图 3 所示：

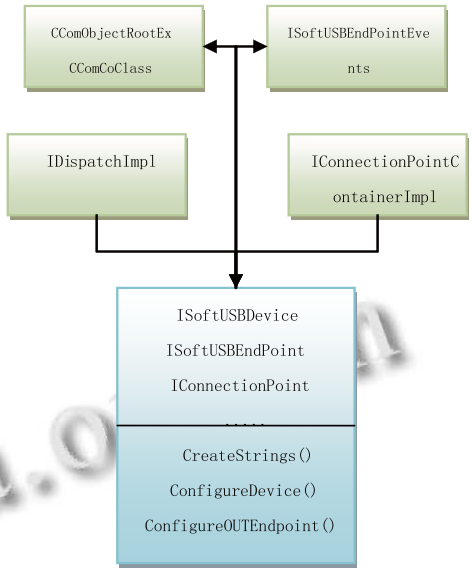

图 3 USB 虚拟设备实现类结构图

其中，CComObjectRootEx 是一个 COM 类的标准接口，所有实现 COM Server 的类必须实现该接口或者 CComObjectRoot 接口，该接口主要为相应的对象提供引用计数。同时，CComCoClass 也是 COM 编程中的常用接口，该接口提供获取对象 CLSID(类 ID)的方法接口，并且提供该类的工厂。IConnectionPoint ContainerImpl 则是实现了一个容纳链接点的容器，以便 COM 的客户端能够方便的查询对象的对外接口。另外，IDispatchImpl 则是对象双接口的一种实现。以上几个接口都是 COM 服务器编程中需要使用的标准接口，而 ISoftUSBEndPointEvent 则是 DSF 中实现虚拟 USB 设备的专用接口，它主要定义了如下重要方法：

OnSetupTransfer：当虚拟控制器向该虚拟设备对象发出设置指令时该方法被触发，用户可根据需要在该方法内进行相关的处理。

OnWriteTransfer,OnReadTransfer：前者在虚拟控制器向设备发出写命令时被触发，后者在控制器向设备发出读命令的时候被触发，用户需根据自己的需求进行相应的实现。

OnDeviceRequest,OnDeviceRequestComplete： 前者在设备收到从虚拟控制器发出的设备请求时被触发，后者在收到控制器发出的请求状态确认时被触发，通过后者，用户可以在设备请求完成时设置该次传输

的完成状态。

以上方法用户都必须给出自己的实现。对于实现虚拟设备的类来说，最重要的组成部分就是 ISoftUSBDevice 接口，该接口实现了 SoftUSBDevcie 对象，后者是描述虚拟 USB 设备的基本对象，该对象中大部分的属性是按照 USB 2.0 标准规范的对应属性命名。该对象的示意图如图 4 所示[3]：

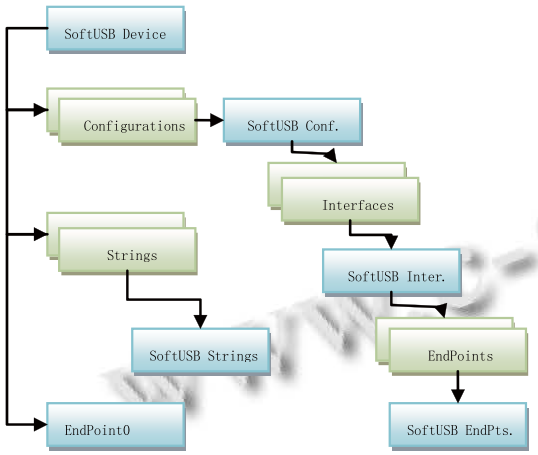

图 4　SoftUSB 对象结构图

根据 USB 2.0 规范[4]，一个 USB 设备可以有多个配置，并以 Configurations 指向该系列配置的集合。在 DSF 架构中，每个配置对应一个 SoftUSBConfiguration 对象。而每个配置具有一个可以容纳 SoftUSBInterface 的容器类，即 Interfaces，相应的每个 SoftUSBInterface 可以定义多个端点(EndPoints)，每一个端点对应一个 SoftUSBEndPoint 对象。端点是用来表示 USB 设备与控制器连接的抽象对象，一般来说，每个设备至少有两个端点，一个用于输入一个用于输出。Strings 是一个保存若干 SoftUSBString 对象集合的容器，与之对应，每一个 SoftUSBString 对象都存有可标识该 USB 设备的字符串，比如序列号，设备制造厂商编号等等。所有这些设备的属性可以在 COM 对象进行初始化时进行设置，但是也可以通过脚本进行设置，前提是 COM 服务器中提供了相应的设置服务。

4.2 控制脚本(COM 客户端)的创建

在 DSF 架构中,虚拟 USB 设备的控制脚本作为 COM 组件的客户端，而 4.1 节中所介绍的动态链接库作为 COM 的服务器端。客户端(脚本)向服务器端发出各种控制命令，包括设备的插入拔出等相关命令，而服务器端则处理这些指令，以控制虚拟 USB 设备的行为。实际上，脚本的功能并不限于仅仅控制设备被的物理行为，正如上文所述，脚本是 COM 的客户端，如果在 COM 服务端增加相应的功能，那么在脚本中就可以实现更多设备的控制功能，例如设置虚拟 USB 设备的生产厂商编号，甚至是改变 USB 设备的某些功能，而一个典型 USB 虚拟设备客户端应该包含如图 5 的流程：

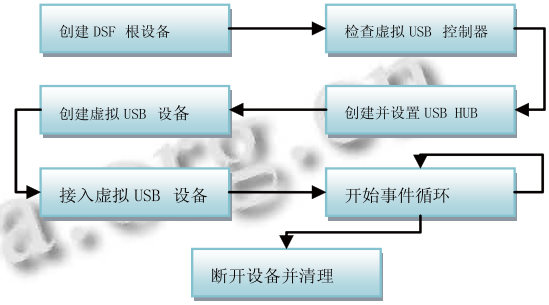

图 5　控制脚本流程图

控制脚本在驱动测试中将扮演重要角色，驱动测试人员主要通过编写该脚本来达到控制硬件行为与测试驱动的目的，该脚本可以用符合 COM 规范的语言编写，如 VBScript 等.

5　总结

本文介绍了一种采用微软 DSF 架构实现虚拟 USB 设备的方法，实施该方法的过程主要包括实现描述设备属性的 COM 组件服务器端，以及模拟设备行为的 COM 客户端，后者通过 COM 的规范调用前者提供的服务(组件)，实现对设备行为的模拟。用该架构模拟出的 USB 设备在操作系统层面以上和物理设备没有区别，可以用该设备进行产品驱动的开发与测试，加快产品的开发流程。

参考文献

1 Oney W. Programming the Microsoft Windows Driver Model. 2nd ed., Microsoft Press, 2005. 362－370.

2 张帆,史彩成,等.Windows 驱动开发技术详解.北京:电子工业出版社,2008.430－450.

3 Shier P. Using The Device Simulation Framework For Software Simulation of USB Device. Microsoft WinHEC, 2006. 20－36.

4 Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. Universal Serial Bus Specification. Rev. 2.0, 2000: 261－274.