第 34 卷 第 9 期 计 算 机 工 程 2008 年 5 月
Vol.34 No.9 Computer Engineering May 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)09—0273—03 文献标识码：A 中图分类号：TP391

Linux 蓝牙协议栈的 USB 设备驱动

梁军学，郁 滨
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：基于对 Linux 下蓝牙协议栈 BlueZ 源代码的分析，给出 BlueZ 的组织结构和特点。分析蓝牙 USB 传输驱动机制和数据处理过程，给出实现蓝牙设备驱动的重要数据结构和流程，并总结 Linux 下开发蓝牙 USB 设备驱动的一般方法和关键技术。
关键词：Linux 系统；蓝牙协议栈；设备驱动

USB Device Driver for Linux Bluetooth Stack

LIANG Jun-xue, YU Bin
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】This paper depicts the structure and characteristics of BlueZ based on analyzing the source code of Linux bluetooth stack BlueZ. It analyzes the implementation of bluetooth USB transport driver scheme and data processing procedure in detail, and gives the key data structure and implementation of bluetooth device driver. It summarizes the approach of developing Linux bluetooth USB device driver and the key technology.
【Key words】Linux system; bluetooth stack; device driver

1 概述

蓝牙技术是开放式通信规范，而 Linux 是开放源码的操作系统。廉价设备与免费软件的结合，促进了蓝牙技术和 Linux 的发展与融合。

Linux 最早的蓝牙协议栈是由 Axis Communication Inc 在 1999 年发布的 OpenBT 协议栈。随后，IBM 发布了 BlueDrekar 协议栈，但没有公开其源码。Qualcomm Incorporated 在 2001 年发布的 BlueZ 协议栈被接纳为 2.4.6 内核的一部分。此外，Rappore Technology 及 Nokia 的 Affix Bluetooth Stack 都是 Linux 系统下的蓝牙协议栈，应用在不同的设备和领域中。

BlueZ 是 Linux 的官方蓝牙协议栈，也是目前应用最广泛的协议栈，几乎支持所有已通过认证的蓝牙设备。对于基于主机的蓝牙应用，目前常见的硬件接口有 UART, USB 和 PC 卡等，USB 作为 PC 的标准外设接口，具有连接方便、兼容性好和支持高速设备等特点，已广泛应用于蓝牙设备。

目前对Linux下USB设备驱动的研究已较为广泛而深 入[1-4]，但对Linux下的蓝牙设备驱动还没有专门的研究。本文在分析USB设备驱动和蓝牙协议栈的基础上，总结了Linux下开发蓝牙USB驱动程序的一般方法，并深入剖析了其关键技术。

2 Linux 蓝牙协议栈 BlueZ 简介

BlueZ 目前已成为一个开放性的源码工程。它可以很好地在 Linux 支持的各种体系的硬件平台下运行，包括各种单处理器平台、多处理器平台及超线程系统。

BlueZ 由多个独立的模块组成，内核空间主要包括设备驱动层、蓝牙核心及 HCI 层、L2CAP 与 SCO 音频层、RFCOMM, BNEP, CMTP 与 HIDP 层、通用蓝牙 SDP 库和后台服务及面向所有层的标准套接字接口；在用户空间提供了蓝牙配置、测试及协议分析等工具。其组织结构如图 1 所示，BlueZ 没有实现专门的 SDP 层，而是将其实现为运行在后台的蓝牙服务库例程(图 1 没有描述该后台服务)。RFOMM 层支持标准的套接口，并提供了串行仿真 TTY 接口，这使串行端口应用程序和协议可以不加更改地运行在蓝牙设备上，例如通过点对点协议 PPP 可实现基于 TCP/IP 协议簇的所有网络应用。BNEP 层实现了蓝牙的以太网仿真，TCP/IP 可以直接运行于其上。

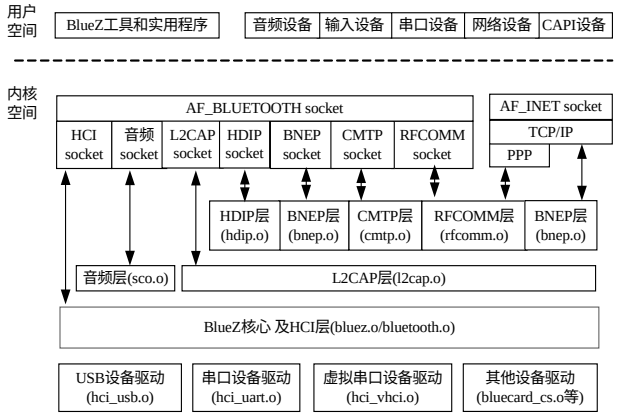

图 1 BlueZ 组织结构

3 蓝牙 USB 设备驱动

设备驱动程序在 Linux 内核中起着重要作用，它使某个硬件能响应一个定义良好的内部编程接口。这些接口隐藏了设备的工作细节，用户通过一组独立于特定驱动程序的标准调用来操作设备。而将这些调用映射到作用于实际硬件设备的特有操作上，则是驱动程序的任务。

3.1 蓝牙 USB 设备驱动架构

3.1.1 Linux USB 驱动简介

USB 是一种被动的总线类型，不能主动向主机发送任何

作者简介：梁军学(1982－)，男，硕士研究生，主研方向：蓝牙技术，嵌入式系统技术；郁 滨，教授、博士
收稿日期：2007-08-02 **E-mail：**ljx413@163.com

消息，USB 设备和主机间的通信由主机端的 USB 控制器实现。主机控制器位于操作系统和设备之间，负责接收操作系统对设备发出的各种控制信息，进行适当处理后再发送给设备，USB 接口是设备和主控制器之间的通信通道，对它所传送的数据没有内容上或结构上的特殊要求。USB 设备与外界的通信通过端点进行，端点单向传送数据，根据数据传输方向的不同，可分为上行端点和下行端点。在 USB 规范中，定义了 4 种不同类型的端点：控制端点，批量端点，中断端点和等时端点。USB 端点被捆绑为接口，每个接口只处理一种 USB 逻辑连接，代表一个基本功能，而每个驱动程序控制一个接口。若干个接口组合成为配置，一个 USB 设备可以有多个配置，且可以在配置之间切换以改变状态，但在一个时刻只能激活一个配置。

USB 驱动程序存在于不同的内核子系统和 USB 主控制器之间，为了屏蔽不同主控制器类型对上层操作的影响，Linux 中定义了一个称为 USB 核心的抽象层，通过定义一套数据结构、宏及函数来抽象所有 USB 硬件设备，为具体的设备驱动程序提供了一个用于访问和控制 USB 硬件的接口，而实际的操作则通过不同的主控制器驱动来完成。

3.1.2　蓝牙 USB 传输层

蓝牙规范中定义了主机控制接口 HCI 协议，它为高层协议访问基带层、LM 层以及状态和控制寄存器等硬件提供了统一的命令格式。蓝牙设备通过模块内的 USB 设备控制器和总线与主机控制器相连，负责与主机的数据交换，它在 USB 固件的控制下，将接收到的 HCI 协议分组交给 HCI 固件处理或将来自链路管理器和基带控制器的数据传送至主机。HCI 固件负责解释从主机接收到的 HCI 分组并交给链路管理器和基带控制器处理，或收集蓝牙模块各部件的状态信息并递交给主机。

根据协议分层观点，主机和蓝牙设备内的每个功能部件都只与对等的实体传输数据，而在主机和蓝牙设备内部，功能部件之间通过接口通信，下层功能部件为上层提供透明的服务(图 2)。蓝牙 USB 驱动的功能是在主机和蓝牙设备之间透明地传递 HCI 协议数据，这就是蓝牙规范中定义的 USB 传输层的功能。

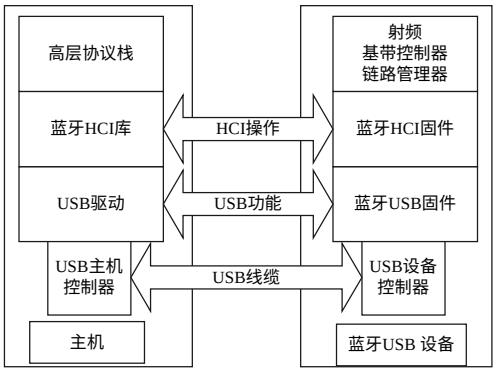

图 2　蓝牙 USB 传输层

3.1.3　蓝牙 USB 设备的数据处理

蓝牙 USB 设备的配置由 2 个接口组成。第 1 个接口即接口 0，由控制、批量和中断端点组成；第 2 个接口即接口 1，包含了可配置的等时端点，它根据不同的带宽需求提供 4 个可选的配置。默认时接口 1 为空，因此，蓝牙设备可以没有等时端点。因为等时端点与其他端点在不同的接口中，所以当调整等时带宽分配时，任何未处理的批量和中断事务不必中止或重新提交。各端点的功能和配置如下：

(1)控制端点，所有 USB 设备都有一个名为“端点 0”的控制端点，USB 核心使用该端点在插入时进行设备的配置和控制。端点 0 允许主机向主机控制器发送 HCI 命令，当 USB 固件在此端点收到具有蓝牙类型码的分组时，应将其视为 HCI 命令分组。

(2)批量端点，用于传输 ACL 分组。ACL 分组的数据完整性很重要，因此，批量端点应具有发现并修正错误的能力。Linux 下的蓝牙设备可支持多个 ACL 链路，经过此通道的数据可能是来自或发往多个不同从设备的数据。为了避免饥饿，主机控制器应采用类似共享端点的机制来进行流量控制。默认的批量最大分组大小为 64 B，根据可获得的总线带宽，批量端点能在 1 ms 的帧中传输 1 个或多个 64 B 的缓冲区数据。

(3)中断端点，用于接收事件分组。为了保证事件可以按预期被及时处理，中断端点保留了足够的带宽以保证可靠的反应时间，对于蓝牙设备，中断端点提供的事件处理时间间隔为 1 ms。

(4)等时端点，用于传输 SCO 分组。SCO 分组用于传输音视频等实时数据，允许数据损坏和丢失，但须提供一个恒定的数据流，这可以通过将这些数据传输到主机控制器的 SCO FIFOs 中来实现，FIFOs 以固定的速率接收或发送数据，如果 FIFOs 已满，则其中的数据被新数据覆盖。蓝牙可支持 3 条 64 Kb/s 的语音信道，并且能接收用不同方式编码的数据(8 bit 或 16 bit 的线性语音编码)。默认最大分组大小为 64 B，如果不能支持 3 条 16 bit 编码的音频信道，则可将最大分组大小设为 32 B。

通过端点传输的 HCI 数据，包含在一个 USB 事务(transaction)中。一个 USB 事务即完成一次 IO 请求，可能包含一个或多个 USB 帧的传输。例如，一个包含 256 B(包括 HCI 头和 HCI 数据)的 ACL 分组将在一次 IO 请求中通过批量端点发送，这个 IO 请求将需要 4 个 64 B 的 USB 帧，才能完成一次 USB 事务。

3.2　重要数据结构

3.2.1　hci_dev 结构体

hci_dev 结构体是在 HCI 层中对设备的一个抽象，不论实际设备是哪种蓝牙设备，通过什么接口连接到主机，都要向 HCI 层或蓝牙核心层注册一个 hci_dev 设备，这通过 hci_register_dev 函数来完成。

该结构体包含的成员主要有以下 5 类：

(1)设备属性，如设备地址、设备名、设备状态、设备类型及所支持的特性等。

(2)链路属性，主要有支持的分组类型、链路策略和链路模式。

(3)分组属性，如最大传输单元、分组数、已接收分组数等。

(4)任务处理，包含事件查询的缓存区、等待队列、数据传输的 tasklet 结构体及相关的锁和信号量。

(5)设备操作，包含 open, close, flush, send, destruct, notify 和 ioctl 共 7 个函数。

此外，还有模块所有者、私有数据区、异常标志等成员。

3.2.2　hci_usb 结构体

hci_usb 结构体通过 2 个分别指向 hci_dev 结构体和

usb_device 结构体的指针将 HCI 层与 USB 设备联系起来。usb_device 是 USB 设备描述符，它是所有 USB 设备的根描述符。

由于蓝牙设备的接口 0 是默认接口，因此该结构体并没有此接口的指针或成员变量。接口 1 用于传输等时数据，有些设备的配置中可能没有提供等时传输能力，因此，hci_usb 中包含了一个指向此接口的指针，在初始化时默认为空。

同样，端点 0 为默认的控制端点，没有包含在该结构体中。包含的 5 个端点指针是 2 个批量端点(in/out)、2 个等时端点和 1 个中断端点(事件只有 in 而没有 out)。

hci_usb 还包含了用于任务请求的套接字缓冲区指针、USB 请求块队列、请求类型及用于同步的读写锁。

3.2.3 _urb 结构体

_urb 结构体中包含了一个指向 urb 结构体的指针，urb 即设备请求块，内核中的 USB 代码通过它与所有 USB 设备通信。这个请求块用 struct urb 结构体来描述，以一种异步的方式往/从特定的 USB 设备上的特定端点发送/接收数据，可以被动态地创建或取消。

_urb 中的其他成员包括请求块队列、指向等待处理数据的链表指针、包含的数据类型和一个私有数据区指针。

3.3 蓝牙 USB 设备驱动的实现

3.3.1 驱动程序的注册与注销

所有 USB 驱动程序都必须向内核注册其驱动程序，这是通过创建一个 usb_struct 结构体来完成的。蓝牙设备创建的 usb_struct 结构体如下：

```
static struct usb_driver hci_usb_driver = {
  .name= "hci_usb",
  .probe= hci_usb_probe,
  .disconnect= hci_usb_disconnect,
  .id_table      = bluetooth_ids,
  };
```

其中，驱动程序的名字为 hci_usb；hci_usb_probe 和 hci_usb_disconnect 分别为其探测函数和断开函数；bluetooth_ids 为全局变量，指向驱动所支持的蓝牙设备列表。

当装载驱动时，需要将该结构体注册到内核，这通过 hci_usb_init 函数来完成，该函数调用了 usb_register_driver。当卸载驱动时，需要将该结构体从内核中注销，这与驱动注册类似，通过封装 usb_deregister 的函数 hci_usb_exit 函数来完成。

3.3.2 设备探测

当一个 USB 设备连接到主机时，主机会给这个 USB 设备分配一个在 1～127 间唯一的设备号，并读取设备的描述符、查找合适的驱动程序，如果与 hci_usb 驱动程序注册到 USB 核心的信息相匹配，则将设备与 hci_usb 驱动程序绑定，并立即调用此驱动的设备探测(probe)函数。探测函数的主要任务是初始化可能用于控制 USB 设备的一些局部结构体并向内核注册设备信息，图 3 为蓝牙设备驱动探测函数 hci_usb_probe 的流程。

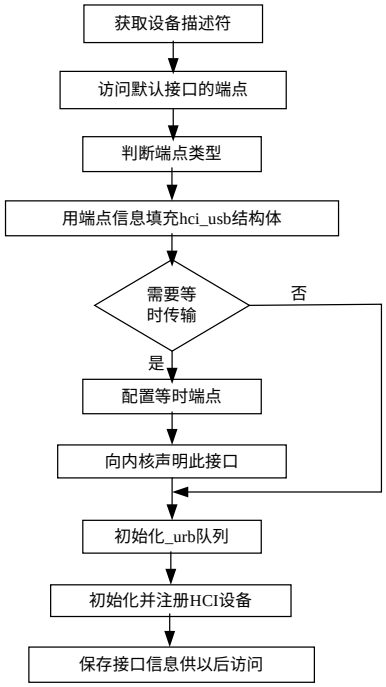

图 3 设备探测过程

3.3.3 设备断开

当蓝牙设备被拔出或断开连接时，需要清理与该设备相关的资源，这是通过调用 hci_usb_disconnect 入口函数完成的。该函数除了清除相关接口信息外，还要从内核中注销设备对应的 HCI 设备并释放其占用的资源。

3.3.4 设备操作

在设备探测期间向内核中注册了 HCI 设备，并初始化设备操作的函数入口点，即 3.2.1 节所述 hci_dev 结构体中除 ioctl 外的 6 个函数。ioctl 主要用于各种类型的硬件控制，各个蓝牙厂商提供的设备操作能力不同，目前还没有统一的实现。6 个设备操作与一般的文件操作类似，由于蓝牙设备既可以是字符型设备，也可以是块设备(通过上层的抽象还可以作为网络设备)，因此须根据设备的特性决定其是否需要某些特殊的函数调用，例如是否支持等时传输。

4 结束语

不同蓝牙设备有不同带宽需求和缓冲区设置，同一个蓝牙设备可支持多种配置和操作模式，因此，蓝牙设备驱动的关键是将底层的设备抽象为 HCI 设备，为上层调用提供统一的接口，而实际操作则通过调用 USB 核心功能来完成。

参考文献

[1] Jonathan C, Alessandro R, Greg K H. Linux Device Driver[M]. [S. l.]: O'Reilly Media Inc., 2005.

[2] Detlef F. Programming Guide for Linux USB Device Drivers [EB/OL]. (2000-10-25). http://usb.cs.tum.edu/usbdoc.

[3] 巍 骜, 张焕强, 方贵明. 基于 Linux 的 USB 驱动程序实现[J]. 计算机应用, 2002, 22(8): 17-19.

[4] 梁正平, 毋国庆, 肖 敬. Linux 中 USB 设备驱动程序研究[J]. 计算机应用研究, 2004, 21(6): 70-72.