Broadview
www.broadview.com.cn

做程序员的最高境界
就要像和尚研究佛法
一样研究算法

这次函数调用就像流星，
短暂地划过
却能照亮整个天空

它的心只有编译器才懂

终于，四大传输也就
这样结束了，
而我们的故事也即将
ALT+F4了。
我只是说也许

Linux那些事儿之我是USB 第2版

·任桥伟　肖季东　肖林甫　编著·

本书
不仅能带给你
一趟愉悦的USB代码之旅，
更能给你
一套终生受益的高效内核学习法宝！

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Linux那些事儿之我是USB 第2版

·任桥伟　肖季东　肖林甫　编著·

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书基于2.6.22内核，对USB子系统的大部分源代码逐行进行分析，系统地阐释了Linux内核中USB子系统是如何运转的，子系统内部的各个模块之间是如何互相协作、配合的。本次改版修改了第1版中出现的错误，增加了一个附录，主要内容是关于Linux内核的学习方法，是作者的经验总结，值得一读。

本书使用幽默诙谐的笔调对Linux内核中的USB子系统源代码进行了分析，形象且详尽地介绍了USB在Linux中的实现。本书从U盘、Hub、USB Core到主机控制器覆盖了USB实现的方方面面，被一些网友誉为USB开发的“圣经”。

对于Linux初学者，可以通过本书掌握学习内核、浏览内核代码的方法；对于Linux驱动开发者，可以通过本书对设备模型有形象深刻的理解；对于USB开发者，可以通过本书全面理解USB在一个操作系统中的实现；对于Linux内核开发者，也可以通过本书学习到很多Linux高手开发和维护一个完整子系统时的编程思想。

图书在版编目（CIP）数据

Linux那些事儿之我是USB / 任桥伟，肖季东，肖林甫编著. —2版. —北京：电子工业出版社，2012.3
ISBN 978-7-121-15817-9

Ⅰ. ①L… Ⅱ. ①任… ②肖… ③肖… Ⅲ. ①Linux操作系统－程序设计 ②USB总线－串行接口－程序设计 Ⅳ. ①TP316.89 ②TP334.7

中国版本图书馆CIP数据核字（2012）第016530号

责任编辑：孙学瑛
印　　刷：北京天宇星印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编100036
开　　本：787×1092　1/16　印张：28　字数：717千字
印　　次：2012年3月第1次印刷
印　　数：3000册　定价：79.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

从写Linux那些事儿系列内容开始，到如今已有四年多了，而从整理出版第1版到现在也已经一载有余了。期间不断有认识或不认识的朋友问我，怎么会想起写这么多如此可爱的文字，我的回答都是：娱乐自己，娱乐大家而已！

或许，大家早已经默认技术本是一个沉重或者枯燥的话题，我们无法用一种娱乐的心态去看待它，甚至说很多人早已丧失了从中获取乐趣的能力。但是，一切本不该如此的，对于不管什么原因踏入这个行业的我们，愿意或不愿意，技术都已经是我们生命不可分割的一部分。

既如此，又何不放轻松些，把它当成朋友，用我们自己的方式去与它交流，而不是仅仅把它当成一堆堆死气沉沉的代码，亦或一些枯燥的名词。而针对这本书的内容，我要说的就是：把内核当朋友。笑来老师有本书，叫《把时间当做朋友》，告诉我们只有把时间当做朋友，才能更好地利用自己的时间做些有益的事情。眼睛一闭一睁，一天就过去了；眼睛一闭不睁，一辈子就过去了。只有善待时间，时间才能善待我们。同样，我们只有把内核当朋友，当成一个有生命的实体，把它放在对等的地位上，我们才能够更好地认识和理解到它的精髓。

具体到这本书，您可以把它当成一本内核源码分析的书，甚至仅仅当成内核USB实现源码分析的书，但是我更希望您把它当成展现如何学习Linux内核，展现如何与内核进行平等交流的一个范例，起码它体现了我们应该用什么样的态度去对待Linux内核源码。也就是说，分析内核源码，态度决定一切。我们很多人或许有这样的困惑，也分析浏览了很多内核的源码，可总是觉得分析、浏览后，脑子里还是空空的，并没有感觉到多大的收获。这个时候我们或许可以去看看是不是自己在分析代码时的态度出现了问题。我们在分析内核源码时，只有遵循严谨的态度，而不是抱着走马观花、得过且过的态度，最终才会有很大的收获。

然后还有一句曾小范围流传的话：技术水平的高低不是决定于C，或者C++等用得有多么熟练，而是决定于你掌握的资源有多少。所以，我们还要以内核源码为中心，坚持学习资源建设。在我们学习内核的过程中，内核源码本身就是最好的参考资料，其他任何经典或非经典的书最多只是起辅助作用，不能也不应该取代内核代码在我们学习过程中的主导地位。但是这些辅助的作用也是不可忽视的，我们需要以内核源码为中心，坚持各种学习资源的长期建设不动摇。

再次感谢孙学瑛编辑，没有她的努力，这本书的内容将会一直偏居网络一隅，将不可能被出版，从而去帮助更多需要的人。

目 录

第 1 篇 Linux 那些事儿之我是 USB Core

第 2 篇 Linux 那些事儿之我是 HUB

第 3 篇 Linux 那些事儿之我是 U 盘

附录 A　Linux 那些事儿之我是 sysfs

附录 B　Linux 内核高效学习法

第 1 篇

Linux 那些事儿之我是 USB Core

1. 引子

老夫子们痛心疾首地总结说，现代青年的写照——自负太高，反对太多，商议太久，行动太迟，后悔太早。上天戏弄，我不幸地混进了“80 后”的革命队伍里，成了一名现代青年，前有老夫子的忧心忡忡，后有“90 后”的轻蔑嘲弄，终日在“迷失”与“老土”这样的两极词汇里徘徊。

这里我就讲一讲 USB，让他们看一看“80 后”还知道什么叫 USB。

还是要说在前面，在这里耗费青春写 USB，并不是因为喜欢它，相反，对它毫无感觉可言，虽然每天都必须和它相依为伴，不离不弃，不过那可是丝毫没有办法的事情，非我所愿。是不是特说到心坎儿里去了？不过您别多想，咱这里只谈 USB。

一句话总结：哥写的不是 USB，是寂寞。

2. 它从哪里来

“你从哪里来，我的朋友，好像一只蝴蝶，飞进我的窗口。”

在嘹亮的歌声中，USB 好像一只蝴蝶飞进了千家万户。它从哪里来，它从 Intel 来。Intel 不养蝴蝶，而是做 CPU，它只是在蝴蝶的翅膀上烙上“Intel inside”，蝴蝶让咱们的同胞去养了，然后带着 Intel 飞进了千万家。

不过，与 PCI、AGP 属于 Intel 单独提出的硬件标准不同，Compaq、IBM、Microsoft 等也一起参与了 USB 这个游戏。他们一起于 1994 年 11 月提出了 USB，并于 1995 年 11 月制定了 0.9 版本，1996 年制定了 1.0 版本。不过 USB 并没有因为有这些大佬的支持立即迎来它的春天，只怪它诞生在了冬季，生不逢时啊！

因为缺乏操作平台的良好支持和大量支持它的产品，这些标准都成了空谈。1998 年，USB 1.1 的出现，忽如一夜春风来，它就像春天里的一朵油菜花，终于涂上了浓重的一抹黄色。

为什么要开发 USB？

在 USB 出现以前，电脑的接口处于“春秋战国时代”，串口、并口等多方割据，键盘、鼠标、MODEM、打印机、扫描仪等都要连接在这些不同种类的接口上，一个接口只能连接一个设备。不过咱们的电脑不可能有那么多接口，所以扩展能力不足，而且速度也确实很有限。还有关键的一点是，热插拔对它们来说也是比较危险的操作。

USB 正是为了解决速度、扩展能力、易用性等问题应景而生的。

3. PK

USB 的一生也充满了“PK”，不过 USB 还不够老，说一生还太早了，发哥说得好：“我才刚上路呢！”

USB 最初的设计目标就是替代串行、并行等各种低速总线，以一种单一类型的总线连接各种不同的设备。它现在几乎可以支持所有连接到 PC 上的设备，1999 年提出的 USB 2.0 理论上可以达到 480 MB/s 的速度，2008 年公布的 USB 3.0 标准更是提供了十倍于 USB 2.0 的传输速度。

因此，USB 与串口、并口等的这场“PK”从一开始就是不平等的，这样的开始也注定了以什么样的结果结束，只能说命运选择了 USB。我们很多人都说命运掌握在自己手里，但是从 USB 充满“PK”的一生中可以知道，只有变得比别人更强，命运才能掌握在自己手里。

有了 USB 在这场 PK 中的大获全胜，才有了 USB 键盘、USB 鼠标、USB 打印机、USB 摄像头、USB 扫描仪、USB 音箱等。至于将来，“PK 自己的，让别人去说吧！”USB 如是说。

4. 漫漫辛酸路

USB 的一生充满了“PK”，并在“PK”中发展，从 USB 1.0、USB 1.1、USB 2.0 到 USB 3.0，漫漫辛酸路，一把辛酸泪。

USB 2.0 的高速模式（High-Speed）最高已经达到了 480 MB/s，也就是说，以这个速度，你将自己从网上下载的短片备份到自己的移动硬盘上的时间长约为一秒钟。而 USB 3.0 的 Super-Speed 模式比这个速度提高了几乎 10 倍，达到了 4.8GB/s。

USB 走过的这段辛酸路，对咱们来说，最直观的结果也就是传输速度提高了，过程很艰辛，结果很简单。

USB 的各个版本都是兼容的。每个 USB 2.0 控制器带有 3 个芯片，根据设备的识别方式，将信号发送到正确的控制芯片。我们可以将 USB 1.1 设备连接到 USB 2.0 的控制器上使用，不过它只能达到 USB 1.1 的速度。同时也可以将 USB 2.0 的设备连接到 USB 1.1 的控制器上，不过不能指望它能以 USB 2.0 的速度运行。

显然，Linux 对 USB 1.1 和 USB 2.0 都是支持的，并抢在 Windows 前，在 2.6.31 内核中率先对 USB 3.0 进行了支持。

5. 我型我秀

USB 既然能一路“PK”走过来，也算是一个挺能“秀”的角色了，不然也不会有那么多的拥护者。

USB 为所有的 USB 外设都提供了单一、标准的连接类型，这就简化了外设的设计，也让我们不用再去想哪个设备对应哪个插槽的问题，就像种萝卜，一个萝卜一个坑，但是哪个萝卜种到哪个坑里是不用我们关心的。

USB 支持热插拔，而其他的比如 SCSI 设备等只有在关掉主机的前提下才能增加或移走外围设备。所以说，USB 的一生不仅仅是“PK”的一生，也是丰富多彩的一生，可以不用关机就能更换不同种类的外设。

USB 在设备供电方面提供了灵活性。USB 设备可以通过 USB 电缆供电，不然移动硬盘、

IPod 等常备外设也用不了了。相对应，有的 USB 设备也可以使用普通的电源供电。

USB 能够支持从每秒几十千字节到几十兆字节的传输速率，来适应不同种类的外设。它可以支持多个设备同时操作，也支持多功能的设备。多功能的设备当然指的就是一个设备同时有多个功能， 比如 USB 扬声器。这通过在一个设备中包含多个接口来支持，一个接口支持一个功能。

USB 可以支持多达 127 个设备。

USB 可以保证固定的带宽，这个对视频/音频设备是利好。

6. 我是一棵树

“我是一棵树，静静地站在田野里，风儿吹过，我不知它的去向，人儿走过，我不知谁会为我停留。”

如图 1.6.1 所示，USB 子系统的拓扑也是一棵树，它并不以总线的方式来部署。

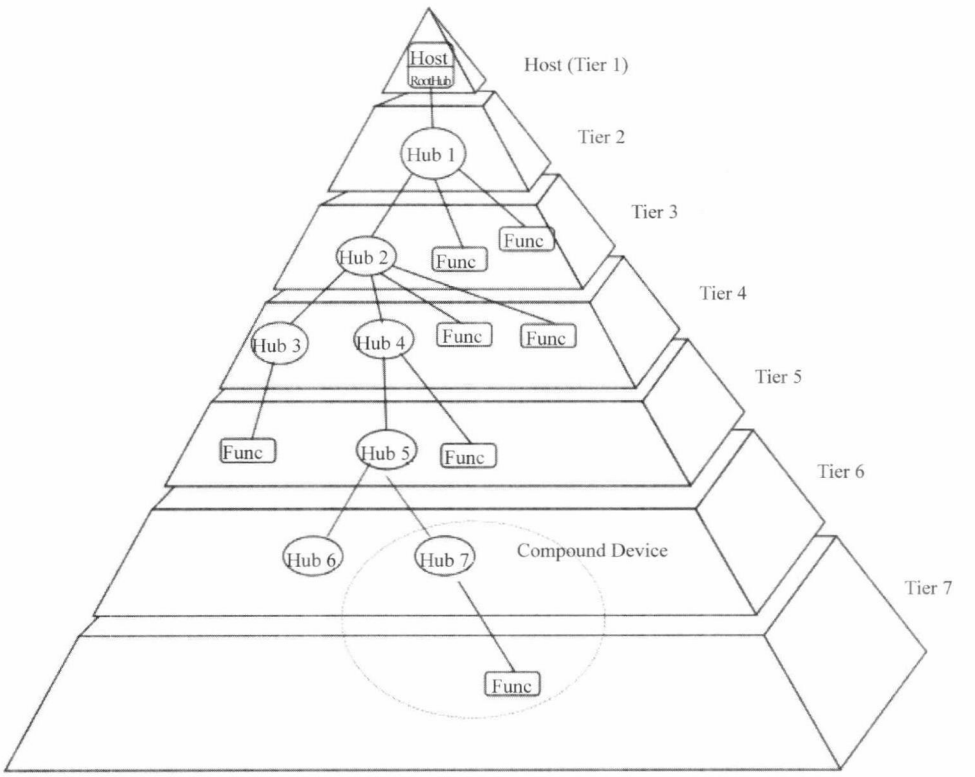

图 1.6.1　USB 子系统的树形结构

我曾经指着路边一棵奇形怪状的树问朋友：“这是什么树？”朋友的回答让我很晕：“大树。”那图 1.6.1 指的是什么树？自然也是大树了，不过却是 USB 的大树。这棵大树主要包括 USB 连

接、USB Host Controller（USB 主机控制器）和 USB 设备三个部分。而 USB 设备还包括了 Hub 和功能设备（也就是图 1.6.1 中的 Func）。

什么是 USB 主机控制器？控制器，顾名思义，用于控制。控制什么？控制所有的 USB 设备的通信。通常，计算机的 CPU 并不是直接和 USB 设备打交道，而是和控制器打交道。它要对设备做什么，它会告诉控制器，而不是直接把指令发给设备。然后控制器再去负责处理这件事情，它会去指挥设备执行命令，而 CPU 就不用管剩下的事情。控制器替它去完成剩下的事情，事情办完了再通知 CPU。否则，让 CPU 去盯着每一个设备做每一件事情，那是不现实的。

那么 Hub 是什么？在大学里，有的宿舍里网口有限，所以会有网口不够用的情况出现，于是有人会使用 Hub，让多个人共用一个网口，这是以太网上的 Hub。而 USB 的世界里同样有 Hub，其实原理是一样的，任何支持 USB 的计算机不会只允许你只能一个时刻使用一个 USB 设备，比如，你插入了 U 盘，同样还可以插入 USB 键盘，然后再插一个 USB 鼠标，因为你会发现你的计算机里并不只是一个 USB 接口。这些接口实际上就是所谓的 Hub 口。

而现实中经常是一个 USB 控制器和一个 Hub 绑定在一起，专业一点称为“集成”，而这个 Hub 也被称做 Root Hub。换而言之，和 USB 控制器绑定在一起的 Hub 就是系统中最根本的 Hub，其他的 Hub 可以连接到它这里，然后可以延伸出去，外接别的设备，当然也可以不用别的 Hub，让 USB 设备直接接到 Root Hub 上。

而 USB 连接指的就是连接 USB 设备和主机（或 Hub）的四线电缆。电缆中包括 VBUS（电源线）、GND（地线）和两根信号线。USB 系统就是通过 VBUS 和 GND 向 USB 设备提供电源的。主机对连接的 USB 设备提供电源供其使用，而每个 USB 设备也能够有自己的电源，如图 1.6.2 所示。

现在，如图 1.6.3 所示的 USB 大树里只有 Compound Device 还没有说。那么，Compound Device 又是什么样的设备？其实，在 USB 的世界里，不仅仅有 Compound Device，还有 Composite Device，简单的中文名字已经无法形象地表达它们的区别。正如图 1.6.3 所示，Compound Device 是将 Hub 和连在 Hub 上的设备封装在一起所组成的设备。而 Composite Device 则是包含彼此独立的多个接口的设备。从主机的角度看，一个 Compound Device 和单独的一个 Hub，然后连接了多个 USB 设备是一样的，它里面包含的 Hub 和各个设备都会有自己独立的地址，而一个 Composite Device 里不管有多少接口，它都只有一个地址。

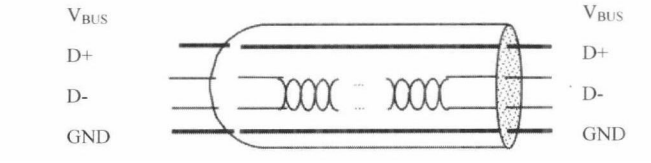

图 1.6.2　USB 四线电缆

USB 大树要想茁壮成长，离不开 USB 协议。USB 总线是一种轮询式总线。协议规定所有的数据传输都必须由主机发起，由主机控制器初始化所有的数据传输，各种设备紧紧围绕在主机周围。

USB 通信最基本的形式是通过 USB 设备中一个叫 Endpoint（端点）的东西，而主机和端点之间的数据传输是通过 Pipe（管道）。

端点就是通信的发送点或者接收点，要发送数据，只需把数据发送到正确的端点就可以了。而管道，实际上只是为了让我们能够找到端点，就相当于我们日常说的邮编地址。

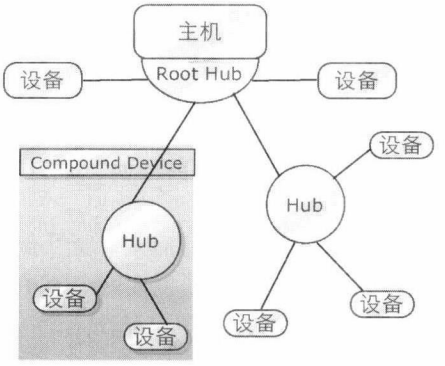

图 1.6.3　Compound Device

比如一个国家，为了通信，我们必须给各个地方取名，然后给各条大大小小的路取名字。严格来说，管道的另一端应该是 USB 主机，USB 协议也是这么说的，协议说管道代表着在主机和设备上的端点之间移动数据的能力。

端点不但是有方向的，而且这个方向还是确定的，要么是 in，要么是 out，没有既是 in 又是 out 的，都是生来就注定的。

有没有特殊的端点呢？看你怎么去理解 0 号端点了，协议规定了，所有的 USB 设备必须具有端点 0，它可以作为 in 端点，也可以作为 out 端点。USB 系统软件利用它来实现默认的控制管道，从而控制设备。

端点也是限量供应的，不是想要多少就有多少，除了端点 0，低速设备最多只能拥有两个端点，高速设备也最多只能拥有 15 个 in 端点和 15 个 out 端点。这些端点在设备内部都有唯一的端点号，这个端点号是在设备设计时就已经指定的。

为什么端点 0 就特殊呢？这还是有内在原因的。管道的通信方式其实有两种：一种是 stream 的，一种是 message 的。message 管道要求从它那儿过的数据必须具有一定的格式，不是随便传的，因为它主要就是用于主机向设备请求信息的，必须得让设备明白请求的是什么。而 stream 管道就没这么苛刻，随和多了，对数据没有特殊的要求。协议中规定，message 管道必须对应两个相同号码的端点：一个用来 in，一个用来 out，默认管道就是 message 管道。当然，与默认管道对应的端点 0 就必须是两个具有同样端点号 0 的端点。

USB 端点有四种类型，分别对应了四种不同的数据传输方式。它们是控制传输（Control Transfers）、中断传输（Interrupt Data Transfers）、批量传输（Bulk Data Transfers）和等时传输（Isochronous Data Transfers）。

- 控制传输用来控制对 USB 设备不同部分的访问，通常用于配置设备，获取设备信息，发送命令到设备，或者获取设备的状态报告。总之，就是用来传送控制信息的，每个 USB 设备都会有一个名为“端点 0”的控制端点，内核中的 USB Core 使用它在设备插入时进行设备的配置。
- 中断传输用来以一个固定的速率传送少量的数据，USB 键盘和 USB 鼠标使用的就是这种方式，USB 的触摸屏也是使用这种方式，传输的数据包含了坐标信息。

- 批量传输用来传输大量的数据，确保没有数据丢失，但不保证在特定的时间内完成。U 盘使用的就是批量传输，用它备份数据时需要确保数据不能丢，而且也不能指望它能在一个固定的比较快的时间内复制完。
- 等时传输同样用来传输大量的数据，但并不保证数据是否到达，以稳定的速率发送和接收实时的信息，对传送延迟非常敏感，显然是用于音频和视频一类的设备。这类设备期望能够有一个比较稳定的数据流，比如在用 QQ 视频聊天时，肯定希望每分钟传输的图像/声音速率是比较稳定的，不能说这一分钟前对方看到你在向她或向你深情表白，可是下一分钟却看见画面停滞在那里，只能看到你在那里一动不动，这不是浪费感情吗？

如图 1.6.1 所示的树形结构描述的是实实在在的物理拓扑，对于内核中的实现来说，没有这么复杂，所有的 Hub 和设备都被看做是一个个的逻辑设备（Logical Device），如图 1.6.4 所示，好像它们本来就直接连接在 Root Hub 上一样。

如图 1.6.5 所示，一个 USB 逻辑设备就是一系列端点的集合，它与主机之间的通信发生在主机上的一个缓冲区和设备上的一个端点之间，通过管道来传输数据。也就是说，管道的一端是主机上的一个缓冲区，一端是设备上的端点。

那么图 1.6.5 中的接口又是指什么？简单地说，USB 端点被捆绑为接口（Interface），一个接口代表一个基本功能。有的设备具有多个接口，像 USB 扬声器就包括一个键盘接口和一个音频流接口。在内核中，一个接口要对应一个驱动程序，USB 扬声器在 Linux 里就需要两个不同的驱动程序。到目前为止，一个设备可以包括多个接口，一个接口可以具有多个端点，当然以后我们会发现并不仅仅止于此。

图 1.6.4　USB 逻辑拓扑结构

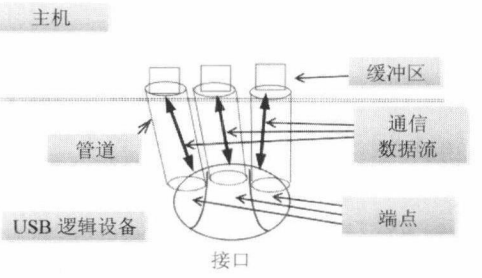

图 1.6.5　USB 数据通信

7. 我是谁

我是谁？USB 也一遍一遍地问着自己，当然它不会真的是一棵树，它也不会是太阳，Linux 里没有太阳，真要有的话，也只能是 Linus。USB 子系统只是 Linux 庞大家族里的一个小部落，主机控制器是它们的族长，族里的每个 USB 设备都需要被系统识别，被我们识别，而 sysfs 就是它们对外的窗口，我们可以从 sysfs 里了解认识每一个 USB 设备。以一个仅包含一个 USB 接口的 USB 鼠标为例，如图 1.7.1 所示，就是该设备对应的 sysfs 目录树。

```
/sys/devices/pci0000:00/0000:00:09.0/usb2/2-1
|-- 2-1:1.0
|   |-- bAlternateSetting
|   |-- bInterfaceClass
|   |-- bInterfaceNumber
|   |-- bInterfaceProtocol
|   |-- bInterfaceSubClass
|   |-- bNumEndpoints
|   |-- detach_state
|   |-- iInterface
|   `-- power
|       `-- state
|-- bConfigurationValue
|-- bDeviceClass
|-- bDeviceProtocol
|-- bDeviceSubClass
|-- bMaxPower
|-- bNumConfigurations
|-- bNumInterfaces
|-- bcdDevice
|-- bmAttributes
|-- detach_state
|-- devnum
|-- idProduct
|-- idVendor
|-- maxchild
|-- power
|   `-- state
|-- speed
`-- version
```

图 1.7.1　USB 鼠标的 sysfs 目录树

其中：

```
/sys/devices/pci0000:00/0000:00:09.0/usb2/2-1
```

表示鼠标。

下层目录：

```
/sys/devices/pci0000:00/0000:00:09.0/usb2/2-1/2-1:1.0
```

表示鼠标的 USB 接口。sysfs 里 USB 设备都是类似的表示，设备的目录下包括表示设备接口的目录。目录里的各个文件表示设备或接口的描述，大都对应了设备描述符、接口描述符等相应值，可以通过这些值获得您感兴趣的信息。什么是设备描述符和接口描述符？我们这里要暂时忽略它的存在，先关心关心 USB 设备在 sysfs 里是如何命名的，弄清它是谁，也就是说，弄清上面路径的含义。

USB 系统中的第一个 USB 设备是 Root Hub，前面已经说了它是和主机控制器绑定在一起的。这个 Root Hub 通常包含在 PCI 设备中，是连接 PCI 总线和 USB 总线的 bridge，控制着连接到其上的整个 USB 总线。所有的 Root Hub，内核的 USB Core 都分配有独特的编号，在上面的例子里就是 USB 2。

USB 总线上的每个设备都以 Root Hub 的编号作为其名字的第一个号码。这个号码后跟着一个“-”字符，以及设备所插入的端口号。因此，上面例子中的 USB 鼠标的设备名就是 2-1。因为该 USB 鼠标具有一个接口，导致了另外一个 USB 设备被添加到 sysfs 路径中。因为物理 USB 设备和单独的 USB 接口在 sysfs 中都将表示为单独的设备。USB 接口的命名是设备名直到该接口，上面就是 2-1 后面跟一个“:”和 USB 配置（Configuration）的编号，然后是一个“.”和该接口的编号。因此，上面的鼠标 USB 接口就是 2-1:1.0，表示使用的是第一个配置，接口编号为 0。

sysfs 并没有展示 USB 设备的所有部分，设备可能包含的可选配置都没有显示，不过这些

可以通过 usbfs 找到，该文件系统被挂在/proc/bus/usb 目录中，从/proc/bus/usb/device 文件可以知道系统中存在的所有 USB 设备的可选配置。

这里既然提到了 USB 设备的配置，还是先简要说说。一个设备可以有一种或者几种配置，这能理解吧？没见过具体的 USB 设备？那么手机见过吧，每部手机都会有多种配置，或者说“设定”。比如，笔者的这款 Nokia 6300 手机，手机语言可以设定为 English、繁体中文、简体中文，一旦选择了其中一种，那么手机里显示的所有信息都是该种语言/字体。再举一个最简单的例子，手机的操作模式也有好几种，标准、无声、会议等。如果我设为“会议”模式，那么就是只振动不发声：要是设为“无声”模式，那么就什么动静也不会有。USB 设备的配置也是如此，不同的 USB 设备当然有不同的配置了，或者说需要配置哪些东西也会不一样。

8. 好戏开始了

首先要去 drivers/usb 目录下走一走、看一看。

```
atm/  class/  core/  gadget/  host/  image/  misc/  mon/  serial/  storage/
Kconfig  Makefile  README usb-skeleton.c
```

ls 命令的结果就是上面的 10 个目录和 4 个文件。usb-skeleton.c 是一个简单的 USB driver 的框架。那么首先应该关注什么？那就是 Kconfig、Makefile、README。

README 里有关于这个目录下内容的一般性描述。再说了，面对“读我吧！读我吧！”这么热情奔放的呼唤，善良的我们是不可能无动于衷的，所以先来看一看 README 里面都有些什么内容。

```
Here is a list of what each subdirectory here is, and what is contained in
them.

core/       - This is for the core USB host code, including the
          usbfs files and the hub class driver ("khubd").

host/       - This is for USB host controller drivers.  This
          includes UHCI, OHCI, EHCI, and others that might
          be used with more specialized "embedded" syste ms.

gadget/     - This is for USB peripheral controller drivers and
          the various gadget drivers which talk to them.


Individual USB driver directories.  A new driver should be added to the
first subdirectory in the list below that it fits into.

image/      - This is for still image drivers, like scanners or
          digital cameras.
input/      - This is for any driver that uses the input subsystem,
          like keyboard, mice, touchscreens, tablets, etc.
media/      - This is for multimedia drivers, like video cameras,
          radios, and any other drivers that talk to the v4l
          subsystem.
net/    - This is for network drivers.
serial/     - This is for USB to serial drivers.
storage/- This is for USB mass-storage drivers.
```

```
class/      - This is for all USB device drivers that do not fit
          into any of the above categories, and work for a range
          of USB Class specified devices.
misc/       - This is for all USB device drivers that do not fit
          into any of the above categories.
```

drivers/usb/README 文件描述了前面使用 ls 命令列出的那 10 个文件夹的用途。那么什么是 USB Core？Linux 内核开发人员们专门写了一些代码，负责实现一些核心的功能，为别的设备驱动程序提供服务，比如申请内存，实现一些所有的设备都会需要的公共函数，并美其名曰为“USB Core”。

时代总在发展，早期的 Linux 内核，其结构并不是如今天这般有层次感，远不像今天这般错落有致，那时候 drivers/usb/目录下放了很多文件，USB Core 与其他各种设备驱动程序代码都堆砌在这里，后来，在 drivers/usb/目录下面出来了一个 core 目录，就专门放一些核心的代码，比如初始化整个 USB 系统，初始化 Root Hub，初始化主机控制器的代码，再后来甚至把主机控制器相关的代码也单独建了一个目录，叫 host 目录。这是因为 USB 主机控制器随着时代的发展，也开始有了好几种，不再像刚开始那样只有一种。所以，设计者们把一些主机控制器公共的代码仍然留在 core 目录下，而一些各主机控制器单独的代码则移到 host 目录下面负责各种主机控制器的人去维护。

那么 USB gadget 呢？gadget 说白了就是配件的意思，主要就是一些内部运行 Linux 的嵌入式设备，比如 PDA，设备本身有 USB 设备控制器（USB Device Controller），可以将 PC，也就是我们的主机作为 master 端，将这样的设备作为 slave 端和主机通过 USB 进行通信。从主机的观点来看，主机系统的 USB 驱动程序控制插入其中的 USB 设备，而 USB gadget 的驱动程序控制外围设备作为一个 USB 设备和主机通信。比如，我们的嵌入式主板上支持 SD 卡，如果我们希望将主板通过 USB 连接到 PC 之后，这个 SD 卡被模拟成 U 盘，那么就要通过 USB gadget 架构的驱动。

gadget 目录下大概可以分为两个模块：一个是 udc 驱动，这个驱动是针对具体 CPU 平台的，如果找不到现成的，就要自己实现；另外一个就是 gadget 驱动，主要有 file_storage、ether、serial 等。另外还提供了 USB gadget API，即 USB 设备控制器硬件和 gadget 驱动通信的接口。PC 及服务器只有 USB 主机控制器硬件，它们并不能作为 USB gadget 存在，而对于嵌入式设备，USB 设备控制器常被集成到处理器中，设备的各种功能，如 U 盘、网卡等常依赖这种 USB 设备控制器来与主机连接，并且设备的各种功能之间可以切换，比如可以选择作为 U 盘或网卡等。

剩下的几个目录分门别类地放了各种 USB 设备的驱动，U 盘的驱动在 storage 目录下，触摸屏和 USB 键盘鼠标的驱动在 input 目录下等。另外，在 USB 协议中，除了通用的软硬件电气接口规范等，还包含各种各样的 Class 协议，用来为不同的功能定义各自的标准接口和具体的总线上的数据交互格式和内容。这些 Class 协议的数量非常多，比如最常见的支持 U 盘功能的 Mass Storage Class，以及通用的数据交换协议 CDC Class。此外，还包括 Audio Class、Print Class 等。理论上讲，即使没有这些 Class，通过专用驱动也能够实现各种各样的应用功能。但是，正是 Mass Storage Class 的使用，使得各个厂商生产的 U 盘都能通过操作系统自带的统一驱动程序来使用，对 U 盘的普及起了极大的推动作用，制定其他的 Class 也是同样的目的。

我们响应了 README 的呼唤，它给予了我们想要的，通过它，我们了解了 USB 目录里的那些文件夹都有着什么样的角色。到现在为止，就只剩下 Kconfig 和 Makefile 两个文件了，它们又扮演着什么样的角色？就好像我吃东西时总是喜欢把好吃的留在最后享受一样，我也习惯