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。

2012 年 3 月第 1 次印刷

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。

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。

# 前 言

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，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。，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**

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

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

# USB Core

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

USB 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。

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）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。

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

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。

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。

## 2．它从哪里来············································2

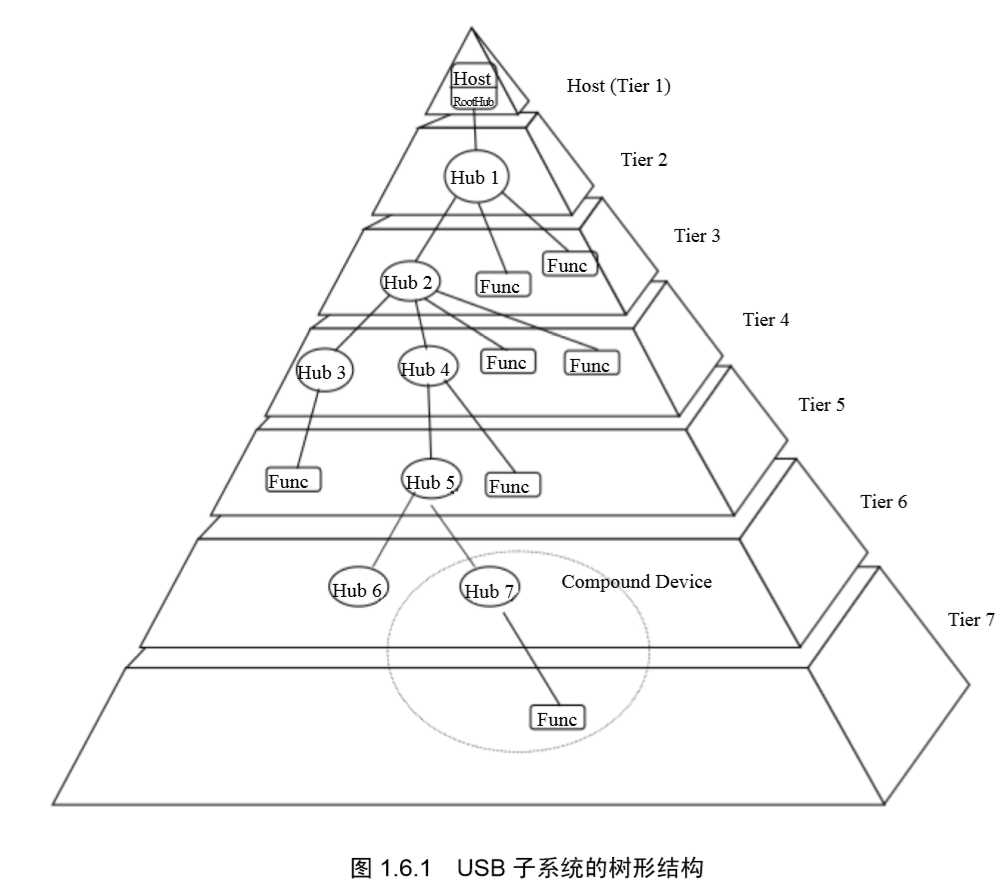
## 3．PK··························································2

## 4．漫漫辛酸路············································3

## 5．我型我秀

## 6．我是一棵树

USB 子系统的拓扑也是一棵树，它并不以总线的方式来部署



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） 。

## 7．我是谁····················································7

## 8．好戏开始了············································9

## 9．不一样的 Core·····································11

## 10．从这里开始········································14

## 11．面纱····················································17

## 12．模型，又见模型································19

## 13．繁华落尽············································23

## 14．接口是设备的接口····························24

## 15．设置是接口的设置····························28

## 16．端点····················································30

## 17．设备····················································32

## 18．配置····················································38

## 19．向左走，向右走································41

## 20．设备的生命线（一） ························45

## 21．设备的生命线（二） ························48

## 22．设备的生命线（三） ························52 2

## 3．设备的生命线（四） ························57

## 24．设备的生命线（五） ························63

## 25．设备的生命线（六） ························69

## 26．设备的生命线（七） ························75

## 27．设备的生命线（八） ························81

## 28．设备的生命线（九） ························86

## 29．设备的生命线（十） ························89

## 30．设备的生命线（十一）·····················94

## 31．驱动的生命线（一） ······················105

## 32．驱动的生命线（二） ······················110

## 33．驱动的生命线（三） ······················113

## 34．驱动的生命线（四） ······················117

## 35．字符串描述符··································119

## 36．接口的驱动······································127

## 37．还是那个 match·······························129

## 38．结束语··············································134

# 附录 A – sysfs

# 附录 B Linux 内核高效学习法

既然有高效，相对的就有低效。学习本身就是一件很玄乎的事情，有些人整天潇潇洒洒没 见怎么用心就能够获得很好的成绩，而有些人则相反，即使头悬梁锥刺骨也还是成绩平平收获 无几。这里面很大一部分的原因就是学习的方法

，因为基于每个人不同的情况，并没有那样一个标准的 方法存在，所以讲起来就很容易成为一场大忽悠。找到自己最为合适的方法才是最好的。很像一句句的口号，我们也可以将它们看做内核学习的大字 报。

## B.1 高效学习 Linux 内核

首先是第一句话：把内核当朋友。

我们只有把内核当朋友，把它放在对等的地位上，而不仅 仅是一堆死气沉沉的代码，我们才能够更好地认识和理解到它的精髓。

然后是第二句话：先学会使用它。 意思就是我们在学习内核前首先要会用 Linux，依照一个由上至下循序渐进的过程，在能 够熟练地使用 Linux 操作系统之后再去研究内核中的实现。这也是 linus 本人的观点。

第三句是依照四个层次进行内核学习。 笛卡儿在 17 世纪的某一天，闲极无聊写了这么一本书，书名就叫《方法论》，在这本目前 来说绝大部分人都不知道的书里将方法上升到了理论的高度。笛卡儿在他的这本书里将研究问 题的方法归纳为简单的一句话，就是“复杂问题要简单化”。就是说要将复杂的问题分解为很多 个简单的小问题，一个个地分开解决。这句话当然可以借鉴运用到内核的学习上，不过需要做 些改动，不是分解为多个简单的小问题，而是将内核学习这么一件很复杂的事情划分为由低到 高多个不同的层次，每一层次都有自己需要达到的目标和要求。这也是我自己认为比较好的认 识学习内核的方法。

第四句是走出心理误区。 对于学习这种复杂的事情来说，无论是我们在学校的课堂学习，还是这里说的内核学习， 它的效果好与坏，主要取决于两个方面：一个是学习的方法，另一个就是学习时的心理。注 意，在这儿我无视了智商的差异，智商这玩意儿太玄了，可以将它归于迷信的范畴。而我们在 学习时经常会产生一系列的问题或者说误区，只有走出这些误区，在学习中养成一个坚强的心 理，我们才能够真正做到高效。

第五句是使用 vim+cscope+ctags 浏览内核源码。 其实这句话更主要的意思是说我们需要一个好的工具去浏览内核的代码。在 Windows 下 面，我们或许可以很容易地找到很多比较好的 IDE，用来浏览代码，比如 source insight，它可 以很方便地在代码之间进行关联阅读。但是对于 Linux 新人来说，有没有一个功能类似的浏览 代码的工具就成为一个很常见的问题。

第六句是使用 kernel 地图定位目标代码。 应该说学习内核就是学习内核的源代码，但是内核代码千千万，又到处像个迷宫一样，不 迷路都很难，又怎么去直面它？这时我们就需要这样的一幅内核地图来帮助我们去定位要分析 的目标代码，并缩小目标代码的范围与代码量。

接下来是第七句话：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，可总是觉得分析完浏览完 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，去理解每一段代 码的实现，多问多想多记，而不是抱着走马观花、得过且过的态度，终必然会有很大的收获。

后一句是：以内核源码为中心，坚持学习资源建设。 在我们内核学习的过程中，内核源码本身就是好的参考资料，其他任何经典或非经典的 书多只是起到个辅助作用，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。 除了这里的八句话，其他的可能会对大家有帮助的感悟或者方法还有很多。

## B.2 Kernel 地图：Kconfig 与 Makefile

基本上， Linux 内核中每一个目录下边都会有一个 Kconfig 文件和一个 Makefile 文件。 对于一个希望能 够在 Linux 内核的汪洋代码里看到一丝曙光的人来说

去香港，通过海关的时候，总会有免费的地图和各种指南拿，有了它们在手里我们才 不至于无头苍蝇般迷茫地行走在陌生的街道上。即使在内地出去旅游的时候一般来说也总是会 首先找份地图，当然了，这时就是要去买了，拿是拿不到的。不同的地方有不同的特色， 只不 过有的特色是服务，有的特色是索取

比如我们打算研究 U 盘驱动的实现，因为 U 盘是一种 storage 设备，所以我们应该先进入 到 drivers/usb/storage/目录。但是该目录下的文件很多，那么究竟哪些文件才是我们需要关注 的？

## B.3 分析内核源码如何入手·················423

### B.3.1 分析 README····················423

### B.3.2 分析 Kconfig 和 Makefile····425

### B.3.3 态度决定一切：从初始化函数 开始······································427

## B.4 内核学习的心理问题·····················432

## B.5 高效学习 Linux 驱动开发··············433

## B.6 设备模型（上）·····························434

## B.7 设备模型（下）·····························438

### B.7.1 内核中 USB 子系统 的结构··································438

### B.7.2 USB 子系统与设备模型······440

## B.8 驱动开发三件宝·····························440