# 如何实现Linux下的U盘（USB Mass Storage）驱动

## 版本：v0.7

# How to Write Linux USB MSC (Mass Storage Class) Driver


## Crifan Li



### 摘要

本文主要介绍了USB Mass Storage的相关的各种协议之间的关系，以及如何在Linux的USB驱动框架下实现U盘驱动


### 本文提供多种格式供：

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 在线阅读 | HTML [1] | HTMLs [2] | PDF [3] | CHM [4] | TXT [5] | RTF [6] | WEBHELP [7] |
| 下载（7zip压缩包） | HTML [8] | HTMLs [9] | PDF [10] | CHM [11] | TXT [12] | RTF [13] | WEBHELP [14] |

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/html/usb_disk_driver.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/usb_disk_driver/

| 修订历史 | | |
|---|---|---|
| 修订 0.4 | 2011-07-01 | crl |
| 1. 介绍如何在Linux下实现U盘驱动 | | |
| 修订 0.7 | 2013-09-05 | crl |
| 1. 通过Docbook发布 | | |

[1] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/html/usb_disk_driver.html
[2] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/pdf/usb_disk_driver.pdf
[4] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/chm/usb_disk_driver.chm
[5] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/txt/usb_disk_driver.txt
[6] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/rtf/usb_disk_driver.rtf
[7] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/html/usb_disk_driver.html.7z
[9] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/pdf/usb_disk_driver.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/chm/usb_disk_driver.chm.7z
[12] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/txt/usb_disk_driver.txt.7z
[13] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/rtf/usb_disk_driver.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/usb_disk_driver/release/webhelp/usb_disk_driver.webhelp.7z

2. 更新所有xml:id

# 如何实现Linux下的U盘（USB Mass Storage）驱动: How to Write Linux USB MSC (Mass Storage Class) Driver

Crifan Li

版本：v0.7

出版日期 2013-09-05


[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

# 目录

# 插图清单

# 缩略词

| | |
|---|---|
| MSC (MSC) | Mass Storage Class<br>大容量存储类型<br>常说的大容量存储设备，就是此处的MSC设备，最常见的例子就是U盘 |
| SAM4 (SAM4) | ISO/IEC 14776-414, SCSI Architecture Model-4 (SAM-4) (ANSI INCITS 447:2008)<br>SCSI架构的Mode-4 |
| Spec (Spec) | Specification<br>规范 |

# 正文之前

## 1. 此文目的

关于U盘，估计大家都用过。

比如，笔者手上的宇瞻AH320的8G的U盘：

**图 1. U盘**


最常见的用法就是，直接将此8GU盘插到电脑的USB口上，然后系统（Windows的XP或者Linux）就会自动检测到你的U盘然后生成一个移动盘符，然后你就可以打开对应盘符，读写文件数据了。

而此文呢，目的就是，要搞懂，作为驱动开发者来说，对于这样一个U盘，如何在Linux平台下，去实现U盘驱动，即USB Mass Storage驱动，实现驱动时，需要做哪些事情，以及如何去实现这些事情。

关于USB，其实网上也有不少相关的文章，但是笔者觉得太多帖子，很多帖子，也只是介绍USB协议，而如何在Linux下面实现驱动，却很少提及。或者说是，理论多，实践少，东一块，西一块，很少能把相关知识有机的结合起来，尤其是软件，硬件，系统框架等结合起来一起说明的，导致看了很多这样的帖子，还是似懂非懂。

关于USB或者说多数计算机方面的技术文章，如果有说得明白的，往往都是老外写的。

所以，为了实现有中文的帖子，也能把问题说明白，所以才有此文的诞生。

所以，简述此文目的：

1. 首先，算为自己学习USB的过程，做个记录和总结，以备后查。
2. 对于其他不懂Linux和USB的人，看了此文后，可以对Linux，USB等有个基本的认识。
3. 对于了解Linux和USB的人，搞开发的人，尤其是Linux下USB驱动开发的人，看了此文后，真正能搞懂Linux下USB的Mass Storage的框架，和自己去实现对应的U盘驱动的时候，数据读写的前后流程，而其中，系统做了哪些事情，需要我们自己做哪些事情。

总的说来，本人写任何帖子，要么不写，要么就写的逻辑清晰，让人看得明白。

就像某人说的，看了我写的东西，能达到“醍醐灌顶”的效果，这才是我写东西的终极目标。

## 2. 阅读此文所需要的前提知识

尽可能使得大家不需要有太多的基础，也能看懂此文。

即不需要对USB以及USB Mass Storage以及Linux有太多知识，然后看了此文，看了此文后，大概清楚这三者是什么，然后想要在Linux下面实现USB Mass Storage的驱动的话，自己需要做哪些事情，以及大概怎么做。

而如果想要实现真正的Linux下U盘驱动开发，那么最基本的一些知识，包括Linux是啥，USB大概是啥，之类的，那你至少多少有些了解，这也才能看懂后续内容。

# 3. 声明

此文不可能面面俱到，但是旨在把问题说清楚的目的，又会在具体内容方面，尽量做到面面俱到，因此，也会使得整个文章显得很臃肿，不过也因此方便了，不了解的人去看懂后续所要解释的内容。

提及一下，文中一些内容的表述，是中英文掺杂，主要是因为有些含义，用英文表述更加贴切，就懒得再去费时费力翻译为中文了。



水平有限，难免有误，欢迎任何意见和建议：admin (At) crifan.com

# 第 1 章 USB基本知识

了解计算机行业的技术，最好的资料，是官方的规范，其实就是一堆文档，多数是PDF格式的文件。

关于USB的基础知识，不像其他多数协议，只是和软件相关，USB协议，总的来说，应该是涉及非常多的规范和协议，涉及太多的软件，硬件，机械等方方面面。

之所以出现USB协议涉及内容太多太广，看起来太繁杂。

关于USB相关的基础知识，可以参考笔者的另一篇文章：USB基础知识概论[1]

## 1.1. USB的硬件

## 1.2. USB相关的协议

## 1.3. USB相关的软件实现

[1] http://www.crifan.com/files/doc/docbook/usb_basic/release/html/usb_basic.html

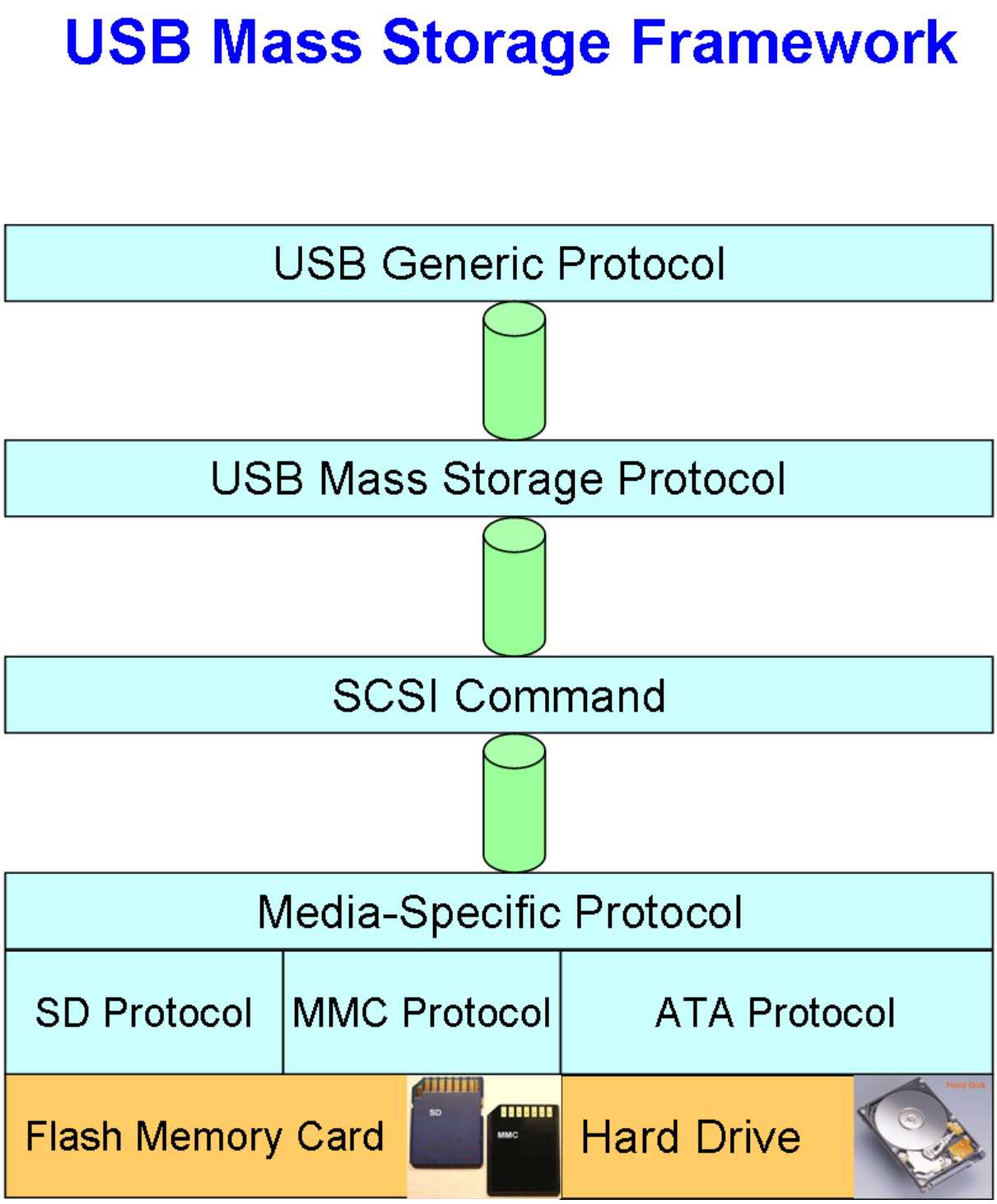
# 第 2 章 USB Mass Storage大容量存储的基本知识

【todo】待整理：Linux USB MASS Storage driver流程图[1]

USB Mass Storage所对应的层次和要实现哪些东西：

[1] http://www.yaabou.com/blog/?p=15

**图 2.1. USB Mass Storage Framework**




PC电脑和U盘之间的关系，以及物理上的组成，可以用下图表示：

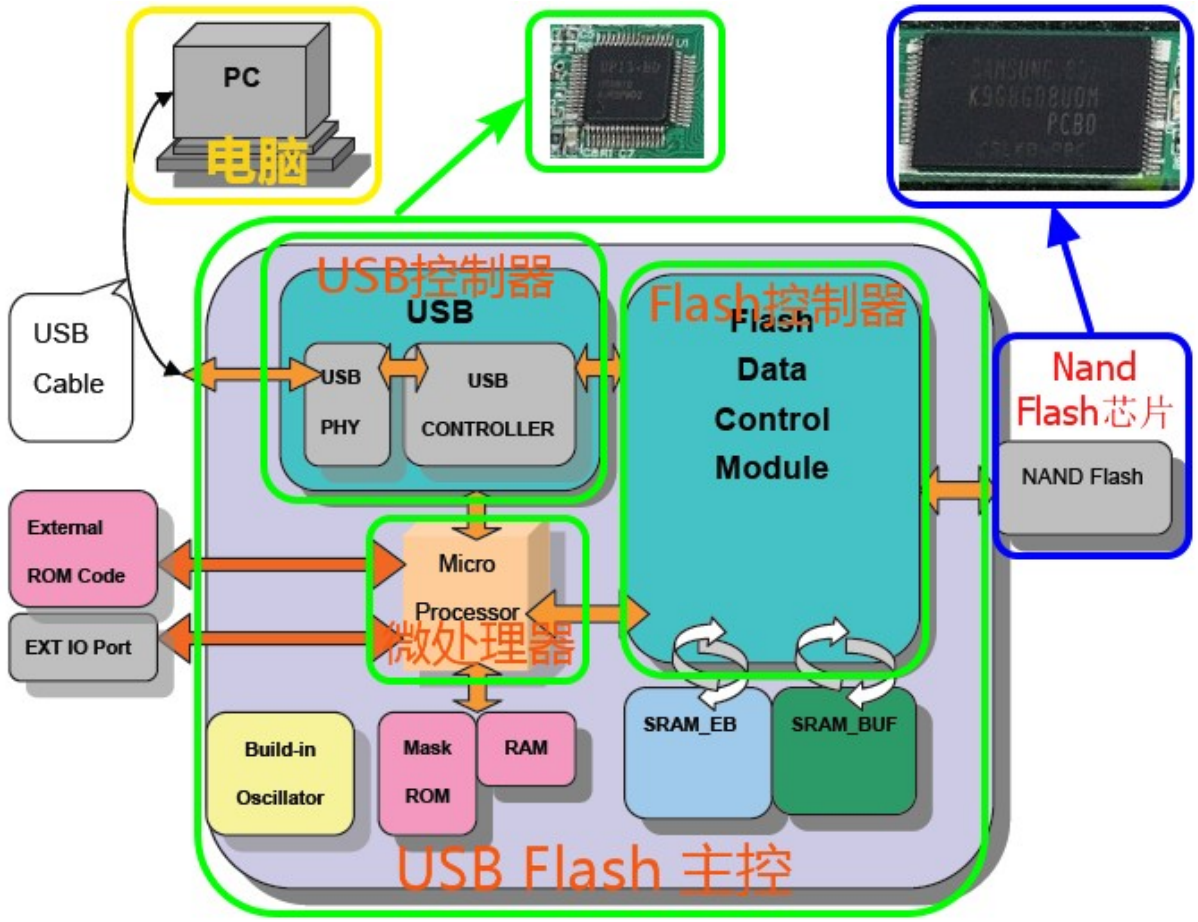
**图 2.2. PC和U盘**


更深入的剖析，对于普通U盘的内部结构，则是一个USB物理接口，加上对应的控制芯片（微控制器（含Nand Flash的控制器）+ USB设备控制器）和一个Nand Flash芯片：

**图 2.3. PC和U盘的芯片内部结构**


上述PC电脑和U盘的物理关系，以群联的PS2251-50 USB 2.0 Flash Controller为例，对应的逻辑关系为：

**图 2.4. PC和U盘的内部逻辑框图**



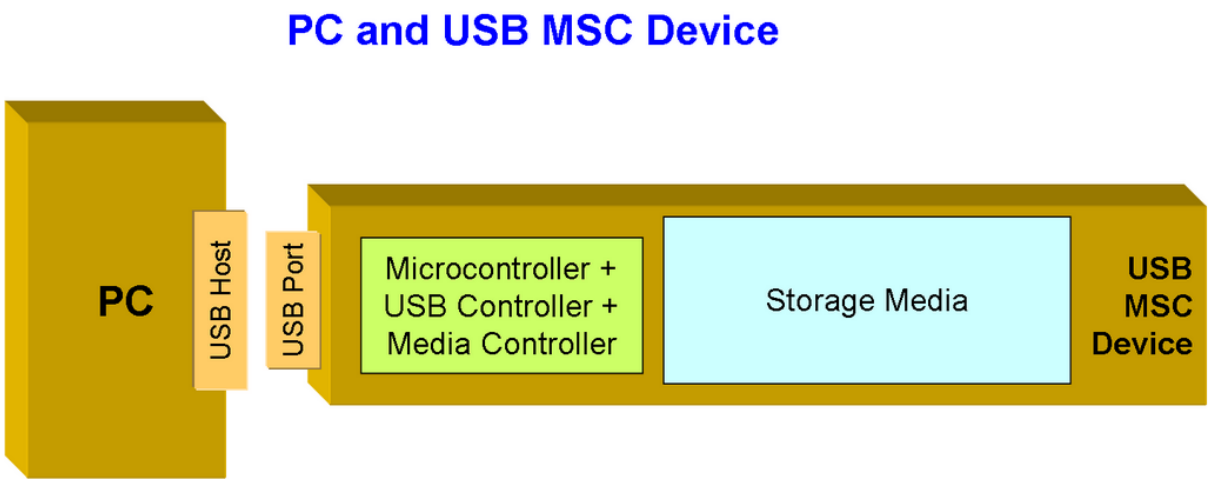

关于U盘容量，再多解释一句：

一般U盘的大小，就是对应着这个Nand Flash芯片的容量，比如2GB，4GB，8GB等。

当然，比如一个8GB的U盘，内部也可以用两块4GB的Nand Flash芯片来构成。

PC和U盘的之间的抽象的逻辑关系，可用下图来表示：

**图 2.5. PC和USB MSC设备**

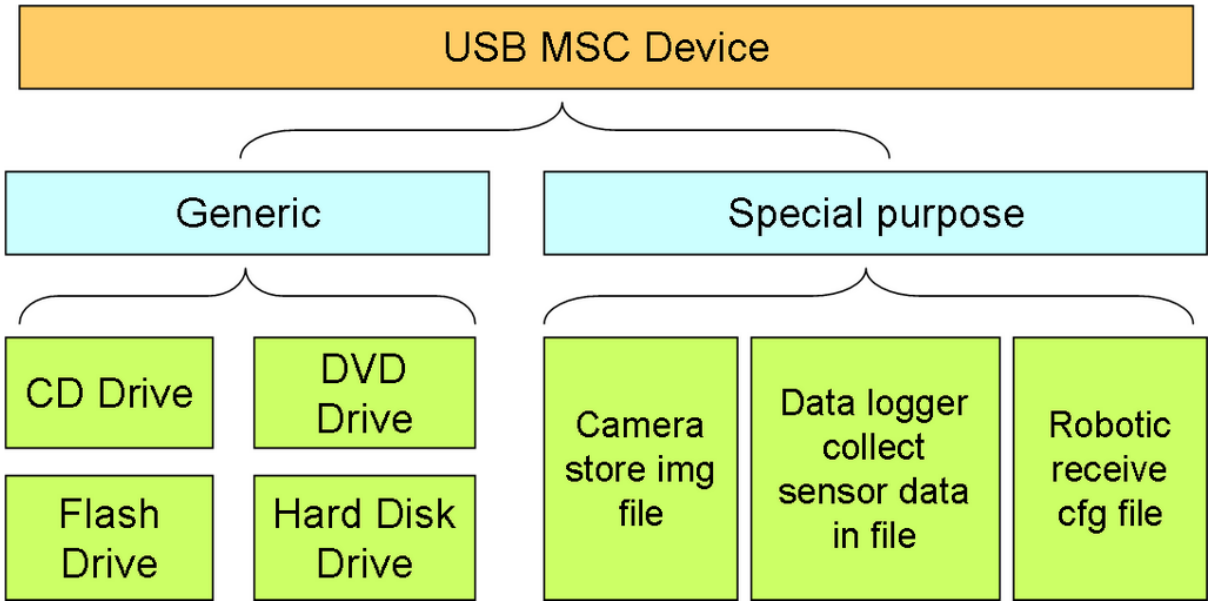



上图中，Storage Media，就是我们例子中的Nand Flash芯片。

而例子中的那个控制芯片，是Microcontroller with embedded USB device controller 和Media Controller的集合。

而上图中的USB MSC（Mass Storage Class） Device，从应用领域来说，可以分为以下几类：

**图 2.6. USB MSC的分类**




而像上述例子中那样的常用的U盘，属于上图中的Flash Drive，即，物理上存储数据的介质用的是Flash Memory，比如例子中的Nand Flash芯片，对应的，Media Controller，也就是Nand Flash的Controller，负责从Nand Flash芯片中读写数据。

USB MSC设备中的固件（firmware）或者硬件（hardware），必须要实现下面这些功能：

1. 检测和响应通用的USB Request和USB总线上的事件。
2. 检测和响应来自USB设备的关于信息或者动作的USB Mass Storage Request。
3. 检测和响应，从USB Transfer中获得的SCSI Command。这些业界标准的命令，是用来获得状态信息，控制设备操作，向存储介质块中读取（read block）和写入（write block）数据的。

另外，设备如果想要向存储介质中，创建/读取/写入，文件/文件夹的话，那么就涉及到文件系统，还要实现对应的文件系统。嵌入式系统中常见的文件系统有FAT16或FAT32。

# 2.1. USB Mass Storage相关的协议

除了本身USB的协议之外，Mass Storage作为其中USB的一种，USB Mass Storage自己又有相关的协议，对应的协议可以去官网下载：

http://www.usb.org/developers/devclass_docs/

中有关于Mass Storage相关的，一堆的协议：

- Mass Storage Class Specification Overview 1.4 [2]
- Mass Storage Bulk Only 1.0 [3]
- Mass Storage Control/Bulk/Interrupt (CBI) Specification 1.1 [4]

[2] http://www.usb.org/developers/devclass_docs/Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf
[3] http://www.usb.org/developers/devclass_docs/usbmassbulk_10.pdf
[4] http://www.usb.org/developers/devclass_docs/usb_msc_cbi_1.1.pdf

- Mass Storage UFI Command Specification 1.0 [5]
- Mass Storage Bootability Specification 1.0 [6]
- Lockable Mass Storage Specification 1.0 and Adopters Agreement - Lockable Mass Storage IP Disclosure [7]
- USB Attached SCSI Protocol (UASP) v1.0 and Adopters Agreement [8]

说实话，咋一看，这么多协议，不知道驴年马月才能看完和真正理解。

不过，等待了解了这些协议的关系之后，会发现，其实需要特别关注和研究的协议，并不是很多，至少很多协议可以不用太关心。

# 2.1.1. USB Mass Storage相关协议简介

凡事至少得对整体系统有个大致了解后，才能继续下一步的深入的开发。

所以，我们的目的，首先是要搞懂这么多协议之间都是啥关系，以及具体写U盘驱动的话，要看哪些协议。

从上面那一堆协议的名词上，我们就能看到，第一个：

Mass Storage Class Specification Overview 1.4

就是对于这么多协议的概述，其中介绍了各个协议的关系。

下面就把其中的主要内容摘出来，解释如下：

关于USB Mass Storage相关的一些协议，都是由一个叫做USB Mass Storage Class Working Group (CWG)的组织定义的，如上所述，包括下面一些协议：

- USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport
- USB Mass Storage Class Bulk-Only (BBB) Transport
- USB Mass Storage Class Universal Floppy Interface (UFI) Command Specification
- USB Mass Storage Class Bootability Specification
- USB Mass Storage Class Compliance Test Specification
- USB Lockable Storage Devices Feature Specification (LSD FS)
- USB Mass Storage Class USB Attached SCSI Protocol (UASP)

对于这些协议，我们一个个的简单解释和分析一下：

## 2.1.1.1. USB MSC Control/Bulk/Interrupt (CBI) Transport

我们所关注的U盘，就是所谓的MSC设备，大容量存储设备。

U盘的功能，就是数据存储。而对应的数据传输，用的是USB中的Bulk Transfer。而Control Transfer是用来发送Class-Specific的信息和Clear Stall。而其他方面的信息交换，是用Bulk-only协议。

[5] http://www.usb.org/developers/devclass_docs/usbmass-ufi10.pdf
[6] http://www.usb.org/developers/devclass_docs/usb_msc_boot_1.0.pdf
[7] http://www.usb.org/developers/devclass_docs/Lockable_Mass_Storage_Spec_and_Adopters_Agreement.zip
[8] http://www.usb.org/developers/devclass_docs/uasp_1_0.zip

而对于CBI，算是Bulk-only的替代品，也可以用来信息交换，但是只能用于，full-speed的软盘（Floppy drive），也不推荐将CBI用于其它新出的MSC设备上。

【总结】

USB MSC Control/Bulk/Interrupt (CBI) 主要用于Floppy设备，对于我们关心的U盘，用不到，不需要太关心，可以忽略之。

## 2.1.1.2. USB MSC Bulk-Only (BBB) Transport

如上所述，Bulk-only是USB设备端，此处的U盘和USB Host端，即普通PC，之间信息交换的协议。

Bulk Only Transport，也被简称为BOT。

### 2.1.1.2.1. 为何USB MSC中Bulk-only Transport被叫做 BBB

USB MSC中的Bulk-Only 常被叫做BBB，是相对于前面所说的CBI来说的。

看起来，USB MSC的CBI，是Control/Bulk/Interrupt的简写，但是其具体含义是：

1. Control

   Control端点用于，除了传送标准的USB请求之外，还用于通过Class-specific的请求，将命令块（command block）传给设备；即Control端点传送命令块

2. Bulk

   Bulk In和Bulk Out端点用于在主机（Host）和设备（Device）之间数据的传输。即，Bulk用于传送数据

3. Interrupt

   Interrupt端点用于（设备向主机）通知命令完成。即Interrupt用于传送状态信息

因此，上面的USB MSC的Control/Bulk/Interrupt，才被简称为CBI。

和CBI中的用三种不同的端点来传送三种类型的信息，而不同的是，Bulk-only传送这些全部的信息，都之用Bulk端点。

即用Bulk端点来传送命令块，数据，状态，因此，才类似于Control/Bulk/Interrupt被简称为CBI一样，而Bulk/Bulk/Bulk被简称为BBB。

【总结】

USB MSC传输协议分CBI和BOT，而BOT又称为BBB。

### 2.1.1.2.2. 为何已经有了CBI，又再弄出个BBB

既然，对于USB MSC设备来说，USB设备和USB主机之间的通信，已经定义了一个CBI规范，那么为何还要再新定义一个Bulk-only（BBB）呢？

我的理解是，那是因为，最开始USB协议定义的时候，那时候市场上的Floppy Disk还是用的很多的。

所以针对Floppy设备特点，分别定义了多个端点来传输不同的信息，即Control端点传命令块，Bulk传数据，Interrupt传状态信息，

而后来计算机行业的发展了，Floppy类的设置很少用了，存储数据的话，多数开始用Flash Memory了，再加上通过合理规划，可以用同一种端点，即Bulk端点，传输上述三种信息，即命令块，数据，状态，

因此，只需要物理上实现一个Bulk端点，节省掉了其他两个端点：Control端点和Interrupt端点，达到了物理上实现起来方便和节省资源，而达到同样的信息传输的目的。

此部分理解，有待进一步考证。

【总结】

USB MSC Bulk-Only (BBB) 协议，是我们要重点关注的对象。因为我们的U盘和USB主机（PC端）直接信息交互，主要是用此协议。

## 2.1.1.3. USB MSC UFI Command Specification

UFI，即Universal Floppy Interface，因此看名字就知道，是关于软盘的。

此USB MSC UFI规范，定义了UFI命令集合（Command Set），设计出来此规范，就是用于软盘的。此UFI命令集合，是基于SCSI-2 和SFF-8070i命令集合的。

【总结】

看完上面的解释，就明白了，我们此处关心的是U盘，是USB Flash Memory类型的，不是Floppy Disk类型的，所以，此处也可以不关心，暂忽略之。

## 2.1.1.4. USB MSC Bootability Specification

目前常见的电脑启动，很多都是从MSC大容量存储设备中启动的，比如硬盘。

因此，设计了这个规范，以使得操作系统可以从USB MSC设备上启动。关于此规范的具体内容，主要是定义了一些命令和相关的一些数据的定义。

即，如果你想要实现让操作系统从你的这个MSC设备启动，那么你就要实现对应的CDB（Command Descriptor Block，命令描述符块）或者Data数据。

【总结】

我们主要关心U盘如何和主机数据交互，暂不关心是否能否从此U盘启动，所以也可不太关心，暂忽略之。

## 2.1.1.5. USB MSC Compliance Test Specification

【总结】

无须多解释，看名字就知道，是一个关于兼容性测试的规范，和我们此处所关心的U盘和主机的数据交互，关系不大，暂忽略之。

## 2.1.1.6. USB Lockable Storage Devices Feature Specification

“Lockable Storage Devices Feature Specification”，简称为 LSD FS。

“Lockable”的意思是，可锁定，即锁定以防止其他人访问或者写入，即变成只读，甚至不允许其他人再访问。

说白了就是对USB存储设备的安全控制方面的规范。

而基于目前还没有一个标准的规范去定义，如何去控制那些对于USB存储设备的访问，所以此处才定义了一个这么个和访问控制相关的协议。

此协议可实现允许主机Host或者设备Device去锁定Lock或者解锁Unlock 对应的存储设备。

【总结】

和安全存储和权限控制相关的协议，我们此处也是暂不关心，可忽略之。

## 2.1.1.7. USB MSC USB Attached SCSI Protocol (UASP)

“Attached”顾名思义，是附在某个上面的，此处即附在SCSI协议的上面的，即SCSI协议的补充部分。

UASP规范，定义了关于如何在USB 2.0和USB 3.0中，UAS的传输标准是如何实现的，并且给出了一些范例和一些推荐的做法。

### 2.1.1.7.1. 已有SCSI协议，为何还要再弄一个UASP

既然已经有了对应的SCSI协议，用于发送对应命令，实现对应功能。作为U盘等应用的话，直接实现对应的协议，符合对应的规范，不是也就可以实现对应功能了吗，为何还要另外再弄出一个SSCI的附属协议UASP？

那是因为，原先的BOT（Bulk Only Transport），虽然协议简单已实现，适合用于大容量存储设备中，但其就像一个单线程，不能同时并行执行多个传输。

即，对于BOT，每一个由Host发起数据传输（transaction），都必须等待设备完成，然后设备再返回对应的已完成状态信息，然后才能开始下一次数据传输。这样的话，对于整个数据传输过程的话，就造成了一个很大（大概有20%）的浪费（overhead）。

而对于USB 3.0来说，速度从USB 2.0的480Mb/s变成了 5.0Gb/s，而如果继续用BOT的话，那么相对来说CPU性能的利用率很低，USB传输速度也不太高，例如，有研究表明，2.4 GHz Core Duo™的CPU，利用率只要大概12%，CPU传输速度只有大约250MB/s，而USB 3.0的理论速度是5.0Gb/s=640MB/s，即还不到理论最大速度的一半。

因此，才有了这个UASP，对于SCSI协议进行了补充，以提高USB 2.0的USB总线利用率，和充分利用USB 3.0的全双工能力，可以使得传输速度达到大约400MB/s。此新的协议UASP的实现，也需要对应的新的Host端的软件，新的Device端的固件（Firmware）。

为了实现设备的向后兼容性，Device同时支持BOT和UAS。

而此UASP规范，定义了就是如何在USB 2.0和USB 3.0上实现对应的UAS协议。

当Host和Device都实现了此UAS协议的话，那么Host将通过Host端的SCSI Software Stack去访问Device，而USB的Interface也将从功能上看，变成Host Stack中的另外一个SCSI Host Adapter。Device需要实现SAM4的架构模型，这样Host也就可以查询（Queue）Device中的命令了，以及对应性能的提升。

【总结】

为了克服旧的BOT协议的总线利用率不高的缺点，所以定义了新的UAS协议，即UASP，来提升USB的传输效率，提升USB速度。

当然，我们此处为了实现U盘功能的话，当然希望性能越高越好了，所以，这个协议也是我们应该好好研究的。

## 2.1.2. USB MSC的各个协议之间关系总结

为了说明清楚USB Mass Storage各个协议的关系，我们先给这些协议编个号：

①USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport

②USB Mass Storage Class Bulk-Only (BBB) Transport

③USB Mass Storage Class Universal Floppy Interface (UFI) Command Specification

④USB Mass Storage Class Bootability Specification

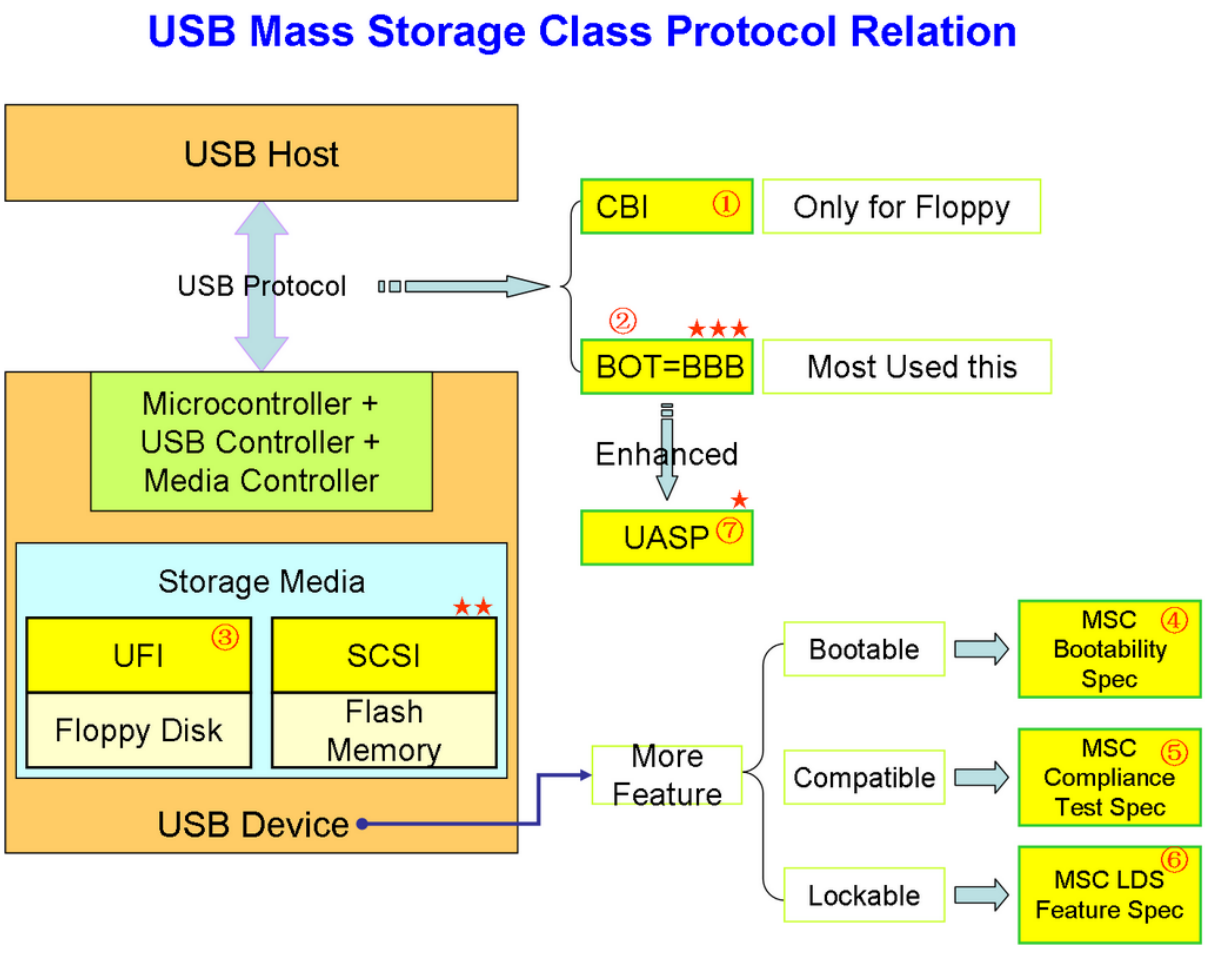
⑤USB Mass Storage Class Compliance Test Specification

⑥USB Lockable Storage Devices Feature Specification (LSD FS)

⑦USB Mass Storage Class USB Attached SCSI Protocol (UASP)

直接用图来表示USB MSC各个协议之间的关系，显得更加直观：

**图 2.7. USB Storage Class Protocol Relation**




如上图，我们U盘实现的功能，主要就是数据的读写，而Device和Host之间的数据通信，主要有两种：

1. CBI：主要用于Floppy设备，所以新的设备，都很少用此协议
2. BOT：Bulk-Only Transport，也称BBB（Bulk/Bulk/Bulk），

而对于BOT/BBB来说，对其提高USB总线利用率，提高了USB速度后，就是对应的UASP协议，故此处称UASP为BOT的增强版的协议。

协议方面说完了，再来看看USB Device这一端。

而USB的Device端，根据内部数据存储的介质类型不同，又分两种：

1. 一种是Floppy设备，对应用的是UFI Command Set；
2. 而另外一种，就是我们常见的Flash Memory，对应的是用SCSI Command Set。

而SCSI协议，本身就是有的了，所以不是属于USB MSC协议范畴，即SCSI只是和USB MSC相关的协议。

同样的，对于USB Device本身，如果需要一些用到其他的特性，比如可启动性，兼容性，可锁定性等等，那么分别对应的规范是

④USB Mass Storage Class Bootability Specification

⑤USB Mass Storage Class Compliance Test Specification

⑥USB Lockable Storage Devices Feature Specification (LSD FS)

【总结】

至此，各个协议和规范之间的关系，就很容易理解了。上面这么多协议中，其中我们所要关心的，只有三个规范，如前面的图中，已经用星号标识出来了：

1. 最需要关心的是BOT，即Host和Device间数据通讯的协议

   ★★★ ②USB Mass Storage Class Bulk-Only (BBB) Transport

2. 其次，需要关心USB Device内部和数据存储介质之间通信的协议

   ★★ SCSI - Small Computer System Interface

3. 最后，对于，如果要实现更好的性能，那么需要关心BOT的升级版

   ★⑦USB Mass Storage Class USB Attached SCSI Protocol (UASP)

## 2.1.3. U盘与USB中的Class，Subclass和Protocol的对应关系

对应的，了解USB的都知道，每个设备的描述符中，都有对应下面这几个域：

- bInterfaceClass
- bInterfaceSubClass
- bInterfaceProtocol

分别对应着USB的Class，Subclass，Protocol。

而对于我们此处的U盘：

### 2.1.3.1. bInterfaceClass=0x08=Mass Storage

Class就是USB Mass Storage Class，

### 2.1.3.2. bInterfaceSubClass=0x06=SCSI Transparent

Subclass，所支持的列表如下：

**图 2.8. SubClass Codes Mapped to Command Block Specifications**

**Table 1 — SubClass Codes Mapped to Command Block Specifications**

| Subclass | Command Block Specification | Comment |
|---|---|---|
| 00h | SCSI command set not reported | De facto use |
| 01h | RBC | Allocated by USB-IF for RBC. RBC is defined outside of USB. |
| 02h | MMC-5 (ATAPI) | Allocated by USB-IF for MMC-5. MMC-5 is defined outside of USB. |
| 03h | Obsolete | Was QIC-157 |
| 04h | UFI | Specifies how to interface Floppy Disk Drives to USB |
| 05h | Obsolete | Was SFF-8070i |
| 06h | SCSI transparent command set | Allocated by USB-IF for SCSI. SCSI standards are defined outside of USB. |
| 07h | LSD FS | LSDFS specifies how host has to negotiate access before trying SCSI |
| 08h | IEEE 1667 | Allocated by USB-IF for IEEE 1667. IEEE 1667 is defined outside of USB. |
| 09h - FEh | Reserved | Reserved |
| FFh | Specific to device vendor | De facto use |

### 2.1.3.3. bInterfaceProtocol=0x50=Bulk Only Transport

Protocol，所支持的列表如下：

**图 2.9. Mass Storage Transport Protocol**

**Table 2 — Mass Storage Transport Protocol**

| *bInterfaceProtocol* | Protocol Implementation | Comment |
|---|---|---|
| 00h | CBI (with command completion interrupt) | USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport |
| 01h | CBI (with no command completion interrupt) | USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport |
| 02h | Obsolete | |
| 03h-4Fh | Reserved | Reserved |
| 50h | BBB | USB Mass Storage Class Bulk-Only (BBB) Transport |
| 51h - 61h | Reserved | Reserved |
| 62h | UAS | Allocated by USB-IF for UAS. UAS is defined outside of USB. |
| 63h-FEh | Reserved | Reserved |
| FFh | Specific to device vendor | De facto use |

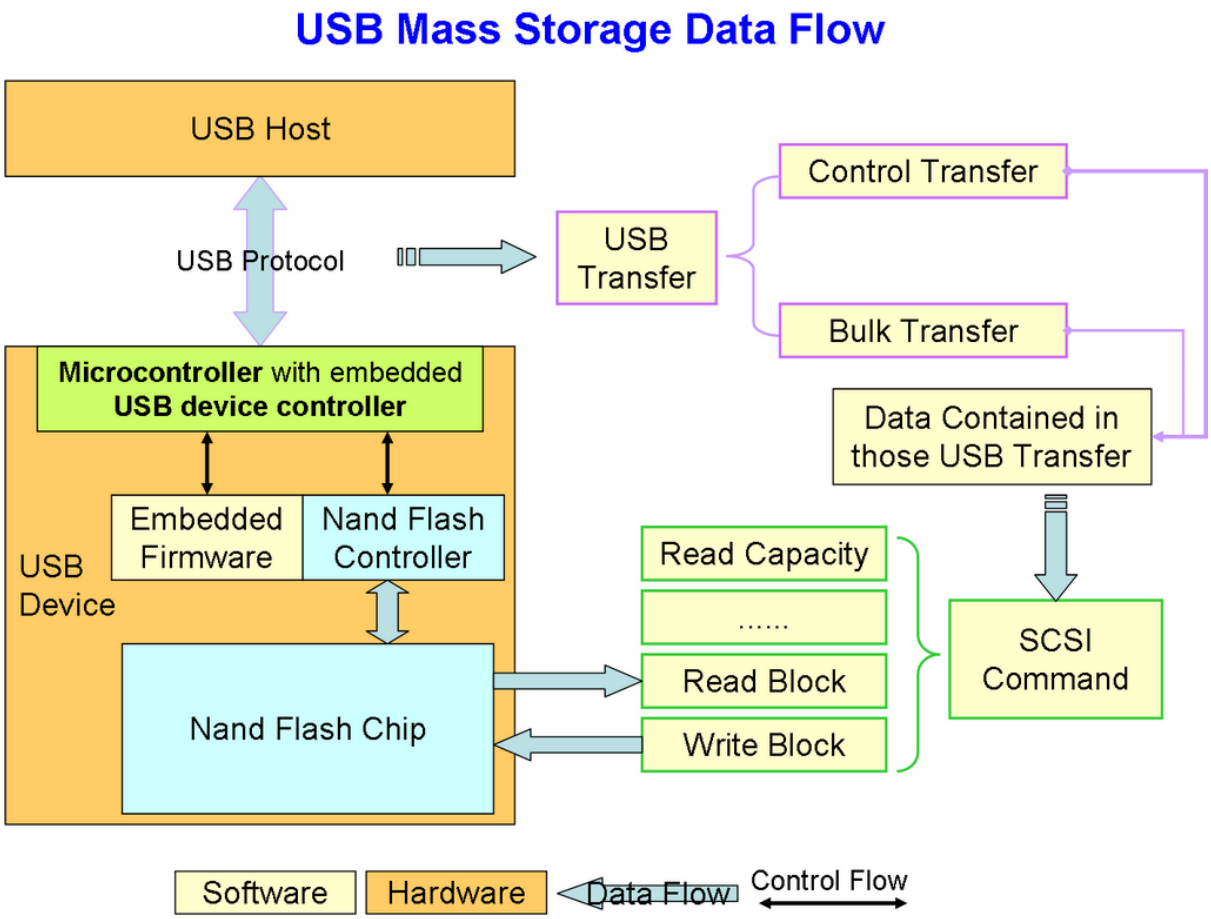
从上面这些规范中所定义的支持的协议来看，加上颜色框的那几个，也就是我们前面所解释过的，需要我们关心和研究的协议，即SCSI，BBB和UAS。

## 2.2. USB Mass Storage相关的软件实现

# 第 3 章 实现U盘驱动的整个流程是什么样的

USB数据流向图：

**图 3.1. USB数据流向图**



# 第 4 章 Linux系统下，USB驱动的框架已经做了哪些事情

# 第 5 章 Linux下实现U盘驱动，自己需要做哪些事情以及如何做

# 参考书目


[1] USB Mass Storage - Designing and Programming Devices and Embedded Hosts[1]

[2] USB Mass Storage - From the Introduction[2]

[3] Mass Storage Basics[3]

[4] 群联 Phison UP19 / PS2251-50 Datasheet[4]

[5] USB Approved Class Specification Documents - Mass Storage[5]

[6] USB Complete 4th Edition[6]

[7] Mass Storage Class Specification Overview 1.4[7]


[1] http://www.lvr.com/usbms.htm
[2] http://www.lvr.com/usbmsintr.htm
[3] http://www.lvr.com/files/usb_mass_storage_chapter_1.pdf
[4] http://www.mydigit.net/read.php?tid=261431
[5] http://www.usb.org/developers/devclass_docs/
[6] http://download.csdn.net/source/1592740
[7] http://www.usb.org/developers/devclass_docs/Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf