# 前言 本文结构

第一章

第二章

第三章

第四章

第五章

# 第一章 扫描USB设备

usb\_stor\_scan（参数仅和显示有关）此函数从USB设备列表usb\_dev[ ]中探测存储器设备，并对存储器设备依次进行初初始化配置，并依次将其存放在存储器设备列表中。

首先禁用异步传输，然后将最大可用设备数量（在此处为USB存储器设备数量）重置为0。其次，用for循环遍历设备列表下的设备，其中i即为设备列表第i个存储位置，在该循环中先调用usb\_get\_dev\_index（位置i）函数获取该位置下的USB设备。然后调用usb\_stor\_probe\_device（USB设备）函数探测刚刚获取到的某USB设备进行判断和配置。最后启用异步传输。

usb\_stor\_scan（）的核心在遍历和判断，如下主要针对usb\_stor\_probe\_device（）展开进一步讨论。

# 第二章 判别USB存储器并进行初始配置

本章小节结构图如下：

1

★

☆

2.1

2.2

3.1

3.2

3.3

☆

☆

**配置**

**存储**

**设备**

**框架**

注：带★的表示调用底层函数

注：带☆的表示调用已封装的传输函数

1 usb\_stor\_probe\_device（USB设备）先判断USB大容量存储器设备连接数量是否已经达到了上限，如果是的话就不进行后续步骤了。否则调用函数usb\_storage\_probe（）检测当前USB设备是否是大容量存储器设备，不是的话不进行后续步骤，否则usb\_storage\_probe函数将有用USB存储设备的数据信息记录下来。

到这里我们能确认他是一个存储设备了，然后调用usb\_get\_max\_lun（★）函数向存储器发送命令，获取当前存储器设备的最大逻辑单元号max\_lun。【注：一个磁盘空间被划分为若干个小逻辑单元以供主机使用】最后迭代其lun，将接口类型、设备号、分区类型、目标小型计算机接口ID、设备类型、**读写操作**、目标逻辑单元号以及当前USB设备整体封装在一个块存储器设备中，保存在块存储器设备列表。

最后调用函数usb\_stor\_get\_info（）尝试与该存储器建立连接，访问存储器信息并输出。

2.1 usb\_storage\_probe（USB设备，接口数量，us\_data）首先对该数据类型为usb\_device的USB设备进行检验，根据设备类型、接口类型与接口子类型判断该设备是不是大容量存储器设备，如果不是的话不进行后续操作直接返回。否则记录存储器的标志、接口数量、当前USB设备、子类别以及协议。

根据协议（协议类型为：US\_PR\_CB、US\_PR\_CBI、US\_PR\_BULK）设置处理程序的指针，该处理程序包括**控制传输程序**、**批量传输程序**、**控制重置程序**以及**批量重置程序**等。然后识别端点类型（批量传输输入端点/批量传输输出端点/中断传输端点）配置端点地址。如果是中断端点的话，还需要配置中断间隔、中断管道以及中断句柄。

2.2 usb\_stor\_get\_info（USB设备，us\_data，块存储器设备）。指定数据缓存区、目标设备的设备号以及逻辑单元号。首先调用函数usb\_inquiry（）向该设备发送查询指令，拷贝制造商信息、产品号以及设备号到blk\_desc中。其次调用函数usb\_test\_unit\_ready（）以测试存储器连接是否准备就绪。最后调用函数usb\_read\_capacity（）读取存储器的容量，并将块存储器数量、块存储器大小以及设备类型写入blk\_desc之中。

3.1 usb\_inquiry（小型计算机接口命令，us\_data）先配置小型计算机接口命令：具体命令、数据长度、命令长度。循环调用transport（☆）函数尝试连接存储器，如果在这五次尝试之内存储器有应答成功则返回正常，否则返回失败。

3.2 usb\_test\_unit\_ready（小型计算机接口命令，us\_data）先配置小型计算机接口命令，再循环调用transport（☆）向存储器发送指令并用usb\_request\_sense（）来检查“密钥代码限定符”判断连接是否就绪，在十次尝试中有一次成功就代表连接就绪。注：usb\_request\_sense本质还是调用transport（☆）函数。

3.3 usb\_read\_capacity（小型计算机接口命令，us\_data）先配置小型计算机接口命令，再循环调用transport（☆）向存储器发送指令以获取存储器的容量，在三次尝试中有一次成功即可。

# 第三章 读写操作

本章小结结构图如下：

1.1

1.2

2.1

2.2

☆

☆

**读写**

**操作**

**框架**

注：☆表示调用已封装的传输函数

1.1 usb\_stor\_read（USB块存储器设备，块存储器数量，块存储器大小，数据缓存区）大容量存储器设备读取操作。先根据设备号block\_dev->devnum从数组usb\_dev\_desc[ ]之中查找到对应的USB设备udev，再根据此设备找到类型为us\_data的设备相关数据。调用函数usb\_disable\_asynch禁用异步传输，配置逻辑单元号、缓存区地址、块存储器数量以及大小。

循环进行以下操作：设置尝试次数为两次，如果块存储器大小超出一次能传输的最大值就把块存储器拆分成数个小区间来传输。设置拆分后数据长度和拆分后数据缓存区起始地址，然后调用usb\_read\_10（）开始读大容量存储器。传输完成一个数据片段后，开始位置后移、未读取区域减小、数据缓存区指针后移，继续判断是否完全读取是的话退出并使能异步传输，否则继续执行当前循环继续读取存储器。

1.2 usb\_stor\_write（USB块存储器设备，块存储器数量，块存储器大小，数据缓存区）大容量存储器设备写入操作。写操作和上述读操作有异曲同工之妙，大体框架均相同，只是核心的函数为usb\_write\_10（），后续仅对其展开叙述其他的见本章1.1。

2.1 usb\_read\_10（小型计算机接口命令，设备数据us\_data，开始读取的位置，读取大小）先清空小型计算机接口命令并重新配置，并赋值命令长度。最后调用transport（☆）开始传输。

2.2 usb\_write\_10（小型计算机接口命令，设备数据us\_data，开始读取的位置，读取大小）也还是先配置小型计算机接口命令，然后赋值命令长度，最后调用transport（☆）开始传输。和usb\_read\_10（）的区别在于小型计算机接口命令不同。

# 第四章 传输程序与重置程序

本章小结结构图如下：

1.1

1.2

1.3

1.4

2.1

☆

3

★

★

★

2.2

☆

★

★

★

**传输程序与重置程序框架**

注：带★的表示调用底层函数

注：带☆的表示调用已封装的传输函数

1.1 usb\_stor\_CB\_transport（小型计算机接口命令，设备数据us\_data）首先调用usb\_stor\_CB\_comdat（）函数发送控制传输/批量传输的命令，判断上一步获取到的协议类型是不是CBI协议，如果是CBI协议的话调用函数usb\_stor\_CBI\_get\_status（★）获取中断请求IRQ与us->ip\_data。根据上述结果调用函数transport\_reset（☆）进行传输复位。

重新配置相关命令：具体命令、数据长度、数据缓存区地址以及命令长度。再调用usb\_stor\_CB\_comdat（）重新发送命令，调用usb\_stor\_CBI\_get\_status（★）重新获取中断请求IRQ，最后再次检查结果是否符合规范。

1.2 usb\_stor\_CB\_reset（设备数据us\_data）这个重置功能并没有真正重置端口，真正的重置是要在物理上重置端口，此处只是发送指令软复位。先设备命令向存储器设备发送小型计算机接口诊断的命令，然后调用底层函数usb\_control\_msg（★）向存储器设备发送命令，同时等待较长时间让设备完成重置。最后调用usb\_clear\_halt（★）函数向下位机发送指令清除输入输出端点停止状态。

1.3 usb\_stor\_BBB\_transport（小型计算机接口命令，设备数据us\_data）首先从小型计算机接口命令解码出传输方向，整个过程有三大阶段命令阶段、数据阶段、状态阶段。

命令阶段：先调用usb\_stor\_BBB\_comdat（）函数向设备发送设置命令，发生错误就调用usb\_stor\_BBB\_reset（☆）复位。然后配置输入输出管道，完成命令阶段。

数据阶段：根据命令阶段的数据，选定输入管道还是输出管道。调用底层传输函数usb\_bulk\_msg（★）开始数据传输，如果该端点被挂起就调用usb\_stor\_BBB\_clear\_endpt\_stall（★）清除设备端点上的停顿并进入状态阶段，如果数据传输出错就复位设备并返回。

状态阶段：调用底层函数usb\_bulk\_msg（★）获取设备状态，如果端点被挂起就清除端点停顿并重试，如果状态传输发生错误就调用usb\_stor\_BBB\_reset（☆）复位设备。最后判断数据传输有没有获取多余信息等，如果有就复位存储器设备。

1.4 usb\_stor\_BBB\_reset（设备数据us\_data）实现存储器复位，调用usb\_control\_msg（★）向存储器发送控制信息，等待较长时间让存储器复位成功。再调用usb\_clear\_halt（★）函数清除输入输出端点的停止。

2.1 usb\_stor\_CB\_comdat（小型计算机接口命令，设备数据us\_data）设置尝试次数为5次，从传递参数小型计算机接口命令解码出传输方向输入还是输出，再根据传输方向配置相应的发送管道或接收管道。

做完这些准备工作后开始多次尝试：调用底层函数usb\_control\_msg(★)向存储器发送命令，然后检查命令的返回码是否被挂起，如果是调用usb\_clear\_halt（★）清除停止。如果该上述过程全都没出错，那就调用函数us\_one\_transfer（）开始传输数据。

2.2 usb\_stor\_BBB\_comdat（小型计算机接口命令，设备数据us\_data）进行设置设备。根据小型计算机接口命令解码出传输方向，配置管道方向总是向端点输出。继续配置具体命令：签名、标签、发送数据长度、标志、逻辑单元号以及命令长度。最后调用函数usb\_bulk\_msg（★）向存储器发送控制命令。

3 us\_one\_transfer（设备数据，管道，数据缓存区，传输长度）根据该管道获取并配置单次最大传输数据包的大小，对该大小进行拆分后通过usb\_bulk\_msg（★）多次传输，并监测每次传输状态（传输被挂起就清除挂起并返回、如果从设备拒绝或者出错也返回，直到所有数据包传输完成。

# 第五章 获取存储器设备

get\_usb\_msc\_blk\_dev（位置编号）根据编号cur\_msc在结构体数组usb\_dev\_desc[ ]中找到相应的大容量存储器设备描述符并返回。

存储器初始化与获取的流程总结：

（1）usb\_init先在总线上扫描所有USB设备，一个设备对应一个结构体usb\_device，并将其填入结构体数组usb\_dev[ ]之中。

（2）usb\_stor\_scan遍历结构体数组usb\_dev[ ]之中的设备，先判断该设备是不是存储器设备，如果是的话就创建USB设备数据结构体us\_data并将其填入结构体数组usb\_stor[ ]之中，**该结构体中还包含已经封装的传输程序与重置程序**。

（3）usb\_stor\_scan获取设备最大逻辑单元号max\_lun，然后迭代其lun将上述us\_data中的数据写入块存储器设备描述符结构体blk\_desc，并将其填入结构体数组usb\_dev\_desc[ ]之中，**该结构体中还包含已经封装的对存储器设备的读写操作**。

（4）get\_usb\_msc\_blk\_dev根据编号cur\_msc在结构体数组usb\_dev\_desc[ ]中找到相应的大容量存储器设备描述符，以便后续操作。