



嵌入式Linux下USB设备驱动开发解析

华清远见

Copyright 2007-2008 Farsight.
All rights reserved.

LINUX的USB设备驱动程序开发

- } USB及驱动框架简介
- } USB主机端驱动
- } USB设备端驱动



usb 驱动程序功能演示

- } 步骤1: 插入MMC卡到fs2410开发板,出现设备
/dev/mmcblk0
 - } 步骤2: 插入4GB Kingston优盘到fs2410开发板的usb
host接口.fs2410将识别这个插入过程并出现设备
/dev/uba1(或者/dev/sda1)
 - } 步骤3: 将fs2410开发板的usb device接口插入
windows USB口,使得fs2410的本机nandflash
/dev/mtdblock3和上述两个设备(mmc卡/4GB优盘)都
能在电脑上通过优盘形式来访问(出现3个盘符).
-




usb 驱动程序功能演示

- } /dev/mtdblock3 => fs2410开发板上nandflash
- } /dev/mmcblk0 => 接在fs2410开发板上的MMC卡
- } /dev/uba1 => 接在fs2410开发板上的Kingston
优盘

- } => 这个演示,涉及了usb host和usb device功能(也涉
及了sd卡驱动的功能).

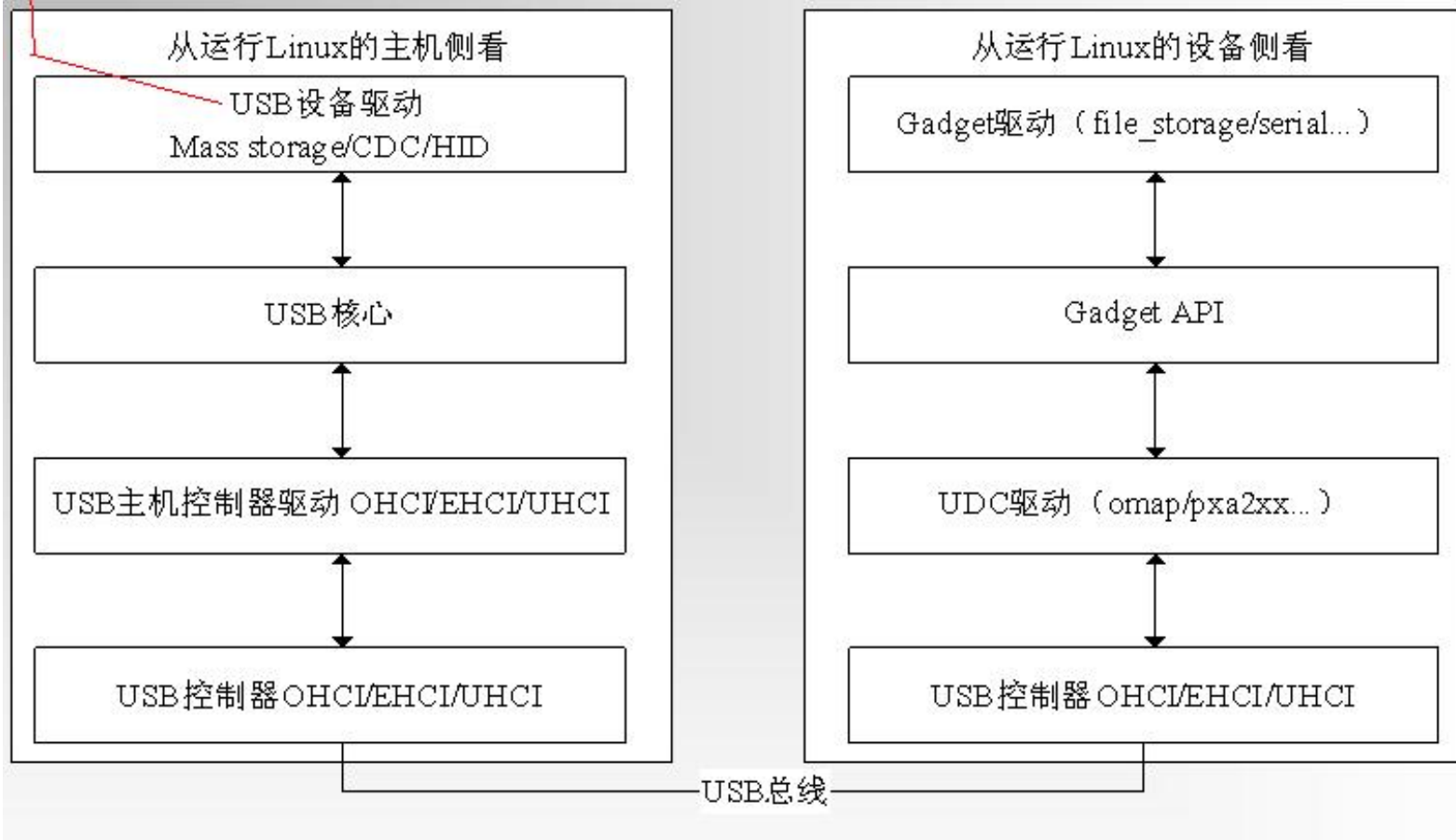


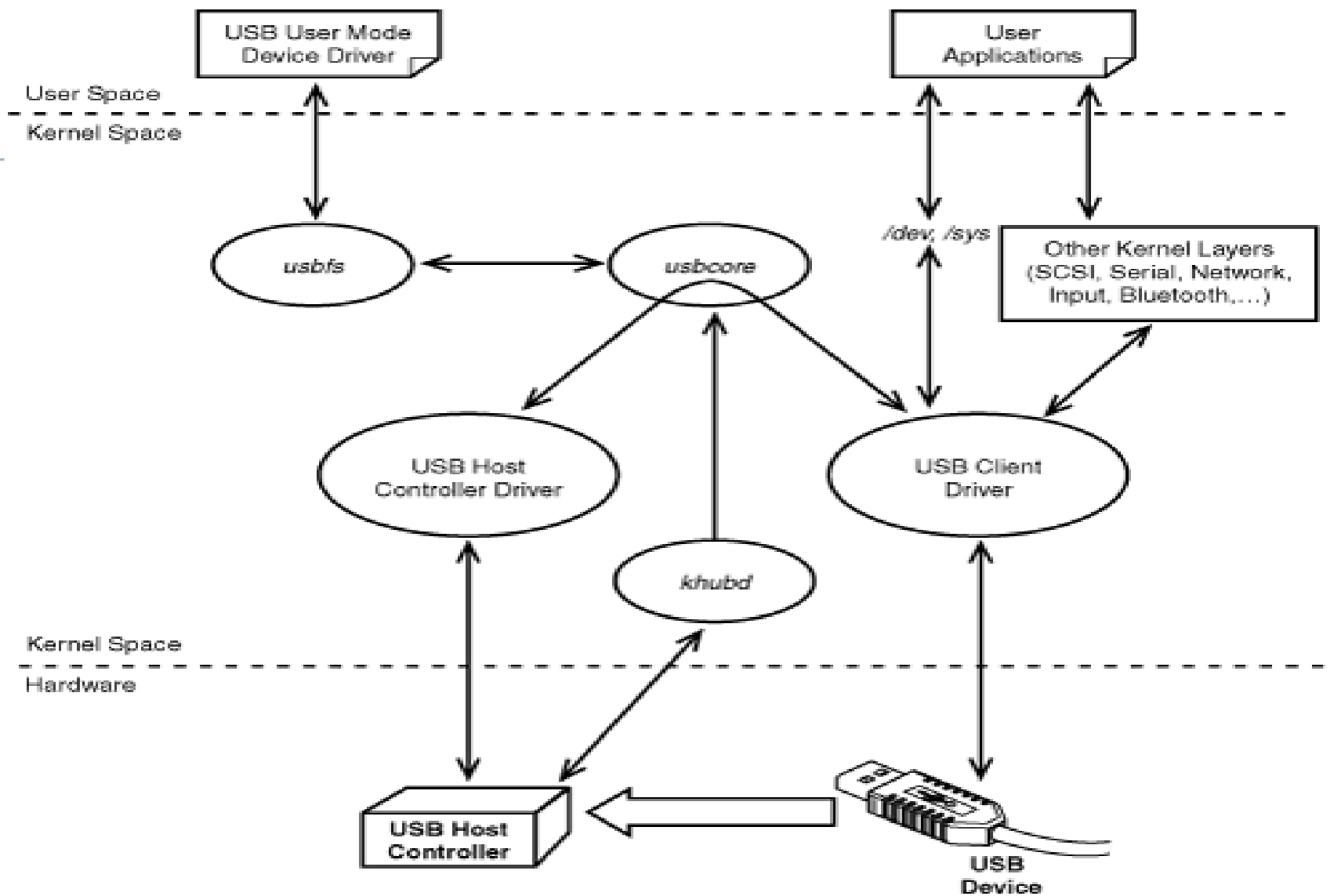
usb 驱动程序功能演示: 解释

- } fs2410 usb host : 插入优盘到fs2410 usb主机端,fs2410 usb主机端检测到插入优盘设备并完成枚举和初始化过程.然后调用一个具体的设备驱动(如storage设备驱动)并产生一个设备节点/dev/sda1
 - } fs2410 usb device : usb设备端驱动在用户的要求下将3个设备(mmcblk0/sda1/mtdblock3)作为优盘设备接入windows usb主机端.并对windows发起的枚举过程作出正确的响应,返回三个设备的相关信息,使得最终windows能正确识别出这三个设备,并出现3个优盘盘符供用户方便的访问这些存储介质.
-
- 

插入usb host上的U盘、鼠标、USB转串口等设备驱动

Linux USB驱动层次





Linux-USB 子系统


}




Linux对USB规范的支持

- } USB-通用串行总线是目前使用最广泛的外部总线
- } USB是采用单一的主从设备通信模式。总线上的唯一的主机负责轮询设备并发动各种传送，因此实现简单，成本相对低廉
- } USB从拓扑上讲类似于主机同外设之间点对点连接，设备连接汇集于集线器上
- } USB最新的规范是USB2.0版本，定义了三种传输速率
 - } Low speed —— 1.5Mbps
 - } Full speed —— 12Mbps
 - } High speed —— 480Mbps


USB接口

- } 按照物理接口特性，USB接口可以分为
 - } 主机（USB HOST）端
 - } USB集线器（USB HUB）
 - } USB设备（USB DEVICE）端
 - } USB集线器其实就是一类特殊的USB设备。
 - } 在一个完整的USB拓扑结构上，必须有且仅有一个USB主机，一个或多个USB集线器和USB设备。
-
- 

USB1.1规范

- } USB1.1规范支持低速（1.5Mb/s）和全速（12Mb/s）两种不同速率的数据传输和4种不同类型的数据传输方式：
 - } 控制传输（CONTROL TRANSFER）
 - } 中断传输（INTERRUPT TRANSFER）
 - } 批量传输（BULK TRANSFER）
 - } 等时传输（ISOCRONOUS TRANSFER）
-
- 

USB2.0规范

- } USB2.0在兼容USB1.1低速（1.5Mb/s）和全速（12Mb/s）数据传输基础上，支持高速（480Mb/s）数据传输。
 - } 对于USB2.0规范，同样支持控制传输、中断传输、批量传输和等时传输4种类型的数据传输方式。
 - } 在物理结构和拓扑结构上，USB2.0与USB1.1也是完全相同的。
-
- 

USB OTG规范



- } USB OTG规范是作为对USB2.0规范的补充而出现的，其目的是为了**满足便携式设备对USB接口性能的需求**。
- } 根据USB OTG规范，一个USB接口可同时具有**USB主机和USB设备**两种功能，根据与其连接的其他设备属性，USB OTG接口会自动转换成为**适合USB总线需求**的接口类型。



USB HOST规范

- } 关于USB HOST接口，在符合USB规范的基础上，不同的厂商开发的USB HOST器件可能有着不同的结构特性。当前流行的USB HOST规范有：
 - } OHCI (OPEN HOST CONTROL INTERFACE)
 - } UHCI (UNIVERSAL HOST CONTROL INTERFACE)
 - } EHCI (ENHANCED HOST CONTROL INTERFACE)
-
- 

EHCI与OHCI

- } ehci-hcd模块支持的是USB2.0控制器的高速模式，它本身并不支持全速或低速模式
- } ohci-hcd或uhci-hcd模块提供对USB1.1设备的支持
- } 如果我们只配置了EHCI，就没有办法使用usb的鼠标键盘

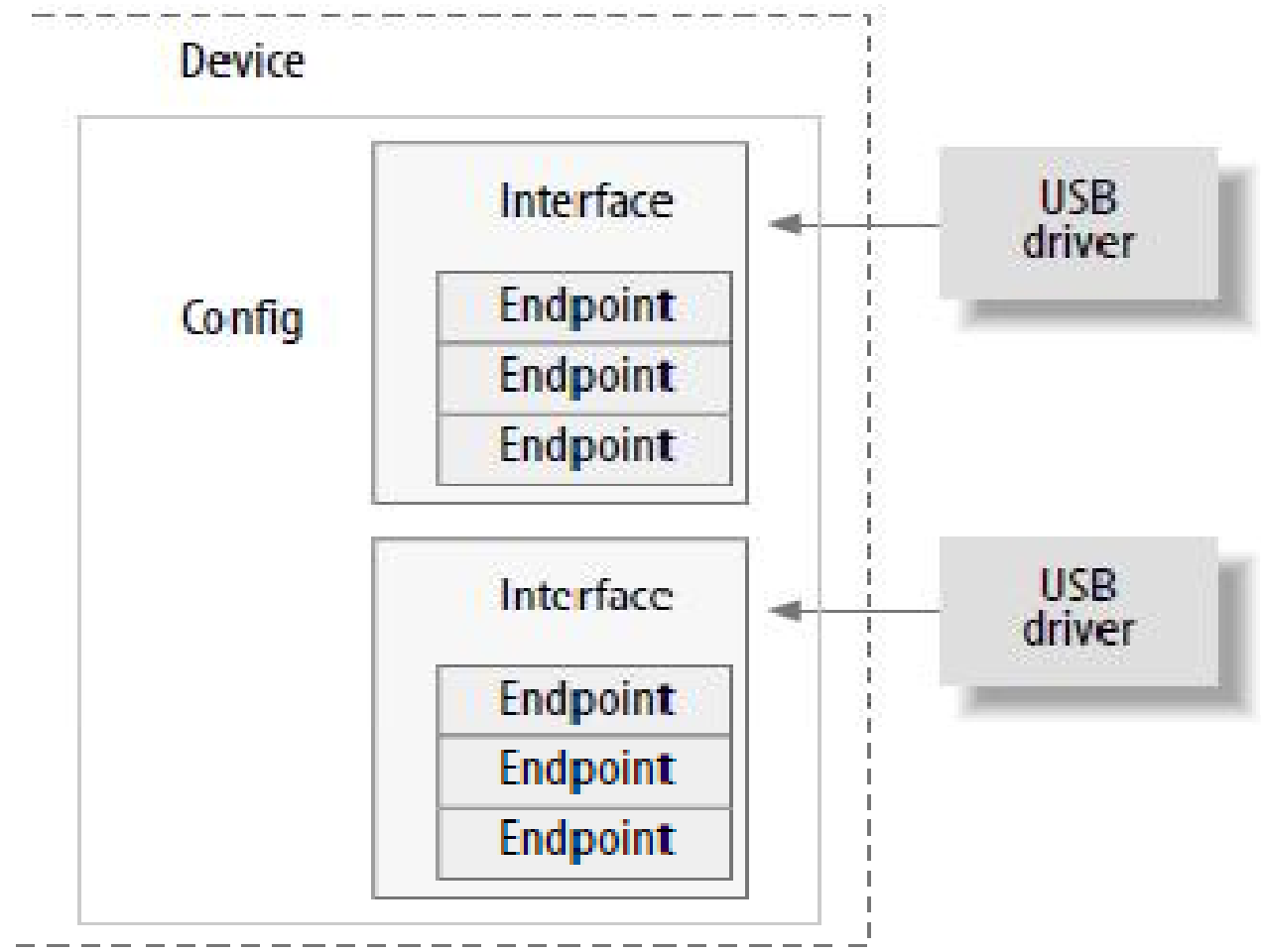


USB设备基本概念


- } USB规范中规定的标准概念由Linux USB core来实现处理
 - } 端点endpoints
 - } 端点是USB总线传输最基本的概念，一个端点可以单方向传输数据。可以把端点看作是一个单方向的管道。端点有4种类型
 - } 接口interfaces
 - } 若干端点可以捆绑起来，成为一个端口。端口可以作为完整的逻辑设备连接，例如鼠标设备、键盘设备。
 - } 需要指出的是，一个硬件设备可能包含多个逻辑设备
 - } 接口可以有多个预设值，用来指定不同的参数
 - } 配置configurations
 - } 接口捆绑起来成为配置。一个USB设备可以在不同的配置之间切换，一次只能激活一个配置
-



USB设备基本概念示意图



USB端点分类

- } **USB 总线中的通信可以使用下面四种数据传输类型中的任意一种：**
 - } **控制传输：**这些是一些短的数据包，用于设备控制和配置，特别是在设备附加到主机上时。
 - } **批量传输：**这些是数量相对大的数据包。像扫描仪或者 SCSI 适配器这样的设备使用这种传输类型。
 - } **中断传输：**这些是定期轮询的数据包。主控制器会以特定的间隔自动发出一个中断。
 - } **等时传输：**这些是实时的数据流，它们对带宽的要求高于可靠性要求。音频和视频设备一般使用这种传输类型。
-
- 

USB Host Controller驱动程序

- } 主机控制器驱动程序负责USB总线通信基本的职责
 - } 处理USB状态。管理状态并报告状态信息
 - } 数据串行/解串行，将设备申请传输的数据转换成比特流
 - } 生成frame或microframe
 - } 处理数据传输的请求
 - } 处理USB总线协议
 - } 进行差错检测和控制
 - } 处理能源管理请求，把总线置为suspended状态以及响应wakeup事件
 - } 提供root hub功能，让设备可以连接到主机控制器
-
- ▶

USB请求块

- } 在Linux内核中，USB驱动程序之间通过USB request block—urb来进行异步数据交换。
- } 设备和多个端点之间可以使用同样或不同的urb，端点可以处理一个urb队列。使用urb的流程如下：
 - } USB设备驱动创建一个urb
 - } 设置urb，将它关联到某个端点
 - } 设备驱动程序将urb提交给USB core
 - } USB core解析urb关联的设备，并把它发送到适当的USB主控器
 - } 主控器按照urb的内容，驱动总线设备完成传输。当传输完成时，主控器将告知设备驱动程序。
 - } 发送urb的设备驱动程序或USB core也可以取消urb



创建和销毁urb

} 因为USB core要对urb进行引用计数，所以驱动程序不能静态的创建urb。应该使用下面的调用来动态的获得和释放urb

```
} struct urb *usb_alloc_urb(int iso_packets, int mem_flags);
```

```
} void usb_free_urb(struct urb *urb);
```

} 获得urb后，在使用之前我们还要对它初始化

```
} usb_fill_int_urb
```

```
} usb_fill_bulk_urb
```

```
} usb_fill_control_urb
```

```
} 而等时urb需要手工地来填充各项
```



提交和完成urb

- } 初始化好一个urb之后，驱动程序就可以将它提交给USB core
 - } `int usb_submit_urb(struct urb *urb, int mem_flags);`
- } 提交过后，驱动程序在complete调用之前就不要再访问urb
- } urb完成时，该urb的complete handler就会被调用。完成分3种情况
 - } urb成功完成，设备返回正确的应答。这种情况下，urb的status会被置0
 - } 发生了错误，urb的status值将会被置为某个错误代号
 - } urb被取消，有可能是驱动程序或USB core取消了urb，或者是该设备已经从总线上拔出
- } 取消一个urb，可以通过下列调用来实现
 - } `int usb_kill_urb(struct urb *urb);`
 - } `int usb_unlink_urb(struct urb *urb);`



编写USB设备驱动

- } USB设备驱动程序首先向USB子系统注册自己，然后通过 vendor id和device id来判断硬件是否插入总线
- } 驱动程序需要创建一个usb_driver结构，该结构包含了下列项
 - } struct module *owner
 - } const char *name
 - } const struct usb_device_id *id_table
 - } int (*probe) (struct usb_interface *intf, const struct usb_device_id *id)
 - } void (*disconnect) (struct usb_interface *intf)
 - } int (*ioctl) (struct usb_interface *intf, unsigned int code, void *buf)
 - } int (*suspend) (struct usb_interface *intf, u32 state)
 - } int (*resume) (struct usb_interface *intf)



编写USB设备驱动

- } USB骨架程序：
drivers/usb/usb-skeleton.c
- } USB mass storage：
drivers/usb/storage/



USB设备端驱动



- } S3c2410 Usb device控制器驱动:
drivers/usb/gadget/s3c2410_udc.c
- } Usb gadget驱动:
drivers/usb/gadget/file_storage.c



实验思考：usb host主机端驱动部分




- } usb host是怎样检测到我插入了一个优盘的？她是怎么实现检测的？
- } usb host怎么知道我插入的优盘是Kingston的？usb host怎么知道我的优盘大小是4GB的？
- } usb host是怎样对这个优盘进行枚举和初始化的？
- } usb host控制器是通过一个什么结构和usb外设通信的？
- }

实验思考：usb device设备端驱动部分



- } usb设备端驱动让3个设备同时出现在了windows上(3个盘符),她是怎么做到的?
- } 为什么在windows上向一个盘符拷贝了数据,这个数据就写到了/dev/mtdblock3或者/dev/sda1或者/dev/mmcblk0中. 她是怎么做到的?
- } usb设备端驱动对主机端发起的枚举过程,是怎样一步一步响应,使得主机端最终正确的识别到自己的型号和容量大小的?
- } RBC/CBW/CSW在usb设备端驱动中又是怎么具体来实现的?
- }

嵌入式Linux驱动开发班

- } LINUX字符设备驱动程序开发
 - } LINUX块设备驱动程序开发
 - } LINUX网络设备驱动程序开发
 - } LINUX下FRAMEBUFFER设备驱动程序开发
 - } LINUX下SD卡设备驱动程序的开发
-
- 

- } 了解Linux中USB设备驱动的特点及层次结构，了解USB设备的初始化流程，从而进行USB设备驱动的简单开发
- } 理解Linux内核USB子系统，掌握USB驱动程序编写技术。经过学习后的学员能够移植或编写USB主机OHCI和设备驱动程序，最后通过具体的USB设备，实现高级的USB驱动程序。



www.farsight.com.cn

Thanks !