关于USB的一点思考(1)

关于USB，我一直觉得体现了软硬件协同的最高境界，正所谓一通则百通，对USB设计背后体现的思想如果能够窥探一斑，学习其他总线可能进益更快，这篇文章先从几个USB概念切进，来了解USB的世界。

1. Hub

Hub是分为roothub和普通hub的，roothub集成在usb控制器中，并会虚拟成usb设备由usb driver管理，其主要负责管理usb port的状态管理；而普通hub则是以真实的usb 设备的形式被usb driver管理的。

roothub在软件层面有对应的struct usb\_device和struct usb\_inferface结构体，并且和struct usb\_bus,struct usb\_hcd唯一 对应，在usb\_add\_hcd时分配；struct usb\_bus、struct usb\_hcd 和struct usb\_device(roothub)的结构体关系如下所示:

struct usb\_hcd {

struct usb\_bus self;

}；

Struct usb\_bus {

Struct device \*controller;

int busnum;

struct usb\_device \*root\_hub;

};

Struct usb\_device {

Struct usb\_bus \*bus;

u8 portnum;

struct usb\_device \*parent;

struct usb\_device\_descriptor descriptor;

struct usb\_host\_endpoint;

struct usb\_host\_endpoint \*ep\_in[16];

struct usb\_host\_endpoint \*ep\_out[16];

};

struct usb\_bus随struct usb\_hcd一起分配，在usb\_add\_hcd中调用usb\_register\_bus初始化struct usb\_bus，struct usb\_bus的busnum成员从1开始；接着在usb\_add\_hcd中调用usb\_alloc\_dev分配struct usb\_device，struct usb\_device的bus成员即被置为struct usb\_hcd的self，struct usb\_device的portnum被置为0，struct usb\_device的parent被置成NULL，struct usb\_device的name被置为usb1，其中的1即为struct usb\_bus的busnum；最后struct usb\_bus的root\_bub成员被置为刚刚分配的struct usb\_device。

有意思的一点是在struct usb\_add\_hcd中，会初始化struct usb\_device ep0成员并将其赋值给dev->ep\_out[0]和dev->ep\_in[0]。

既然roothub被抽象成了struct usb\_device，那枚举struct usb\_device时的device descriptor等信息从哪里获取？我们看一下如下调用流程:

Usb\_add\_hcd->

Register\_root\_hub->

usb\_get\_device\_descriptor->

Usb\_get\_descriptor->

Usb\_control\_msg->

Usb\_submit\_urb->

Usb\_hcd\_submit\_urb

在usb\_hcd\_submit\_urb函数中会调用is\_root\_hub判断是否是roothub，is\_root\_hub根据struct usb\_device 的parent成员是否为空来判断，如果是roothub，则进入rh\_urb\_enqueue流程，由于获取device descriptor的端点是control 类型（USB\_ENDPOINT\_XFER\_CONTROL的值为0），所以会进入rh\_call\_control函数，在rh\_call\_control函数中会根据urb的typereq分别走到case DeviceRequest | USB\_REQ\_GET\_DESRIPTOR和case USB\_DT\_DEVICE<<8分支，注意DeviceRequest定义为((USB\_DIR\_IN | USB\_TYPE\_STANDARD | USB\_RECIP\_DEVICE)<<8)，而USB\_TYPE\_STANDARD和USB\_RECIP\_DEVICE的值都为0，所以DeviceRequest即为USB\_DIR\_IN <<8，可以看到， roothub的device descriptor是保存在一个数组中的，并且每个速率都对应不同的数组，同理configuration descriptor也是通过同样的函数调用流程从数组中获取。

按照上述方式，可以分别获取struct usb\_device 的 device descriptor和configuration descriptor，接着进入generic\_probe中，进而choose configuration，set configuration，根据configuration生成struct usb\_inferface,进入hub\_probe流程，生成struct usb\_hub。

在hub\_probe过程中，