iTOP-4412-驱动-usb 文档 07-鼠标驱动注册

在 Linux 的 USB 驱动中，鼠标驱动是最简单，最适合用来学习的。USB 鼠标驱动是内核源码目录“drivers/hid/usbhid/”下的“usbmouse.c”文件，这个驱动在任意版本的内核中都有。

我们将“usbmouse.c”文件进行分解，提取其中一部分来分析，本文档主要分析 USB 鼠标驱动的注册部分。

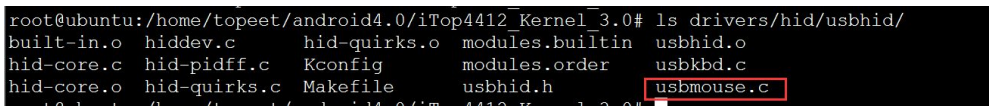
1 USB 鼠标驱动的配置

我们要分析的 USB 鼠标驱动源码是“usbmouse.c”，这里要注意的是要支持 USB 鼠标驱动有多种配置方法。

这里给大家介绍一个概念“人机接口”HMI，人机接口在 linux 中包含很多设备，例如：鼠标、键盘、显示屏...，凡是用来直接和用户直接通信的设备都可以叫做人机接口。从原始的 DOS 控制台，到手机触屏，未来的人脑控制等等也都是人机接口。

在 linux 中，默认使用的是 HID，也就是人机接口设备，内核中默认配置的是 HID 驱动，并没有配置 USB 鼠标驱动。配置部分涉及到 Makefile、menuconfig 和 Kconfig，我们一起来研究一下。

在内核配置编译之后，如下图所示，使用命令“ls drivers/hid/usbhid/ ”，可以看到“usbmouse.c”并没有被编译，可以我们的内核明显是可以支持 USB 鼠标的。



接着使用命令“vim drivers/hid/usbhid/Makefile”，打开 Makefile 文件，可以看到编译条件是宏“CONFIG_USB_MOUSE”。

```
endif
ifeq ($(CONFIG_HID_PID),y)
        usbhid-y        += hid-pidff.o
endif

obj-$(CONFIG_USB_HID)           += usbhid.o
obj-$(CONFIG_USB_KBD)           += usbkbd.o
obj-$(CONFIG_USB_MOUSE)         += usbmouse.o

~
```

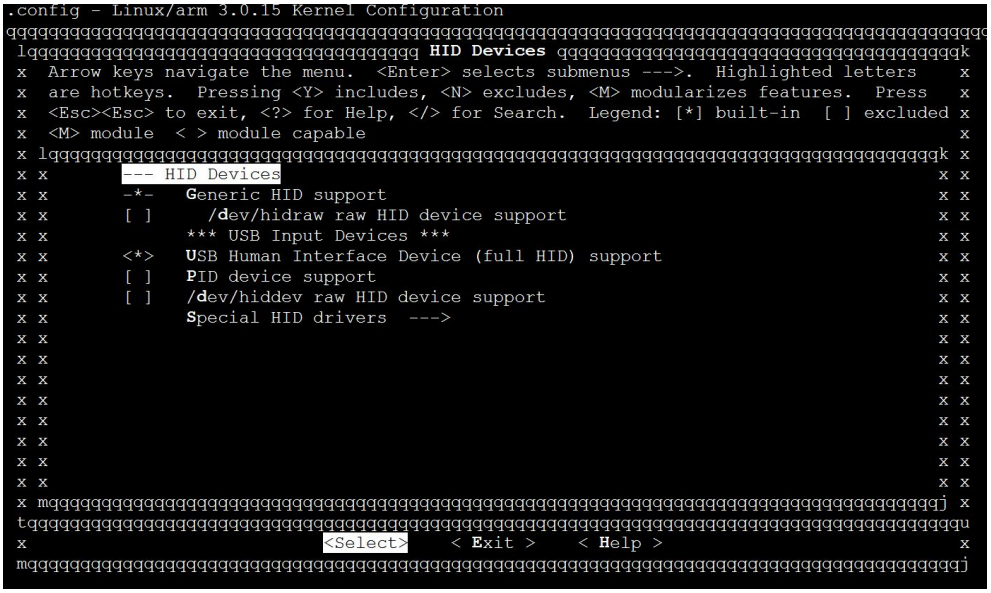
然后使用“vim drivers/hid/usbhid/Kconfig”命令，可以看到“config USB_MOUSE”Kconfig 的配置，如下图所示。

```
config USB_MOUSE
        tristate "USB HIDBP Mouse (simple Boot) support"
        depends on USB && INPUT
        ---help---
          Say Y here only if you are absolutely sure that you don't want
          to use the generic HID driver for your USB mouse and prefer
          to use the mouse in its limited Boot Protocol mode instead.

          This is almost certainly not what you want.  This is mostly
          useful for embedded applications or simple mice.

          To compile this driver as a module, choose M here: the
          module will be called usbmouse.

          If even remotely unsure, say N.
```

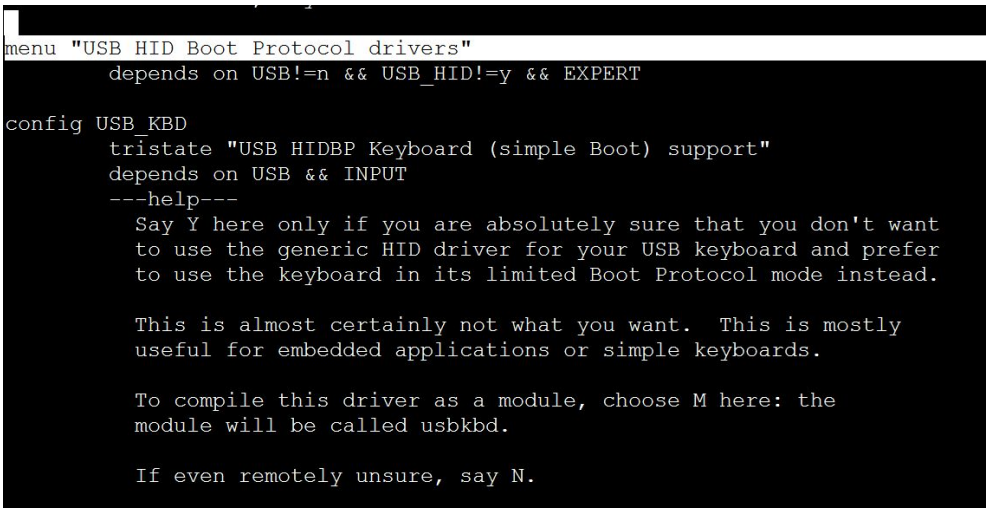
接着我们使用命令“make menuconfig”，然后在其中搜索“USB_MOUSE”宏，如下图所示。

```
Symbol: USB_MOUSE [=n]
Type  : tristate
Prompt: USB HIDBP Mouse (simple Boot) support
  Defined at drivers/hid/usbhid/Kconfig:66
  Depends on: HID_SUPPORT [=y] && USB_HID [=y]!=y && EXPERT [=y] && USB [=y] && INPUT
  Location:
    -> Device Drivers
      -> HID Devices (HID_SUPPORT [=y])
        -> USB HID Boot Protocol drivers
```

如下图所示，我们进入到“HID Devices”配置界面之后，却没有发现有“USB HID Boot Protocol drivers”这个菜单。



那么到底是什么原因呢？我们回到“drivers/hid/usbhid/Kconfig”文件，如下图所示，可以看到“USB_MOUSE”是在菜单“USB HID Boot Protocol drivers”下。

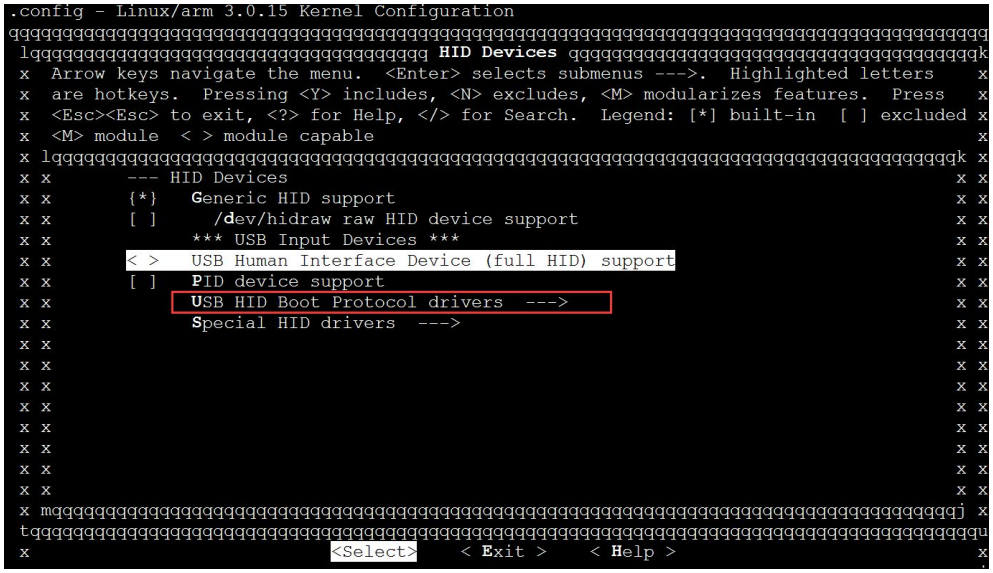


```
menu "USB HID Boot Protocol drivers"
        depends on USB!=n && USB_HID!=y && EXPERT

config USB_KBD
        tristate "USB HIDBP Keyboard (simple Boot) support"
        depends on USB && INPUT
        ---help---
          Say Y here only if you are absolutely sure that you don't want
          to use the generic HID driver for your USB keyboard and prefer
          to use the keyboard in its limited Boot Protocol mode instead.

          This is almost certainly not what you want.  This is mostly
          useful for embedded applications or simple keyboards.

          To compile this driver as a module, choose M here: the
          module will be called usbkbd.

          If even remotely unsure, say N.
```

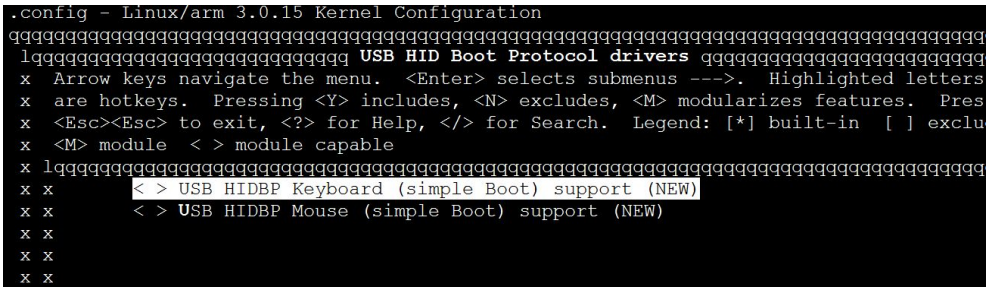
如上图所示，注意以下的提示，这部分意思是，菜单“USB HID Boot Protocol drivers”需要依赖“USB!=n && USB_HID!=y && EXPERT”，也就是必须定义“USB”，没有定义“USB_HID”，这部分菜单才会显示出来。请注意这种配置在内核中经常会遇到。

```
menu "USB HID Boot Protocol drivers"
        depends on USB!=n && USB_HID!=y && EXPERT
```

我们到 menuconfig 中将 USB_HID 取消配置，然后就可以看到 "USB HID Boot Protocol drivers"配置菜单了，如下图所示。

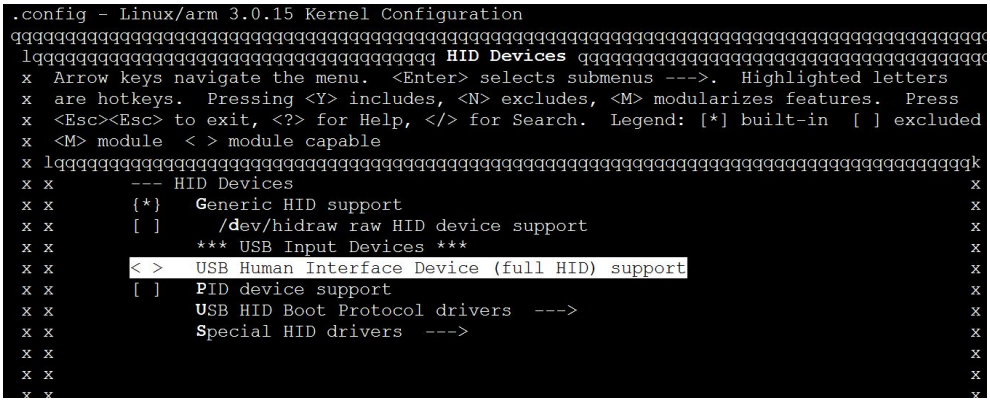


如上图所示，接着进入"USB HID Boot Protocol drivers"，如下图所示，USB 鼠标和键盘都是没有配置的。



前面介绍这么多，主要是给大家介绍更多的缺省文件配置的知识。

为了进行后面的实验，只需要去掉“USB Human Interface Device (full HID) support ”就可以了，如下图所示，其它地方都不需要动，这样是为了我们动态加载 USB 鼠标驱动，一步一步的实现 USB 鼠标驱动。



取消配置“USB Human Interface Device (full HID) support” ，然后退出保存，重新编译内核，烧写到开发板中，准备后续的实验。

2 USB 鼠标驱动的注册

做好准备工作之后，我们来查看一下 USB 鼠标驱动的注册部分。

USB 设备驱动的驱动注册的函数为 usb_register 和 usb_deregister，如下图所示。

```
static int __init usb_mouse_init(void)
{
    int retval = usb_register(&usb_mouse_driver);
    if (retval == 0)
        printk(KERN_INFO KBUILD_MODNAME ": " DRIVER_VERSION ":"
                DRIVER_DESC "\n");
    return retval;
}

static void __exit usb_mouse_exit(void)
{
    usb_deregister(&usb_mouse_driver);
}

module_init(usb_mouse_init);
module_exit(usb_mouse_exit);
```

这部分内容就不再赘述，它和前面介绍的字符驱动以及 I2C、SPI 驱动等等类似，由内核提供专门的注册和卸载函数。

我们需要注意的是，USB 驱动是可以热拔插的，再插入和拔掉设备的时候，肯定都需要对应的代码，如下图所示，可以看到 struct usb_driver usb_mouse_driver 中有定义 usb_mouse_probe 和 usb_mouse_disconnect。

```
static struct usb_driver usb_mouse_driver = {
    .name       = "usbmouse",
    .probe      = usb_mouse_probe,
    .disconnect = usb_mouse_disconnect,
    .id_table   = usb_mouse_id_table,
};
```

插入 USB 设备，进行初始化则进入“usb_mouse_probe”；拔掉 USB 卸载则进入“usb_mouse_disconnect”；其中的参数 name，则是驱动名称“usbmouse”，既然有驱动名称，那一定有设备名称，请注意前面介绍过的 USB 描述符，USB 描述符的具体内容是在 USB 设备中的，相当于设备注册是在实体的“USB 设备”中！另外 usb_mouse_id_table，

这部分用来匹配，在初始化 probe 的代码中我们再进行打印测试，这里我们先使用默认的配置即可。

我们将代码进行简化，只进行 USB 设备的初始化和卸载部分，代码如下所示。

```
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/usb/input.h>
#include <linux/hid.h>

MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
MODULE_LICENSE(DRIVER_LICENSE);


static int usb_mouse_probe(struct usb_interface *intf, const struct usb_device_id *id)
{
	printk("usb mouse probe!\n");
	return 0;
}

static void usb_mouse_disconnect(struct usb_interface *intf)
{
	printk("usb mouse disconnect!\n");
}

static struct usb_device_id usb_mouse_id_table [] = {
	{ USB_INTERFACE_INFO(USB_INTERFACE_CLASS_HID, USB_INTERFACE_SUBCLASS_BOOT,
		USB_INTERFACE_PROTOCOL_MOUSE) },
	{ }	/* Terminating entry */
};

MODULE_DEVICE_TABLE (usb, usb_mouse_id_table);

static struct usb_driver my_usb_mouse_driver = {
	.name		= "usbmouse",
	.probe		= usb_mouse_probe,
	.disconnect	= usb_mouse_disconnect,
	.id_table = usb_mouse_id_table,
```

```
};

static int __init my_usb_mouse_init(void)
{
    //注册 usb 驱动
    return usb_register(&my_usb_mouse_driver);
}

static void __exit my_usb_mouse_exit(void)
{
    //卸载 USB 驱动
    usb_deregister(&my_usb_mouse_driver);
}

module_init(my_usb_mouse_init);
module_exit(my_usb_mouse_exit);
```

上面这段代码要实现的功能很简单，就是加载驱动之后，如果有 USB 鼠标插入，则会打印“usb mouse probe!”，如果有 USB 鼠标拔出，则会打印“usb mouse disconnect!”。

如下图所示，我们加载该驱动，串口控制台提示如下。

```
[root@iTOP-4412]# insmod my_usb_mouse.ko
[ 1469.347854] usbcore: registered new interface driver usbmouse
[root@iTOP-4412]#
```

如上图所示，这部分是 usbcore 打印的，和我们的驱动中的代码没有直接的关系，在加载 usb 鼠标驱动之后，打印“注册了新的驱动，叫 usbmouse”。

接着我们接上鼠标，看到打印信息如下。

```
[root@iTOP-4412]# insmod my_usb_mouse.ko
[ 1469.347854] usbcore: registered new interface driver usbmouse
[root@iTOP-4412]# [ 1577.980679] usb 1-3.1: new low speed USB device number 5 using s5p-ehci
[ 1578.096092] usb 1-3.1: New USB device found, idVendor=046d, idProduct=c077, bcdDevice=7200
[ 1578.104863] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 1578.112481] usb 1-3.1: New USB device Class: Class=0, SubClass=0, Protocol=0
[ 1578.118250] usb 1-3.1: Product: USB Optical Mouse
[ 1578.122907] usb 1-3.1: Manufacturer: Logitech
[ 1578.156256] usb mouse probe!

[root@iTOP-4412]#
```

如上图所示，红色框中的就是驱动中的打印信息“usb mouse probe!”，其它部分全部是由 USB 主控制器完成。这说明匹配成功进入了 usb 鼠标的 probe。

接着我们拔掉 USB 鼠标，如下图所示，打印了“usb mouse disconnect!”这部分信息，是和我们 disconnect 函数中的打印对应。

```
[root@iTOP-4412]# [ 1698.098576] usb 1-3.1: USB disconnect, device number 6
[ 1698.102800] usb mouse disconnect!

[root@iTOP-4412]#
```

如上图所示，其它部分打印信息也是由主控制器来完成。

主控制器会统一管理每一个 usb hub 以及每一个 usb 接口，大部分工作其实都是有主控制器完成的！

至此，本文档要介绍的内容完成，后面的文档我们继续分析进入 probe 之后，我们需要进行的配置等工作。

联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需平板电脑案支持，请访问迅为平板方案网“http://www.topeet.com”，我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2018 年 2 月