基于 ARM9 系统的 USB 无线网卡驱动程序设计①

Design of USB WLAN Device Driver Based on ARM9

郭 磊 廖启征 魏世民 蔡 坤 李 伟 (北京邮电大学 自动化学院 100876)

摘要:USB 无线网卡驱动程序设计嵌入式系统的无线局域网接入的关键环节。本文介绍了基于 ARM902T 内核的硬件平台和 uclinux 环境下的一种 USB 接口无线网卡驱动程序的实现。同时对于开发中遇到自锁的问题进行了描述并提出了解决方案。并编写了应用程序,进行嵌入式系统和以太网的无线数据传输实验。实验结果验证了驱动程序的有效性。

关键词:uclinux 系统 ARM902T 内核 USB 接口无线网卡 驱动程序 嵌入式系统

1 引言

嵌入式系统的无线局域网接入可以实现嵌入式系统的无线控制无线数据传输。从而使人嵌入式系统的控制,运行,以及开发更加方便,并可以满足一些特殊的应用场合。本文通过对无线局域网 IEEE802.11b 协议规范和嵌入式系统的深入理解和分析,利用基于 ARM9 内核的嵌入式处理器、嵌入式操作系统 uClinux 和 802.11b 的无线网卡来实现嵌入式系统的无线局域网接入。

2 开发环境介绍

开发环境分为硬件平台和软件平台两个方面。硬件平台选择了基于 S3C2410 处理器的 HHARM9 - EDU 实验系统,无线网卡选用了 corega 公司生产的 Wireless Lan USB - 11 mini 型号无线网卡进行开发。这块无线网卡为 USB1.1 接口,符合 802.11b 无线传输协议,传输速率为 11M/s,主控芯片位 atmelb503。市面上大多数厂商包括 TP - link,3com 等公司的无线网主控芯片都以该芯片为主控芯片,故选择 atmelb503 芯片的无线网卡开发具有一定的一般性。

软件平台方面,选择了开放原代码及网络资源丰富的 uclinux 式操作系统。通过对 linux 操作系统网络协议实现以及驱动程序开发得深入分析,对系统进行了必要裁剪、编译、移植,以便于进行驱动程序和应用的程序的编写。

3 交叉编译环境的建立过程

3.1 目标板设置

根据目标板的硬件情况和开发的需求适当地裁减 Linux 内核,然后调试,编译得到针对开发板的 uclinux 内核。

3.2 宿主机设置

在宿主机上需建立交叉编译环境,首先完全安装的 Redhat9.0(内核为 2.4)操作系统,在 PC 机(宿主机)的根目录下安装了 HHARM9 - EDU 的目录和 opt 目录,其中 HHARM9 - EDU 是开发套件的源代码、驱动、以及相应的应用程序。opt 是 ARM 的编译器存放的目录。HHARM9 - EDU 所用的交叉编译工具都放在 opt/host/armv4l 下。

将目标板和宿主机的得 IP 设置为同一网段内,以便于连接。启动 NFS 服务,通过 telnet 登陆到目标板,然后用将宿主机挂载到目标板进行开发调试,并通过串口终端显示结果。交叉编译的开发环境如图 1 所示。

4 USB 接口无线网卡驱动程序设计

设备的驱动程序往往只支持某一类设备,在驱动

① 项目支持:国家自然科学基金项目(50475161);国家 973 项目(2004CB318000);
2004 年教育部科学技术研究重点项目(104043)和高等学校博士学科点专项科研基金资助课题(20050013006)。

程序中会列出该驱动程序所支持的的设备。对于不同厂商生产的同类设备，驱动程序根据设备的生产代号和产品代号（VENDOR ID，DEVICE ID）来区分他们的不同，并决定是否支持。因此在编写设备驱动程序时就必须首先搞清楚该设备的生产和产品 ID。

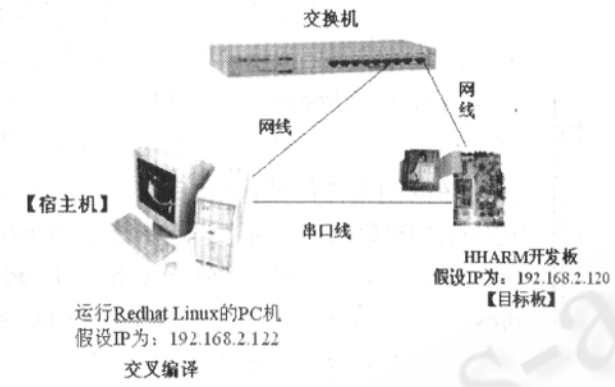

图 1　交叉编译环境

为了获得该无线网卡的 VENDOR ID 和 DEVICE ID，以便设备驱动程序对于该无线网卡硬件的识别。先在 Windows 操作系统下插入该无线网卡，然后进入设备属性，就可以看到该无线网卡的 VENDOR ID 和 DEVICE ID 分别为 0x07aa，0x0011。获得此 ID 后，在驱动程序中加入以下语句，对设备进行声明：

```
#define VENDOR_ID_MY 0x07aa
#define PRODUCT_ID_MY 0x0011
```

并在如下的设备列表的结构体中加入该设备的声明：

```
static struct usb_device_id dev_table[ ] = {
{USB_DEVICE( VENDOR_ID_MY, PRODUCT_ID_MY )
//声明该设备的生产代号和产品代号
}
}
```

该无线网卡的驱动程序分为如下的三个模块：（1）USB 接口驱动程序；（2）Atmel76C503 芯片的驱动程序；（3）固件升级程序模块。它们分别对应着 at76c503. c，usbdfu. c，at76c503 – rfmd. c 三个文件。由于 Atmel76C503 芯片中集成了 usb1.1 接口单元并提供无线传输协议 802. 11b 的支持。故在文件 at76c503. c 中包含了：芯片的初始化、usb 初始化，以及 802.11b 协议的实现。在网卡驱动加载系统检查到设备后由 at76c503 – rfmd. c 控制下载固件。

驱动程序首先需要在内核中进行注册，以告诉系统自己的模块名以及函数的入口点等信息，结构体 usbdfu_driver 在系统中注册一个 USB 设备驱动：

usb_driver 结构体必须由驱动程序填写，它向 usb 核心代码描述了 USB 驱动程序，在此结构体中，对驱动程序的整体作了定义。usbdfu_probe 为驱动程序探测函数，当设备被安装，USB 核心认为该设备应被处理时，探测函数被调用。它探测设备的端点地质和缓冲大小等信息。当探测完成后，驱动程序调用注册函数 usbdfu_register 将设备注册到 USB 核心，只要该函数被调用，就确保该设备和驱动程序都处于可以处理用户访问设备的要求的状态。在设备被拔出，或者发生异常时，驱动程序需要调用断开函数，将设被断开断开，同时需要调用 usbdfu _deregister 解除注册，以释放系统资源。

5　编写 Makefie 文件

首先设置 Makefile 中的 CC 变量，即设定编译器。在华恒 ARM9EDU 嵌入式开发平台环境中需要将它设置为：

```
CC = /opt/host/armv4l/bin/armv4l – unknown – linux
– gcc
//将编译器设置为交叉编译器
```

对于编译所需的头文件，系统文件，内核调用等都设置为 ARM9EDU 平台下 uCLinux 的相应文件。

当 Makefile 编写完毕后，对 USB 无线网卡驱动程序进行交叉编译。系统没有返回任何错误，说明 Makefile 文件初步成功。

编译通过后，就应该完成驱动程序的载入。在宿主机终端以 telnet 方式登陆目标板（支持 NFS），然后 mount 进入宿主机根目录，找到交叉编译的三个. o 文件，在超级终端中使用“insmod”命令将它们插入内核，注意，由于三个内核间有依赖关系故在插入内核时，必修按照以下顺序：

```
#insmod at76c503. o
# insmod usbdfu. o
# insmod at76c503 – rfmd. o
```

插入完毕后，察看插入模块是否成功：

```
#lsmod
```

得到如图 2 所示的信息：

从图 2 中可以看到模块列表中有 at76c503，usbd-

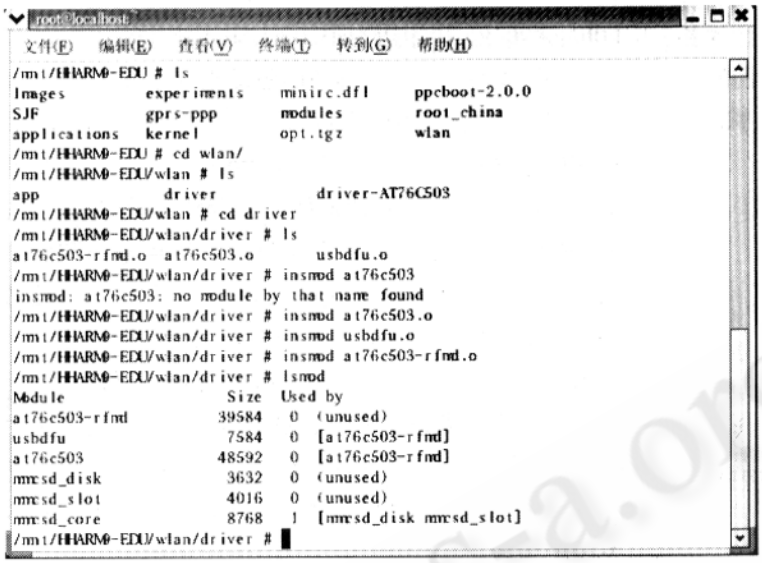

图 2　模块显示结果

fu，at76c503 – rfmd 三项说明驱动已加载。将无线网卡与 HHARM9 – EDU 的 USB 接口相连，看到网卡信号指示灯点亮，说明连接无误，在终端中键入：#dmesg ，得到的设备信息如图 3 所示。

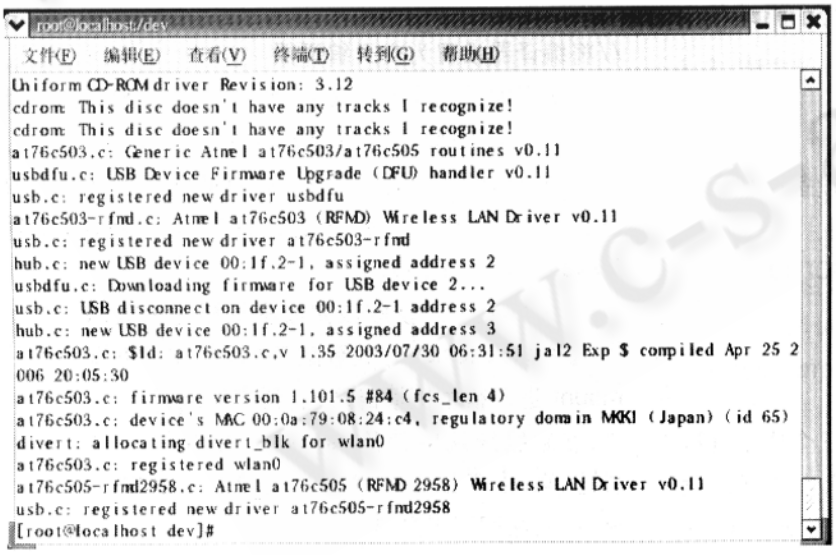

图 3　设备信息显示

在图 3 显示的倒数第七行，我们看到 at76c503.c: device's MAC 00:0a:79:08:24: c4 ，regulatory domain in MKK1 (jappan) (id 65)，这说明 usb 无线网卡成功得被驱动程序检测到，系统得到了网卡的 Mac 地址，该地址与网卡卡生产厂家所标识的地址一致，至此，基于嵌入式 uclinux 操作系统的无线网卡驱动的开发工作完成。

6　驱动程序出现的问题及其解决方法

6.1　无法正确初始化

在将无线网卡插入目标板加载驱动后，绝大多数时候可正常检测到设备。但在极少出现的情况下 dmesg 后无法得到设备信息。分析该问题，查阅资料最后发现是 ARM9 芯片 sc2410 自身的问题，导致启动时无法正常初始化 UPLLCON 寄存器，于是修改了 driver/usb/usb.c 的如下代码，使得 usb_set_address 失败的时候，再设置一次 UPLLCON，这样就不会再出现此问题了。

6.2　自锁

在目标板上进行网卡的热插拔测试时，将 USB 网卡带电拔下之后，再次连接 usb 设备，则驱动失败，使用 dmesg 命令也无得到设备信息

其原因在于：hub.o 中的：usb reset device( ) 函数（重置 hub）与其他进程间发生了死锁。

在 Linux 等多用户操作系统中提供一种信号灯机制来防止两个或多个进程同时访问共享资源

在我们早期的程序中，出现一种隐性死锁，程序进入互斥区做设备离开后的收尾工作，之后等待另个进程的通知：waite ms(10)，同时会继续保持锁。然而由于进程在两互斥区等待，其他进程无法通知该进程运行结果，因此导致了死锁的产生。收尾工作没有顺利完成，导致错误。

其解决方法为：修改 usb_reset_device( ) 函数，将 wait ms(10) 函数移出互斥区即可。在互斥区外等待其他进程传递信息，将会保证通讯的顺利进行。

7　结论

本文将嵌入式系统与无线局域网相结合，提出了

（下转第 98 页）

(上接第 94 页)

一种嵌入式无线接入网的实现方案,利用 ARM9 嵌入式处理器、嵌入式操作系统 uClinux 和 802.11b 的无线网卡来实现嵌入式系统的的无线接入。通过对软硬件平台的测试和实际应用,验证了该方案的有效性。

参考文献

1 Alessandro Rubini&Jonathan Corbet, Linux Device Driver, 3rd Edition,2005.1:327 –361.

2 魏洪兴、胡亮、曲学楼编著,嵌入式系统设计于实例开发实验教材 II ,清华大学出版社 2005.12:‖1 –14 127 –146 244 –249.

3 李玉波、朱自强、郭军编著,Linux C 编程,清华大学出版社 2005.9 ,68 –128.

4 毛德操、胡希明著,Linux 内核源代码情景分析,浙江大学出版社,2001.5.