# 嵌入式 linux 下 USB 设备自动加载

董光

（长春理工大学 计算机科学技术学院，长春 130022）

**摘 要：**分析了 Linux 驱动程序的结构及特点，对 USB 驱动程序的结构进行了深入研究。在分析了 USB Host 的 OHCI 标准和 USB 热插拔特性的基础上，利用热插拔脚本实现 USB 设备自动加载。在 Windows 下使用虚拟机 VMWare，建立嵌入式 Linux 开发环境，并实现 Linux 系统对 U 盘的自动加载。

**关键词：**嵌入式 Linux；驱动程序；USB；热插拔

**中图分类号：**TP316 **文献标识码：**A **文章编号：**1672-9870（2010）02-0165-04

# Automatic Mount of USB Equipment on Embedded Linux

DONG Guang

(College of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022)

Abstract: The papers analyzes Linux kernel, introduce the subsystem of Linux kernel and the interrelations. It also makes some necessary analysis about Linux document system. Study the structure of driving program, working system and different features of three kinds of driving programs. Through analyzing the structure of driving programs of USB, make some detail notes for the data structure of core driving program of USB. On the base of analyzing OHCI standard of USB Host and the features of USB hotplug, give the ways to automatic mount of USB equipment, and realize the automatic mount of U-disk by Linux system .

Key words: embedded Linux; driving program; USB; hotplug

在 Linux 操作系统中，设备驱动程序是操作系统内核与硬件设备之间的接口，分为驱动程序与操作系统内核的接口、驱动程序与系统引导的接口、驱动程序与设备的接口[1]。它屏蔽了硬件的细节，是内核的一部分，完成对设备初始化和释放；对设备进行管理，包括实时参数设置以及提供对设备的操作接口；负责应用程序和设备文件间的数据传输；检测处理设备出现的错误[2]。在应用程序看来，硬件设备只是一个设备文件，应用程序可以像操作普通文件一样对硬件设备进行操作。

## 1 嵌入式 linux 驱动分析

### 1.1 Linux 设备驱动程序结构及特点

每个设备文件都有其文件属性（c/b），表示是字符设备还是块设备。两个设备号，主设备号标识驱动程序，取值空间是 0 到 255，其中USB设备是 180。从设备号标识使用同一个设备驱动程序的不同的硬件设备[3]。

设备驱动程序由三部分组成：1.自动配置和初始化子程序，负责检测所要驱动的硬件设备是否存在和是否能正常工作。2.服务于 I/O 的子程序，又称驱动程序的上半部分。3.中断服务程序，又称驱动程序的下半部。

在 linux 中，设备驱动程序是核心的一部分，象核心中其它代码一样，出错将导致系统的严重损伤。设备驱动程序必须为 Linux 核心或者其从属子系统提供一个标准接口。多数的 Linux 设备驱动程序可以在核心模块发出加载请求时进行加载，同时在不再使用设备时进行卸载。Linux 设备驱动程序

收稿日期：2009-07-13
基金项目：吉林省科技支撑计划重点项目（20090307）
作者简介：董光（1963-），男，副研究员，主要从事智能软件、图像处理与模式识别、嵌入式软件环境方面的研究。

可以连接到核心中。当核心被编译时，哪些驱动程序被连入核心是用户自己可以进行配置的。当系统启动及设备驱动程序初始化的时候查找它所控制的硬件设备。

### 1.2 USB 设备驱动

Linux 为 USB 创建了 USB 文件系统，所有的 USB 设备驱动程序和任何连接到主机的 USB 设备的接口都被 USB 文件系统管理。USB 文件系统使得所有 USB 设备看起来只是一个普通文件，并且对 USB 总线上的设备的增加和删除进行动态监控[4]。

USB 系统包括硬件电路和软件两部分，硬件上通过 USB 芯片实现。软件部分由 USB 主机控制器驱动程序 HCD（Host Controller Driver）、USB 核心驱动 USBD（Universal Serial Bus Driver）和 USB 设备驱动组成。其中 HCD 和 USBD 就是我们通常说的协议栈，这两部分共同处理与协议相关的操作，如图 1 所示。客户端驱动程序可以包含多个，不同的功能接口对应不同的驱动程序，它们不直接与 USB 设备硬件打交道，而是通过协议栈的抽象处理来完成与设备的不同功能接口之间的通信[5]。

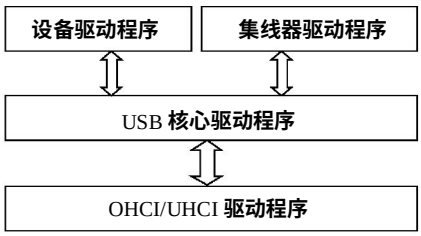


图 1 USB 系统的层次结构

Fi.1 Systematic adecker of USB

USB 设备驱动是最终与应用程序交互的软件模块，其主要为应用程序提供访问接口、提供驱动程序管理、设备接口的插拔处理和对设备控制与驱动等功能。但是，对于不同设备驱动程序，其功能的具体实现不同，在开发时，须对 USB 提供类协议进行分析。

## 2 热插拔脚本实现 USB 设备自动加载

### 2.1 USB Host 的 OHCI 标准

1995 年底，Compaq、Microsoft、National Semiconductor 公司推出了 USB OHCI（Open Host Control Interface，开放式主控制器接口）标准。OHCI 是一个完全开放的标准，它定义了一系列寄存器和相关的数据结构，以统一 USB 主机控制器和驱动程序接口，其结构如图 2 所示。

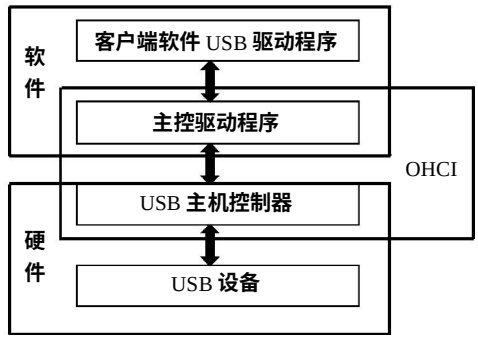


图 2 OHCI 在 USB 中的层次

Fig.2 OHCI is in arrangement of ideas in USB

### 2.2 Linux 的 USB OHCI 驱动程序接口分析

因为有 OHCI 定义的寄存器标准，Linux 下也提供相应的驱动程序，所以只要符合 OHCI 标准的 USB 主机接口，很容易在 Linux 下被驱动起来。

常用的 USB 设备主要有 U 盘、USB 键盘/鼠标、USB 摄像头等，它们在 Linux 内核中都有相应的驱动程序。可通过配置 Linux 内核时在 USB support 菜单中添加对各种 USB 设备的支持。U 盘的驱动程序是 USB Mass Storage support; USB 键盘/鼠标是 USB Human Interface Devices（HID）。使用 Preliminary USB device filesystem 选项，可开启 USB 设备文件系统的支持。系统启动后通过命令 Mount -t usbdevfs none /proc/bus/usb 加载 USB 设备文件系统。

从上述可以看到，USB 总线上有两个设备：最底层是 OHCI 标准的根 Hub（Product=USB OHCI Root Hub），在它上面是一个 USB 大容量存储设备（Product=USB Mass Storage Device），也就是 U 盘。

### 2.3 USB 热插拔脚本

USB 是一种热插拔总线，它允许在系统运行时动态地添加或删除设备。但是，还不能实现 USB 设备的自动加载，而必须通过命令挂载后才能正常使用。从 linux2.4 内核开始，引入了 hotplug 的方法实现对热插拔的支持。在配置内核时，选择 General setup 中的选项 Sopport hot-pluggable devices 可实现内核对 hotplug 的支持。

有了热插拔的支持后，就可以插入一个新设备并立即使用该设备了，因为系统内核可主动调用一个热插拔的时间来自动配置新插入的设备。

热插拔事件可用来调用一个代理任务来配置适配器、子系统或程序。代理任务利用脚本来处理并

管理这种配置。这种脚本的框架就称为热插拔策略代理。这些代理与/sbin/hotplug 脚本中的一项绑定在一起。这个脚本可用来调试和记录事件的变化，并控制系统的热插拔代理。热插拔代理是目录/etc/hotplug 中的一些脚本文件。一个热插拔时间可调用一个或多个代理类型，其名称格式为 type.agent。对于 USB 设备来说就是/etc/hotplug/usb.agent。系统上电启动正常以后，一个 USB 设备的 hotplug 调用过程如下：

1.用户插入 USB 设备。

2.USB 总线报告有新设备插入，读取设备的 PID(product ID)、VID(vendor ID)等相关信息。

3.如果找到驱动程序，则 Linux 会自动加载对应的 USB 设备驱动程序。

4.内核调用文件系统中的/sbin/hotplug 脚本，并传递相关的环境变量。

5./sbin/hotplug 脚本进行解析后，发现是 USB 设备，它会调用 USB 设备配置代理脚本/etc/hotplug/usb.agent。

6./etc/hotplug/usb.agent 脚本根据读取到的设备信息 PID、VID 和 Class 等判断设备类型，实现相应的动作，如模块加载对应的驱动程序等。

从这个过程中可见，hotplug 是内核主动发起调用应用程序的过程。/sbin/hotplug 脚本是所有 hotplug 的入口。

其关键脚本如下：

```
AGENT= /etc/hotplug/ $ 1.agent
# $ 1 为系统调用时传递的第一个参数，就 USB 设备来说是$ 1 =“usb”
if ［-x $ AGENT］; then
# 如果存在 xxx.agent 且类型为可执行，则继续进行
shift
if ［“$ DEBUG”!=“”］; then
mesg“invoke $ AGENT ($ @)”
fi
exec $ AGENT“$ @”
# 执行 xxx.agent，执行后退出，下面的代码不会被执行到
  mesg“couldn't exec $ AGENT”
  exit 1
fi
```

如果有 USB 设备插入系统内核，就会相当于以下面命令行的方式调用 hotplug 脚本：

/sbin/hotplug usb

经过/sbin/hotplug，/etc/hotplug/usb.agent 就会被调用。

## 2.4　U 盘的自动加载实现

本文的开发环境是在 Windows 下安装虚拟机 VMWare5.5，并在虚拟机中安装 Linux Red Hat 9.0 操作系统，所有工作都是在此环境下完成的。

在安装 Linux 系统时，选择“内核开发”选项，内核源码就直接安装在/usr/src 的目录下，版本号为 2.4.20-8。在 Linux 系统中配置 U 盘驱动程序（USB Mass Storage support），按照 Mass Storage 协议，把 U 盘作为一个虚拟的 SCSI 磁盘设备来使用。所以，要想正常使用 U 盘，还需要内核支持 SCSI 设备，在配置内核时，选择对 SCSI 设备（SCSI support）的支持。

系统上电启动正常以后，当一个 U 盘插入时，系统会发生如下动作：

1.用户插入 U 盘。

2.USB 总线报告有新设备插入，读取设备相关信息，得知是 U 盘设备。

3.找到并启用 Mass Storage 驱动程序。

4.系统会添加一个新的 SCSI 设备，并通过设备文件系统在/dev/scsi 目录下创建节点。

5.内核调用文件系统中的/sbin/hotplug 脚本，并传递相关的环境变量。

6./sbin/hotplug 脚本进行解析后，发现是 USB 设备，它会调用 USB 设备配置代理脚本/etc/hotplug/usb.agent。

通常，当 U 盘插入以后，可通过命令：

mount -t vfat /dev/scsi/host0/bus0/target0/lun0/part1 /mnt/udisk

把 U 盘挂载到/mnt/udisk 目录下，或者也可使用/etc/fstab 中的路径。

要想实现 U 盘插入以后的自动挂载，需在/etc/hotplug/usb.agent 脚本中添加如下代码：

```
case $ ACTION in
add)
……
USB_MASS_INT_CLASS =“8”
DIR=“/dev/scsi”
if ［$ USB_MASS_INT_CLASS -eq $ usb_bInterfaceClass］; then
for I in“ $ {DIR}/”* ; do
if ［-d $ I -a -e $ I/bus0/target0/lun0/part1］; then
/bin/mount -t vfat -o iocharset= cp936\
$ I/bus0/target0/lun0/part1 /mnt/udisk;
fi
done
```

```
fi
;;
remove)
……
```

其中，环境变量$ACTION为add，表示有设备插入；remove 表示设备被拔出。“USB_MASS_INT_CLASS=”8“ ”代表 USB Mass Storage 设备（按照 USB Mass Storage 协议，所有的 U 盘设备 class 都为 8）。内核在调用 hotplug 时，把 USB 的 class 通过参数传递给/sbin/hotplug 脚本，接着被传递到usb.agent脚本中。usb.agent脚本通过分析传递进来的参数，把当前 USB 设备的 class 记录在环境变量 $usb_bInterfaceClass 中。通过它与 USB_MASS_INT_CLASS 的比较，就可以判断 U 盘设备。接着，在“/dev/scsi”目录下搜索/bus0/target0/lun0/part1 设备，搜索到以后进行自动加载工作。

## 3 结束语

根据 USB 支持热插拔特性，本文提出了利用热插拔脚本实现 USB 设备自动加载的方法，并对此方法做了详细的说明。最后利用热插拔脚本实现 Linux 系统对 U 盘的自动加载，完成对 Linux 驱动程序的改进。

实现 USB 设备的自动加从结果上来看，利用热插拔脚本载，无论使用装有 Linux 系统的 PC 机还是装有 Linux 系统的嵌入式设备，用户在使用 USB 设备时都不再需要手动加载。不仅提高工作效率，同时也避免手动加载时输入命令容易出错的问题。

### 参考文献


[1] 李驹光,郑耿. 基于嵌入式 Linux 的设备驱动程序开发[J].电脑编程技巧与维护,2005,11(11):15-19.

[2] Klaus Wehrle,Frank Pahlke. The Linux Networking Architecture:Design and Implementation of Network Protocols in the Linux Kernel[J]. Prentice Hall,2004: 99-104.

[3] 巍骂,张焕强,方贵明. 基于 Linux 的 USB 驱动程序实现[J]. 计算机应用,2002,22(8):17-19.

[4] 郑智. 嵌入式 Linux 下 USB 驱动程序开发研究[J]. 武汉理工大学学报信息与管理工程版,2006,28(7):117-120.

[5] Imai,Yoshiro. A Linux-based engineering education with hardware implementation,devicedrivers' programming and network literacy learning[J].ITHET 2004: 463-46.


（上接第 164 页）

的数据采集过程。采集联邦成员订购所有的数据并在数据到达时记录数据。采集联邦成员在整个联邦中被动地收集数据，它只是订购在 FOM 中定义的所有数据，将反射属性和接收到的交互类存储到分布式缓冲区中，然后触发SWITCH中间件，将数据写入到当前活跃的 Oracle 数据库服务器中。

## 4 结论

本文分析了光电对抗半实物仿真系统采集数据的类型，发现了实时数据存储这一难点；提出了使用分布式存储方法来实现对大批量实时数据进行存储的方法，同时将使用分布式存储方法进行存储和直接向数据库进行数据插入这两种方法作了对比，通过比较分析了使用分布式存储方法的优点，解决了 Oracle 数据库管理系统的磁盘 I/O 瓶颈问题，大大提高了仿真系统的数据实时存储性能，同时该方法通过多个服务器的使用大大提高了系统数据存储的可靠性；最后实现了这一方法在光电对抗半实物仿真系统中的应用。

### 参考文献


[1] 冯贵江,李言俊,张科,等.基于 HLA 光电对抗仿真系统的设计与实现[J].计算机工程设计 2008(4):2055-2058.

[2] 厉明,纪勇,贾宏光,等.基于快速仿真原型的飞行器半物理仿真系统[J]. 光学 精密工程,2008,16(10):1949-1955.

[3] 赵炜渝.光电对抗仿真试验技术[J].红外与激光工程,2001,30(3):171-175.

[4] 李艳峰,刘延斌,金光.机载光电平台地面测试系统目标模拟分系统的建模与半物理仿真实现[J]. 光学精密工程,2004(2):193-196.

[5] 蒋夏军,李蔚清,吴慧中.高级分布式仿真中的数据收集技术研究[J].系统仿真学报,2004(8):1758-1767.

[6] 向方.基于内存数据库的 HLA 仿真数据收集方法研究[J].数据库及信息管理,2007(7):596-597.