Tekkaman Ninja

tekkamanninja.blog.chinaunix.net

tekkamanninj

博客访问： 75935
博文数量： 263
博客积分： 15936
博客等级： 上将
技术积分： 13951
用 户 组： 普通用户
注册时间： 2007-03-27 11:22

加关注 短消息
论坛 加好友

个人简介

Fedora-ARM

文章分类

文章存档

Linux设备驱动程序学习（14）-Linux设备模型（各环节的整合） 2008-01-02 13:27:50

分类： LINUX

Linux设备驱动程序学习（14）
-Linux设备模型（各环节的整合）

通过一个设备在内核中生命周期的各个阶段，可以更好地理解Linux设备模型。我将通过分析lddbus和sculld的源码来了解Linux设备模型中各环节的整合。《LDD3》中的（PCI总线）各环节的整合这部分内容作为参考资料，因为嵌入式Linux比较少用到PCI总线。看这部分内容一定要先熟悉一下 lddbus 和 sculld 的源码。

一、lddbus模块：添加总线、导出总线设备和设备驱动的注册函数。

lddbus子系统声明了一个bus_type结构，称为ldd_bus_type 。源码是在编译时初始化了这个结构体，源码：

```
/*
 * And the bus type.
 */
struct bus_type ldd_bus_type = {
    .name = "ldd",
    .match = ldd_match,
    .uevent  = ldd_uevent,
};
```

在将lddbus子系统装载到内核和从内核卸载的源码如下：

```
static int __init ldd_bus_init(void)
{
    int ret;

    ret = bus_register(&ldd_bus_type); /*注册总线，在调用这个函数之后ldd_bus_type 结构体将向内核注册，在/sys/bus中出现ldd文件夹，其中包含两个目录：devices 和 drivers */
    if (ret)
        return ret;
    if (bus_create_file(&ldd_bus_type, &bus_attr_version)) /*添加总线属性,将在/sys/bus/ldd目录中出现version属性文件*/
        printk(KERN_NOTICE "Unable to create version attribute ! \n");
    ret = device_register(&ldd_bus);/*将总线作为设备注册。因为总线也可以是一个设备，比如在S3C2440中SPI总线控制器相对于ARM920T核心来说，其实就是一个外设。调用此函数后，就会在/sys/devices中出现ldd0目录*/
    if (ret)
        printk(KERN_NOTICE "Unable to register ldd0 ! \n");

    printk(KERN_NOTICE "Mount lddbus ok !\nBus device is ldd0 !\nYou can see me in sys/module/ , sys/devices/ and sys/bus/ ! \n");

    return ret;
}
```

```
static void ldd_bus_exit(void)
{
    device_unregister(&ldd_bus);
    bus_unregister(&ldd_bus_type);
}

module_init(ldd_bus_init);
module_exit(ldd_bus_exit);
```

lddbus模块的主要部分就是这些，很简单。因为这只不过是一个虚拟的总线，没有实际的驱动。模块还导出了加载总线设备和总线驱动时需要用到的注册和注销函数。对于实际的总线，应该还要导出总线的读写例程。

将总线设备和驱动注册函数放在lddbus模块，并导出给其他的总线驱动程序使用，是因为注册总线设备和驱动需要总线结构体的信息，而且这些注册函数对于所有总线设备和驱动都一样。只要这个总线驱动一加载，其他的总线驱动程序就可以通过调用这些函数注册总线设备和驱动，方便了总线设备驱动的作者，减少了代码的冗余。

这些注册函数内部调用driver_register、device_register 和 driver_unregister、device_unregister 这些函数。

二、sculld模块：在scull的基础上添加设备和驱动注册和注销函数。

sculld模块基本和scull模块实现的功能一致，我参考《LDD3》提供的sculld，将以前实验过的功能较全的scull进行修改。主要的修改如下（其他还有些小改动）：

```
//*******在源码的声明阶段添加如下代码，以增加设备和驱动的结构体*****
struct sculld_dev *sculld_devices; /* allocated in scull_init_module */

/* Device model stuff */
static struct ldd_driver sculld_driver = {
    .version = "$Revision: 1.21-tekkamanninja $",
    .module = THIS_MODULE,
    .driver = {
        .name = "sculld",
    },
};
//************************************************************

//******增加设备注册函数和设备号属性*********************************
static ssize_t sculld_show_dev(struct device *ddev, struct
device_attribute *attr , char *buf)
{
    struct sculld_dev *dev = ddev->driver_data;
    return print_dev_t(buf, dev->cdev.dev);
}

static DEVICE_ATTR(dev, S_IRUGO, sculld_show_dev, NULL);

static void sculld_register_dev(struct sculld_dev *dev, int index)
{
    sprintf(dev->devname, "sculld%d", index);
    dev->ldev.name = dev->devname;
    dev->ldev.driver = &sculld_driver;
    dev->ldev.dev.driver_data = dev;
    register_ldd_device(&dev->ldev);
    if (device_create_file(&dev->ldev.dev, &dev_attr_dev))
    printk( "Unable to create dev attribute ! \n");
}
//************************************************************

/*还要在模块的初始化函数和模块清除函数中添加设备和驱动的注册和注销函数*/
```

```
    sculld_register_dev(sculld_devices + i, i);
    register_ldd_driver(&sculld_driver);
    unregister_ldd_device(&sculld_devices[i].ldev);
    unregister_ldd_driver(&sculld_driver);
```

修改后好模块就可以实现向sysfs文件系统导出信息。

三、分析设备和驱动注册和注销核心函数，了解一般的注册、注销过程。

以下也参考了《LDD3》中的PCI驱动分析

（1）设备的注册

在驱动程序中对设备进行注册的核心函数是：

```
int device_register(struct device *dev)
{
    device_initialize(dev);
    return device_add(dev);
}
```

在 device_register 函数中，驱动核心初始化 device 结构体中的许多成员，向 kobject 核心注册设备的 kobject （ 导致热插拔事件产生），接着添加设备到其 parent 节点所拥有的设备链表中。此后所有的设备都可通过正确的顺序被访问，并知道其位于设备层次中的哪一点。

设备接着被添加到总线相关的设备链表（包含了所有向总线注册的设备）中。接着驱动核心遍历这个链表，为每个驱动程序调用该总线的match函数。

match函数主要是将驱动核心传递给它的 struct device 和 struct device_driver转换为特定的设备、驱动结构体 ，检查设备的特定信息，以确定驱动程序是否支持该设备：

若不支持，函数返回 0 给驱动核心，这样驱动核心移向链表中的下一个驱动；

若支持，函数返回 1 给驱动核心，使驱动核心设置struct device 中的 driver 指针指向这个驱动，并调用在 struct device_driver 中指定的 probe 函数.

probe 函数(又一次) 将驱动核心传递给它的 struct device 和 struct device_driver转换为特定的设备、驱动结构体 ，并再次验证这个驱动是否支持这个设备，递增设备的引用计数，接着调用总线驱动的 probe 函数：

若总线 probe 函数认为它不能处理这个设备，则返回一个负的错误值给驱动核心，这样驱动核心移向链表中的下一个设备；

若这个 probe 函数能够处理这个设备，则初始化这个设备，并返回 0 给驱动核心。这会使驱动核心添加设备到与这个特定驱动所绑定的设备链表中，并在 /sys/bus的总线目录中的 drivers 目录中创建一个到这个设备符号链接（指向/sys/devices中的设备），使用户准确知道哪个驱动被绑定到了哪个设备。

（2）设备的注销

在驱动程序中对设备进行注销的核心函数是：

```
void device_unregister(struct device * dev)
```

在 device_unregister 函数中，驱动核心将删除这个设备的驱动程序(如果有)指向这个设备的符号链接，并从它的内部设备链表中删除该设备，再以 device 结构中的 struct kobject 指针为参数，调用 kobject_del。kobject_del 函数引起用户空间的 hotplug 调用，表明 kobject 现在从系统中删除，接着删除所有该 kobject 以前创建的、与之相关联的 sysfs 文件和目录。kobject_del 函数也去除设备自身的 kobject 引用。此后，所有的和这个设备关联的 sysfs 入口被去除，并且和这个设备关联的内存被释放。

（3）驱动程序的注册

在驱动程序中对驱动程序进行注册的核心函数是：

```
int driver_register(struct device_driver * drv)
```

driver_register 函数初始化 struct device_driver 结构体（包括 一个设备链表及其增删对象函数 和一个自旋锁），然后调用 bus_add_driver 函数。

bus_add_driver进行如下操作：

（1）查找驱动关联的总线：若未找到，立刻返回负的错误值；
（2）根据驱动的名字和关联的总线，创建驱动的 sysfs 目录；
（3）获取总线的内部锁，遍历所有的已经注册到总线的设备，为这些设备调用match函数，若成功,进行

剩下的绑定过程。（类似注册设备，不再赘述）

（4）驱动程序的注销

删除驱动程序是一个简单的过程，在驱动程序中对驱动程序进行注销的核心函数是：

```
void driver_unregister(struct device_driver * drv)
```

deiver_unregister 函数通过清理在 sysfs 树中连接到这个驱动入口的 sysfs 属性，来完成一些基本的管理工作。然后遍历所有属于该驱动的设备，为其调用 release 函数（类似设备从系统中删除时调用 release 函数）。

> 在所有的设备与驱动程序脱离后,通常在驱动程序中会使用下面两个函数:
>
> down(&drv->unload_sem);
>
> up(&drv->unload_sem);
>
> 它们在函数返回给调用者之前完成。这样做是因为在安全返回前，代码需要等待所有的对这个驱动的引用计数为 0。
>
> 模块卸载时，通常都要调用 driver_unregister 函数作为退出的方法。 只要驱动程序被设备引用并且等待这个锁时，模块就需要保留在内存中。这使得内核知道何时可以安全从内存删除驱动。

四、ARM9开发板实验

实验源码：http://blogimg.chinaunix.net/blog/upfile2/080109150139.rar

实验过程：

```
[Tekkaman2440@SBC2440V4]#insmod /lib/modules/lddbus.ko
Mount lddbus ok !
Bus device is ldd0 !
You can see me in sys/module/ , sys/devices/ and sys/bus/ !
[Tekkaman2440@SBC2440V4]#tree -AC /sys/module/lddbus/ /sys/devices/ldd0/ /sys/bus/ldd/
/sys/module/lddbus/
├── holders
├── initstate
├── refcnt
└── sections
    ├── __ksymtab
    └── __ksymtab_strings
/sys/devices/ldd0/
├── power
│   └── wakeup
└── uevent
/sys/bus/ldd/
├── devices
├── drivers
├── drivers_autoprobe
├── drivers_probe
└── version

5 directories, 9 files
[Tekkaman2440@SBC2440V4]#cat /sys/bus/ldd/version
Revision: 1.9-tekkamanninja
[Tekkaman2440@SBC2440V4]#rmmod lddbus
The LDD bus device
 ldd_bus_release : lddbus
[Tekkaman2440@SBC2440V4]#ls /sys/module /sys/devices /sys/bus
/sys/bus:
i2c ide mmc platform serio spi usb

/sys/devices:
platform system

/sys/module:
8250 loop rd snd_seq usbnet
```

```
atkbd mac80211 redboot snd_seq_oss v4l1_compat
cdrom mousedev rfd_ftl snd_soc_core vt
dm9000 nfs rtc_ds1307 snd_timer yaffs
hid ohci_hcd s3c2410_wdt spidev zc0301
ide_cd printk snd sunrpc
keyboard psmouse snd_pcm tcp_cubic
lockd rcupdate snd_pcm_oss usbcore
[Tekkaman2440@SBC2440V4]#insmod /lib/modules/sculld.ko
sculld: Unknown symbol register_ldd_device
sculld: Unknown symbol register_ldd_driver
sculld: Unknown symbol unregister_ldd_driver
sculld: Unknown symbol unregister_ldd_device
insmod: cannot insert '/lib/modules/sculld.ko': Unknown symbol in module (-1): No such
file or directory
[Tekkaman2440@SBC2440V4]#insmod /lib/modules/lddbus.ko
Mount lddbus ok !
Bus device is ldd0 !
You can see me in sys/module/ , sys/devices/ and sys/bus/ !
[Tekkaman2440@SBC2440V4]#insmod /lib/modules/sculld.ko
[Tekkaman2440@SBC2440V4]#tree -AC /sys/module/lddbus/ /sys/devices/ldd0/ /sys/bus/ldd/
/sys/module/sculld/
/sys/module/lddbus/
├── holders
│   └── sculld -> ../../../module/sculld
├── initstate
├── refcnt
└── sections
    ├── __ksymtab
    └── __ksymtab_strings
/sys/devices/ldd0/
├── power
│   └── wakeup
├── sculld0
│   ├── bus -> ../../../bus/ldd
│   ├── dev
│   ├── driver -> ../../../bus/ldd/drivers/sculld
│   ├── power
│   │   └── wakeup
│   ├── subsystem -> ../../../bus/ldd
│   └── uevent
├── sculld1
│   ├── bus -> ../../../bus/ldd
│   ├── dev
│   ├── driver -> ../../../bus/ldd/drivers/sculld
│   ├── power
│   │   └── wakeup
│   ├── subsystem -> ../../../bus/ldd
│   └── uevent
├── sculld2
│   ├── bus -> ../../../bus/ldd
│   ├── dev
│   ├── driver -> ../../../bus/ldd/drivers/sculld
│   ├── power
│   │   └── wakeup
│   ├── subsystem -> ../../../bus/ldd
│   └── uevent
├── sculld3
│   ├── bus -> ../../../bus/ldd
│   ├── dev
│   ├── driver -> ../../../bus/ldd/drivers/sculld
```

```
│   ├── power
│   │   └── wakeup
│   ├── subsystem -> ../../../bus/ldd
│   └── uevent
└── uevent
/sys/bus/ldd/
├── devices
│   ├── scculld0 -> ../../../devices/ldd0/scculld0
│   ├── scculld1 -> ../../../devices/ldd0/scculld1
│   ├── scculld2 -> ../../../devices/ldd0/scculld2
│   └── scculld3 -> ../../../devices/ldd0/scculld3
├── drivers
│   └── sculld
│   ├── bind
│   ├── scculld0 -> ../../../../devices/ldd0/scculld0
│   ├── scculld1 -> ../../../../devices/ldd0/scculld1
│   ├── scculld2 -> ../../../../devices/ldd0/scculld2
│   ├── scculld3 -> ../../../../devices/ldd0/scculld3
│   ├── unbind
│   └── version
├── drivers_autoprobe
├── drivers_probe
└── version
/sys/module/scculld/
├── holders
├── initstate
├── parameters
│   ├── scull_major
│   ├── scull_minor
│   ├── scull_nr_devs
│   ├── scull_qset
│   └── scull_quantum
├── refcnt
└── sections
    ├── __ex_table
    └── __param

38 directories, 33 files
[Tekkaman2440@SBC2440V4]#cat /sys/bus/ldd/version /sys/devices/ldd0/scculld*/dev
/sys/bus/ldd/drivers/sculld/version
Revision: 1.9-tekkamanninja
252:0
252:1
252:2
252:3
$Revision: 1.21-tekkamanninja $
[Tekkaman2440@SBC2440V4]#rmmod scculld
[Tekkaman2440@SBC2440V4]#tree -ACd /sys/module/lddbus/ /sys/devices/ldd0/
/sys/bus/ldd/ /sys/module/
/sys/module/lddbus/
├── holders
└── sections
/sys/devices/ldd0/
└── power
/sys/bus/ldd/
├── devices
└── drivers
/sys/module/
├── 8250
│   └── parameters
```

```
├── atkbd
│   └── drivers
│   └── serio:atkbd -> ../../../bus/serio/drivers/atkbd
├── cdrom
├── dm9000
│   └── parameters
├── hid
│   └── parameters
├── ide_cd
│   └── parameters
├── keyboard
│   └── parameters
├── lddbus
│   ├── holders
│   └── sections
├── lockd
│   └── parameters
├── loop
├── mac80211
│   └── parameters
├── mousedev
│   └── parameters
├── nfs
│   └── parameters
├── ohci_hcd
├── printk
│   └── parameters
├── psmouse
│   ├── drivers
│   │   └── serio:psmouse -> ../../../bus/serio/drivers/psmouse
│   └── parameters
├── rcupdate
├── rd
├── redboot
├── rfd_ftl
├── rtc_ds1307
├── s3c2410_wdt
├── snd
│   └── parameters
├── snd_pcm
│   └── parameters
├── snd_pcm_oss
│   └── parameters
├── snd_seq
│   └── parameters
├── snd_seq_oss
│   └── parameters
├── snd_soc_core
├── snd_timer
│   └── parameters
├── spidev
│   └── parameters
├── sunrpc
│   └── parameters
├── tcp_cubic
│   └── parameters
├── usbcore
│   ├── drivers
│   │   ├── usb:hub -> ../../../bus/usb/drivers/hub
│   │   └── usb:usbfs -> ../../../bus/usb/drivers/usbfs
```

```
|   └── parameters
├── usbnet
├── v4l1_compat
|   └── parameters
├── vt
|   └── parameters
├── yaffs
|   └── parameters
└── zc0301
    ├── drivers
    |   └── usb:zc0301 -> ../../../bus/usb/drivers/zc0301
    └── parameters

79 directories
[Tekkaman2440@SBC2440V4]#insmod /lib/modules/sculld.ko
[Tekkaman2440@SBC2440V4]#rmmod lddbus
rmmod: lddbus: Resource temporarily unavailable
[Tekkaman2440@SBC2440V4]#rmmod sculld
[Tekkaman2440@SBC2440V4]#rmmod lddbus
The LDD bus device
 ldd_bus_release : lddbus
[Tekkaman2440@SBC2440V4]#
```

在系统中添加“tree”命令

实验中用到了Linux的常用命令“tree”，若使用busybox可能没有这个命令。如出现这种情况，可以下载此命令的源码，交叉编译一下，再放到根文件系统中的/bin目录中就好。在 CalmArrow 的博客中可以下载：http://blog.chinaunix.net/u/21948/showart_297101.html

（因为源码的下载地址 ftp://mama.indstate.edu/linux/tree/我一直进不去，在这里谢谢CalmArrow）

上一篇：Linux设备驱动程序学习（13）-Linux设备模型（总线、设备、驱动程序和类）
下一篇：Linux设备驱动学习15-设备模型(热插拔,mdev,firmware)

0