文章编号:1001 - 9081(2005)01 - 0076 - 02

Linux 2.6 内核的内核对象机制分析

丁晓波,桑 楠,张 宁
(电子科技大学 计算机科学与工程学院,四川 成都 610054)
(dxb1995@sina.com)

摘 要:文中介绍了 Linux 2.6 内核中管理设备及其驱动程序的内核对象机制,重点分析了该机制的主要数据结构、工作原理和操作函数。提出了基于嵌入应用时简化目录结构的方法。

关键词:Linux 2.6 内核;设备管理;内核对象机制;Kobject
中图分类号: TP311 **文献标识码**:A

Analysis of kernel object of Linux 2.6

DING Xiao-bo, SANG Nan, ZHANG Ning
(College of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu Sichuan 610054, China)

Abstract: The new mechanism of Linux kernel v2.6 - kernel object was analyzed, including the structure, principle and functions of it. A method was provided to simplify the structure of directories.

Key words: Linux kernel v2.6; devices management; mechanism of kernel object; Kobject

0 引言

经过 Linux 开发团队的不懈努力,Linux 内核终于在 2003 年 8 月发布了新的测试版 v2.6。与原来 2.4 内核相比,新内核在很多方面进行了较大改进,主要有:支持更多种类的 CPU、线程级的抢占式内核、更大的内存空间、超线程支持、更可靠的模块子系统、内核对象抽象技术等等。该版本发行近一年来,其性能上的改进和提高,已经得到了 Linux 爱好者们的广泛认同和一致赞赏。这些新的技术,使 Linux 变得更为可靠而稳定,具备了更好的适应性和可用性。

Linux 2.6 内核采用了抢占式的调度机制,使其实时性能得到了大幅的提升,因此在嵌入式应用领域已经受到前所未有的关注。由于嵌入式应用首先需要解决硬件驱动问题,因此掌握 2.6 内核的设备管理和驱动技术是做好内核配置和嵌入式应用开发的首要任务。

本文针对 Linux 2.6 内核中新的设备管理机制 —— 内核对象机制进行了分析和研究,阐述了内核对象机制的基本原理和应用方法,为深入理解和进一步研究 Linux 2.6 内核的设备和驱动程序管理提供了十分必要的技术支持。

1 内核对象机制

内核对象机制是 Linux 2.6 内核引入的新的设备管理机制 ——一种底层数据结构。通过这个数据结构,可以使所有设备在底层都具有一个公共接口,便于设备或驱动程序的管理和组织。具体来说,是将一个数据结构 Struct Kobject 作为一种公共连接部件定义到更高层的其他数据结构中去,而各个高层数据结构之间的关系通过 Kobject 结构以不同的链表方式表示,从而形成结构紧密的上下层次关联。通过这种 Kobject 结构,使得总线、设备、设备上的接口等相互间的关系变得更有层次,也更容易管理和组织。同时也使得加载或卸载某个设备的驱动程序更加方便。

下面从内核对象涉及的关键数据结构、数据结构相互关系两个方面来阐述内核对象机制原理。

1.1 内核对象机制的关键数据结构

2.6.0 内核中的内核对象机制有三个主要的数据结构。

1.1.1 Struct Subsystem 内核对象子系统

```
struct subsystem {
  struct ksetkset;                          该子系统的内核对象集合
  Struct rw_semaphorerwsem;                 互斥访问信号量
};
```

它是内核对象机制最上层的数据结构,用于描述一类设备子系统。所有属于一类的设备,将通过描述设备的数据结构中的 Kobject 成员挂接到其所属类的 subsys.kset.list 链中。

1.1.2 Struct Kset 内核对象集合

```
struct kset {
  struct subsystem * subsys;                所在的 subsystem 的指针
  struct kobj_type * ktype;                 所属的对象类型
  struct list_head list;                    属于该集合的对象链表头
  struct Kobject kobj;                      本集合自身相关信息的内核对象
  struct kset_hotplug_ops * hotplug_ops;    该集合的热插拔函数
};
```

该结构用于描述一类内核对象的集合,它可以是 subsystem 结构的一个成员,也可以单独定义。其成员包括:设备对象链的链头指针 list,设备的类型描述结构指针 ktype,此类设备热插拔的公共操作的集合指针 hotplug_ops。该结构中的 Kobject 用于描述该 kset 自身的节点路径、类型、引用计数等参数。

1.1.3 Struct Kobject 结构

```
struct kobject {
  char name[KOBJ_NAME_LEN];                 设备名称
```

收稿日期:2004 - 06 - 16;**修订日期**:2004 - 11 - 29 **基金项目**:国家 863 计划资助项目(2003AA1Z2210)

作者简介:丁晓波(1974 -),男,硕士研究生,主要研究方向:嵌入式 Linux 技术; 桑楠(1964 -),男,副教授,主要研究方向:嵌入式实时操作系统; 张宁(1975 -),男,博士研究生,主要研究方向:嵌入式实时操作系统.

```
atomic_t refcount;                    引用计数
struct list_head entry;               挂接到所在集合中去的单元
struct kobject * parent;              上级对象的指针
struct kset * kset;                   所属的对象集的指针
struct kobj_type * ktype;             所属的对象类型指针
Struct dentry * dentry;               文件节点在 sysfs 系统中的路径指针
};
```

这是内核对象机制中最底层的数据结构,包含该结构的上层数据结构间的相互依赖和所属关系,都通过该结构的连接关系来描述。也就是说包含该结构的上层数据的注册等操作均通过对这个数据结构的注册操作来完成。

1.2 内核对象机制数据结构的相互关系

内核对象机制中数据结构相互关系,描述了设备管理的整个体系架构,体现了设备或驱动程序注册过程在内核对象机制中最终完成的任务。

内核对象机制,实际上是内存中的一种数据结构相互关系。这种数据结构相互关系,首先是从系统初始化时开始的,在系统初始化时由 driver_init( ) (drivers/base/init. c 中) 函数,将几个代表不同子系统的 subsystem 数据结构注册到内核,注册的主要工作是初始化子系统数据结构,并在 sys 文件系统下建立各自子系统的目录路径。此时注册的子系统有:设备子系统 devices_subsys,总线子系统 bus_subsys,设备基类子系统 class_subsys,固件子系统 firmware_subsys,在设备子系统下又定义了系统虚拟总线子系统 system_subsys。

这五个子系统就为后面设备注册搭起了框架,设备初始化时,初始化程序通过调用内核对象注册函数,将设备数据结构中的成员 Kobject 结构注册到相应子系统中。注册要完成的任务:一是将 Kobject. entry 加入到 Kobject. kset. list 链中,二是在 Sysfs 系统中,Kobject. parent 指向的对象所在目录下创建名字为 Kobject. name 的文件节点。

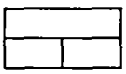


图 1　Kobject 结构

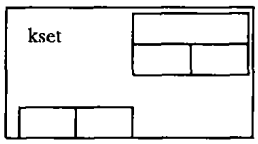


图 2　kset 结构

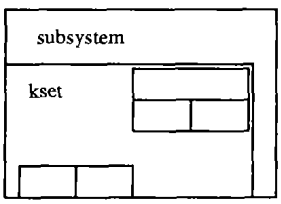


图 3　subsystem 结构

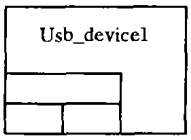
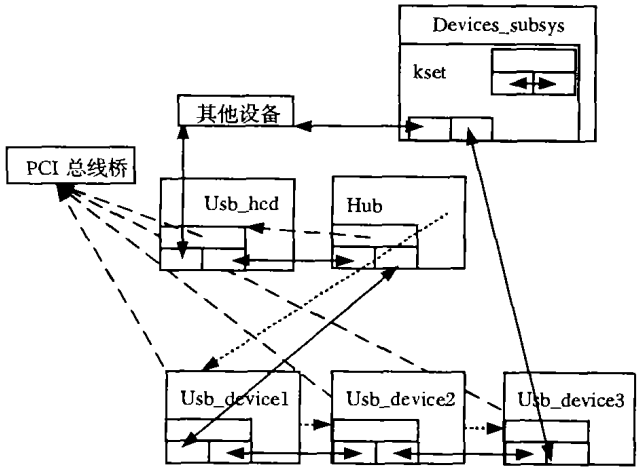


图 4　USB_device 结构

下面以 USB 总线及其设备在内核中的层次结构为例,介绍内核对象的作用和相互关系。首先将 Kobject, kset, subsystem 表示为图 1 ~ 图 3。其中,kset 嵌入的 Kobject 成员,既描述 kset 自身属性,又为相互链接提供了连接单元。当 kset 属于某个 subsystem 结构的成员时,这个内嵌于 kset 的 Kobject 实际上就是对 subsystem 属性的描述,并为 subsystem 提供连接单元。

USB 设备描述为 USB_device,每个 USB_device 中都嵌入了一个 Kobject 结构,如图 4 所示。

USB 总线控制器定义为 USB_hcd,根集线器定义为 Hub。则整个 USB 总线上设备相互关系,在 Kobj 层上的描述为:



说明:
- ····► 表示设备之间的上下关系
- – – ► 表示Kobject描述的父子关系，也就是在Sysfs中反映出来的目录或代表设备的节点之间的层次关系
- ◄——► 表示在一类设备Kset中设备间的链接指针

图 5　USB 设备相互关系

图 5 描述了在 USB 总线中所有设备的都属于 Devices_subsystem,而所有根 Hub 上接入的 USB 设备,虽然在设备层都属于根 Hub 的设备链,但它们的 Kobject 指针都与 USB 总线控制器的父指针指向同一设备:PCI 桥。这样在进行 USB 设备的 Kobject 注册时,将会在 PCI 桥所在目录下创建与设备相关的节点,而不是在某个 Hub 所在的目录下。这里 Kobject 将实际的设备依赖关系通过父指针的方式屏蔽了,在体现设备层次结构的同时也减少了目录结构的复杂性。

2　内核对象机制主要函数功能和使用方法

针对内核对象机制不同层次的数据结构,Linux2. 6 内核定义了各数据结构的操作函数,这些函数均定义于 Lib/Kobject. c 文件中。

2.1 内核对象机制中的主要相关函数

2.1.1 subsystem 相关函数

void subsystem_init(Struct subsystem *);

subsystem 初始化函数,完成互斥信号量、调用 kset 初始化函数初始化 ksetd 的相关值

int subsystem_register(Struct subsystem *);

subsystem 注册函数,该函数通过调用 kset 的注册函数,向其上层 subsystem 注册

void subsystem_unregister(Struct subsystem *);

撤销注册函数,功能与注册函数相反

2.1.2 kset 相关函数

void kset_init(Struct kset * k);

kset 结构初始化函数,该函数只完成 list 表头初始化和 kset 内嵌 Kobject 结构的初始化

int kset_add(Struct kset * k);

kset_register 函数的辅助函数,主要完成 kset 内嵌 Kobject 结构注册

(下转第 84 页)

语义关系。

4）将确定的对象信息重构为符合既定 Schema 形式的 XML 文档。

转换后的 XML 文档如下：

```
<?xml version = "1.0"? >
<weatherreport >
    <temperature >
        <date >2004-4-15 </date >
        <topmost >28℃ </topmost >
        <lowermost >23℃ </lowermost >
        <wind power >3 </wind power >
    </temperature >
    <city >xi'an </city >
    <nation >china </nation >
</weatherreport >
```

4 结语

异构数据源的集成问题包括数据模型的转换，查询翻译。HISQ 针对关系型数据库，XML 和 HTML 数据源采用不同的转换机制，并使用基于中间模式的用户查询，数据仍保存在局部的数据源中，简化了集成方法，提高了查询的精度。下一步的工作将在查询的优化和查询响应速度方面开展。

参考文献：

[1] SEVILLA D, GARCIA JM, GóMEZ A. CORBA Lightweight Components: A Model for Distributed Component-BasedHeterogeneous Computation[A]. Euro-Par 2001[C], 2001. 845 - 854.

[2] XU XZ, *et al*. Knowledge-based intelligent query processing system [A]. ICYCS'2001[C], 2001. 1048 - 1051.

[3] ASLAN G, MCLEOD D. Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution[J]. The VLDB Journal, 1999, (8): 120 - 132.

[4] KNOBLOCK GA, MINTON S, *et al*. The Ariadne Approach To Web-based Information Integration[J]. International Journal of Cooperative Information Systems, 2001, 10(1 - 2): 145 - 169.

[5] ASHISH N, KNOBLOCK C. Semi-automaticWrapper Generation for Internet Information Sources[EB/OL]. http://www.isi.edu/info-agents/papers/ashish97-coopis.pdf.

[6] JACKSON J, MYLLYMAKI J. Web-based data mining[EB/OL]. http://www-900.ibm.com/developerWorks/cn/xml/x-wbdm/index_eng.shtml.

[7] COLLINS SR, NAVATHE SB, MARK L. XML Schema Mappings for Heterogeneous Database Access[J]. Information & Software Technology, 2002.44(4): 251 - 257.

[8] CHUNG TS, KIM HJ. Techniques for the evaluation of XML queries: a survey[J]. Data & Knowledge Engineering, 2003, 46(2): 225 - 246.

（上接第 77 页）

int kset_register(Struct kset * k);

kset 注册函数，通过调用 kset_init 和 kset_add 完成向子系统链接和相应目录下文件节点的创建

void kset_unregister(Struct kset * k);

kset 撤销注册函数，功能与注册函数相反

2.1.3 Kobject 相关函数

void Kobject_init(Struct Kobject *);

Kobject 初始化函数，它是上层结构内核对象初始化的基础，完成对引用计数置 1，链接单元 entry 指向自身，上层 kset 引用计数加 1

void Kobject_cleanup(Struct Kobject *);

Kobject 结构清除函数，它将通过其所属类型的 release 函数释放它所占用的空间（当对象引用计数为 0 时）

int Kobject_add(Struct Kobject *);

Kobject 注册辅助函数，完成任务包括：挂入上层 kset 结构的 list 链中，修改父目录各级 Kobject 的引用计数，在其 parent 指向的目录中创建文件节点，启动该类型内核对象的 hotplug 函数

void Kobject_del(Struct Kobject *);

Kobject 撤销注册辅助函数，完成任务包括对该类型内核对象 hotplug 函数的终止，从上层 kset 链中删除该对象，修改上层各级引用计数，删除所属文件节点

int Kobject_register(Struct Kobject *);

内核对象注册函数，通过调用 Kobject_init 和 Kobject_add 完成对 Kobject 的初始化和注册工作

void Kobject_unregister(Struct Kobject *);

内核对象撤销注册函数，通过调用 Kobject_del 和 Kobject_put 完成对 Kobject 从内核的撤销工作

Struct Kobject * Kobject_get(Struct Kobject *);

本函数将返回调用参数的指针，但同时对调用参数引用计数加 1

void Kobject_put(Struct Kobject *);

本函数将调用参数的引用计数减 1，然后调用 Kobject_cleanup 释放该 Kobject 对象（当引用计数为 0 时）

2.2 内核对象机制的应用和简化

在嵌入式应用，由于设置的数量和种类都相对较少，内核对象机制提供的树形管理结构，显得过于复杂。在实际使用中应加以简化，通过以上分析可以发现，在设备注册时通过改变设备数据结构所包含的 Kobject 结构的父结构指针，就可以将系统创建的设备文件安排成一个平面形结构，达到简化目录层次的目的。实际实现时，在设备或驱动程序注册程序中，通过递归搜索上级结构中 Kobject 的父指针的方法来找到最上层的设备文件目录，然后在这个目录下创建待注册的设备文件。

3 结语

本文通过分析大量 Linux2.6 内核代码，详细论述了 2.6 内核在设备和驱动程序管理中的新机制——内核对象机制的数据结构、工作原理、设计思想等相关问题，并针对嵌入式应用环境提出了一种简化设备目录结构的方法。为学习、理解和设计 Linux 2.6 内核下的设备驱动程序提供了有益的参考。

参考文献：

[1] Examing a Kobject hierarchy[EB/OL]. http://lwn.net/Articles/55847/, 2003.

[2] MOCHEL P. The Kobject InfraStructure[EB/OL]. http://cvs.sourceforge.net/viewcvs.py/linux-vax/kernel-2.5/Documentation/kobject.txt?rev=1.1.1.3, 2003 - 1 - 7.

[3] PRANEICHJ. Linux 2.6 内核的精彩世界[EB/OL]. 穆荣均，等. http://www-900.cn.ibm.com/developerWorks/cn/linux/kernel/l-kernel26/index.shtml, 2003 - 9.

[4] 毛德操，胡希明. Linux 内核源代码情景分析（下）[M]. 杭州：浙江大学出版社，2001.

[5] 郭学理，潘松，韦智. Linux 网络驱动程序分析[J]. 计算机应用，2001，21(11)：23 - 24.