第xxx章

# 第14章 Linux设备模型

2.5开发周期的目标之一是为内核创建统一的设备模型。以前的内核没有单一的数据结构，用来获取有关系统如何把设备放在一起的信息。尽管缺乏这方面的信息，但一段时间以来，内核仍然运行良好。但是，具有更复杂拓扑结构，并需要支持电源管理等功能的新系统，为了使其更清晰简洁，就需要一种系统结构的抽象描述。

2.6内核设备模型提供了这种抽象描述。在内核中，被用来支持各种任务，如下：

* 电源管理和系统关机  
  这需要了解系统的结构。例如，在处理连接到USB主机适配器的所有设备之前，适配器无法关闭。设备模型确保能够以正确的顺序遍历系统的硬件。
* 和用户空间进行通信  
  sysfs虚拟文件系统的实现与设备模型紧密相关，并揭示了它所代表的结构。将系统信息提供给用户空间和用于改变操作参数的“旋钮”越来越多地通过sysfs完成，因此也是通过设备模型完成。
* 可热插拔设备  
  计算机硬件越来越多的是动态设备，用户可以随意插拔外设。内核里用来处理热插拔设备，尤其是和其通信的hotplug机制，是通过设备模型管理的。
* 设备类  
  系统的许多部分对如何连接设备几乎没有兴趣，但他们需要知道哪些设备可用。设备模型包括一个用于将设备分配给类的机制，这些类可以在更高的功能级别上描述这些设备，并允许从用户空间中发现它们。
* 对象生命周期  
  上面描述的许多功能，包括热插拔支持和sysfs，都会使内核中对象的创建和操作复杂化。设备模型的实现需要创建一组处理对象生命周期的机制，它们之间的关系以及它们在用户空间中的表示。

Linux设备模型是一个复杂的数据结构。例如，如图14-1所示，该图简化显示了有关USB鼠标的设备模型结构的一小部分。沿着图的中心，可以看到，“设备”树展示了鼠标是怎样连接到系统的。“总线”树跟踪每条总线的连接情况，而“类”的子树则关心设备提供的功能，而不管他们是如何连接的。即使在一个简单的系统，设备树模型也包含上百个上述这样的节点，所以这是一个非常难以整体可视化的数据结构。

图14-1，设备模型的一小部分

对于大多数情况下，Linux设备模型考虑了所有应该考虑的事情，而不是把这些交给驱动程序的开发者去完成。它在后台运行，与设备模型的直接交互也是通过总线或者其它内核子模块完成。结果，许多驱动程序的开发者能够忽略设备模型，并相信它会自行处理。

但是，对设备模型的理解总是一件好事情的。有时候设备模型会从其它层“泄漏出去”；例如，使用struct device的通用DMA代码（我们在第15章中遇到）。您可能需要使用设备模型提供的某些功能，例如kobjects提供的引用计数和相关功能。通过sysfs与用户空间进行通信也是一种设备模型功能；本章解释了该通讯的工作原理。

我们对设备模型实行自下而上演示。设备模型的复杂性使得从顶层视图理解变得非常困难。我们希望，通过展示底层设备组件的工作方式，我们可以帮助您准备应对这些组件如何用于构建更大结构的挑战。

对于很多读者来说，本章可以被看作是第一次不需要阅读的高级资料。但是，我们鼓励那些对Linux设备模型如何工作感兴趣的同学们，随我们一起进入更低层次的细节。

## 14.1 Kobjects，Ksets和Subsystems

kobject是将设备模型整合在一起的基本结构。它最初被设想为一个简单的引用计数器，但它的职责随着时间的推移而增长，其内容也是。现在由struct kobject及其支持代码处理的任务包括：

* 对象的引用计数  
  通常，在创建内核对象时，无法知道它将存在多长时间。跟踪这些对象生命周期的一种方法是通过引用计数。当内核中没有代码持有给定对象的引用时，该对象已经完成了它的使用寿命并且可以被删除。
* Sysfs表示  
  每个在sysfs中显示的对象都有一个kobject，它与内核交互以创建其可见的表示形式。
* 数据结构胶水  
  整个设备模型是一个非常复杂的数据结构，由多个层次组成，并且它们之间有很多链接。 kobject实现了这个结构并且把它保存在一起。
* 热插拔事件处理  
  kobject子系统负责有关系统硬件插拔时，通知用户空间的事件的产生。

从前面的内容中可以看出：kobject是一个复杂的结构。但是，我们可以通过每次一小块的方式，尽可能地理解这种数据结构和它的工作原理。

### 14.1.1 Kobject基础知识

一个kobject对象的类型是struct kobject；它在<linux/kobject.h>中定义。该文件还包含许多与kobjects相关的其它结构的声明，当然还包括一系列用于操作它们的函数。

**1. 嵌入式内核对象Kobject**

在详细讨论细节之前，花点时间了解如何使用Kobjects。如果回顾一下由kobjects处理的函数列表，您会发现它们都是代表其他对象执行的服务。换句话说，kobject对象对它本身并不感兴趣。它只存在于将更高级别的对象绑定到设备模型中。

因此，内核代码很少（甚至不知道）创建独立的kobject;相反，kobject对象被用来控制对更大的，特定领域对象的访问。为此，kobject对象被发现总是嵌入在其它结构中。如果您习惯于以面向对象的方式来思考事物，那么kobjects可以被看作是其它类派生的顶级抽象类。 kobject实现了一组本身并不特别有用的功能，但在其它对象中很有用。C语言不允许直接表示继承，所以必须使用其它技术 - 例如将一个结构嵌入另一个结构中。

作为一个例子，让我们回头看看在第3章中遇到的struct cdev。

这个结构，在2.6.10内核中，是这样的：