华清远见
FARSIGHT®
始于2004
嵌入式培训专家

linux设备驱动开发概述

潘友华, 成都中心

专业始于专注　　卓识源于远见

版权

内容提纲

- 1、为什么要学习驱动开发
- 2、驱动开发涉及内容
- 3、内核面向对象编程
- 4、驱动框架
- 5、内核技术
- 6、调试技术
- 7、可移植性

为什么要学习驱动开发

- 学习linux设备驱动设计有诸多的理由，简述如下：
  - 1、现在硬件更新很快，linux系统中需要有设备驱动来控制这些设备。
  - 2、基于linux的嵌入式系统设计实现中，平台搭建需要移植内核。在移植内核时会发现，主要的工作是添加设备驱动及相关的内核逻辑模块。
  - 3、计算机是怎么工作的，设备是怎么工作的，这样的疑问，会因为熟悉设备驱动而不再是个问题。
  - 4、应用程序和内核的关系好比政府和百姓，百姓（应用程序）处理不了的事情，会交给政府（内核）去完成。所以熟悉设备驱动有助于应用程序的设计。
  - 5、linux是一款非常优秀的系统，其设计实现使用的很多优秀的设计方法，遵循了很多优秀的设计思想。通过设备驱动设计去学习、理解、实践这些优秀的设计，是一种有效的手段。

驱动开发涉及内容

- 1、硬件知识
  - 1.1、软件如何控制硬件
  - 主要涉及体系结构及接口编程。学习重点是掌握软件如何控制硬件涉及的一些基本知识：具备体系结构的一些基础知识（硬件平台的基本特性、指令系统、异常中断处理及接口编程），看懂原理图、芯片手册。
  - 1.2、分工
  - 把控制硬件的过程拆分为：打开设备、关闭设备、读设备、写设备等等。
- 2、内核编程技术
  - 2.1、框架
  - 这里所谓的框架就是组织代码的一种设计形式，内核关于设备管理控制已经设计出具体的一些框架，我们只需要学习这些框架，就可以完成对设备的管理控制。
  - A、框架形式
    - 模块：三个基本组成部分。
    - 初始化函数：构造并初始化对象、注册对象。
    - 卸载函数：回收资源、注销对象。
  - B、学习重点：内核结构（设备对象类）、内核函数。
  - 2.2、技术
  - 涉及内容：阻塞、并发控制、异步通知、延迟执行、分配内存、中断等等。

内核面向对象编程

- 1、面向对象编程（Object Oriented Programming，OOP）
  - OOP 是一种计算机编程架构。OOP 的一条基本原则是计算机程序是由单个能够起到子程序作用的单元或对象组合而成。
  - OOP主要目标：重用性、灵活性和扩展性。

内核面向对象编程

- 2、内核编程涉及的C面向对象设计基础
  - A、C面向对象设计
    - OOP仅仅是一种设计方法，讨论的是代码组织的一种形式。C语言本身语法不支持面向对象这样的特征，所以需要程序员自己来设计实现。内核主要使用C编写出来的，又使用了面向对象技术。
  - B、回调
    - 函数通过函数指针来调用，这样的间接调用就是回调。
  - C、封装
    - 使用struct把一组函数及数据局部化，封装为一个整体。
  - D、注册（对象管理）
    - 由于内核是分层设计的，设备驱动一般在最底层，驱动相当于内核里面的库代码，总是被动的被上层调用。如何让内核知道驱动对象，就有一个注册的概念。

驱动框架

- 1、接口分类
  - A、字符设备驱动
    - 字符设备是指在I/O传输过程中以字符为单位进行传输的设备，例如键盘，打印机等。请注意，以字符为单位并不一定意味着是以字节为单位，因为有的编码规则规定，1个字符占16比特，合2个字节。控制字符设备的驱动就是字符设备驱动。
  - B、块设备驱动
    - 块设备将信息存储在固定大小的块中，每个块都有自己的地址。数据块的大小通常在512字节到32768字节之间。块设备的基本特征是每个块都能独立于其它块而读写。常见设备有：硬盘、光盘、u盘、flash等等。控制块设备的驱动就是块设备驱动。
  - C、网络设备驱动
    - 网卡是一种特殊的设备，用于网络通讯。负责控制网卡的驱动就是网络设备驱动。
  - 备注：
    - 字符设备和块设备都属于IO设备，所以没有明显的界限。应该根据设备的实际特征来决定采用字符接口还是块接口。

驱动框架

- 2、分层设计
  - A、优点
    - 分散关注：分工，各司其职
    - 松散耦合：各层间依赖性降低，仅仅接口交互
    - 逻辑复用：相同功能的层次代码可以直接复用
    - 标准定义：各层接口标准化
  - B、内核一般情况下的分层
    - 接口层：字符、块及网络框架
    - 核心层：衔接接口及设备层，管理设备对象等等
    - 设备层：控制设备

驱动框架

- 3、主机与设备分离思想
  - A、优点：
    - 一个设备接入主机前，预先并不知道IO地址及中断号，无法在驱动中直接写死这些信息，那么驱动编写就存在困难。譬如热插拔设备，其驱动就得先编写好，但是又不能写死硬件信息。推而广之，一些逻辑信息也可以这样来设计。
  - B、涉及概念：
    - 设备
      - 提供设备信息：IO地址、中断号、设备参数及逻辑信息
    - 驱动
      - 提供使用设备信息的代码
    - 总线
      - 管理设备对象列表及驱动对象列表，负责两者之间的关联

驱动框架

- 4、学习重点：
  - A、内核结构
    - 主要是表征一个对象，面向对象设计中，类就有public和private的概念，使用类就只需要知道public就行了。
  - B、内核函数
    - 功能（干什么）
    - 参数（怎么用）
    - 返回值（执行结果）

内核技术

- 这里所谓的内核技术，是想概述下设备驱动设计中除框架外涉及控制硬件操作的一些工作方式或是硬件相关的技术。主要的归类如下：
- 1、阻塞
  - 一般的，设备在进行I/O的时候会存在阻塞和非阻塞两种方式。阻塞就是指在执行I/O操作时，如果不能获得资源，则挂起进程，直到能够获得资源时才唤醒进程。
- 2、并发控制
  - Linux是并发系统，所以对共享资源访问会存在竞态，为了实现同步互斥，linux内核设计了很多并发控制技术：中断屏蔽、原子操作、自旋锁、信号量、互斥体等等。
- 3、异步通知
  - 异步通知就是内核通过信号来主动通知的一种内核异步通讯机制。
- 4、延迟执行
  - 有些任务可能存在要在延迟一定时间后，来执行特定的任务。

内核技术

- 5、分配内存
  - 内核中涉及内存分配释放的技术。
- 6、中断
  - 一般的情况下，硬件事件的发生系统并不知道，如果让CPU不断查询的话，系统性能就会很差。中断就是让硬件在需要CPU处理的时候主动发出信号，从而实现异步通讯。
- 7、内存管理
  - linux是虚拟内存系统，涉及内存分布、虚实地址转换、内存池、内存映射等等。
- 8、IO
  - 如何与硬件通讯，内核也提供了管理操作IO地址的函数。
- 9、DMA
  - DMA(Direct Memory Access，直接内存存取)，DMA 传输将数据从一个地址空间复制到另外一个地址空间。当CPU 初始化这个传输动作，传输动作本身是由 DMA 控制器来实行和完成。

调试技术

- 内核调试难度还是比较大的，但是有一些简单有效的调试手段，一般情况下，可以帮助我们解决很多问题。
  - 1、打印跟踪
  - 2、执行验证
  - 3、Oops
  - 4、proc及sysfs接口
  - 5、调试工具

可移植性

- 设备驱动主要是管理控制外围设备的。一般的，为访问特定设备专门编写的程序通常是不具有可移植性，也就是说，要使代码具有可移植性，应该极力规避与具体硬件相关的代码出现。所以这里讨论的可移植性设计，跟应用层还是有很大的区别。大概需要注意的有如下几点：
  - 1、内核函数及宏
  - 2、字节对齐
  - 3、大小端
  - 4、规范编程