Linux 设备驱动的开发---开发客户定制的嵌入式Linux驱动

John Madieu

1: 内核开发介绍

。第一章 1： 内核开发介绍

。环境搭建

。内核惯例

。总结

2：设备驱动基础

。 第二章： 设备驱动基础

。 用户空间和内核空间

。 驱动架构

。错误和消息打印

。模块参数

。构建第一个模块

总结

3：内核基础设施和有用的函数

。第三章： 内核基础设施和有用的函数

。理解 container\_of 宏

。链表

。内核休眠机制

。延迟和定时器管理

。内核锁定机制

。工作延迟机制

。线程化的中断

。从内核调用用户空间程序

总结

4：字符设备驱动

。第四章：字符设备驱动

。在主要和次要后面的概念

。介绍设备文件操作

。分配和登记一个字符设备

。写文件操作

总结

5：平台设备驱动

。第五章：平台设备驱动

。平台驱动

。平台设备

。设备、驱动和总线匹配

总结

6：设备树的概念

。第六章：设备树的概念

。设备树机制

。表示和寻址设备

。资源处理

。平台驱动和DT

总结

7：I2C 客户端驱动

。第七章：I2C 客户端驱动

。驱动架构

。I2C 和设备树

总结

8：SPI 设备驱动

。第八章：SPI设备驱动

。驱动架构

。对客户端的访问和通信

。放到一起来说

。SPI 用户模式驱动

总结

9：Regmap API 接口： 寄存器的抽象映射

。第九章：Regmap API 接口： 寄存器的抽象映射

。使用regmap API 编程

。总结

10：IIO 框架

。第十章 IIO 框架

。IIO 数据结构

。触发缓存支持

。IIO 数据访问

。总结

11： 内核内存管理

。第十一章 内核内存管理

。系统内存布局-内核地址空间和用户地址空间

。地址转译和MMU

。内存分配机制

。用 I/O 内存来访问硬件

。内存（再）映射

。Linux 缓存系统

。受管理设备资源：Devres

总结

12: DMA: 直接内存访问

。第十二章：DMA: 直接内存访问

。建立DMA 映射

。理解‘completion’的含义

。放到一起来说：NXP SDMA（i.MX6）

。总结

13：Linux 设备模型(LDM)

。第十三章：Linux 设备模型(LDM)

。LDM 数据结构

。深入 LDM

。设备模型和 sysfs

。总结

14：PIN 脚控制 和 GPIO 子系统

。第十四章：PIN 脚控制 和 GPIO 子系统

。PIN 脚控制子系统

。GPIO 子系统

。总结

15：GPIO 控制器驱动：gpio\_chip

。第十五章：GPIO 控制器驱动：gpio\_chip

。驱动架构和数据结构

。PIN脚控制器指南

。GPIO 控制器的 Sysfs 接口

。GPIO 控制器 和 DT

。总结

16：高级中断管理

。第十六章：高级中断管理

。中断复用和中断控制器

。高级外设中断管理

。中断请求和 传播

17：Input 设备驱动

。第十七章：Input 设备驱动

。Input设备结构

。分配和登记一个Input设备

。产生和报告一个 Input 事件

。用户空间接口

。放到一起来说

。总结

18：RTC 驱动

。第十八章：RTC 驱动

。RTC 架构数据结构

。RTC 和 用户空间

。总结

19：PWM 驱动

。第十九章：PWM 驱动

。PWM 控制器驱动

。PWM 消费者接口

。在 Sysfs 接口中用 PWM

。总结

20：稳定器框架

。第二十章：稳定器框架

。PMIC/产生器 驱动接口

。稳定器消费者接口

。稳定器绑定

。总结

21：帧缓存驱动

。第二十一章：帧缓存驱动

。驱动数据结构

。设备各方法

。驱动各方法

。用户空间访问帧缓存

。总结

22：网卡驱动

。第二十二章：网卡驱动

。驱动数据结构

。设备各方法

。驱动各方法

。总结

1. . 内核开发介绍

Linux 始于一个叫Linus Torvalds的芬兰学生在1991年的个人爱好项目。这个项目逐渐成长，到现在已经在全世界有大约1000个贡献者。现在 Linux 成了嵌入式系统和服务器的必选。内核是操作系统的核心部分，其开发不是那么显而易见。

Linux 有相对于其他操作系统的很多优点：

.免费

.文档丰富且有一个大的社区

.可以在不同平台上移植

.源码可访问

.大量的免费的开源软件

本书想尽量通用。 有个特别的主题，设备树，还没有完全应用到X86架构。这个主题就专门针对 ARM 处理器和所有完全支持设备树的处理器。为什么选择这些架构？因为它们是在桌面机和服务器（x86）和嵌入式系统（ARM）里用的最多的。

本章也描述其它的：

.开发环境建立

.获得，配置，编译内核源码

.内核源码组织

.介绍内核编程风格

环境建立

在想开始开发前，需要建立一个环境。 至少在基于 Debian 的系统里针对Linux开发的环境是特别简单的：

$ sudo apt-get update$ sudo apt-get install gawk wget git diffstat unzip texinfo \gcc-multilib build-essential chrpath socat libsdl1.2-dev \xterm ncurses-dev lzop

本书的部分代码是同ARM 片上系统（SoC）兼容 。所以也需要安装 gcc-arm：

sudo apt-get install gcc-arm-linux-gnueabihf

我在一台 ASUS RoG 上跑 Ubuntu 16.04,配置是 Intel i7核心（8核），16GB 内存，256G SSD硬盘，以及 1TB 的普通硬盘。我喜欢的编辑器是 Vim，但是也可以选用自己觉得最顺手的编辑器。

获得源码

在内核早期岁月（直到 2003年），使用奇数-偶数版本号；奇数版本号是稳定版本，偶数版本号是不稳定版本。2.6 版本发布以来，版本号方案转为 X.Y.Z:

.X: 实际内核版本号，也叫主版本号，当有后向不兼容的API接口改变就会加大此版本号

.Y: 这个是次版本号，它...

内核惯例

内核代码在进化的过程中一直想遵循标准规则。在本章我们会介绍一下。这些规则也会在后续一些章节里讨论，从第三章*内核基础设施和有用的函数* 到 第十三章 *Linux 设备模型* 会对内核开发过程和诀窍有一个很好的了解。

代码风格

在深入这个章节前，一定要读一下在内核代码目录 Documentation/CodingStyle下的内核代码风格手册。如果想让提交的代码补丁能为内核开发者接受，这种代码风格是一套必须遵循的规则。

其中一些规则关心缩进，程序流程，命名传统，等等。

最常用的几个是：

. 总是用8个字符的tab 缩进，一行要长80列。如果缩进影响到写函数，这通常是因为代码里有了太多的嵌套。可以用内核源码中scripts/cleanfile 脚本来改动tab 缩进并验证一行的大小：

scripts/cleanfile my\_module.c

.也可以用缩进工具 indent来正确缩进代码：

**sudo...**

**总结**

本章很简短解释了如何下载Linux 源码并进行第一次构建。本章也讲了一些基本概念。说起来，本章相当简短可能讲得不够，但不要担心，这只是一个介绍。这也是为什么下一章会更深入到内核构建的细节，怎么在外面或者作为内核一部分来实际编译一个驱动，还有在开始内核开发的漫长旅程之前必须学习的基础。