嵌入式 Arm Linux 入门必读书籍推荐

老吴嵌入式 2020-01-08 08:28

编者荐语：

Q群里有人发了一份书单，我浏览了一下，发现归纳的挺好，里面确实有不少经典的书籍值得阅读。另外，标题里的“入门”感觉实在太谦虚了，在我看来，能看完这些书的人肯定会是一个优秀的工程师。OK，开始 enjoy 这份书单吧！

以下文章来源于HackforFun ，作者HackforFun

HackforFun

Hack the CS world。

嵌入式 Arm Linux 入门必读书籍推荐

前段时间有个刚开始学习 Arm Linux 的同学问我：对于还处于入门阶段的新手，有什么建议。并让我推荐一些好的书籍。

嵌入式 Linux 是一个庞大的系统，涉及到硬件和计算机科学，是横跨电子和计算机的综合学科。很多从电子相关专业切入的学生对计算机原理和操作系统不了解，而从计算机相关专业切入的同学对硬件、电路也很迷惑，所以入门确实是有一定的难度的。

然后我告诉这位同学：首先得有兴趣，兴趣是克服一切困难的源动力。然后要善于利用网络和身边的资源，多尝试、多搜索、多讨论。

我本人也是电子相关专业(自动化)毕业的，大四找工作之前在实验室一直玩 51 单片机 和 Cortex-M3，可是机缘巧合却面试进了一家做 Android 的公司——面试之前，我看招聘要求上只写了要熟悉 Arm 体系结构，Cortex-M3 也是 Arm 啊，所以我就去面试了，最后竟然拿到了 offer。签三方的时候 HR 说：你回去后要开始学 Linux，这是你进来后工作的重点技术。我点头说：好的、好的……

出来后我就蒙了——Linux！WTF！我要是看到招聘要求上有写 Linux，再给我个胆子我也不敢去面试啊！

回来后去实验室，老师知道我签了这家公司，还是很高兴的！说你先学会怎么在 Linux 系统下工作吧，不然你连门的入不了。

于是老师给我找了一台台式机，装了 Ubuntu，然后又从实验室仓库的最里面翻出来了一块 S3C2410 的开发板—— 没错，我没写错，就是 2410。因为我们是自动化院系，大部分老师的项目都是和各种厂矿、车间的控制相关，这里面大部分都是用 PLC、单片机、MCU 来做控制，跑 Linux，那是不存在的！老师说，一个控制不当，机毁人亡，那是要去坐牢的。所以院系里也没人玩 Linux、这块开发板还是好几年前一家公司来做推广的时候送的。我会玩玩 Ubuntu 已经是比较潮流的了。

于是后面的几个月我就在这块 S3C2410 的板子上开始了 Arm Linux 的入门学习：不停的重复实验、不停的上网搜索……

后面就毕业，顺利入职。工作后蛮顺利，在各种芯片上折腾 Android、Linux，乐此不疲。

得益于入门时期的特殊经历，我对业内的相关书籍一直比较关注，看到哪位大牛出版了好书，就忍不住要买回来，看到网上有对应的 PDF 版本，也一定会想办法下下来。

所以刚好这几天有空，我就把这些我看过的书单整理了出来，希望对大家有所帮助：

这些书籍我按照 Arm Linux 系统所需要的知识结构，做了分类：

1. **Linux 系统环境**
2. **Git 版本管理**
3. **C 语言和汇编**
4. **Arm 体系结构**
5. **Linux 应用编程**
6. **Linux 内核基础原理**
7. **Linux 内核开发与实践**
8. **操作系统理论与基础**

前三类是基础，无论从事哪个方向的开发，这些都是必须要熟练掌握的基本功。后面五类是相辅相成的，理论基础相当于内功心法，开发实践相当于招式，要相互结合相互印证。招式好上手，使起来也漂亮，可以让你快速在江湖上闯出一定的小名头，但是要想走的更远、飞的更高、跻身真正的高手行列，还是离不开深厚的内功心法。

1、Linux 系统环境

熟悉 Linux 系统环境是一切的前提 —— Linux 系统环境中，最强大的开发武器是命令行和 Shell 脚本。

为什么？因为大量(几乎所有)的开源项目的编译、配置、都是通过命令行实现；在涉及海量源码的工程下，命令行的 find、grep 命令可以提供强大、高效的搜索功能，能帮忙快速的定位、理解源码；通过 Shell 脚本能够让大量重复的工作自动化，节省时间和生命。我曾经向 U-Boot 和 Linux Kernel mainline 提交过两个补丁，涉及到大几十个文件中雷同代码的修改，如果一步步手工修改，可能要几天时间，我花了大半天的时间写了个脚本利用 sed 命令来处理，跑一把就搞定了。

我听到过很多还处在入门阶段的同学说：以后的趋势是桌面化、图形化、所以我不需要命令行。其实这种看法是不对的，至少在可以看到的将来，命令行没有被取代的趋势。至少我现在还经常看到有同学问：我照着教程敲的这个命令，为什么报错了？我问他：你明白这个命令是什么意思吗？你看懂它报错的提示信息了吗？他说不知道。所以你要熟悉命令行，至少你要知道你敲下去的每个命令是什么意思，能看懂它的错误提示信息。能会用 help 命令行去查阅相关工具的用法。

所以这部分我推荐了四本书(资料)：

- 《LINUX命令行与SHELL脚本编程大全》第三版

这本书是一个外国人写的，然后中国人翻译，是一本非常详细的书籍，详细的讲解了 Linux 系统中各种常用命令的使用，以及使用 Linux 系统的一些通用原理。可以先粗读一遍，后面当作工具书来用，需要的时候随时查阅。

- 《Shell 编程大全》

  这本书是在网上下的，我大部分的 Shell 脚本技和命令行技巧都是从这本书书里面学的，是我经常查阅的工具书。

- 《Shellbook》

  也叫做《Shell编程范例》，是泰晓科技(也是魅族前 BSP 总监)的吴章金写的开源书籍，虽然只有100多页，但是在网络上广受欢迎。

- 《Shell 命令行操作》

  这份 14 页的 pdf 列举了一些常用命令行的使用范例，可以做快速参考。

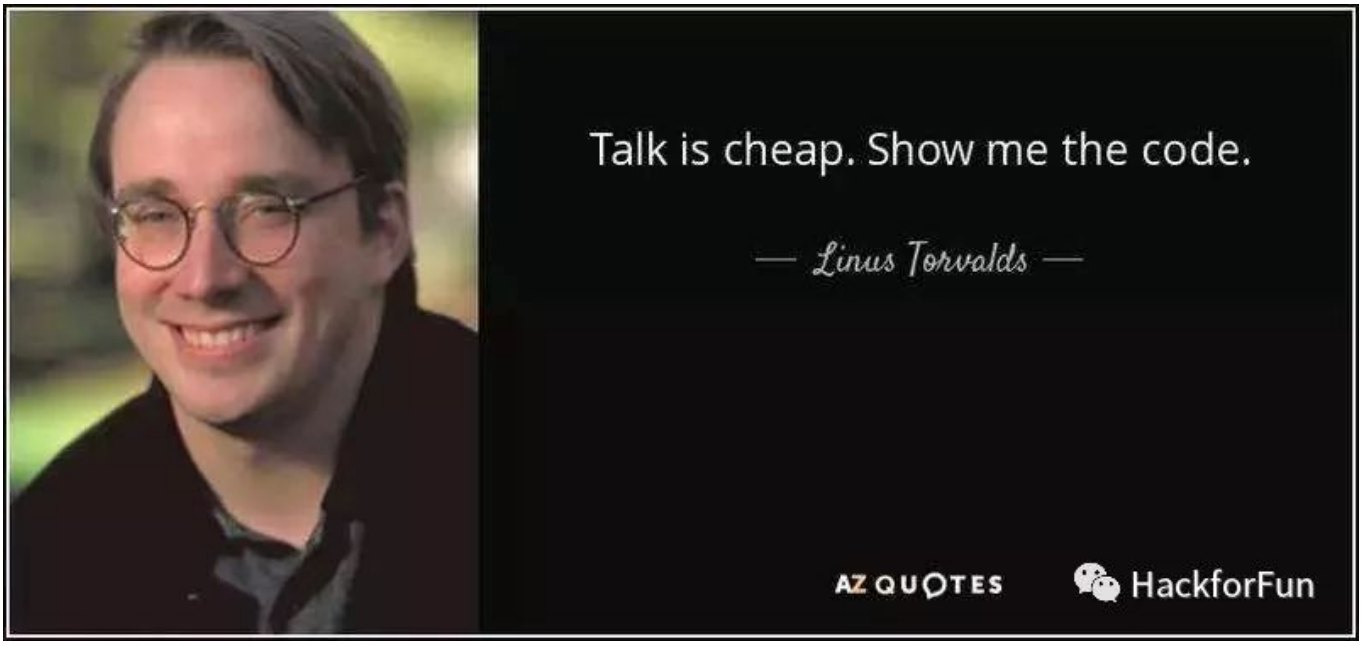
2、Git 版本管理

我把 Git 版本管理放在了第二位、是想凸显它的重要性，在现在这种团队化、快速迭代的开发模式中，用不好版本管理，开发工作也很难做的干净高效。我现在写文档都是用 Git + Markdown + Github 做版本管理，发现错误随时迭代修改。

据传说，中国的第一代的程序员——求伯君、王江民，他们晚上下班之前会把主机上存储自己写的程序的软件盘拆下来，带回家——怕发生意外，自己写的程序丢了。现在依托 Git 这种版本管理工具，似乎再也没有人有这种忧虑了。

其实版本管理工具很多，推荐 Git 是因为它用的广泛，Android 项目用的是 Git做版本管理，Linux 内核、U-Boot 这些知名的开源项目都是用 Git 做版本管理，还有 Github、Gitlab、Gitee 这种代码托管网站都是以 Git 作为基石，可见在版本管理上，Git 以及形成了一种生态级别的存在。所以你应该毫不犹豫的拥抱它。

对了，Git 的原型是 Linus 大神(就是写出 Linux kernel 的那位)在一个星期内写出来的。



我推荐了两本书，对，两本：

- 《GitHub 入门与实践》

  GitHub 是一个以 Git 技术作为依托实现的代码托管网站，你可以在上面选择以公开或者非公开的方式托管自己的代码，公开的方式就是所有的人都可以浏览下载你的代码，非公开的私有仓库只有你自己才能看到你的代码。

  这本书主要讲解了如何使用 GitHub，当然也列举了常用的 Git 基础操作。我把 Git 基础操作这部分单独列了出来，就是那份《Git基础功能》，看完这份资料基本可以应付日常的版本管理了。

- 《Pro Git》第二版

  这也是一本开源书籍，详细讲述了 Git 内部的基本原理，如果你想对 Git 有进一步的深入了解，去解决大型项目，多分支多仓库管理中的更复杂问题，可以读读这本书。

3、C 语言和汇编

做嵌入式开发、C 语言的重要性就不用多说了，从 U-Boot 到 Linux kernel，已经各种基础组件，基本都是用 C 语言写的，所以要做一个好的嵌入式 Hacker，C 语言是必须会的。

不像 C 这种跨平台的语言，汇编是和 CPU 体系结构强相关的一门语言，不同的架构 汇编都不一样，比如 32 位的 Armv7 和 64 位的 Armv8 汇编就不一样，比较难以掌握。

对于入门者来说，刚开始不会汇编也没关系，对你的工作影响不大，因为绝大部分场景都用不到。

但是当你的技能提高的一定层次，掌握了汇编就会让你更能深入系统内部，看清程序运行的内部机理。

U-Boot 和 Linux Kernel 中的第一段启动代码，基本都是汇编写的，Cache、MMU 相关的控制，也只能通过汇编实现。

有时候一段 C 代码运行异常，或者效率低下，或者你的 Linux 系统跑崩溃了，这时候把对应的代码反汇编，从汇编级别的去分析，更容易发现问题之所在。我平时经常的一部分工作就是通过汇编去发现或者分析前线部门或者客户报过来的各种疑难杂症。

当然了，汇编不用把每一句都搞懂，只要能大概看懂常用的指令就行，具体遇到问题的时候再去查对应的汇编手册，然后就是熟能生巧的问题了。

这里我推荐了三本 C 语言的书和 三份 Arm64 的汇编指令简介：

- 《C 和 指针》

  老外写的书，适合有一定 C 语言基础后提高进阶用。

- 《C 语言深度剖析》

  这本书只有 130 多页，作者是中国人，是个牛人，而且已经出版发行到第二版，大家如果有机会可以考虑买正版支持一下，搞懂了这本书，面试啥的应该不成问题。

- 《高质量C++编程指南》

  作者林锐博士，这本书用来培养自己的编码规范比较好，也是在大学实验室的适合，老师让我打印出来学习的一份资料。

- 另外三份 PDF 是关于 Arm64 指令集的，都很简短，最长的一份只有 135 页，适合做 Arm 汇编手册。用的时候拿来做参考，我没有加 Arm32 指令集相关的，Arm32 有更好的书籍推荐。

4、Arm 体系结构

要想把 Arm 嵌入式 玩的溜，尤其是想了解整个系统的，对 Arm 体系结构还是要有一定的学习。如果深入到一定的层次，Arm 体系结构相关的，也是必须要熟悉的，因为很多优化做到极致都是

和体系结构紧密相关的。

各位小伙伴，某国产手机大厂要招聘多名Linux内核工程师，薪水很诱人哟，工作地点：北京或者上海

Essential Functions

Design and develop new features for Android system to improve power consumption and performance.
Implement specific customization for EAS scheduler, Memory, and IO core layer in Android system.

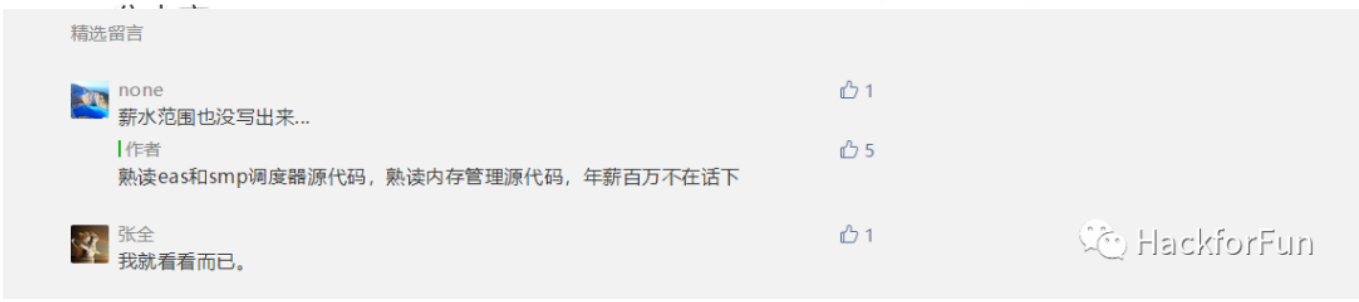
Key Requirement

Experienced in C and script language programming.
Experienced in developing and debugging Embedded Linux driver and application.
Broad knowledge in ARM architectures, Linux kernel
Being familiar with software development process, debugging tools and methodology in embedded Linux system.
Being familiar with one of subsystem of scheduler, memory and IO in Linux.
Good communication skills and team working.
Self-motivated in study and work.
5 years or more working experience in Linux kernel.

职位要求

笨叔简单分析了一下，这个职位的要求：

1. 要求熟悉ARM体系结构，至少ARMv8体系结构那个7000多页的手册应该翻过，对ARM内存管理，MMU，TLB，cache，页表管理，异常和中断管理等机制比较熟悉。
2. 熟练使用调试工具和issue定位的方法，比如笨叔死机黑屏专题里讲的东西。
3. 熟读EAS调度器源代码。笨叔觉得，这里应该需要候选者对进程管理，CFS调度器、SMP调度、EAS调度器等源代码应该熟读。
4. 熟读内存管理源代码。笨叔觉得内存管理是linux内核最难，最复杂的部分，涉及内容最多，特别是多核并发以及内存压力情况下的复杂场景的思考。这是候选者应该关注的部分。
5. 熟悉电源管理和性能调优。电源管理包括了EAS，cpufreq，cpuidle，DVFS等部



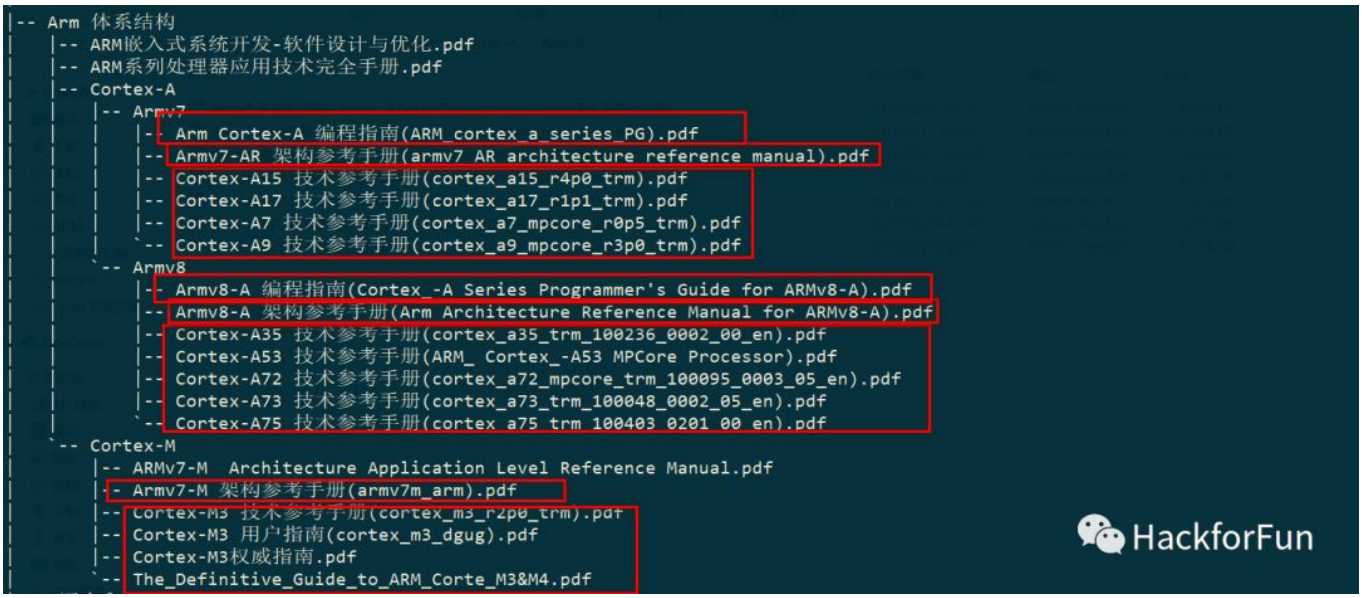
精选留言

none 1
薪水范围也没写出来...

作者 5
熟读eas和smp调度器源代码，熟读内存管理源代码，年薪百万不在话下

张全 1
我就看看而已。

上面是《奔跑吧 Linux Kernel》的作者笨叔叔前两天在朋友圈帮国内手机大厂打的招聘信息，里面有对 Arm 体系结构的要求，大家可以自行感受下。



现阶段的 Arm 处理器，应用最广泛的大概分为三类：定位于 MCU 的 Cortex-M 系列，定位于应用处理器的 Cortex-A 系列：Armv7-A 和 Armv8-A，Armv7-A 是 32 位的 CPU，包括 Cortex-A7/A9/A15/A17 这些，比如大家常见的 i.MX6ULL 芯片就属于 Arm Cortex-A7，Armv8-A 是 32 位的 CPU，包括 Cortex-A35/A53/A57/A72/A73 市面上在售 的 i.MX8，RK3399、以及现在的主流手机 CPU，都属于 Armv8-A。

这部分我推荐了两本书，剩下的都是 Arm 官方发布的文档：

- 《ARM 嵌入式系统开发-软件设计与优化》

  我印象中这本书的作者就是 Arm 公司的工程师，中文版是北航的一位老师翻译的，本书介绍了 Arm11 之前的历代经典架构，对 Arm 汇编指令，MPU、MMU、Cache、中断等关键模块的管理和应用介绍的十分详细，甚至还专门用了两章来介绍 Bootloader 和嵌入式操作系统的设计。

  因为这本书写作的时间比较早，所以没有设计 Cortex-A 系列的处理器，但是它任然是一本值得强烈推荐的书，因为对于 32 位的 Arm 来说，他们的架构和汇编指令保持了很大的延续性，把这本书看懂了，对 Arm 架构的理解也就不存在什么障碍了。

- 《ARM系列处理器应用技术完全手册》

  这本书是华清远见出的，主要内容是对 ARM 处理器的简介和 ARM 汇编程序设计，对学习 ARM 指令还是很有帮助的。

- ARM 官方文档

Arm 官方文档主要分为三类：编程指南(pg)，芯片的技术参考手册(trm)，架构参考手册(ARM)。这三类文档，gp-》trm-》ARM，前面的更容易懂，越到后面越贴近芯片内部，越难懂。所以建议大家按顺序阅读。

- 编程指南：
  - 《The_Definitive_Guide_to_ARM_Corte_M3&M4》
  - 《Arm Cortex-A 编程指南》
  - 《Armv8-A 编程指南》

    他们主要描述 Arm 处理器上各个模块的应用原理，和软件开发比较贴近，内容只有几百页，是最实用最 容易上手的文档。

- 技术参考手册(trm)

  trm 是 Technical Reference Manual 的缩写，它是针对特定 CPU 的单独文档，比如 Cortex-A7/A9/A53/A72 技术参考手册，内容主要包括对 CPU 上重要模块的介绍，比如 Cache、Timer、系统控制，性能监控、Debug 这些模块，多则 300 页，少的只有 100 多页，当你拿到特具体的开发板芯片，可以有针对性的去阅读。

- 架构参考手册(arm)

  这里的 arm 是 Architecture Reference Manual，Arm 官方称这份文档为 Arm arm。这份文档详细描述具体架构的内部细节，比如 《Armv7-AR 架构手册》是针对 32 位的 Arm Cortex-A/R 系列处理器架构的详细描述，《Armv8-A架构手册》是针对 64 位 Arm Cortex-A 系列处理器的架构详细描述，这份文档主要适用于 SOC 设计工程师和进行底层芯片开发的软件工程师，Armv7-AR 的架构文档 2000 多页，Armv8-A 的架构文档将近 9000 页，堪称宏篇巨著。了解大概结构后，我一般把它当作工具书，要用到相关模块的知识就去找出来看看。

5、Linux 应用编程

从应用编程的角度切入 Linux 世界，是一个很好的方式。相比内核开发，应用编程容易上手很多，甚至都不需要开发板，也不用担心写错程序把整个系统弄崩溃。而且透过应用程序的各种机制，我们也能感知到 Linux Kernel 在背后运行的机理，毕竟 Kernel 还是为应用服务的，每一个产品都要靠应用来实现和终端用户交互的接口。应用程序写好了，再来开发 Linux kernel，你会更容易理解，它为什么要这么实现。

关于应用编程我推荐了五本书：

- 《Linux 环境编程：从应用到内核》
- 《GNU Linux 编程指南》
- 《Linux 程序设计》
- 《Linux 系统编程》
- 《UNIX 环境高级编程》

除了第一本是中国人写的，剩下的四本都是老外写的，而且部头都比较大，所以选一本你自己喜欢的认真去实践即可。

6、Linux 内核基础原理

很多学习嵌入式 Linux 开发的同学一上来就呆住一个驱动就开始较劲，然后发现看着像天书一样，各种奇怪的 API 完全不知道为什么需要这样写，这时候你需要了解一些 Linux 内核的基础原理。

这种书一般不是上来就给你分析某个驱动时怎么写的，它侧重于描述 Linux 内核的基础框架，基础数据结构，基本驱动模型，CPU 调度、内存管理这些机制。

我推荐了五本书：

- 《Linux 内核设计与实现》
- 《Linux 设备驱动程序》
- 《深入 Linux 内核架构》
- 《深入Linux 设备驱动程序内核机制》
- 《Mastering Linux Kernel Development》

当然，你不用希望一上来就能熟读他们，第一遍你可能只能看个大概，而且看的很容易犯困。没关系，瞌睡了就去睡觉，或者去找个简单的驱动写一写，过几天遇到疑问了再返回来看一看。这样周而复始，总会有看明白的一天。

这里面的前两本我从大四就开始看，最开始看着就犯迷糊，到后来工作了，结合平时写的一些驱动，再返回去看，总算不再会打瞌睡。现在偶尔还会拿起来翻一翻。

我知道有的同学会说 《Linux 设备驱动程序》这本书太老了，它是针对 2.6 内核的，连设备树都没有。其实在技术里面，也有一个道和术的问题，这本书讲的是驱动的基础原理，虽然 Linux kernel 一直在发展，现在 5.4 都快发布了，但是那些基本的内在逻辑还是一致的，驱动最底层的结构和原理都还没有变化。Linux Kernel 开发的哲学就是这样的，不允许你上来就直接推到之前的设计重新搞一套，你必须在原有的基础上做持续性的改进，对于一个拥有庞大用户群体的系统，前向兼容性比什么都重要。至于设备树(dts)这些东西，只是技术实现上的细枝末节，找到规律很快就能掌握。

《深入 Linux 内核架构》这本书是一个德国人写的，其实我没看过，但是我知道这本书很出名，讲的很深入，在我的阅读计划之内。

《深入 Linux 设备驱动程序内核机制》这本书是一个中国人写的，基于 Linux 2.6.39，也是讲 Linux 设备驱动的基本原理和组件，对于做驱动开发来说，还是值得一读，而且中国人写的书在思维也和大家更接近。

《Mastering Linux Kernel Development》这本书比较新，基于 Linux 4.9 ，所以这本书也只有英文版本。

7、Linux 内核开发与实战

第六部分给大家推荐的书都比较偏理论，这部分推荐到是比较偏实践的，基本就是告诉你开发环境怎么搭建，内核怎么编译，文件系统怎么构建，驱动怎么编写，所以建议六、七两部分要结合起来看。

还是五本书：

- 《Linux 设备驱动开发详解》

  业内大名鼎鼎的宋宝华老师写的，这本书我附了两个版本，第二版和第三版，第二版是基于 Linux 2.6，第三版是基于 Linux 4.x，两本书我都买了，第二版我看的比较多，两本书都很不错，比较注重理论和实践的结合。

- 《精通 Linux设备驱动开发》

  这是一本老外写的书，宋宝华老师参与了中文版的翻译，这本书也是基于 Linux 2.6，涵盖了 Linux 内核中重要模块的驱动编写，比如字符设备，块设备，网卡，串口，framebuffer，alsa，usb。

- 《嵌入式 Linux 基础教程》

这本书讲是一个老外写的，华清远见翻译的，讲的比较泛，但是涉及到了一个嵌入式系统的方方面面，比如 U-Boot、Linux Kernel、文件系统、toolchain，GDB。对于了解整个嵌入式系统是如何构建的还是比较有价值的

- 《构建嵌入式 Linux 系统》

  这本书也是一个老外写的，目前已经出到第二版，但是我没找到，这本书和 《嵌入式 Linux 基础教程》比较类似，两本可以对照着看。

8、操作系统理论与基础

Linux Kernel 是一个综合性的工程，是大量科学理论的具体实现。对这些基础理论有个基础的认识，会让我们学的更好。当然，短时间内不了解这些知识，也不会影响到你去调试一个驱动，Bringup 一块板子，但是要想让更深入的去研究一些问题，这些知识还是要补充的。尤其是对于电子相关专业的学生，可能连计算机基本原理，进程、线程、虚拟内存、Cache这些概念都没有，那下边这两本书可以很好的帮到你。

- 《深入理解计算机系统》

  这本书很出名，作者任教于卡内基-梅隆大学。是我刚工作的时候我老板送我的一本书，刚开始没当回事，后面随着工作中遇到疑问的逐渐增多，来来回回翻了好多遍，前段时间又拿来出来翻了翻，为了查一个 Cache 相关的问题。

- 《Operating Systems：Three Easy Pieces》

  这本书是宋宝华老师推荐的，它涉及了操作系统中非常难的一些知识点：比如调度和内存管理，但是这本书却能用简单易懂的方式把这些原理解释清楚，读了这本书我对操作系统中的调度，内存管理总算有了一些稍微清晰的认识，感觉我又可以去读一读内核中相关的代码实现了。

写在最后

这份书单非常的长，53 份！

几乎不可能全部读完！

其实你也用不着把他们详细的读完！

如果从这篇介绍文章中，你能区分出哪些是要精读的，哪些是当作工具书来需要的时候查阅的，它们就会对你有一定的帮助了。

我也会感到很高兴。

另外，写作不宜，这里面提到的大部分书，都已出版，如果可能，去买一本，支持下作者，也算是对知识的一种尊重。

我每年都会买很多书，有些书买来都没来得及看，但是看着就感觉是好书，买回来，摆在案头，看着就开心。

杂七杂八 | 一些有用的网站

GeoLab 219