搞嵌入式Linux，做底层还是应用？底层要掌握哪些技能？

大鱼机器人 2021-02-01 11:37

ID：技术让梦想更伟大

整理:李肖遥

很多学习嵌入式的新人、工程师，在学习到半途中，总会感觉到迷茫，不知道自己从哪方面入手、进阶，也不知道自己目前处于学习的哪个层次，不清楚往后从哪方面来提升自己。

针对这些工程师常见的情况，今天编者就以一个8年工作经验的嵌入式工程师来分享，来看一下嵌入式学习，下一阶段的你需要掌握些什么。

◇ 嵌入式工程师的职业方向 ◇

学习嵌入式，大致来说可以分为四个方向：

一、嵌入式硬件开发：熟悉电路等知识，非常熟悉各种常用元器件，掌握模拟电路和数字电路设计的开发能力。熟练掌握嵌入式硬件知识，熟悉硬件开发模式和设计模式，熟悉ARM32位处理器嵌入式硬件平台开发、并具备产品开发经验。精通常用的硬件设计工具：Protel/PADS(PowerPCB)/Cadence/OrCad。一般需要有4~8层高速PCB设计经验。

二、嵌入式驱动开发：熟练掌握Linux操作系统、系统结构、计算机组成原理、数据结构相关知识。熟悉嵌入式ARM开发，至少掌握Linux字符驱动程序开发。具有单片机、ARM嵌入式处理器的移植开发能力，理解硬件原理图，能独立完成相关硬件驱动调试，具有扎实的硬件知识，能够根据芯片手册编写软件驱动程序。

三、嵌入式系统开发：掌握Linux系统配置，精通处理器体系结构、编程环境、指令集、寻址方式、调试、汇编和混合编程等方面的内容;掌握Linux文件系统制作，熟悉各种文件系统格式(YAFFS2、JAFFS2、RAMDISK等)；熟悉嵌入式Linux启动流程，熟悉Linux配置文件的修改;掌握内核裁减、内核移植、交叉编译、内核调试、启动程序Bootloader编写、根文件系统制作和集成部署Linux系统等整个流程;、熟悉搭建Linux软件开发环境(库文件的交叉编译及环境配置等);

四、嵌入式软件开发：精通Linux操作系统的概念和安装方法、Linux下的基本命令、管理配置和编辑器，包括VI编辑器，GCC编译器，GDB调试器和 Make 项目管理工具等知识;精通C语言的高级编程知识，包括函数与程序结构、指针、数组、常用算法、库函数的使用等知识、数据结构的基础内容，包括链表、队列等;掌握面向对象编程的基本思想，以及C++语言的基础内容;

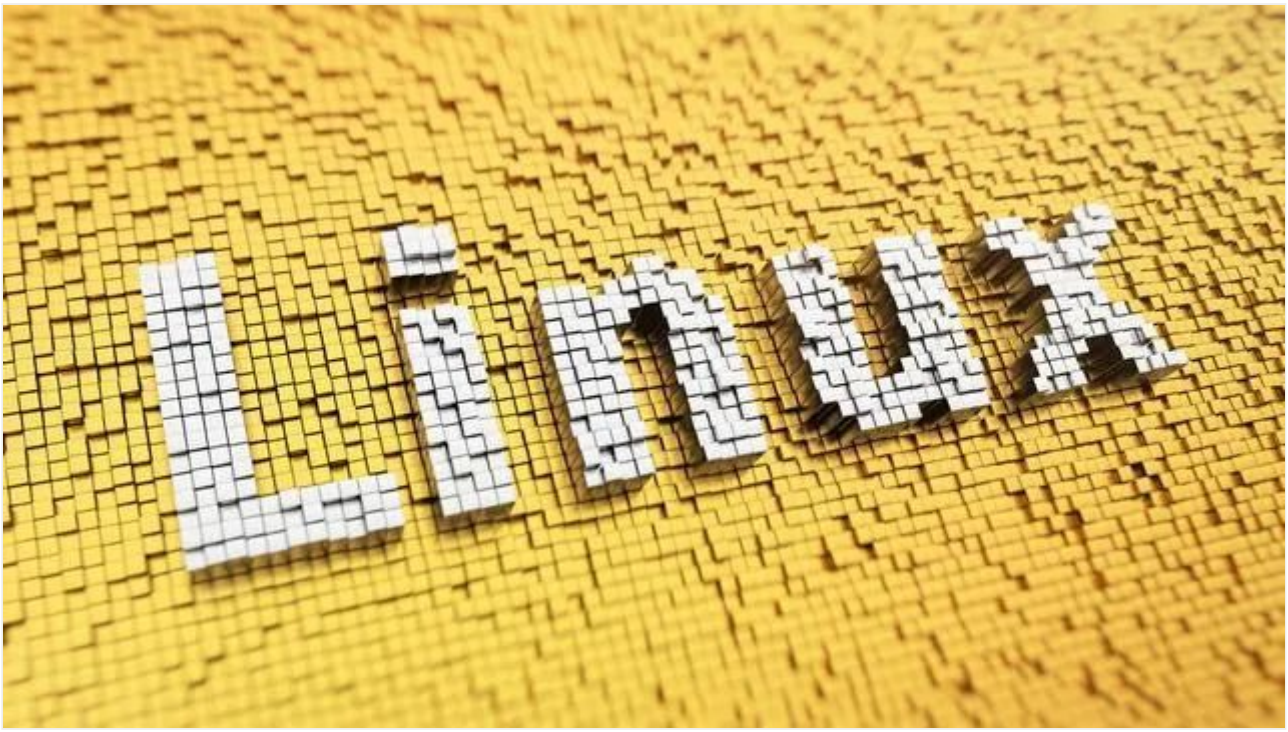
精通嵌入式Linux下的程序设计，精通嵌入式Linux开发环境，包括系统编程、文件I/O、多进程和多线程、网络编程、GUI图形界面编程、数据库;熟悉常用的图形库的编程，如QT、GTK、miniGUI、fltk、nano-x等。

公司的日常活动还是看公司的规模，大一点的一般只是让你负责一个模块，这样你就要精通一点。若是公司比较小的话估计要你什么都做一点。还要了解点硬件的东西。

说了这么多，依据我个人经验而言，做嵌入式和纯软件的最大区别在于：

纯软学习的是一门语言，例如C,C++,java,甚至Python，语言说到底只是一门工具，就像学会英语法语日语一样。

但嵌入式学习的是软件+硬件，通俗的讲，它学的是做系统做产品，讲究的是除了具体的语言工具，更多的是如何将一个产品分解为具体可实施的软件和硬件，以及更小的单元。



不少人问，**将来就业到底是选驱动还是选应用？只能说凭兴趣，并且驱动和应用并不是截然分开的。**具体原因有如下几点：

1）我们说的驱动，其实并不局限于硬件的操作，还有操作系统的原理、进程的休眠唤醒调度等概念。想写出一个好的应用，想比较好的解决应用碰到的问题，这些知识大家应该都懂。

2）做应用的发展路径个人认为就是业务纯熟。比如在通信行业、IPTV行业、手机行业，行业需求很了解。

3）做驱动，其实不能称为“做驱动”，而是可以称为“做底层系统”，做好了这是通杀各行业。比如一个人工作几年，做过手机、IPTV、会议电视，但是这些产品对他毫无差别，因为他只做底层。当应用出现问题，解决不了时，他就可以从内核角度给他们出主意，提供工具。做底层的发展方向，应该是技术专家。

4）其实，做底层还是做应用，之间并没有一个界线，有底层经验，再去做应用，会感觉很踏实。有了业务经验，再了解一下底层，很快就可以组成一个团队。

----------◇ **嵌入式Linux底层系统包含哪些东西？** ◇----------

嵌入式Linux里含有bootloader, 内核, 驱动程序、根文件系统这4大块。

一、bootloader

它就是一个稍微复杂的裸板程序。但是要把这裸板程序看懂写好一点都不容易。Windows下好用的工具弱化了我们的编程能力。很多人一玩嵌入式就用ADS、KEIL。能回答这几个问题吗?

Q:一上电，CPU从哪里取指令执行?
A:一般从Flash上指令。

Q:但是Flash一般是只能读不能直接写的，如果用到全局变量，这些全局变量在哪里?
A:全局变量应该在内存里。

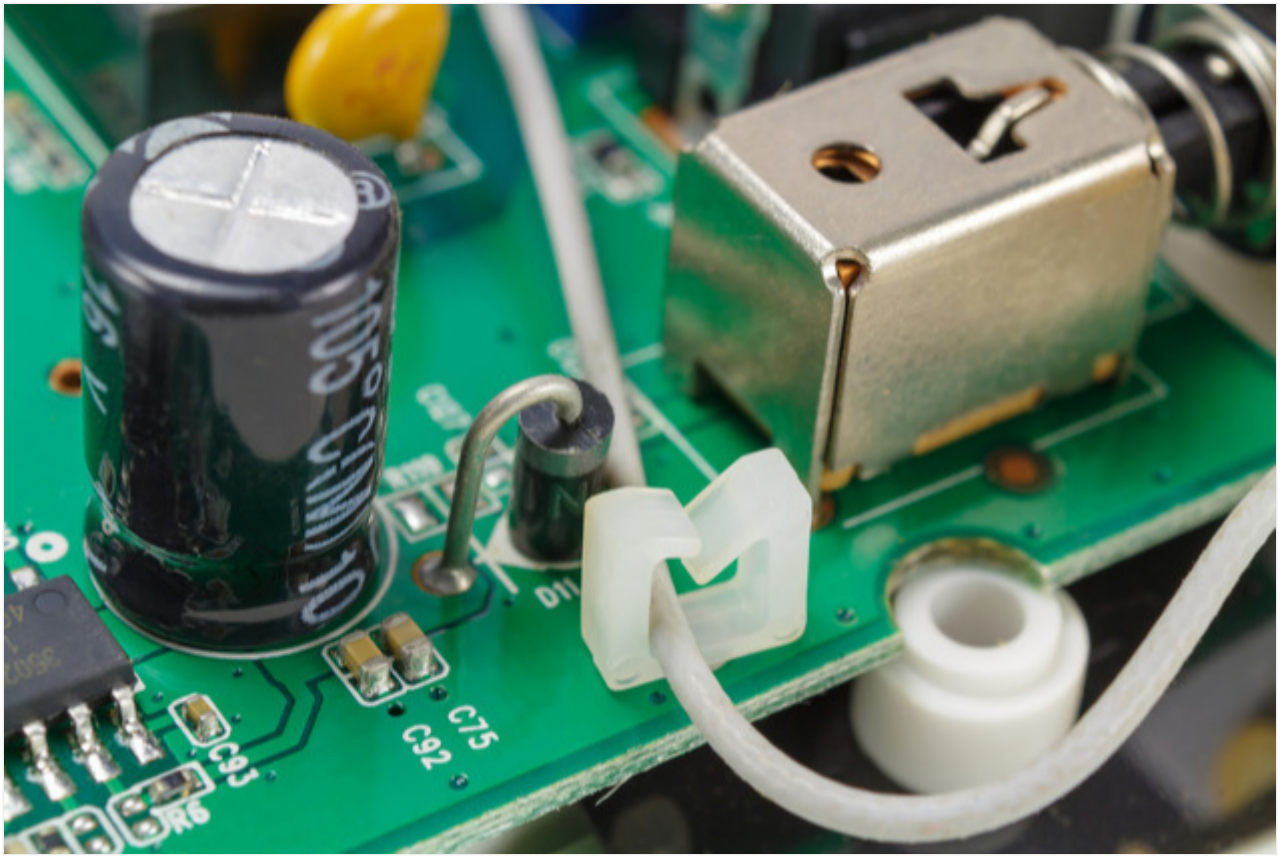
Q:那么谁把全局变量放到内存里去?
A:长期用ADS、KEIL的朋友，你能回答吗？这需要”重定位”。在ADS或KEIL里，重定位的代码是制作这些工具的公司帮你写好了。你可曾去阅读过?

Q:内存那么大，我怎么知道把”原来存在Flash上的内容”读到内存的”哪个地址去”?
A:这个地址用”链接脚本”决定，在ADS里有scatter文件，KEIL里也有类似的文件。但是，你去研究过吗?

Q:你说重定位是把程序从Flash复制到内存，那么这个程序可以读Flash啊?
A:是的，要能操作Flash。当然不仅仅是这些，还有设置时钟让系统运行得更快等等。

先自问自答到这里吧，对于bootloader这一个裸板程序，其实有**3部分要点：**

①对硬件的操作



对硬件的操作，需要看原理图、芯片手册。这需要一定的硬件知识，不要求能设计硬件，但是至少能看懂; 不求能看懂模拟电路，但是要能看懂数字电路。这方面的能力在学校里都可以学到，微机原理、数字电路这2本书就足够了。

想速成的话，就先放掉这块吧，不懂就GOOGLE、发贴。另外，芯片手册是肯定要读的，别去找中文的，就看英文的。开始是非常痛苦，以后就会发现那些语法、词汇一旦熟悉后，读任何芯片手册都很容易。

②对ARM体系处理器的了解

对ARM体系处理器的了解,可以看杜春蕾的<ARM体系架构与编程>，里面讲有汇编指令，有异常模式、MMU等。也就这3块内容需要了解。

③程序的基本概念：重定位、栈、代码段数据段BSS段等

程序的基本概念，王道当然是去看编译原理了。可惜，这类书绝对是天书级别的。若非超级天才还是别去看了。可以看韦东山的<嵌入式Linux应用开发完全手册>。

对于bootloader，可以先看<ARM体系架构与编程>，然后自己写程序把各个硬件的实验都做一遍，比如GPIO、时钟、SDRAM、UART、NAND。把它们都弄清楚了，组在一起就很容易看懂u-boot了 。

总结一下，看懂硬件原理图、看芯片手册，这都需要自己去找资料。

二、内核

想速成的人，先跨过内核的学习，直接学习怎么写驱动。

想成为高手，内核必须深刻了解。注意，是了解，要对里面的调度机制、内存管理机制、文件管理机制等等有所了解。

推荐两本书：

1. 通读<Linux内核完全注释>,请看薄的那本
2. 选读<Linux内核情景分析>, 想了解哪一块就读哪一节

三、驱动

驱动包含两部分：硬件本身的操作、驱动程序的框架。又是硬件，还是要看得懂原理图、读得懂芯片手册，多练吧。

①硬件本身的操作

说到驱动框架，有一些书介绍一下。LDD3,即<Linux设备驱动>，老外写的那本，里面介绍了不少概念，值得一读。但是，它的作用 也就限于介绍概念了。入门之前可以用它来熟悉一下概念。

②驱动程序的框架

驱动方面比较全的介绍，应该是宋宝华的<Linux设备驱动开发详解>了。要想深入了解某一块，<Linux内核情景分析>绝对是超5星级推荐。别指望把它读完，1800多页，上下两册呢。某一块不清楚时，就去翻一下它。任何一部分，这书都可以讲上2、3百页，非常详细。并且是以某个目标来带你分析内核源码。它以Linux2.4为例，但是原理相通，同样适用于其它版本的Linux。

把手上的开发板所涉及的硬件，都去尝试写一个驱动吧。有问题就先”痛苦地思考”，思考的过程中会把很多不相关的知识串联起来，最终贯通。

四、根文件系统

大家有没有想过这2个问题：

Q:对于Linux做出来的产品，有些用作监控、有些做手机、有些做平板。那么内核启动后，挂载根文件系统后，应该启动哪一个应用程序呢?

A:内核不知道也不管应该启动哪一个用户程序。它只启动init这一个应用程序，它对应/sbin/init。

显然，这个应用程序就要读取配置文件，根据配置文件去启动用户程序(监控、手册界面、平板界面等等，这个问题提示我们，文件系统的内容是有一些约定的，比如要有/sbin/init，要有配置文件 。

Q:你写的hello,world程序，有没有想过里面用到的printf是谁实现的?

A:这个函数不是你实现的，是库函数实现的。它运行时，得找到库。

这个问题提示我们，文件系统里还要有库。

简单的自问自答到这里，要想深入了解，可以看一下busybox的init.c，就可以知道init进程做的事情了。

当然，也可以看<嵌入式Linux应用开发完全手册>里构建根文件系统那章。

----------◇ **嵌入式Linux学习书籍推荐** ◇----------

1. 硬件方面的书: 微机原理、数字电路，高校里的教材。

2. Linux方面的书：
<ARM体系架构与编程>
<嵌入式Linux应用开发完全手册>

<Linux设备驱动>，老外写的那本

<Linux设备驱动开发详解>

<Linux内核完全注释>

<Linux内核情景分析>

在做驱动的时候，肯定会用到与内核相关的东西，或者需要和内核中的某些模块配合，这样你也要理解内核的某些部分是如何实现的，最后，你应该可以很好的掌握Linux的内核整体框架是什么。

这些都是进步，都是在你一次又一次的开发中需要总结的东西，如果你不总结，永远都是从头开始（或者说永远都是还没看懂别人代码为什么这么做的时候，就去改它，然后可以工作了），就完事了，这样你永远也不可能提高，最后你就有了现在的这种感觉，觉得自己什么都不是，什么都不懂。

还有一点要说明的，现在有许多人搞Linux开发，却不去用Linux系统做为自己工作的平台，在这种情况下，你很难理解Linux内核的实现机制，以及为什么要采用这种方式实现。

你都没用过Linux系统，就想去实现一个与Linux运行机理相符合的项目，这是不可能的。就是你这个项目成功了，它也肯定不是最优的，或者是不符合Linux的使用习惯的（包括内核的扩展和应用程序的实现）。

-END-

猜你喜欢（点击下划线阅读）

嵌入式的坑在哪方面?

五年，我成为了一名嵌入式工程师。

嵌入式开发中的三种程序构架

最 后

若觉得文章不错，**转发分享**，也是我们继续更新的动力。

5T资源大放送！包括但不限于：C/C++，Linux，Python，Java，PHP，人工智能，PCB、FPGA、DSP、labview、单片机、等等！

在公众号内回复「更多资源」，即可免费获取，期待你的关注~


点分享


点点赞　点在看