嵌入式 linux 入门六步曲

由于很多人总问这个问题，所以这里做一个总结供大家参考。这里必须先说明，以下的步骤都是针对 Linux 系统的，并不面向 WinCE。也许你会注意到，现在做嵌入式的人中，做 linux 研究的人远比做 WinCE 的人多，很多产家提供的资料也是以 linux 为主。我一直很难理解，其实 WinCE 的界面比 linux 的界面好看多了，使用起来也很方便，更为重要的是，WinCE 的开发和 Windows 下的开发基本一样，学起来简单得多，但是学 linux 或者使用 linux 做嵌入式的人就是远比 WinCE 多。在和很多工作的人交流时我了解到，他们公司从没考虑使用 WinCE，因为成本高，都是使用 linux 进行开发。我读研究生的实验室中也没有使用 WinCE 的，大都研究 linux，也有少部分项目使用 vxwork，但是就没有听说过使用 WinCE 的，原因就是开源!当然现在 WinCE6.0 听说也开源，不过在成本和资源上 linux 已经有了无人能挡的优势。与此相对应的是，越来越多的电子厂商已经开始使用 linux 开发产品。举个例子，Google 近期开发的智能手机操作系统 Android 其实就是使用 linux-2.6.23 内核进行改进得到的。

第一、学习基本的裸机编程

对于学硬件的人而言，必须先对硬件的基本使用方法有感性的认识，更必须深刻认识该硬件的控制方式，如果一开始就学 linux 系统、学移植那么只会马上就陷入一个很深的漩涡。我在刚刚开始学 ARM 的时候是选择 ARM7(主意是当时 ARM9 还很贵)，学 ARM7 的时候还是保持着学 51 单片机的思维，使用 ADS 去编程，第一个实验就是控制 led。学过一段时间 ARM 的人都会笑这样很笨，实际上也不是，我倒是觉得有这个过程会好很多，因为无论做多复杂的系统最终都会落实到这些最底层的硬件控制，因此对这些硬件的控制有了感性的认识就好很多了学习裸机的编程的同时要好好理解这个硬件的构架、控制原理，这些我称他为理解硬件。所谓的理解硬件就是说，理解这个硬件是怎么组织这么多资源的，这些资源又是怎么由 cpu、由编程进行控制的。比如说，s3c2410 中有 AD 转换器，有 GPIO(通用 IO 口)，还有 nandflash 控制器，这些东西都有一些寄存器来控制，这些寄存器都有一个地址，那么这些地址是什么意思？又怎么通过寄存器来控制这些外围设备的运转？还有，norflash 内部的每一个单元在这个芯片的内存中都有一个相应的地址单元，那么这些地址与刚刚说的寄存器地址又有什么关系？他们是一样的吗？而与 norflash 相对应的 nandflash 内部的储存单元并不是线性排放的，那么 s3c2410 怎么将 nandflash 的地址映射在内存空间上进行使用？或者简单地说应该怎么用 nandflash？再有，使用 ADS 进对 ARM9 行编程时都需要使用到一个初始化的汇编文件，这个文件究竟有什么用？他里面的代码是什么意思？不要这个可以吗？诸如此类都是对硬件的理解，理解了这些东西就对硬件有很深的理解了，这对以后更深一步的学习将有很大的帮助，如果跳过这一步，我相信越往后学越会觉得迷茫，越觉得这写东西深不可测。因为，你的根基没打好。

对于这部分不久将提供一份教程，这个教程中的例程并不是我为我们所代理的板子写的，是我在我们学院实验室拿的，英培特为他们自己的实验箱写的，不过很有借鉴意义，可以作为一份有价值的参考。

第二、使用 linux 系统进行一些基本的实验

在买一套板子的时候一般会提供一些 linux 的试验例程，好好做一段时间这个吧，这个过程也是很有意义的，也是为进一步的学习积累感性认识，你能想象一个从没有使用过 linux 系统的人能学好 linux 的编程吗？好好按照手册上的例程做一做里面的实验，虽然有点娃娃学走路，有点弱智，但是我想很多高手都会经历这个过程。

在这方面我们深蓝科技目前没有计划提供相应的例程，主要是开发板的提供商会提供很丰富的例程，我们不做重复工作，只提供他们没有的、最有价值的东西给大家。

第三、研究完整的 linux 系统的的运行过程

所谓完整的 linux 系统包括哪些部分呢？

三部分：bootloader、linux kernel(linux 内核)、rootfile(根文件系统)。

那么这 3 部分是怎么相互协作来构成这个系统的呢？各自有什么用呢？三者有什么联系？怎么联系？系统的执行流程又是怎么样的呢？搞清楚这个问题你对整个系统的运行就很清楚了，对于下一步制作这个 linux 系统就打下了另一个重要的根基。介绍这方面的资料网上可以挖掘到几吨，自己好好研究吧。

第四、开始做系统移植

上面说到完整的 linux 有 3 部分，而且你也知道了他们之间的关系和作用，那么现在你要做的便是自己动手学会制作这些东西。

当然我不可能叫你编写这些代码，这不实现。事实上这个 3 者都能在网下载到相应的源代码，但是这个源代码不可能下载编译后就能在你的系统上运行，需要很多的修改，直到他能运行在你的板子上，这个修改的过程就叫移植。在进行移植的过程中你要学的东西很多，要懂的相关知识也很多，等你完成了这个过程你会发现你已经算是一个初出茅庐的高手了。

在这个过程中如果你很有研究精神的话你必然会想到看源代码。很多书介绍你怎么阅读 linux 源代码，我不提倡无目的地去看 linux 源代码，用许三多的话说，这没有意义。等你在做移植的时候你觉得你必须去看源代码时再去找基本好书看看，这里我推荐一本好书倪继利的《linux 内核的分析与编程》，这是一本针对 linux-2.6.11 内核的书，说得很深，建议先提高自己的 C 语言编程水平再去看。

至于每个部分的移植网上也可以找到好多吨的资料，自己研究研究吧，不过要提醒的是，很多介绍自己经验的东西都或多或少有所保留，你按照他说的去做总有一些问题，但是他不会告诉你怎么解决，这时就要靠自己，如果自己都靠不

住就找我一起研究研究吧，我也不能保证能解决你的问题，因为我未必遇到过你的问题，不过我相信能给你一点建议，也许有助你解决问题。

这一步的最终目的是，从源代码的官方主页上(都是外国的，悲哀)下载标准的源代码包，然后进行修改，最终运行在板子上。

盗用阿基米德的一句话：“给我一根网线，我能将 linux 搞定”。

第五、研究 linux 驱动程序的编写

移植系统并不是最终的目的，最终的目的是开发产品，做项目，这些都要进行驱动程序的开发。

Linux 的驱动程序可以说是五花八门，linux2.4 和 linux2.6 的编写有相当大的区别，就是同为 linux2.6 但是不同版本间的驱动程序也有区别，因此编写 linux 的驱动程序变都不是那么容易的事情，对于最新版本的驱动程序的编写甚至还没有足够的参考资料。那么我的建议就是使用、移植一个不算很新的版本内核，这样到时学驱动的编程就有足够的资料了。

第六、研究应用程序的编写

做作品做项目除了编写驱动程序，最后还要编写应用程序。现在的趋势是图形应用程序的开发，而图形应用程序中用得最多的还是 qt/e 函数库。我一直就使用这个函数库来开发自己的应用程序，不过我希望你能使用国产的 MiniGUI 函数库。盗用周杰伦的广告词就是“支持国产，支持 MiniGUI”。MiniGUI 的编程比较相似 Windows 下的 VC 编程，比较容易上手，效果应该说是相当不错的，我曾使用过来开发 ARM7 的程序。不过 MiniGUI 最大的不好就是没有像 qtopia 这样的图形操作平台，这大大限制了他的推广，我曾经幻想过与北京飞漫公司(就是 MiniGUI 的版权拥有者)合作使用 MiniGUI 函数库开发像 qtopia 这样的图形操作平台，不过由于水平有限这只能是幻想了，呵呵。完成这一步你基本就学完了嵌入式 linux 的全部内容了。

还有一个小小的经验想和大家分享。我在学习嵌入式 linux 的过程中很少问人，客观原因是身边的老师、同学师兄都没有这方面的高手，主观原因是我不喜欢问人，喜欢自己研究解决问题。这样做有个好处，就是可以提高自己解决问题的能力，因为做这些东西总有很多问题你难以理解，别人也没有这方面的经验，也不是所有问题都有人给你答案，这时必须要自己解决问题，这样，个人的解决问题能力就显得非常关键了。因此我的建议就是一般的问题到网上搜索一下，确实找不到答案了就问问高手，还是不行了就自己去研究，不要一味去等别人帮你解决问题。记住，问题是学习的最好机会。