嵌入式 Linux+Android 学习路线

版本	日期	作者	说明
V1	2016.07.29	韦东山	第 1 版本，Android 部分未写

我是 1999 年上的大学，物理专业。在大一时，我们班里普遍弥漫着对未来的不安，不知道学习了物理后出去能做什么。你当下的经历、当下的学习，在未来的一天肯定会影响到你。毕业后我们也各自找到了自己的职业：出国深造转行做金融、留校任教做科研、设计芯片、写程序、创办公司等等，这一切都离不开在校时学到的基础技能（数学、IT、电子电路）、受过煅炼的自学能力。

所以，各位正在迷茫的在校生，各位正在尝试转行的程序员，未来一定有你的位置，是好是坏取决于你当下的努力与积累。

我不能预言几年后什么行业会热门，也不能保证你照着本文学习可以发财。我只是一个有十几年经验的程序员，给对编程有兴趣的你，提供一些建议。

程序员的三大方向

程序员的方向，一般可以分为 3 类：**专业领域**、**业务领域**、**操作系统领域**。你了解它们后，按兴趣选择吧。对于专业领域，我提供不了建议。业务，也就是应用程序，它跟操作系统并不是截然分开的：

① 开发实体产品时，应用程序写得好的人，有时候需要操作系统的知识，比如调度优先级的设置、知道某些函数可能会令进程休眠。

② 写应用程序的人进阶为系统工程师时，他需要从上到下都了解，这时候就需要有操作系统领域的知识了，否则，你怎么设计整个系统的方案呢？

③ 做应用程序的人，需要了解行业的需求，理解业务的逻辑。所以，当领导的人，多是做应用的。一旦钻入了某个行业，很难换行业。

④ 而操作系统领域，做好了这是通杀各行业：他只负责底层系统，在上面开发什么业务跟他没关系。这行很多是技术宅，行业专家。

⑤ 操作系统和业务之间并没有一个界线。有操作系统经验，再去做应用，你会对系统知根知底，碰到问题时都有解决思路。有了业务经验，你再了解一下操作系统，很快就可组成一个团队自立门户，至少做个 CTO 没问题。

专业领域

它又可以分为下面 2 类。

学术研究

比如语音、图像处理、人工智能，这类工作需要你有比较强的理论知识，我倾向于认为这类人是“科学家”，他们钻研多年，很多时候是在做学术研究。

在嵌入式领域，需要把他们的成果用某种算法表达出来，针对某种芯片进行优化，这部分工作也许有专人来做。

工程实现

也有这样一类人，他们懂得这些专业领域的概念，但是没有深入钻研。可以使用各类开源资料实现某个目标，做出产品。比如图像处理，他懂得用 opencv 里几百个复杂函数来实现头像识别。有时候还可以根据具体芯片来优化这些函数。

“专业领域”不是我的菜，如果你要做这一块，我想最好的入门方法是在学校学习研究生、博士课程。

业务领域

换句话说，就是应用程序，这又可以分为下面 2 类。

界面显示

做产品当然需要好的界面，但是，不是说它不重要，是没什么发展后劲。
现在的热门词是 Android APP 和 IOS APP 开发。你不要被 Android、IOS 两个词骗了，它们跟以前的 VC、VB 是同一路货色，只是、仅仅是一套 GUI 控件的实现。

希望没有冒犯到你，我有理由。
一个程序需要有 GUI 界面，但是程序的内在逻辑才是核心。Android、IOS 的开发工具给我们简化了 GUI 的开发，并提供了这些控件的交互机制，封装并提供了一些服务(比如网络传输)。
但是程序内部的业务逻辑、对视频图像声音的处理等等，这才是核心。另外别忘了服务器那边的后台程序：怎样更安全地保存数据、保护客户的隐私，怎样处理成千上万上百万的并发访问，等等，这也是核心。
但是，从 Android、IOS APP 入门入行，这很快！如果你是大四，急于找到一份工作，那么花上 1、2 个月去学习 Android 或 IOS，应该容易找到工作，毕竟 APP 的需求永远是最大的，现在这两门技术还算热门。在 2011、2012 年左右，Android 程序员的起薪挺高，然后开始下滑。Android APP 的入门基本只要 1 个月，所以懂的人也越来越多。2013、2014 年，IOS 开发的工资明显比 Android 高了，于是各类 IOS 培训也火曝起来。中华大地向来不缺速成人才，估计再过一阵子 IOS 工程师也是白菜价了。会 Android、IOS 只是基本要求，不信去 51job 搜搜 Android 或 IOS，职位要求里肯定其他要求。

业务逻辑

举个简单例子，做一个打卡软件，你需要考虑这些东西：
① 正常流程是上班下班时都要打卡
② 有人忘记了怎么办？作为异常记录在案，推送给管理员
③ 请假时怎么处理？
④ 加班怎么处理？

对于更复杂的例子，视频会议系统里，各个模块怎么对接，各类协议怎么兼容，你不深入这个行业，你根本搞不清楚。
应用开发的职位永远是最多的，入门门槛也低。基本上只要你会 C 语言，面试时表现比较得体，一般公司都会给你机会。因为：
① 你进公司后，还需要重新培训你：熟悉它们的业务逻辑。
② 你要做的，基本也就是一个个模块，框架都有人给你定好了，你去填代码就可以了。

说点让你高兴的事：软件公司里，做领导的基本都是写应用程序的（当然还有做市场的）。写应用程序的人，对外可以研究市场接待客户，对内可以管理程序员完成开发，不让他做领导让谁做？**如果你的志向是写应用程序，那么我建议你先练好基本功：数据结构、算法是必备，然后凭兴趣选择数据库、网络编程等等进行深入钻研。最后，选择你看好的、感兴趣的行业深耕个 10 年吧**。做应用开发的人选择了某个行业，后面是很难换行业的，选行很重要！

操作系统领域

UCOS 太简单，VxWorks 太贵太专业，Windows 不玩嵌入式了，IOS 不开源，所以对于操作系统领域我们也只能玩 Linux 了。
在嵌入式领域 Linux 一家独大！
Android 呢？Android 跟 QT 一样，都是一套 GUI 系统。只是 Google 的实力太强了，现在 Android 无处不在，所以很多时候 Linux+Android 成了标配。注意，在这里我们关心的是 Android 的整个系统、里面的机制，而不是学习几个 API 然后开发界面程序。
操作系统领域所包含的内容，简单地说，就是制作出一台装好系统的专用“电脑”，可以分为：
① 为产品规划硬件：
按需求、性能、成本选择主芯片，搭配周边外设，交由硬件开发人员设计。
② 给单板制作、安装操作系统、编写驱动
③ 定制维护、升级等系统方案
④ 还可能要配置、安装 Android 等 GUI 系统：
⑤ 为应用开发人员配置开发环境
⑥ 从系统角度解决疑难问题
这个领域，通常被称为“底层系统”或是“驱动开发”。
先解决 2 个常见误区：
① 这份工作是写驱动程序吗？

看看上面罗列的 6 点，应该说，它包含驱动开发，但远远不只有驱动开发。

② 我们还需要写驱动吗？不是有原厂吗？或者只需要改改就可以？

经常有人说，芯片原厂都做好驱动了，拿过来改改就可以了。如果，你的硬件跟原厂的公板完全一样，原厂源码毫无 BUG，不想优化性能、削减成本，不想做一些有特色的产品，那这话是正确的。

但是在这个不创新就是找死的年代，可能吗？！原因有二：

① 即使只是修改代码，能修改的前提是能理解；能理解的最好煅炼方法是从零写出若干驱动程序

② 很多时候，需要你深度定制系统。

以前做联发科手机只需要改改界面就可以出货了，现在山寨厂一批批倒下。大家都使用原厂的方案而不加修改时，最后只能拼成本。举个例子，深圳有 2 家做交通摄像头、监控摄像头的厂家，他们曾经找我做过 4 个项目：

① 改进厂家给的 SD 卡驱动性能，使用 DMA。

② 换了 Flash 型号后，系统经常出问题，需要修改驱动 BUG。

③ 触摸屏点击不准，找原因，后来发现是旁路电容导致的。

④ 裁减成本，把 4 片 DDR 换为 2 片 DDR，需要改 bootloader 对 DDR 的初始化。

这些项目都很急，搞不定就无法出货，这时候找原厂？除非你是中兴华为等大客户，否则谁理你？

我在中兴公司上班时，写驱动的时间其实是很少的，大部分时间是调试：系统调优，上帮 APP 工程师、下帮硬件工程师查找问题。我们从厂家、网上得到的源码，很多都是标准的，当然可以直接用。但是在你的产品上也许优化一下更好。比如我们可以把摄像头驱动和 DMA 驱动揉合起来，让摄像头的数据直接通过 DMA 发到 DSP 去。

我们可以在软件和硬件之间起桥梁作用，对于实体产品，有可能是软件出问题也可能是硬件出问题，一般是底层系统工程师比较容易找出问题。当硬件、软件应用出现问题，他们解决不了时，从底层软件角度给他们出主意，给他们提供工具。再比如方案选择：芯片性能能否达标、可用的 BSP 是否完善等等，这只能由负责整个方案的人来考虑，他必须懂底层。

在操作系统领域，对知识的要求很多：

① 懂硬件知识才能看懂电路图

② 英文好会看芯片手册

③ 有编写、移植驱动程序的能力

④ 对操作系统本身有一定的理解，才能解决各类疑难问题
⑤ 理解 Android 内部机制
⑥ 懂汇编、C 语言、C++、JAVA

它绝对是一个大坑，没有兴趣、没有毅力的人慎选。
① 这行的入门，绝对需要半年以上，即使全天学习也要半年。
② 它的职位，绝对比 APP 的职位少
③ 并且你没有 1、2 年经验，招你到公司后一开始你做的还是 APP。

优点就是：
① 学好后，行业通杀，想换行就换行；想自己做产品就自己做产品。
② 相比做应用程序的人，不会被经常变动的需求搞得天天加班。
③ 门槛高，当然薪水相对就高。
操作系统领域，我认为适合于这些人：
① 硬件工程师想转软件工程师，从底层软件入门会比较好
② 单片机工程师，想升级一下。会 Linux 底层的人肯定会单片机，会单片机的人不一定会 Linux。
③ 时间充足的学生：如果你正读大二大三，那么花上半年学习嵌入式 Linux 底层多有益处。
④ 想掌握整个系统的人，比如你正在公司里写 APP，但是想升为系统工程师，那么底层不得不学。
⑤ 想自己创业做实体产品的工程师，你有钱的话什么技术都不用学，但是如果没钱又想做产品，那么 Linux 底层不得不学。
⑥ 做 Linux APP 的人，没错，他们也要学习。
这部分人不需要深入，了解个大概就可以：bootloader 是用来启动内核，Linux 的文件系统(第 1 个程序是什么、做什么、各目录干嘛用)、APP 跟驱动程序的调用关系、工具链，有这些概念就可以了

本文中，就把操作系统默认为 Linux，讲讲怎么学习嵌入式 Linux+Android 系统。

嵌入式 Linux+Android 系统包含哪些内容

嵌入式 Linux 系统包含哪些东西？不要急，举一个例子你就知道了。
① 电脑一开机，那些界面是谁显示的？

是 BIOS，它做什么？一些自检，然后从硬盘上读入 windows，并启动它。
类似的，这个 BIOS 对应于嵌入式 Linux 里的 bootloader。这个 bootloader 要去 Flash 上读入 Linux 内核，并启动它。
② 启动 windows 的目的是什么？
当然运行应用程序以便上网、聊天什么的了。
这些上网程序、聊天程序在哪？
在 C 盘、D 盘上。
所以，windows 要先识别出 C 盘、D 盘。在 Linux 下我们称之为根文件系统。
③ windows 能识别出 C 盘、D 盘，那么肯定有读写硬盘的能力。
这个能力我们称之为驱动程序。当然不仅仅是操作硬盘，还有网卡、USB 等等其他硬件。
嵌入式 Linux 能从 Flash 上读出并执行应用程序，肯定也得有 Flash 的驱动程序啊，当然也不仅仅是 Flash。

简单地说，嵌入式 LINUX 系统里含有 bootloader、内核、驱动程序、根文件系统、应用程序这 5 大块。而应用程序，我们又可以分为：C/C++、Android。
所以，嵌入式 Linux+Android 系统包含以下 6 部分内容：
① bootloader
② Linux 内核
③ 驱动程序
④ 使用 C/C++编写的应用程序
⑤ Android 系统本身
⑥ Android 应用程序
Android 跟 Linux 的联系实在太大了，它的应用是如此广泛，学习了 Linux 之后没有理由停下来不学习 Android。在大多数智能设备中，运行的是 Linux 操作系统；它上面要么安装有 Android，要么可以跟 Android 手机互联。现在，**Linux+Android 已成标配**。

怎么学习嵌入式 Linux 操作系统

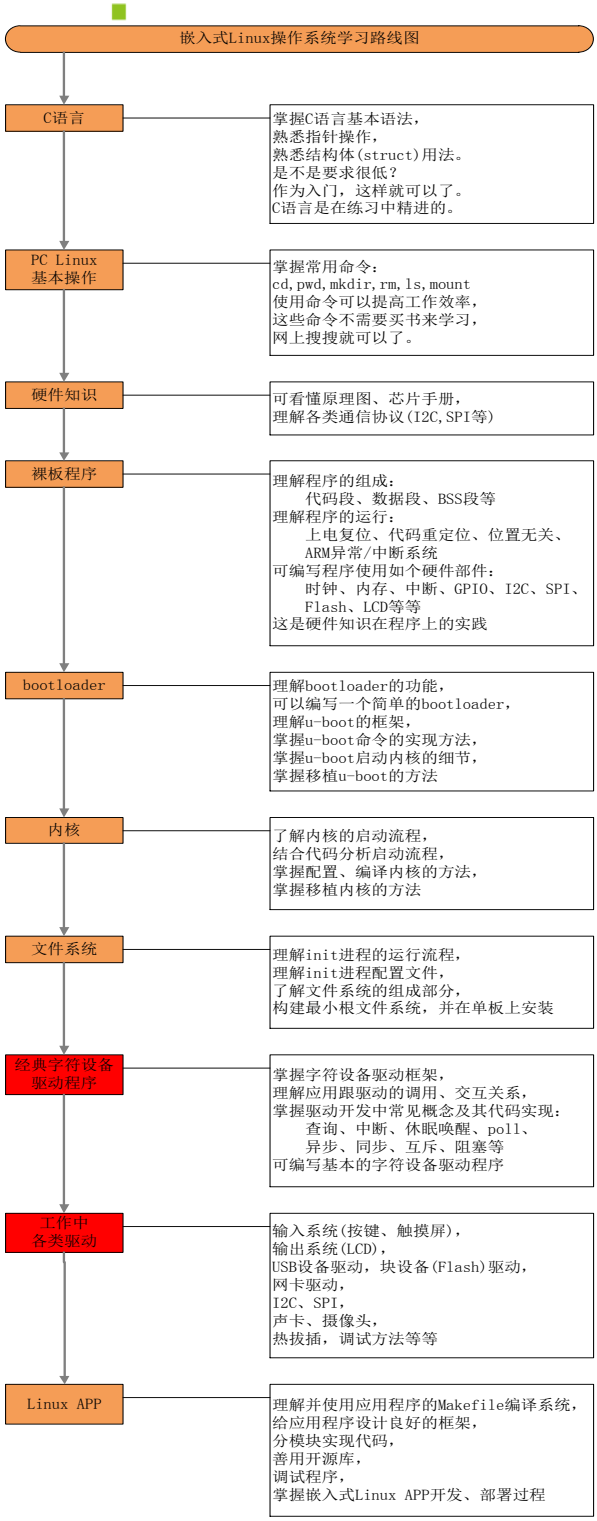
本文假设您是零基础，以实用为主，用最快的时间让你入门；后面也会附上想深入学习时可以参考的资料。在实际工作中，我们从事的是“操作系统”周边的开发，并不会太深入学习、修改操作系统本身。

① 操作系统具有进程管理、存储管理、文件管理和设备管理等功能，这些核心功能非常稳定可靠，基本上不需要我们修改代码。我们只需要针对自己的硬件完善驱动程序
② 学习驱动时必定会涉及其他知识，比如存储管理、进程调度。当你深入理解了驱动程序后，也会加深对操作系统其他部分的理解
③ Linux 内核中大部分代码都是设备驱动程序，可以认为 Linux 内核由各类驱动构成
但是，要成为该领域的高手，一定要深入理解 Linux 操作系统本身，要去研读它的源代码。
在忙完工作，闲暇之余，可以看看这些书：
① 赵炯的《linux 内核完全注释》，这本比较薄，推荐这本。他后来又出了《Linux 内核完全剖析》，太厚了，搞不好看了后面就忘记前面了。
② 毛德操、胡希明的《LINUX 核心源代码情景分析》，此书分上下册，巨厚无比。当作字典看即可：想深入理解某方面的知识，就去看某章节。
③ 其他好书还有很多，我没怎么看，没有更多建议

基于快速入门，上手工作的目的，您先不用看上面的书，**先按本文学习**。

入门路线图

假设您是零基础，我们规划了如下入门路线图。前面的知识，是后面知识的基础，建议按顺序学习。每一部分，不一定需要学得很深入透彻，下面分章节描述。


嵌入式Linux操作系统学习路线图
C语言
掌握C语言基本语法，
熟悉指针操作，
熟悉结构体(struct)用法。
是不是要求很低？
作为入门，这样就可以了。
C语言是在练习中精进的。
PC Linux
基本操作
掌握常用命令：
cd,pwd,mkdir,rm,ls,mount
使用命令可以提高工作效率，
这些命令不需要买书来学习，
网上搜搜就可以了。
硬件知识
可看懂原理图、芯片手册，
理解各类通信协议(I2C,SPI等)
裸板程序
理解程序的组成：
代码段、数据段、BSS段等
理解程序的运行：
上电复位、代码重定位、位置无关、
ARM异常/中断系统
可编写程序使用如个硬件部件：
时钟、内存、中断、GPIO、I2C、SPI、
Flash、LCD等等
这是硬件知识在程序上的实践
bootloader
理解bootloader的功能，
可以编写一个简单的bootloader，
理解u-boot的框架，
掌握u-boot命令的实现方法，
掌握u-boot启动内核的细节，
掌握移植u-boot的方法
内核
了解内核的启动流程，
结合代码分析启动流程，
掌握配置、编译内核的方法，
掌握移植内核的方法
文件系统
理解init进程的运行流程，
理解init进程配置文件，
了解文件系统的组成部分，
构建最小根文件系统，并在单板上安装
经典字符设备
驱动程序
掌握字符设备驱动框架，
理解应用跟驱动的调用、交互关系，
掌握驱动开发中常见概念及其代码实现：
查询、中断、休眠唤醒、poll、
异步、同步、互斥、阻塞等
可编写基本的字符设备驱动程序
工作中
各类驱动
输入系统(按键、触摸屏)，
输出系统(LCD)，
USB设备驱动，块设备(Flash)驱动，
网卡驱动，
I2C、SPI，
声卡、摄像头，
热拔插，调试方法等等
Linux APP
理解并使用应用程序的Makefile编译系统，
给应用程序设计良好的框架，
分模块实现代码，
善用开源库，
调试程序，
掌握嵌入式Linux APP开发、部署过程

学习驱动程序之前的基础知识

C 语言

只要是理工科专业的，似乎都会教 C 语言。我见过很多 C 语言考试 90、100 分的一上机就傻了，我怀疑他们都没在电脑上写过程序。

理论再好，没有实践不能干活的话，公司招你去干嘛？
反过来，实践出真知，学习 C 语言，必须练练练、写写写！
当你掌握基本语法后，就可以在电脑上练习一些 C 语言习题了；
当你写过几个 C 程序后，就可以进入下一阶段的裸机开发了。
① 不需要太深入
作为快速入门，只要你会编写“Hello, world!”，会写冒泡排序，会一些基础的语法操作，暂时就够了。指针操作是重点，多练习；
不需要去学习过多的数据结构知识，只需要掌握链表操作，其他不用学习，比如：队列、二叉树等等都不用学；不需要去学习任何的函数使用，比如文件操作、多线程编程、网络编程等等；这些知识，在编写 Linux 应用程序时会用，但是在操作系统特别是驱动学习时，用不着！永往直前吧，以后碰到不懂的 C 语言问题，我们再回过头来学习。
在后续的“裸机开发”中，会让你继续练习 C 语言，那会更实战化。
C 语言是在写代码中精进的。

② 可以在 Visual Studio 下学习，也可以在 Linux 下学习，后者需要掌握一些编译命令
我们暂时没有提供 C 语言的教程，找一本 C 语言书，网上找找免费的 C 语言视频(主要看怎么搭建环境)，就可以自学了。

PC Linux 基本操作：

对于 PC Linux，我们推荐使用 Ubuntu，在它上面安装软件非常简便。
我们的工作模式通常是这样：在 Windows 下阅读、编写代码，然后把代码上传到 PC Linux 去编译。实际上，Ubuntu 的桌面系统已经很好用了，我们拿到各种智能机可以很快上手，相信 Ubuntu 的桌面系统也可以让你很快上手。为了提高工作效率，我们通常使用命令行来操作 Ubuntu。不用担心，你前期只需要掌握这几条命令就可以了，它们是如此简单，我干脆列出它们：
① cd : Change Directory（改变目录）

```
cd 目录名     // 进入某个目录
```

```
cd ..        // cd “两个点”：返回上一级目录
cd -         // cd “短横”：返回上一次所在目录
```

② pwd : Print Work Directory（打印当前目录 显示出当前工作目录的绝对路径）
③ mkdir : Make Directory（创建目录）

```
mkdir abc          // 创建文件夹 abc
mkdir -p a/b/c     // 创建文件夹 a，再 a 下创建文件夹 b，再在 b 下创建文件夹 c
```

④ rm : Remove（删除目录或文件）

```
rm  file      // 删除名为 file 的文件
rm -rf dir    // 删除名为 dir 的目录
```

⑤ ls : List（列出目录内容）
⑥ mount : 挂载

```
mount -t nfs -o nolock,vers=2 192.168.1.123:/work/nfs_root /mnt
mount -t yaffs /dev/mtdblock3 /mnt
```

⑦ chown : Change owner（改变文件的属主，即拥有者）

```
chown book:book /work -R  // 对/work 目录及其下所有内容，属主改为 book 用户，组改为 book
```

⑧ chmod : Change mode（改变权限），下面的例子很简单粗暴

```
chmod 777 /work -R  // 对/work 目录及其下所有内容，权限改为可读、可写、可执行
```

⑨ vi : Linux 下最常用的编辑命令，使用稍微复杂，请自己搜索用法。
要练习这些命令，你可以进入 Ubuntu 桌面系统后，打开终端输入那些命令；或是用 SecureCRT、putty 等工具远程登录 Ubuntu 后练习。

硬件知识

我们学习硬件知识的目的在于能看懂原理图，看懂通信协议，看懂芯片手册；**不求能设计原理图，更不求能设计电路板**。对于正统的方法，你应该这样学习：
① 学习《微机原理》，理解一个计算机的组成及各个部件的交互原理。
② 学习《数字电路》，理解各种门电路的原理及使用，还可以掌握一些逻辑运算（与、或等）。
③ 《模拟电路》？好吧，这个不用学，至少我在工作中基本用不到它，现在全忘光了。

就我个人经验来说，这些课程是有用的，但是：
① 原理有用，实战性不强。

比如《微机原理》是基于 x86 系统，跟 ARM 板子有很大差别，当然原理相通。
我是在接触嵌入式编程后，才理解了这些课程。
② 每本书都那么厚，内容都很多，学习时间过长，自学有难度。

针对这些校园教材的不足，并结合实际开发过程中要用到的知识点，我们推出了《学前班_怎么看原理图》的系列视频：

学前班第 1 课第 1 节_怎么看原理图之 GPIO 和门电路.wmv
学前班第 1 课第 2.1 节_怎么看原理图之协议类接口之 UART.wmv
学前班第 1 课第 2.2 节_怎么看原理图之协议类接口之 I2C.wmv
学前班第 1 课第 2.3 节_怎么看原理图之协议类接口之 SPI.wmv
学前班第 1 课第 2.4 节_怎么看原理图之协议类接口之 NAND Flash.wmv
学前班第 1 课第 2.5 节_怎么看原理图之协议类接口之 LCD.wmv
学前班第 1 课第 3 节_怎么看原理图之内存类接口.wmv
学前班第 1 课第 4.1 节_怎么看原理图之分析 S3C2410 开发板.wmv
学前班第 1 课第 4.2 节_怎么看原理图之分析 S3C2440 开发板.wmv
学前班第 1 课第 4.3 节_怎么看原理图之分析 S3C6410 开发板.wmv

即使你只具备初中物理课的电路知识，我也希望能通过这些视频，让你可以看懂原理图，理解一些常见的通信协议；如果你想掌握更多的硬件知识，这些视频也可以起个索引作用，让你知道缺乏什么知识。这些视频所讲到的硬件知识，将在《裸板开发》系列视频中用到，到时可以相互对照着看，加深理解。

要不要专门学习 Windows 下的单片机开发

很多学校都开通了单片机的课程，很多人都是从 51 单片机、AVR 单片机，现在比较新的 STM32 单片机开始接触嵌入式领域，并且使用 Windows 下的开发软件，比如 keil、MDK 等。
问题来了，要不要专门学习 Windows 下的单片机开发？
① 如果这是你们专业的必修课，那就学吧
② 如果你的专业跟单片机密切相关，比如机械控制等，那就学吧
③ 如果你只是想从单片机入门，然后学习更广阔的嵌入式 Linux，那么放弃在 Windows 学习单片机吧！理由如下：
① Windows 下的单片机学习，深度不够
Windows 下有很好的图形界面单片机开发软件，比如 keil、MDK 等。
它们封装了很多技术细节，比如：
你只会从 main 函数开始编写代码，却不知道上电后第 1 条代码是怎么执行的；
你可以编写中断处理函数，但是却不知道它是怎么被调用的；
你不知道程序怎么从 Flash 上被读入内存；

也不知道内存是怎么划分使用的，不知道栈在哪、堆在哪；
当你想裁剪程序降低对 Flash、内存的使用时，你无从下手；
当你新建一个文件时，它被自动加入到工程里，但是其中的机理你完全不懂；
等等等。

② 基于 ARM+Linux 裸机学习，可以学得更深，并且更贴合后续的 Linux 学习。
实际上它就是 Linux 下的单片机学习，只是一切更加原始：所有的代码需要你自己来编写；哪些文件加入工程，需要你自己来管理。在工作中，我们当然倾向于使用 Windows 下更便利的工具，但是在学习阶段，我们更想学习到程序的本质。一切从零编写代码、管理代码，可以让我们学习到更多知识：
你需要了解芯片的上电启动过程，知道第 1 条代码如何运行；
你需要掌握怎么把程序从 Flash 上读入内存；
需要理解内存怎么规划使用，比如栈在哪，堆在哪；
需要理解代码重定位；
需要知道中断发生后，软硬件怎么保护现场、跳到中断入口、调用中断程序、恢复现场；
你会知道，main 函数不是我们编写的第 1 个函数；
你会知道，芯片从上电开始，程序是怎么被搬运执行的；
你会知道，函数调用过程中，参数是如何传递的；
你会知道，中断发生时，每一个寄存器的值都要小心对待；
等等等。
你掌握了 ARM+Linux 的裸机开发，再回去看 Windows 下的单片机开发，会惊呼：怎么那么简单！并且你会完全明白这些工具没有向你展示的技术细节。

驱动程序=Linux 驱动程序软件框架+ARM 开发板硬件操作，我们可以从简单的裸机开发入手，先掌握硬件操作，并且还可以：
① 掌握如何在 PC Linux 下编译程序、把程序烧录到板子上并运行它
② 为学习 bootloader 打基础：掌握了各种硬件操作后，后面一组合就是一个 bootloader

为什么选择 ARM9 S3C2440 开发板，而不是其他性能更好的？

有一个错误的概念：S3C2440 过时了、ARM9 过时了。
这是不对的，如果你是软件工程师，无论是 ARM9、ARM11、A8 还是 A9，对我们来说是没有差别的。一款芯片，上面有 CPU，还有众多的片上设备(比如 UART、USB、LCD 控制器)。我们写程序时，并不涉及 CPU，只是去操作那些片上设备。
所以：差别在于片上设备，不在于 CPU 核；差别在于寄存器操作不一样。

因为我们写驱动并不涉及 CPU 的核心，只是操作 CPU 之外的设备，只是读写这些设备的寄存器。之所以推荐 S3C2440，是因为它的 Linux 学习资料最丰富，并有配套的第 1、2 期视频。

怎么学习 ARM+Linux 的裸机开发

学习裸机开发的目的有两个：
① 掌握裸机程序的结构，为后续的 u-boot 作准备
② 练习硬件知识，即：怎么看原理图、芯片手册，怎么写代码来操作硬件
后面的 u-boot 可以认为是裸机程序的集合，我们在裸机开发中逐个掌握各个部件，再集合起来就可以得到一个 u-boot 了。后续的驱动开发，也涉及硬件操作，你可以在裸机开发中学习硬件知识。
注意：如果你并不关心裸机的程序结构，不关心 bootloader 的实现，这部分是可以先略过的。在后面的驱动视频中，我们也会重新讲解所涉及的硬件知识。
推荐两本书：杜春蕾的《ARM 体系结构与编程》，韦东山的《嵌入式 Linux 应用开发完全手册》。后者也许是国内第 1 本涉及在 PC Linux 环境下开发的 ARM 裸机程序的书，如果我说错了，请原谅我书读得少。

对于裸机开发，我们提供有 2 部分视频：
① 环境搭建

第 0 课第 1 节_刚接触开发板之接口接线.wmv
第 0 课第 2 节_刚接触开发板之烧写裸板程序.wmv
第 0 课第 3 节_刚接触开发板之重烧整个系统.wmv
第 0 课第 4 节_刚接触开发板之使用 vmware 和预先做好的 ubuntu.wmv
第 0 课第 5 节_刚接触开发板之 u-boot 打补丁编译使用及建 sourceinsight 工程.wmv
第 0 课第 6 节_刚接触开发板之内核 u-boot 打补丁编译使用及建 sourceinsight 工程.wmv
第 0 课第 7 节_刚接触开发板之制作根文件系统及初试驱动.wmv
第 0 课第 8 节_在 TQ2440,MINI2440 上搭建视频所用系统.wmv
第 0 课第 9 节_win7 下不能使用 dnw 烧写的替代方法.wmv

② 裸机程序开发

第 1 课 环境搭建及工具、概念介绍.wmv
第 2 课 GPIO 实验.wmv
第 3 课 存储管理器实验.wmv
第 4 课 MMU 实验.wmv
第 5 课 NAND FLASH 控制器.wmv

第 6 课 中断控制器.wmv
第 7 课 系统时钟和 UART 实验.wmv
第 8 课 LCD 实验.wmv

要声明的是：

录制上述《裸机程序开发》视频时，本意是结合《嵌入式 Linux 应用开发完全手册》的《第 2 篇 ARM9 嵌入式系统基础实例篇》来讲解，所以视频里没有完全从零编写代码，需要结合书本来学习。
① 书和视频并不是完全配套的，不要照搬，其中的差异并不难解决。
《嵌入式 Linux 应用开发完全手册》发表于 2008 年，使用了很多款开发板，并且那时的开发板配置较低(Nand Flash 是 64M)；《裸机程序开发》视频使用 JZ2440 开发板录制。
② 书和视频，适用于所有 S3C2440 开发板，包括 mini2440、tq2440 等
天下 S3C2440 配置都是相似的，基本也就是 LED、按键所用引脚不同，LCD 型号不同；你学习了书、视频，如果连这些差异都搞不定的话，那就是你我的失败了。

学习方法是这样的：
① 先看《环境搭建》视频来搭建开发环境
② 书（第 2 篇）和视频（裸机程序开发）结合，看完一章，练习一章
一定要编写代码，即使是照抄也要写。
③ 如果对于 ARM 架构相关的知识，觉得模糊或是想了解得更深入，参考《ARM 体系结构与编程》

学习程度：
① 理解一个裸机程序的必要结构：异常向量、硬件初始化、代码重定位、栈
② 知道如何操作 GPIO、Flash、LCD、触摸屏等硬件
③ 很多人觉得 MMU 难以理解，可以放过它

bootloader 的学习

bootloader 有很多种，vivi、u-boot 等等，最常用的是 u-boot。
u-boot 功能强大、源码比较多，对于编程经验不丰富、阅读代码经验不丰富的人，一开始可能会觉得难以掌握。但是，u-boot 的主要功能就是：启动内核。它涉及：读取内核到内存、设置启动参数、启动内核。按照这个主线，我们尝试自己从零编写一个 bootloader，这个程序相对简单，可以让我们快速理解 u-boot 主要功能的实现。

从零编写 bootloader 的视频有：

毕业班第 1 课第 1.1 节_自己写 bootloader 之编写第 1 阶段.wmv
毕业班第 1 课第 1.2 节_自己写 bootloader 之编写第 2 阶段.wmv
毕业班第 1 课第 2 节_自己写 bootloader 之编译测试.wmv
毕业班第 1 课第 3 节_自己写 bootloader 之改进.wmv

分析 u-boot 1.1.6 的视频有：

第 9 课第 1 节 u-boot 分析之编译体验.wmv
第 9 课第 2 节 u-boot 分析之 Makefile 结构分析.wmv
第 9 课第 3 节 u-boot 分析之源码第 1 阶段.wmv
第 9 课第 3 节 u-boot 分析之源码第 2 阶段.wmv
第 9 课第 4 节 u-boot 分析之 u-boot 命令实现.wmv
第 9 课第 5 节 u-boot 分析_uboot 启动内核.wmv

移植一个全新 u-boot 的视频有：

毕业班第 2 课第 1 节_移植最新 u-boot 之初试.wmv
毕业班第 2 课第 2.1 节_移植最新 u-boot 之分析启动过程之概述.wmv
毕业班第 2 课第 2.2 节_移植最新 u-boot 之分析启动过程之内存分布.wmv
毕业班第 2 课第 2.3 节_移植最新 u-boot 之分析启动过程之重定位.wmv
毕业班第 2 课第 3.1 节_移植最新 u-boot 之修改代码之建新板_时钟_SDRAM_UART.wmv
毕业班第 2 课第 3.2 节_移植最新 u-boot 之修改代码支持 NAND 启动.wmv
毕业班第 2 课第 3.3 节_移植最新 u-boot 之修改代码支持 NorFlash.wmv
毕业班第 2 课第 3.4 节_移植最新 u-boot 之修改代码支持 NandFlash.wmv
毕业班第 2 课第 3.5 节_移植最新 u-boot 之修改代码支持 DM9000 网卡.wmv
毕业班第 2 课第 4.1 节_移植最新 u-boot 之裁剪和修改默认参数.wmv
毕业班第 2 课第 4.2 节_移植最新 u-boot 支持烧写 yaffs 映象及制作补丁.wmv

《嵌入式 Linux 应用开发完全手册》上对 u-boot 的讲解有如下章节：

15.1 Bootloader 简介
15.1.1 Bootloader 的概念
15.1.2 Bootloader 的结构和启动过程
15.1.3 常用 Bootloader 介绍
15.2 U-Boot 分析与移植
15.2.1 U-Boot 工程简介
15.2.2 U-Boot 源码结构

15.2.3 U-Boot 的配置、编译、连接过程
15.2.4 U-Boot 的启动过程源码分析
15.2.5 U-Boot 的移植
15.2.6 U-Boot 的常用命令
15.2.7 使用 U-Boot 来执行程序

学习方法如下：

① 先学习《从零编写 bootloader 的视频》，这可以从最少的代码理解 bootloader 的主要功能
② 再看书上对 u-boot 的讲解，并结合《分析 u-boot 1.1.6 的视频》来理解
③ 最后，有时间有兴趣的话，看《移植一个全新 u-boot 的视频》，这不是必须的。

学习程度：
① 理解 u-boot 的启动过程，特别是 u-boot 代码重定位：怎么从 Flash 上把自己读入内存
② 理解 u-boot 的核心：命令
③ 知道 bootloader 如何给内核传递参数
④ 知道 bootloader 是根据“bootcmd”指定的命令启动内核
⑤ 作为入门：只求理解，不要求能移植 u-boot

Linux 内核的学习

前面说过，内核本身不是我们学习的重点，但是了解一下内核的启动过程，还是很有必要的：工作中有可能要修改内核以适配硬件，掌握了启动过程才知道去修改哪些文件。

分析内核的视频有：
第 10 课第 1 节 内核启动流程分析之编译体验.wmv
第 10 课第 2 节 内核启动流程分析之配置.wmv
第 10 课第 3 节 内核启动流程分析之 Makefile.wmv
第 10 课第 4 节 内核启动流程分析之内核启动.wmv

移植内核的视频有：
业班第 3 课第 1 节_移植 3.4.2 内核之框架介绍及简单修改.wmv
毕业班第 3 课第 2 节_移植 3.4.2 内核之修改分区及制作根文件系统.wmv
毕业班第 3 课第 3 节_移植 3.4.2 内核之支持 yaffs 文件系统.wmv
毕业班第 3 课第 4 节_移植 3.4.2 内核之裁剪及 ECC 简介及制作补丁.wmv

《嵌入式 Linux 应用开发完全手册》上对内核的讲解有如下章节：

16.1 Linux 版本及特点
16.2 Linux 移植准备
16.2.1 获取内核源码
16.2.2 内核源码结构及 Makefile 分析
16.2.3 内核的 Kconfig 分析
16.2.4 Linux 内核配置选项
16.3 Linux 内核移植
16.3.1 Linux 内核启动过程概述
16.3.2 修改内核以支持 S3C2410/S3C2440 开发板
16.3.3 修改 MTD 分区
16.3.4 移植 YAFFS 文件系统
16.3.5 编译、烧写、启动内核

学习方法如下：
① 先看书，并结合《分析内核的视频》进行理解
② 如果有兴趣，再根据《移植内核的视频》自己掌握移植内核，这不是必须的

学习程度：
① 知道机器 ID 的作用，根据机器 ID 找到单板对应的文件
② 知道 Makefile、Kconfig 的作用，知道怎么简单地配置内核
③ 知道怎么修改分区
④ 作为入门：只求理解，不要求能移植

根文件系统

在驱动程序开发阶段，我们喜欢搭建一个最小根文件系统来调试驱动；
在开发应用程序时，也需要搭建文件系统，把各种库、配置文件放进去；
在发布产品时，你还需要修改配置文件，使得产品可以自动运行程序；
甚至你想实现插上 U 盘后自动启动某个程序，这也要要修改配置文件；
这一切，都需要你理解根文件系统的构成，理解内核启动后是根据什么配置文件来启动哪些应用程序。

分析根文件系统的视频有：

第 11 课第 1 节 构建根文件系统之启动第 1 个程序.wmv

第 11 课第 2 节 构建根文件系统之 init 进程分析.wmv
第 11 课第 3 节 构建根文件系统之 busybox.wmv
第 11 课第 4 节 构建根文件系统之构建根文件系统.wmv

《嵌入式 Linux 应用开发完全手册》上对文件系统的讲解有如下章节：

17.1 Linux 文件系统概述
17.1.1 Linux 文件系统的特点
17.1.2 Linux 根文件系统目录结构
17.1.3 Linux 文件属性介绍
17.2 移植 Busybox
17.2.1 Busybox 概述
17.2.2 init 进程介绍及用户程序启动过程
17.2.3 编译/安装 Busybox
17.3 使用 glibc 库
17.3.1 glibc 库的组成
17.3.2 安装 glibc 库
17.4 构建根文件系统
17.4.1 构建 etc 目录
17.4.2 构建 dev 目录
17.4.3 构建其他目录
17.4.4 制作/使用 yaffs 文件系统映象文件
17.4.5 制作/使用 jffs2 文件系统映象文件

学习方法：结合书和视频学习。
学习程度：
① 理解配置文件的作用
② 知道根文件系统中 lib 里的文件来自哪里
③ 可以制作、烧写文件系统映象文件

驱动程序的学习

《嵌入式 Linux 应用开发完全手册》对驱动程序的讲解不多，我们推出的“韦东山 Linux 视频第 2 期_驱动现场编写调试”，可以认为完全脱离了书本。
所以，**驱动程序的学习完全按照视频来就可以**了。

第 2 期的视频，对每一个驱动，先讲解硬件原理，然后从零写代码，从简单到复杂，逐渐完善它的功能。我们不会罗列专业术语，会参考日常生活的例子，力争用最形象的比喻让你轻松入门，同时又会很深入。

注意：我们可以让你入门时很轻松，但是要深入理解的话，这需要你跟着视频练习代码，这是个要慢慢思考的过程，不会轻松。

轻松的话，凭什么拿高工资？

再次申明：即使照抄也要写代码！很多人视频看得很高兴，但是写代码时就傻了。

经典字符设备驱动程序

视频中以 LED、按键驱动为例，讲解并练习开发过程中碰到的机制：查询、休眠-唤醒、中断、异步通知、poll、同步、互斥等等。后续更复杂的驱动程序，就是在这些机制的基础上，根据硬件特性设计出精巧的软件框架。相关的视频有（文件名中带“_P”的属于第 2 期加密视频）：

第 12 课第 1 节　字符设备驱动程序之概念介绍.wmv
第 12 课第 2.1 节　字符设备驱动程序之 LED 驱动程序_编写编译.wmv
第 12 课第 2.2 节　字符设备驱动程序之 LED 驱动程序_测试改进.wmv
第 12 课第 2.3 节　字符设备驱动程序之 LED 驱动程序_操作 LED.wmv
第 12 课第 3 节　字符设备驱动程序之查询方式的按键驱动程序.wmv
第 12 课第 4.1 节　字符设备驱动程序之中断方式的按键驱动_Linux 异常处理结构.wmv
第 12 课第 4.2 节　字符设备驱动程序之中断方式的按键驱动_Linux 中断处理结构.wmv
第 12 课第 4.3 节　字符设备驱动程序之中断方式的按键驱动_编写代码.wmv
第 12 课第 5 节　字符设备驱动程序之 poll 机制.wmv
第 12 课第 6 节　字符设备驱动程序之异步通知.wmv
第 12 课第 7 节　字符设备驱动程序之同步互斥阻塞.wmv
第 12 课第 8 节　字符设备驱动程序之定时器防抖动_P.wmv
第 13 课第 1 节　输入子系统概念介绍_P.wmv
第 13 课第 2 节　输入子系统第编写驱动程序_P.wmv

《嵌入式 Linux 应用开发完全手册》上对字符设备驱动程序的讲解有如下章节：

第 19 章　字符设备驱动程序
19.1　Linux 驱动程序开发概述
19.1.1　应用程序、库、内核、驱动程序的关系
19.1.2　Linux 驱动程序的分类和开发步骤
19.1.3　驱动程序的加载和卸载

学习方法：

① 沿着数据流向，从应用程序的对驱动程序的使用进行情景分析。

所谓情景分析，就是假设应用程序发起某个操作，你去分析其中的运作过程。比如应用程序调用 open、read、ioctl 等操作时涉及驱动的哪些函数调用。你要思考一个问题：一个应用程序，怎么获得按键信息，怎么去控制 LED。把其中数据的流向弄清楚了，对字符驱动程序也就基本理解了。

② 学习异常和中断时，可以结合书和视频；对于驱动程序中其他内容的学习，可以不看书。

工作中各类驱动程序

我们的视频中讲解的驱动程序非常多，目的有二：

① 在你工作中遇到同类驱动时提供借鉴

② 供你学习、练习，煅炼阅读驱动程序的“语感”，提升编写程序的能力，增加调试经验

我们还打算扩充驱动视频，把它打造成“Linux 驱动程序大全”视频，基本上都会采取从零现场编写的方式。也许有人说：在工作中我们基本上只是移植、修改驱动而已，很少从头编写。这话没错，但是能修改的前提是理解；想更好地理解，最好的方法是从零写一个出来。在学习阶段，不要怕耗费太多时间，从零开始编写，慢慢完善它，在这过程中你既理解了这个驱动，也煅炼了能力，做到触类旁通。

如果有时间，建议你学完这些所有的视频，直到你自认为：
① 给你一个新板，你可以很快实现相关驱动
② 给你一个新硬件，你可以很快给它编写/移植驱动。

我们录制的视频很多，下面只罗列到“课”，不罗列到“节”。
第 2 期视频：
第 14 课 驱动程序分层分离概念_总线驱动设备模型
第 15 课 LCD 驱动程序
第 16 课 触摸屏驱动程序
第 17 课 USB 驱动程序
第 18 课 块设备驱动程序
第 19 课 NAND FLASH 驱动程序
第 20 课 NOR FLASH 驱动程序
第 21 课 网卡驱动程序
第 22 课 移植 DM9000C 驱动程序
第 23 课 I2C 设备裸板程序
第 24 课 I2C 驱动程序 （不看此课，看第 32 课，第 32 课讲得更好）
第 26 课 声卡驱动程序 （不看此课，看第 3 期的 ALSA 驱动，那讲得更好）
第 27 课 DMA 驱动程序
第 28 课 hotplug_uevent 机制
第 32 课 3.4.2 内核下的 I2C 驱动程序
第 3 期的驱动视频：
摄像头驱动_虚拟驱动 vivi
摄像头驱动_USB 摄像头
摄像头驱动_CMOS 摄像头
WIFI 网卡驱动程序移植
3G 网卡驱动程序移植
ALSA 声卡驱动程序
学习方法：
① 再次强调，不能光看不练：一定要写程序，即使照抄也得写
② 必学：LCD、触摸屏、NAND Flash、Nor Flash、hotplug_uevent 机制
③ 学完之后，强烈建议换一个不同的开发板，尝试在新板上写驱动程序。
按视频学习会一切顺利，很多问题你可能没想到、没想通，换一个新板会让你真正掌握。

调试方法

有一种说法，程序是三分写七分调，我们从操作系统的角度提供了一些很有用的调试方法。
相关的视频有：

第 29 课第 1 节_裸板调试之点灯法_P.wmv
第 29 课第 2 节_裸板调试之串口打印及栈初步分析_P.wmv
第 29 课第 3.1 节_裸板调试之 JTAG 原理_P.wmv
第 29 课第 3.2 节_裸板调试之 JTAG 调试体验_P.wmv
第 29 课第 3.3 节_裸板调试之 JTAG 调试命令行调试_P.wmv
第 29 课第 3.4 节_裸板调试之 JTAG 调试源码级调试_P.wmv
第 30 课第 1.1 节_驱动调试之 printk 的原理_P.wmv
第 30 课第 1.2 节_驱动调试之 printk 的使用_P.wmv
第 30 课第 1.3 节_驱动调试之打印到 proc 虚拟文件_P.wmv
第 30 课第 2.1 节_驱动调试之段错误分析_根据 pc 值确定出错的代码位置_P.wmv
第 30 课第 2.2 节_驱动调试之段错误分析_根据栈信息确定函数调用过程_P.wmv
第 30 课第 3 节_驱动调试之自制工具_寄存器编辑器_P.wmv
第 30 课第 4 节_驱动调试之修改系统时钟中断定位系统僵死问题_P.wmv
第 31 课第 1 节_应用调试之使用 strace 命令跟踪系统调用_P.wmv
第 31 课第 2 节_应用调试之使用 gdb 和 gdbserver_P.wmv
第 31 课第 3 节_配置修改内核打印用户态段错误信息_P.wmv
第 31 课第 4.1 节_应用调试之自制系统调用_P.wmv
第 31 课第 4.2 节_应用调试之使用自制的系统调用_P.wmv
第 31 课第 5.1 节_应用调试之输入模拟器之设计思路_P.wmv
第 31 课第 5.2 节_应用调试之输入模拟器之编写保存功能_P.wmv
第 31 课第 5.3 节_应用调试之输入模拟器之编写测试模拟功能_P.wmv

Linux 应用程序的学习

对于大多数人来说，第 1 个 C 程序是在 Windows 的 Visual Studio C++(简称 VC)上写的，所以你们关心的也许是：嵌入式 Linux 应用程序，跟 VC 应用程序之间的区别：

① 编译方法不同：

在 VC 上点点鼠标即可编译，对于嵌入式 Linux 应用程序，我们需要“交叉编译”：程序要在 PC Linux 上编译，但是运行时要放到单板上。并且，它的编译环境需要你自己搭建：解压出工具链后设计 PATH，还要自己构造一套 Makefile 系统。

② 调试方法不同：

在 VC 上点点鼠标就可以调试，对于嵌入式 Linux 应用程序，你可以更喜欢用打印；或是在 PC Linux 上通过 GDB 观察应用程序在单板上的运行状况。

③ 可用的资源不同：

对于 VC 程序，你可以直接使用微软公司提供的各种类库；对于嵌入式 Linux 应用程序，很多时候需要去寻找、下载、编译、使用开源库。

④ 功能不同：

VC 程序运行在 PC 上，一般是用来解决某些纯软件的问题，比如整理数据、修图、联网播放音乐之类。嵌入式 Linux 应用程序一般都要操作若干种硬件，比如监控设备中要操作摄像头、存储音视频，无人机中要操作 GPS、螺旋桨，POS 机中要操作银行卡等等。它跟单板上的硬件联系很大，很多时候需要你懂点硬件知识，至少是知道怎么通过驱动程序来操作这些硬件。

上述 4 点的不同，花很少的时间就可以掌握。

如果你有志于开发应用程序，那么一定要有算法、数据结构、网络编程等基础，然后再掌握一些设计模式，最后就是多参加一些实际项目的开发了。

基于我们提供的视频，你可以这样学习：

① 先掌握第 1 期讲解的根文件系统：

在后续学习中你会经常构建根文件系统，比如往里面添加库、修改配置文件让你的程序自动运行。

② 掌握怎么编译、烧写 u-boot、内核：

在实际工作中，一般来说不需要你去烧写 u-boot、内核，但是在自学阶段还是自己掌握吧，免得去麻烦别人。按开发板手册即可操作，你甚至不用管里面的原理。

③ 掌握 Makefile：

可以看如下第 3 期视频，以后编译程序时只要执行 make 命令即可：

第 1 课第 4 节_数码相框_编写通用的 Makefile

④ 学习第 1 个项目：数码相框

该项目不使用任何开源的 GUI 项目，完全是自己构建一套 GUI 系统，实现了文件浏览、文件显示(文本和图片)、图片操作(放大、缩小、自动播放)等功能；涉及网络编程、多线程编程、开源库使用等等。

虽然数码相框作为一个产品已经落伍了，但是该项目所涉及的技术，特别是以面向对象的编程思想设计出一个模块化的、易扩展的系统，非常适合没有大型项目开发经验的人。很多同学，都是根据该项目所教会的编程思想找到了心怡的工作。

第 3 期视频取名为“项目开发”，而非“应用开发”，它的第 2、3 个项目跟内核、驱动耦合很大。如果只关心应用开发，或是急于找一份工作，可以先看第 1 个项目。

第 2 个项目涉及摄像头、ALSA 声卡、WIFI 网卡、3G 网卡，这些都是在实际工作过程中经常用到的设备，比如我们后面补充的 QQ 物联就用到摄像头、声卡、WIFI 网卡。
第 3 个项目是电源管理，讲解怎么讲你的单板休眠以省电。

怎么学习整个 Android 系统

待续...等录完安卓系统深度开发视频再细谈。

微信公众号：baiwenkeji
邮箱：weidongshan@qq.com
网站/论坛：www.100ask.net
www.100ask.org