实验 01 内核开发基础

1.1 本章导读

本实验将带您学习一遍 Linux 的框架和源码目录结构。

从任何地方拿到的 Linux 源码，都有几百 M 大小，包含上万个文件。

这么多的文件！那么问题来了，应该从什么地方入手呢？

哪些内容应该“深入研究”？

哪些内容应该“惊鸿一瞥”？

哪些内容应该“束之高阁”？

本期实验“内核开发基础”,带大家快速梳理一遍，把和学习无关的内容剔除掉。

1.1.1 工具

1.1.1.1 硬件工具

PC 机一台

1.1.1.2 软件工具

软件 Source Insight

Linux 源码“iTop4412_Kernel_3.0_xxx”（在光盘目录“/Android 源码”文件夹下,xxx 表示日期）

1.1.2 预备课程

视频教程“01-烧写、编译以及基础知识视频”→“实验 12-使用 Source Insight 加载和阅读内核源码”

使用手册“3.5 Source Insight 的安装和使用”

1.1.3 视频资源

本节配套视频为“视频 01_内核开发基础”

1.2 学习目标

本章需要学习以下内容：

理解 Linux 体系结构

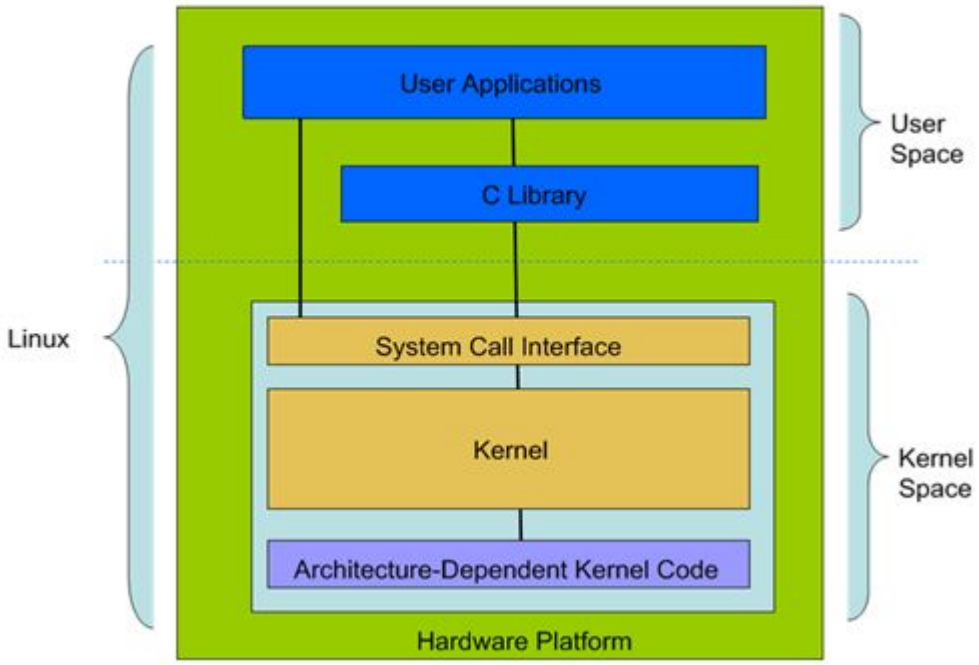
了解 Linux 内核结构

了解 Linux 内核源码目录结构

了解学习 Linux 的大方向→驱动

1.3 Linux 体系结构

如下图所示，Linux 体系结构，从大的方面可以分为用户空间（User Space）和内核空间（Kernel Space）。



用户空间中包含了 C 库，用户的应用程序。在某些体系结构图中还包含了 shell，当然 shell 脚本也是 Linux 体系中不可缺少的一部分。

内核空间包括硬件平台、平台依赖代码、内核、系统调用接口。

在任何一个现代操作系统中，都是分层的。为什么需要分层呢？

从程序员的角度分析，将 linux 底层和和应用分开，将 linux 底层和应用分开，做应用的做应用，做底层的做底层，各干各的。经济学的基本原理是，分工产生效率。

从安全性的角度分析，是为了保护内核。现代 CPU 通常都实现了不同的工作模式。

以 ARM 为例：ARM 实现了 7 种工作模式，不同模式下 CPU 可以执行的指令或者访问的寄存器不同： (1)用户模式 usr (2)系统模式 sys(3)管理模式 svc(4)快速中断 fiq(5)外部中断 irq(6)数据访问终止 abt(7)未定义指令异常。如果任何一个上层应用都可以调用都可以调用寄

存器，那样肯定是无法稳定执行的。而且因为出现了这个问题，出现了一个新的学科“现代操作系统”，如果大家感兴趣可以看一下“现代操作系统”相关文章或者书籍。

以 X86 为例：X86 实现了 4 个不同级别的权限，Ring0—Ring3；Ring0 下可以执行特权指令，可以访问 IO 设备；Ring3 则有很多的限制

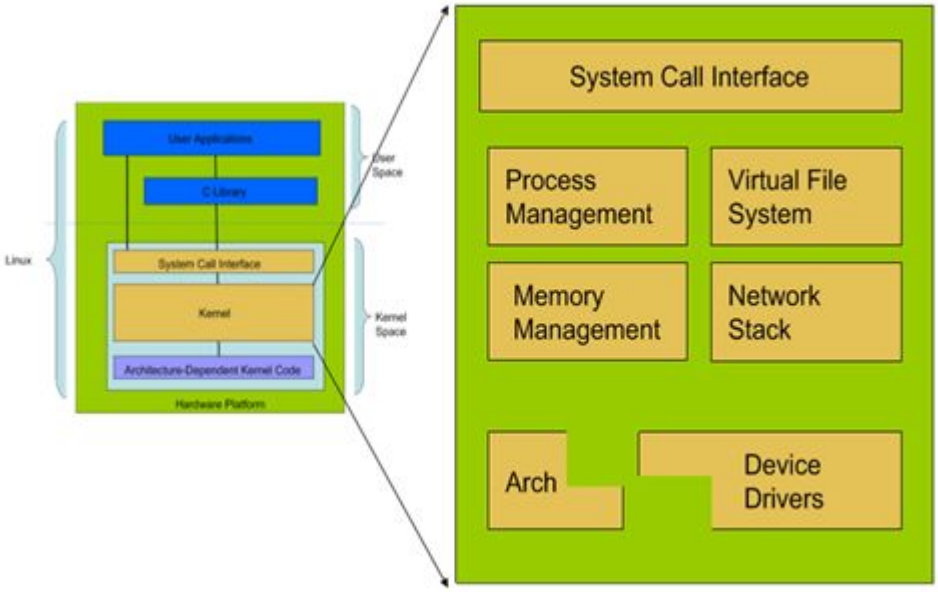
如果分析一下 Android 的，这方面做的更加“丧心病狂”，Android 所有的 APK 应用程序，都是在 Java 虚拟机上面运行，应用程序更加远离底层。

另外，用户空间和内核空间是程序执行的两种不同状态，可以通过“系统调用”和“硬件中断”来完成用户空间到内核空间的转移。

1.4 Linux 内核结构

这一节，分析一下内核结构。

如下图所示，是 Linux 内核结构图。

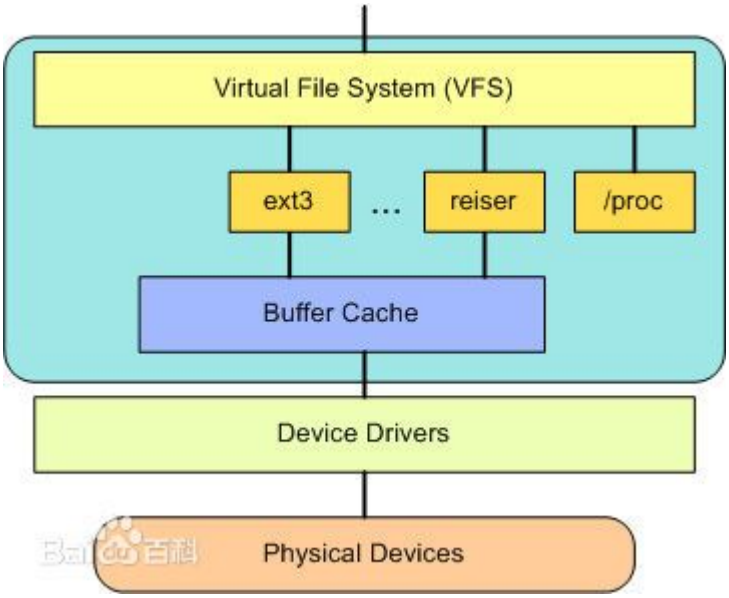


SCI 层（System Call Interface），这一层是给应用用户空间提供一套标准的系统调用函数来访问 Linux。前面分析 Linux 体系结构的时候，介绍过任何一类现代操作系统都不会允许上层应用直接访问底层，在 Linux 中，内核提供了一套标准接口，上层应用就可以通过这一套标准接口来访问底层。

PM（Procees Management），这一部分包括具体创建创建进程（fork、exec）,停止进程（kill、exit）,并控制他们之间的通信（signal 等）。还包括进程调度，控制活动进程如何共享 CPU。这一部分是 Linux 已经做好的，在写驱动的时候，只需要调用对应的函数即可实现这些功能，例如创建进程、进程通信等等。

MM（Memory Management），内存管理的主要作用是控制多个进程安全的共享内存区域。

VFS（Virtual File Systems），虚拟文件系统，隐藏各种文件系统的具体细节，为文件操作提供统一的接口。在 Linux 中“一切皆文件”，这些文件就是通过 VFS 来实现的。Linux 提供了一个大的通用模型，使这个模型包含了所有文件系统功能的集合。如下图所示，是一个虚拟文件系统的结构图。



Device Drivers 设备驱动，这一部分就是需要学习和掌握的。Linux 内核中有大量的代码在设备驱动程序部分，用于控制特定的硬件设备。

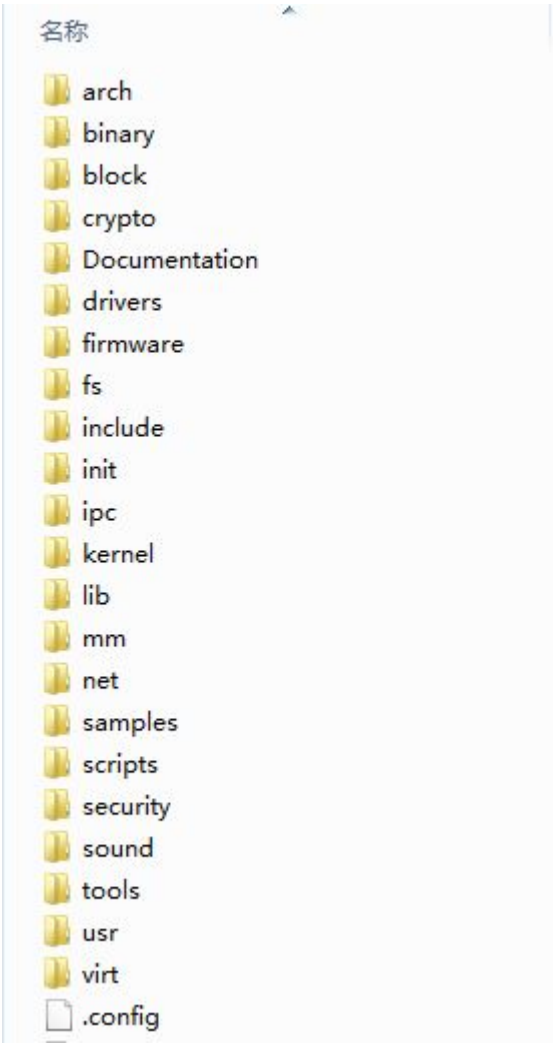
Linux 驱动一般分为网络设备、块设备、字符设备、杂项设备，需要编写的只有字符设备，杂项设备是不容易归类的一种驱动，杂项设备和字符设备有很多重合的地方。

网络协议栈，Linux 内核中提供了丰富的网络协议实现。

1.5 Linux 内核源码目录结构

Linux 内核源码采用树形结构。功能相关的文件放到不同的子目录下面，使程序更具有可读行。

使用 Source Insight 打开源码，如下图所示，可以看到源码是树形结构。



下面来介绍每一个目录的作用。

arch 目录是平台目录。处理器原厂提供一套 Linux 内核的源码，那么在这个目录下都有一套针对具体处理器 CPU 的子目录。每个 CPU 的子目录，又进一步分解为 boot，mm，kernel 等子目录，分别控制系统引导，内存管理，系统调用，动态调频，主频率设置部分等。

在 arch 目录中有关键的平台文件。任何一款支持 Linux 的处理器，都有一部分内核代码是针对特定的处理器来提供的，具体的实现就是通过平台文件。

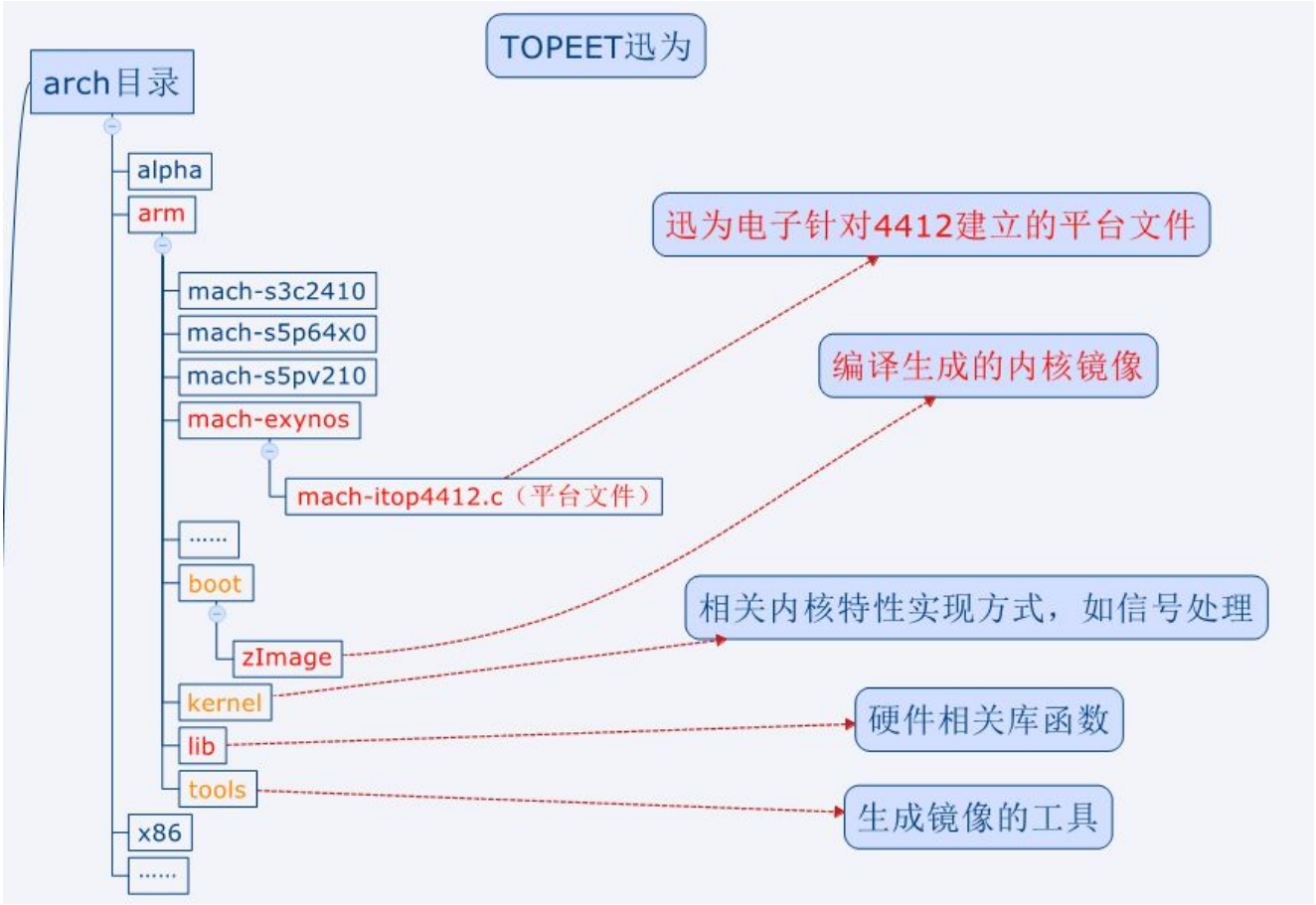
迅为 iTOP-4412 的平台文件，是 arch→arm→mach-exynos→mach-itop4412.c。

arch→arm→boot 目录，默认编译生成的内核镜像是在这个目录下。

在 arch→arm→kernel 目录中，有针对具体 CPU 处理器的代码，有相关内核特性实现方式，如信号处理等。这一部分当然是芯片厂商做好了，4412 的这部分就是三星已经做好的部分。

在 arch→arm→lib 目录中，有一些和硬件相关库函数，后面学习驱动的时候会使用到。

在 arch→arm→tools 目录中，包含了生成镜像的工具。

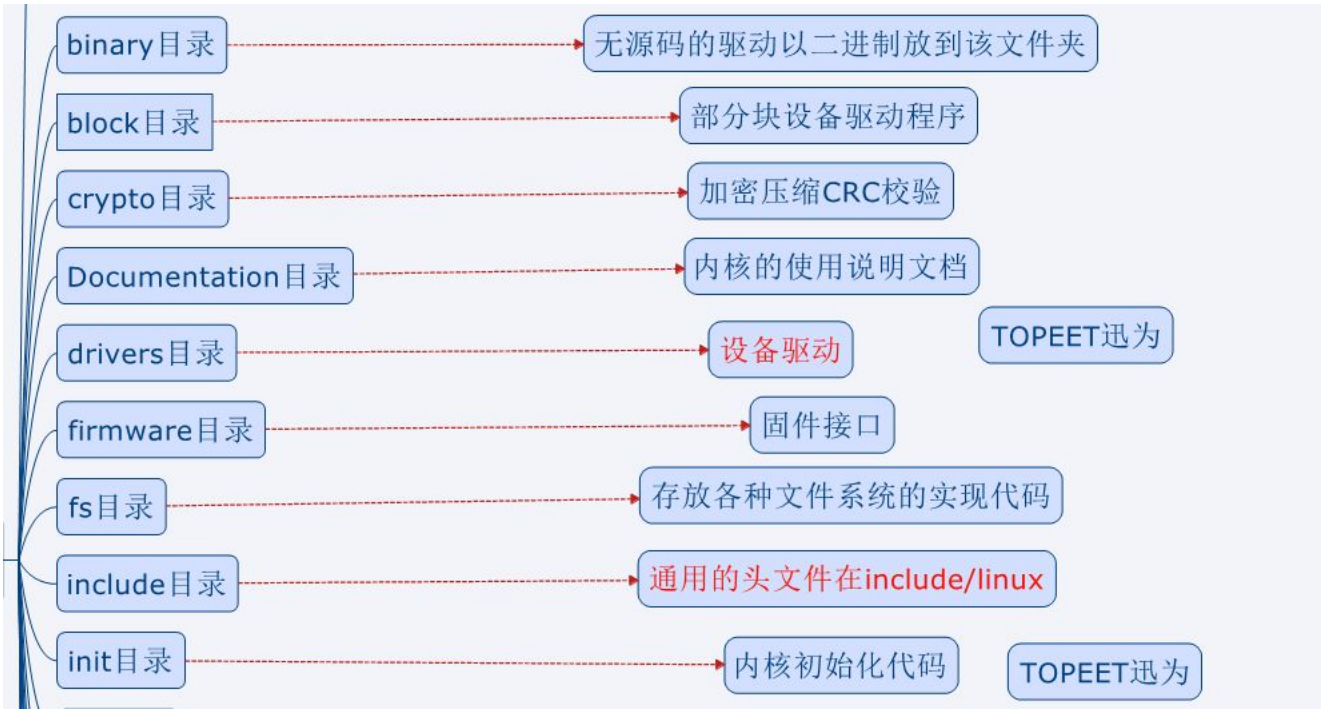


如下图所示。

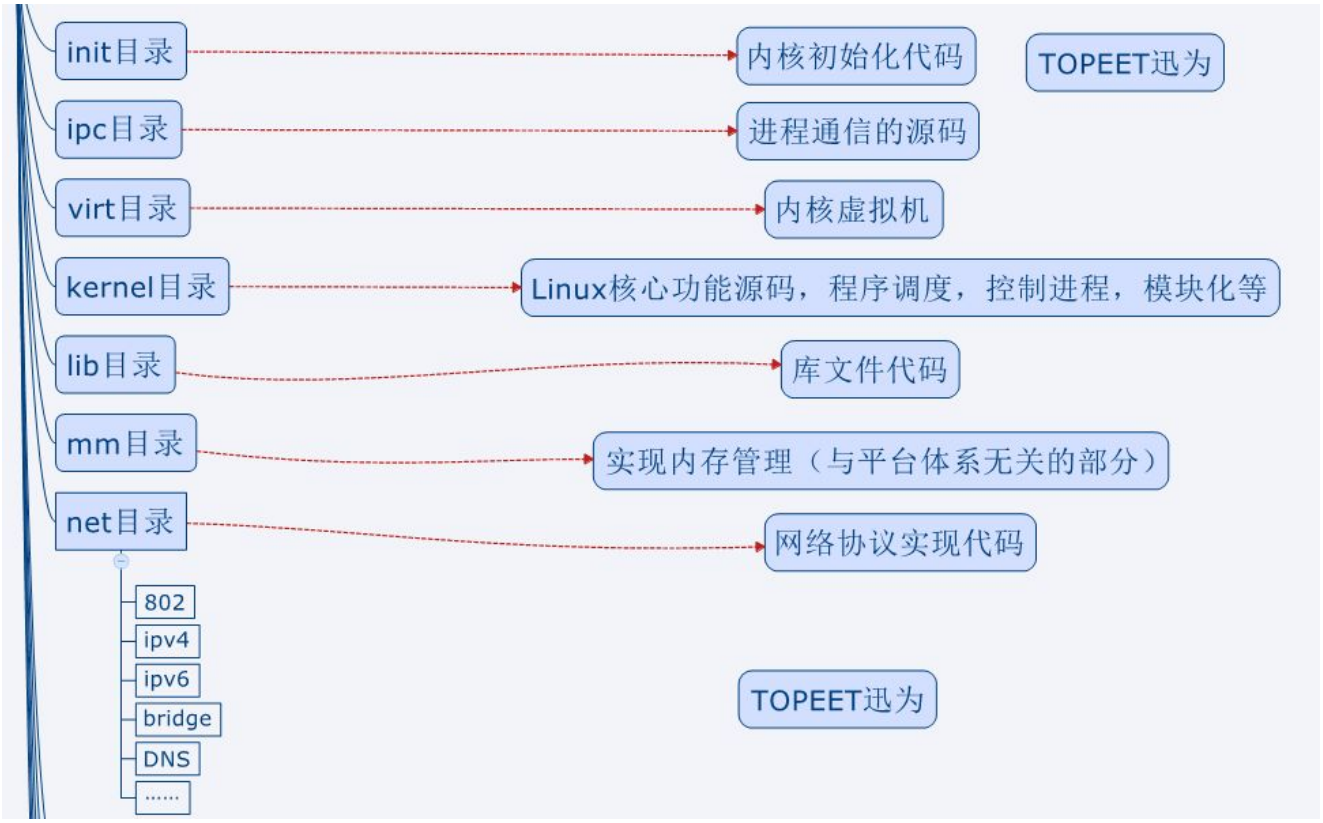
在 binary 目录中，有一些无源码的驱动以二进制放到该文件夹，例如一些测试版本或者不愿意公布源码，都可以将二进制文件放到这个目录中。

在 drivers 目录中，就是需要重点学习的部分，后面的实验都是围绕这一步进行的。

在 include 目录中，通用的 Linux 头文件都在该文件下。

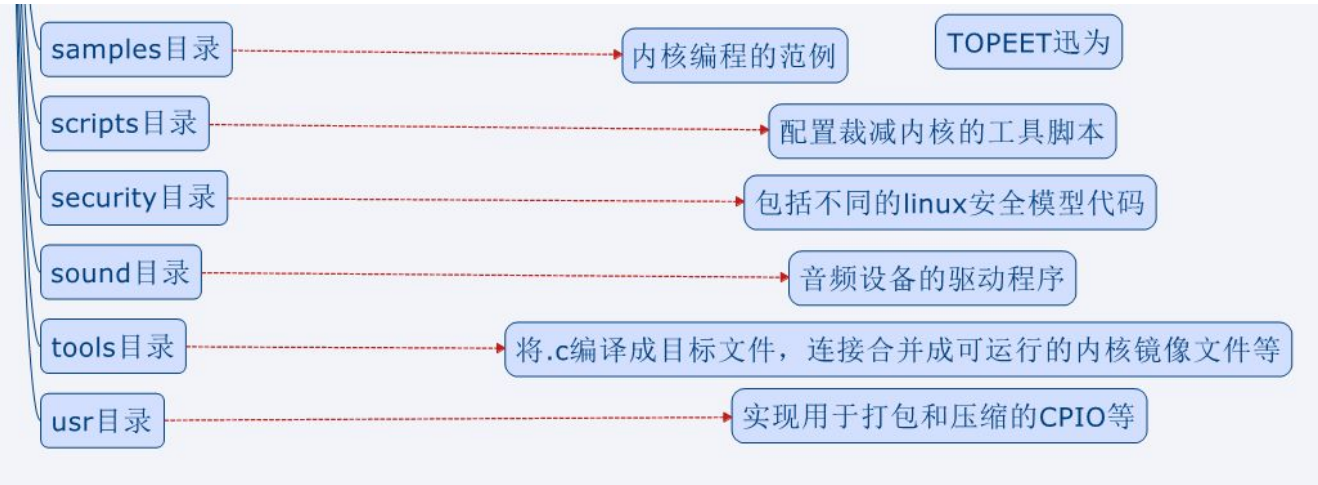


如下图所示，部分目录如下。下面的这些目录，几乎不需要去动其中任何一个文件。



如下图所示，有内核编程的范例，实现安全性的代码，声卡设备驱动等

还有内核裁减配置工具目录 tools，这一部分实现的功能是将.c 编译成目标文件，连接合并成可运行的内核镜像文件等。提供给大家的内核源码一百多 M，最后编译成的 zImage 只有不到 5M，这都是依靠这个工具来实现的，后面会有针对性的实验来教大家如何使用编译工具。



1.6 学习方法介绍

从事 Linux 相关工作的人员中，可以分为以下几类。

1）linux 运维工程师。Linux 服务器运用很广泛，需要一些工程师在后台维护，只需要掌握基本的电脑知识，然后熟悉服务器操作 shell 命令、网络知识就可以胜任。

2）linux 系统工程师。各种软件的安装、服务器的配置、常见故障排除等

3）Linux 应用工程师。实现应用程序，在 linux 系统中编写应用程序，需要很强的 C 语言功底和编程能力。

4）Linux 嵌入式驱动工程师。相比其他种类的 Linux 工程师，Linux 嵌入式驱动工程师需要掌握各种各样的知识，包括“基础的硬件知识”“测量工具万用表示波器等等”“Linux 嵌入式驱动”“Linux 应用的编写”（对于应用掌握的最低要求是能写驱动的测试程序）。

在嵌入式范围内，学习 Linux 可以分为应用和驱动，这个教程的重点不在应用而在驱动，在需要用到应用程序测试的时候，也会介绍如何编写简单的测试代码。

这里先提醒学习者一下，广义的说法驱动是属于内核的，但是不要以为“内核就是驱动”，那样的就犯了“白马非马”的错误。大家不要觉得这问题说的很“幼稚”，作者在和初学者接触的过程中，大量的初学者上来直接问一些内核源码的问题。不可否认，阅读内核源码是提高能力的非常好的一种方式，但是对于初学者来说，如果你一套扎进内核，整天研究“Linux 是怎么启动”，“linux 是怎么引导到内存”，“超级终端打印是怎么实现”等等类似的问题，那么你将在很长一段时间内无法进步。

这里郑重提醒一下，在 Linux 内核中，驱动只是其中非常非常小的一部分，而且在这“非常小的一部分”中，只有极其小的一部分是需要自己写的，这就是字符类驱动。其它的驱动全部都是移植的，而且字符驱动中很大一部分也是移植的。

我们的目标不是“星辰大海”，而是“溺水三千我只取一瓢”，这“一瓢”虽然是很容易掌握，但是要到灵活应用却不是那么容易的。

这里大家要做好心理准备，Linux 不是“单片机”，单片机号称可以 15 天学会（其实不是号称，学习十五天真就能干点活了，本人刚参加嵌入式方面的工作前，就是搞了一周单片机，然后就找了家公司去实习了）。

如果真想踏踏实实的学习 Linux，以后想从事着方面的工作，最少要准备三个月到半年，很可能有些学习者学习到这里的时候，已经都过了三个月了，不过大家不要担心，只要坚持就可以学会的。

如果真要说一个简单的学习方法，那么就是一个问题一个问题的“攻克”，教程中的代码大家好好研究，教程中的内容搞得差不多了，然后在网上找一些相关文章来看，一步一个脚印，夯实了基础，后面的学习就会“顺风顺水”，找到 Linux 相关的工作也不是难事。