

关于Linux系统内核源代码分析经验谈 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/145/2021_2022__E5_85_B3_E4_BA_8E_Linu_c103_145384.htm Linux的最大的好处之一就是它的源码公开。同时，公开的核心源码也吸引着无数的电脑爱好者和程序员；他们把解读和分析Linux的核心源码作为自己的最大兴趣，把修改Linux源码和改造Linux系统作为自己对计算机技术追求的最大目标。Linux内核源码是很具吸引力的，特别是当你弄懂了一个分析了好久都没搞懂的问题；或者是被你修改过了的内核，顺利通过编译，一切运行正常的时候。那种成就感真是油然而生！而且，对内核的分析，除了出自对技术的狂热追求之外，这种令人生畏的劳动所带来的回报也是非常令人着迷的，这也正是它拥有众多追随者的主要原因：首先，你可以从中学到很多的计算机的底层知识，如后面将讲到的系统的引导和硬件提供的中断机制等；其它，象虚拟存储的实现机制，多任务机制，系统保护机制等等，这些都是非都源码不能体会的。同时，你还将从操作系统的整体结构中，体会整体设计在软件设计中的份量和作用，以及一些宏观设计的方法和技巧：Linux的内核为上层应用提供一个与具体硬件不相关的平台；同时在内核内部，它又把代码分为与体系结构和硬件相关的部分，和可移植的部分；再例如，Linux虽然不是微内核的，但他把大部分的设备驱动处理成相对独立的内核模块，这样减小了内核运行的开销，增强了内核代码的模块独立性。而且你还能从对内核源码的分析中，体会到它在解决某个具体细节问题时，方法的巧妙：如后面将分析到了的Linux通过Botoom_half机制来加快系统对

中断的处理。最重要的是：在源码的分析过程中，你将会被一点一点地、潜移默化地专业化。一个专业的程序员，总是把代码的清晰性，兼容性，可移植性放在很重要的位置。他们总是通过定义大量的宏，来增强代码的清晰度和可读性，而又不增加编译后的代码长度和代码的运行效率；他们总是在编码的同时，就考虑到了以后的代码维护和升级。甚至，只要分析百分之一的代码后，你就会深刻地体会到，什么样的代码才是一个专业的程序员写的，什么样的代码是一个业余爱好者写的。而这一点是任何没有真正分析过标准代码的人都无法体会到的。然而，由于内核代码的冗长，和内核体系结构的庞杂，所以分析内核也是一个很艰难，很需要毅力的事；在缺乏指导和交流的情况下，尤其如此。只有方法正确，才能事半功倍。正是基于这种考虑，作者希望通过此文能给大家一些借鉴和启迪。由于本人所进行的分析都是基于2.2.5版本的内核；所以，如果没有特别说明，以下分析都是基于i386单处理器的2.2.5版本的Linux内核。所有源文件均是相对于目录/usr/src/linux的。要分析Linux内核源码，首先必须找到各个模块的位置，也即要弄清源码的文件组织形式。虽然对于有经验的高手而言，这个不是很难；但对于很多初级的Linux爱好者，和那些对源码分析很有兴趣但接触不多的人来说，这还是很有必要的。

- 1、Linux核心源程序通常都安装在/usr/src/linux下，而且它有一个非常简单的编号约定：任何偶数的核心（的第二个数为偶数，例如2.0.30）都是一个稳定地发行的核心，而任何奇数的核心（例如2.1.42）都是一个开发中的核心。
- 2、核心源程序的文件按树形结构进行组织，在源程序树的最上层，即目录/usr/src/linux下有这样一些目

录和文件。 COPYING: GPL版权申明。对具有GPL版权的源代码改动而形成的程序，或使用GPL工具产生的程序，具有使用GPL发表的义务，如公开源代码。 CREDITS: 光荣榜。对Linux做出过很大贡献的一些人的信息。

MAINTAINERS: 维护人员列表，对当前版本的内核各部分都有谁负责。 Makefile: 第一个Makefile文件。用来组织内核的各模块，记录了个模块间的相互这间的联系和依托关系，编译时使用；仔细阅读各子目录下的Makefile文件对弄清各个文件这间的联系和依托关系很有帮助。 ReadMe: 核心及其编译配置方法简单介绍。 Rules.make: 各种Makefilemake所使用的一些共同规则。 REPORTING-BUGS: 有关报告Bug 的一些内容。

- Arch/ : arch子目录包括了所有和体系结构相关的核心代码。它的每一个子目录都代表一种支持的体系结构，例如i386就是关于intel cpu及与之相兼容体系结构的子目录。PC机一般都基于此目录；
- Include/: include子目录包括编译核心所需要的大部分头文件。与平台无关的头文件在include/linux子目录下，与intel cpu相关的头文件在include/asm-i386子目录下,而include/scsi目录则是有关scsi设备的头文件目录。
- Init/ : 这个目录包含核心的初始化代码(注：不是系统的引导代码)，包含两个文件main.c和Version.c，这是研究核心如何工作的好的起点之一。
- Mm/ : 这个目录包括所有独立于cpu体系结构的内存管理代码，如页式存储管理内存的分配和释放等；而和体系结构相关的内存管理代码则位于arch/*/mm/，例如arch/i386/mm/Fault.c。
- Kernel/ : 主要的核心代码，此目录下的文件实现了大多数linux系统的内核函数，其中最重要

的文件当属sched.c；同样，和体系结构相关的代码在arch/*/kernel中。

- Drivers/：放置系统所有的设备驱动程序；每种驱动程序又各占用一个子目录：如，/block 下为块设备驱动程序，比如ide（ide.c）。如果你希望查看所有可能包含文件系统的设备是如何初始化的，你可以看drivers/block/genhd.c中的 device_setup()。它不仅初始化硬盘，也初始化网络，因为安装nfs文件系统的时候需要网络。
- Documentation/：文档目录,没有内核代码，只是一套有用的文档，可惜都是English的，看看应该有用的哦。
- Fs/：所有的文件系统代码和各种类型的文件操作代码，它的每一个子目录支持一个文件系统,例如fat和ext2。
- lpc/：这个目录包含核心的进程间通讯的代码。
- Lib/：放置核心的库代码。
- Net/：核心与网络相关的代码。
- Modules/：模块文件目录，是个空目录，用于存放编译时产生的模块目标文件。
- Scripts/：描述文件，脚本，用于对核心的配置。一般，在每个子目录下，都有一个 Makefile 和一个Readme 文件，仔细阅读这两个文件，对内核源码的理解很有用。对Linux内核源码的分析，有几个很好的入口点：一个就是系统的引导和初始化，即从机器加电到系统核心的运行；另外一个就是系统调用，系统调用是用户程序或操作调用核心所提供的功能的接口。对于那些对硬件比较熟悉的爱好者，从系统的引导入手进行分析，可能来的容易一些；而从系统调用下口，则可能更合适于那些在dos或Unix、Linux下有过C编程经验的高手。

100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com