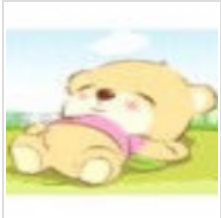
zhangxinrun的专栏

个人资料



zhangxinrun_业余erlang

访问： 2889383次
积分： 28673
等级： BLOG 7
排名： 第146名

原创： 75篇 转载： 1101篇
译文： 0篇 评论： 176条

如何成为一个Linux内核开发者（经典）

标签： linux内核 文档 工作 linux email postscript

2010-09-28 15:53 7280人阅读 评论(1) 收藏 举报

分类： 没有读的文章（85） linux内核（68）

这篇文章将是这个话题的最权威的文档。它将教你如何成为一个Linux内核开发者以及学会如何和Linux内核社区一起工作。它不包含任何有关内核编程的技术细节，但是会帮你在这方面指明方向。
如果这篇文档里任何部分已经过时，请把更新信息以补丁的形式发送给本文的维护者，他的联系方式列在本文档的末尾。
介绍
好了，你想成知道如何成为一个Linux内核开发者么？或者你的老板告诉你，“去为这个设备写一个Linux驱动。“这篇文档的目的，就是通过描述你需要经历的过程和提示你如何和社区一起工作，来教给你为达到这些目的所需要知道的所有知识。本文也尝试解释社区为什么这样工作的一些原因。
内核几乎全是用C写成的，有一些架构相关的部分是用汇编语言写成的。熟练掌握C语言是内核开发的必备条件。汇编语言（任何架构）的了解不是必须的，除非你准备做某个架构的底层开发。虽然下面这些书不能完全代替扎实的C语言教学和/或者成年累月的经验，他们还是不错的参考，如果用得着的话：
- "The C Programming Language" 作者: Kernighan and Ritchie [Prentice Hall]
- "Practical C Programming" 作者: Steve Oualline [O'Reilly]
内核是用 GNU C 和 GNU 工具链写成的。虽然它符合 ISO C89 标准，它还是使用了一些标准中没有的扩展。内核是自成体系的 C 环境，它并不依赖标准C库，所以某些C语言标准是不支持的。任意长度long long类型除法和浮点数是不被允许的。有时候会很难理解内核对于它所使用的工具链和扩展的假定，而且不幸的是也没有关于它们的绝对的参考。请查阅gcc的info页(`info gcc`)以获取有关信息。
请记住你是在尝试学习如何与已经存在的开发社区一起工作。这是一群成分复杂的人们，他们对于代码，风格和步骤有高的标准。这些标准是经过时间检验的。
他们发现遵循这些标准对于这样一个大规模的且地理上分散的团队是最佳的选择。尝试提前学习尽可能多的有关这些标准的知识，因为它们都有很好的文档；不要期望别人会遵照你或者你公司的行事方式。
法律问题
Linux内核源代码依照GPL发布。请参考源代码树下的COPYING文件，以获取有关这个许可证的详细信息。如果你对这个许可证有疑问，请联系你的律师，不要在Linux内核邮件列表里询问。邮件列表里的人们不是律师，你不应该依赖于他们对于法律问题的解释。
欲了解有关GPL的常见问题和答案，请看：
http://www.gnu.org/licenses/gpl-faq.html

文档
Linux内核源代码树有很多文档，它们对于学习如何与内核社区交流来说有不可估量的价值。当新的功能加进内核的时候，通常建议作者把解释这个新功能的文档也加进内核。如果一个内核变动导致了内核对用户空间界面的改变，建议你把这个信息或者一个解释了这个变动的manpage的补丁发送给手册页的维护者mtk-manpages@gmx.NET。
这里有一个内核源代码树里需要阅读的文件列表：
README
这个文件简单介绍了Linux内核的背景，并描述了配置和编译内核需要做哪些事情。内核新手应该从这里开始。
Do*****entation/Changes
这个文件介绍了成功编译和运行内核所需要各种不同软件的列表。
Do*****entation/CodingStyle
这个文件描述了Linux内核代码风格，还有背后的一些原因。所有的新代码的要符合这个文档里的准则。大多数维护者只会接受符合这些规则的补丁，很多人只看符合正确风格的代码。
Do*****entation/SubmittingPatches

VC++ DLL (1)
windows (18)
云计算 (12)
互联网 (14)
压缩解压 (1)
图像技术 (1)
字符集 (5)
学习书籍 (2)
定时器 (3)
工具介绍 (2)
开源库 (5)
技术类（杂） (25)
搜索引擎 (9)
数据库 (37)
新闻记录 (3)
服务配置 (8)
汇编 (32)
没有读的文章 (86)
测试 (1)
版本控制 (8)
生活常识 (12)
破解 (17)
算法 (9)
网络协议 (108)
网页技术 (13)
职场 (15)
英语 (4)
遗留没有试验的问题 (2)
问题篇 (2)
面试题以及经验总结 (29)
黑客技术 (16)
SNMP (8)
胃病 (11)
广告知识 (9)
需要学习的知识 (5)
电脑组成 (19)
手机 (4)
海量存储和算法 (21)
memcache (1)
nosql (16)
负载均衡和高可靠性 (2)
Hadoop (6)
数据挖掘-机器学习-推荐 (6)
python (35)
boost (1)
IM (5)
php (0)
javascript (2)
golang (1)
nginx (6)
zookeeper (3)
物联网 (1)

文章存档

2016年10月 (1)
2016年09月 (1)
2016年06月 (4)

Do*****entation/SubmittingDrivers
这些文件非常详细的介绍了如何成功的创建和发送一个补丁，包括（但不限于）：
－Email内容
－Email格式
－发送给谁
遵守所有这些规则并不能保证成功（对所有的补丁都需要进行内容和风格的详细检查），但是不遵守这些规则就一定不会成功。
其他关于如何创建补丁的很好的文章有：
“The Perfect Patch”
http://www.zip.com.au/~akpm/linux/patches/stuff/tpp.txt
"Linux kernle patch submission format"
http://linux.yyz.us/patch-format.html
Do*****entation/stable_api_nonsense.txt
这个文件解释了有意识的决定－不在内核里使用稳定的API－的原因，包括：
－子系统分隔层（为了兼容？）
－操作系统之间的驱动可移植性
－缓和（或者阻止）内核源代码树的急速变动
这个文档对于了解Linux的开发哲学是非常关键的，对于由开发其他操作系统转而开发Linux人也是很重要的。
Do*****entation/SecurityBugs
如果你感觉到你发现了Linux内核里的一个安全问题，请遵照这个文档里所描述的步骤来提醒内核开发者，并帮助解决问题。
Do*****entation/ManagementStyle
这个文档描述了Linux内核维护者如何运作，以及他们为什么这样做。它对于任何内核开发新手（或者任何对本话题感兴趣的人）来说是非常重要的。
因为它解释了一些惯有的错误概念，可解决有关内核维护者独特行为的疑惑。
Do*****entation/stable_kernel_rules.txt
本文件描述了稳定版本内核释出的规则，还有如果你想对其中的一个版本做一些改动应该做些什么。
Do*****entation/kernel-docs.txt
一个有关内核开发的外部文档的列表。如果你在内核内部文档里没有找到？要找的东西，你可以参考这个列表。
Do*****entation/applying-patches.txt
介绍了对于什么是补丁，以及如何应用补丁于不同的内核开发分支。
内核也有很多可以从源代码自动产生的文档。这包括内核内部API的全面描述，有关如何处理好锁定的规则。这些文档会被创建于Do*****entation/DocBook/文件夹中。在内核主源码树中通过运行下面的命令可以创建出PDF，Postscript， HTML和manpage等不同格式的文档：
make pdfdocs
make psdocs
make htmldocs
make mandocs
成为一个内核开发者
如果你对Linux内核开发一无所知，你可以看看Linux KernelNewbies项目：
http://kernelnewbies.org
它包含一个邮件列表，在那里你可以问任何有关内核开发的基础问题（在问问题之前先搜索一下存档，很可能这个问题已经被解答过了。）它还有一个IRC频道，你可以在里面实时的提问。它还有很多有用的文档，对于学习Linux内核开发很有用。
这个网站有有关代码组织，子系统，当前项目（代码树之内的和之外的）的基本信息。它也描述了一些基本的“物流”信息，比如怎么样编译内核和怎么样打补丁。
如果你不知道从何处起步，但是你想找一些任务来做以加入内核开发社区，请看一下Linux Kernel Janitor项目：
http://janitor.kernelnewbies.org/
这是一个很好的起步的地方。它描述了一些相对来说简单的内核中需要清理的和解决的问题。和负责这个项目的开发者一起工作，你会学到如何令你的补丁进入Linux内核树的基本知识，而且可能会为你指明下一步的发展方向，如果你自己尚不明确的话。
如果你已经有了一段代码想要放到内核树里，但是需要某种形式的帮助，那么kernel-mentors项目就可以帮你的忙了。这是一个邮件列表，可以在下面找到：
http://selenic.com/mailman/listinfo/kernel-mentors
在你对Linux内核代码作任何实际的改动之前，必须要了解相关的代码是如何工作的。为了达到这个目的，没有比直接读它（很多困难的地方都有很好的注释）更好的方法了，甚至可能是在某个特殊工具的帮助下来阅读。很值得推荐的这样一种工具是Linux Cross-Reference项目，它可以把源代码以一种自我引用的、索引的网页形式显示出来。一个非常好的最新的内核代码仓库可以在这里找到： http://sosdg.org/~coywolf/lxr

开发流程
Linux内核开发流程当前包括一些主内核分支，和很多不同的子系统专有的内核分支。它们是：
－ 主 2.6.x 内核树

2016年05月 (3)
2016年04月 (2)
展开

阅读排行
Django模板系统(非常详细 (102428)
linux下启动和关闭网卡命 (75992)
（经典）tcp粘包分析 (34086)
推荐几个开源web自动化 (31923)
cookie的expires属性和m (30166)
ethtool 命令详解 (29437)
TPC,TPCC,TPMC(计算机 (27513)
android配置引用第三方库 (27503)
C++中placement new操作 (25864)
关于Python中以字母r/R, (25237)

评论排行
C++中placement new操作 (12)
（经典）tcp粘包分析 (9)
c语言宏定义 (6)
结构体struct的自然对齐问 (5)
Django模板系统(非常详细 (5)
frameset和div常规布局的 (5)
解决Linux和SecureCRT_ (4)
android配置引用第三方库 (4)
漫谈C语言及如何学习C语 (4)
阿里云面试总结 (4)

推荐文章
* Android 反编译初探 应用是如何被注入广告的
* 凭兴趣求职80%会失败，为什么
* 安卓微信自动抢红包插件优化和实现
* 【游戏设计模式】之四 《游戏编程模式》全书内容提炼总结
* 带你开发一款给Apk中自动注入代码工具icodetools(完善篇)

最新评论
关于Python中以字母r/R，或字母
qq_32804891: 谢谢楼主
C++中placement new操作符（经
piplu: 在C++标准中，对于placement operator new []有如下的说明： placeme...
RTSP协议、RTMP协议、HTTP
l_fish: 谢谢文章，不过上面区别那里，不知是复制后忘了修改，请参考1：HTTP: 即超文本传送协议(ftp即文...
RTSP协议、RTMP协议、HTTP
灿哥哥: 谢谢分享
实现自动登录Cookie

— 2.6.x.y -stable 内核树
— 2.6.x -Git 内核补丁
— 2.6.x -mm 内核补丁
— 子系统专有内核树和补丁

2.6.x 内核树
2.6.x 内核树是有Linus Torvalds维护的，可以在kernel.org的pub/linux/kernel/v2.6目录里找到。它的开发流程是这样的：
— 当一个新的内核发布之后，一个为期两个星期的窗口打开，在这段时间里维护者可以提交大的补丁给Linus，通常是已经在-mm内核中存在了一定时间的补丁。推荐的提交补丁的方式是通过git（有关git的更多信息可以在http://git.or.cz/找到），但是普通的补丁也是可以的—两个星期之后一个-rc1内核发布，然后现在只可以再加入不会为内核添加新功能的补丁，因为那样的补丁可能会影响这个内核的稳定性。请注意这个时候一个整的新驱动（或者文件系统）可以被接受。因为只要这个变动是自成一体的并且不影响它之外的代码的话，就不会有产生回归的危险。在-rc1发布之后，git可以用来发送补丁给Linus，但是这些补丁也需要发到一个公开的邮件列表里以备审查。
— 当Linus确信当前的git（内核代码管理工具）树已经处于一个合理的健全状态，足够测试时，一个新的-rc就会发布了。目标是每周发布一个新的-rc内核。
— 这个过程将会持续到内核被认为可以发布为止，整个流程会持续大概6个星期。
Andrew Morton在linux-kernel邮件列表里写的有关内核发布的一句话值得提一下： “没有人知道什么时候一个内核会发布，因为它发布的依据已经掌握的bug状态，而不是事先设想好的一个时间线。

2.6.x.y -stable 内核树
有四个数字版本号的内核是-stable内核。他们包含一些相对较小的和重要的修正。这些修正针对的是在一个给定2.6.x内核中发现的安全问题或者重大的回归。
对于想使用最新的稳定内核并且对于帮助测试开发/实验版本不感兴趣的用户，这是推荐使用的版本。
如果没有2.6.x.y版本，那么最高版本号的2.6.x内核是当前稳定内核。
2.6.x.y由“stable”团队维护，每周发布一次。

内核树里的文件Do*****entation/stable_kernel_rules.txt描述了什么样的改动可以被-stable树所接受，以及发布流程是怎样工作的。

2.6.x -git 补丁
这些是在git仓库里管理的Linus内核树的每日快照。这些补丁每天发布一次，代表Linus树的当前状态。它们比-rc内核更具实验性质，因为它们是自动生成的，以至没有人曾经瞟上一眼来检查它们是否处于健全状态。

2.6.x -mm 内核补丁
这些是Andrew Morton发布的实验性质的内核补丁。Andrew取得所有不同子系统的内核树和补丁，连同从linux-kernel邮件列表里拉过来的补丁，把它们融合在一起。这个树是新功能和补丁证明自己的场所。如果一个补丁在-mm里证实了自己的价值，Andrew或者子系统维护者就会把它提交给Linus，以求被收录于主线内核中。
强烈建议所有的新补丁在发送给Linus之前都先发到-mm树里测试一下。
打了此种补丁的内核不适用于追求稳定的系统中，运行它们比运行其他任何分支都更具冒险性。
如果你想帮助内核开发流程，请测试并使用这些内核发布，并在linux-kernel邮件列表里提供回馈，如果你发现任何问题的话，哪怕什么问题也没有。
在所有其他实验性质的补丁之外，这些内核补丁通常还会包含在发布时在主线-git内核中已经包含的改动。
-mm内核没有一个固定的发布计划，但是通常在每两个-rc内核发布间歇期会发布一些-mm内核（1到3个都很常见）。

子系统专有内核树和补丁
一些不同的内核子系统开发者公布他们的开发树，这样其他人可以看到在内核的不同领域里正在发生什么。这些树都会被包含在-mm内核发布中。
下面是一些不同的内核树的列表：
git树：
- Kbuild 开发树, Sam Ravnborg
kernel.org:/pub/scm/linux/kernel/git/sam/kbuild.git

- ACPI 开发树, Len Brown
kernel.org:/pub/scm/linux/kernel/git/lenb/linux-acpi-2.6.git

- 块设备开发树, Jens Axboe
kernel.org:/pub/scm/linux/kernel/git/axboe/linux-2.6-block.git

- DRM 开发树, Dave Airlie
kernel.org:/pub/scm/linux/kernel/git/airlied/drm-2.6.git

- ia64 开发树, Tony Luck
kernel.org:/pub/scm/linux/kernel/git/aegl/linux-2.6.git

- ieee1394 开发树, Jody McIntyre
kernel.org:/pub/scm/linux/kernel/git/scjody/ieee1394.git

- infiniband, Roland Dreier
kernel.org:/pub/scm/linux/kernel/git/roland/infiniband.git

- libata, Jeff Garzik
kernel.org:/pub/scm/linux/kernel/git/jgarzik/libata-dev.git

- 网络驱动, Jeff Garzik
kernel.org:/pub/scm/linux/kernel/git/jgarzik/netdev-2.6.git

- pcmcia, Dominik Brodowski
kernel.org:/pub/scm/linux/kernel/git/brodo/pcmcia-2.6.git

- SCSI, James Bottomley
kernel.org:/pub/scm/linux/kernel/git/jejb/scsi-misc-2.6.git

其他git内核树可以在http://kernel.org/git找到

quilt trees:
- USB, PCI, Driver Core, and I2C, Greg Kroah-Hartman
kernel.org/pub/linux/kernel/people/gregkh/gregkh-2.6/

报告Bug
bugzilla.kernel.org是Linux内核开发者追踪内核bug的地方。我们鼓励用户在这个工具中报告他们发现的所有bug。欲知使用内核bugzilla的具体细节，请看： http://test.kernel.org/bugzilla/faq.html
主内核源码目录中的REPORTING-BUGS文件有一个报告可能有的内核bug的模板，并详细讲述了内核开发者需要什么样的信息，以便他们可追踪到问题所在。

邮件列表
就像上面的一些文档所描述的，大多数核心内核开发者参与Linux内核邮件列表的讨论。如何订阅和取消订阅这个列表的细节可以从这里找到：
http://vger.kernel.org/vger-lists.html#linux-kernel
网上很多地方都有这个邮件列表的存档。使用一个搜索引擎来搜索这些存档。比如：
http://dir.gmane.org/gmane.linux.kernel
强烈建议你在向列表发邮件询问之前，先在存档中搜索一下你想问的话题。很多事情都已经详细的讨论过了，它们只在邮件列表存档中有记录。
很多内核子系统也有它们单独的邮件列表，在那里开发者们做他们的开发工作。请查看MAINTAINER文件来找到这些不同子系统的邮件列表。
很多邮件列表放置于kernel.org之上。有关它们的信息可以在这里找到：
http://vger.kernel.org/vger-lists.html
请记住在使用这些列表时请遵守良好的行为习惯。下面的URL有一些如何在邮件列表上与人交流的简单的准则，虽然有一点俗气：http://www.albion.com/netiquette
如果有许多人回复你的邮件，收件人的抄送列表会变得很长。在没有一个合理的理由时不要把任何人从抄送列表中去掉，或者不要只回复被列出的地址。要对于收到同一封信两次感到习惯，一次从发信者，一次从邮件列表。不要为了好看而加上别致的邮件头部，人们不喜欢这样。
记得保证你的回复的上下文和属性的完整性，在你的回复的最上方保留类似 “John Kernelhacker wrote ...“的字样，把你的回复写在被引用的段落之间，而不要写在邮件的最上方。
如果你在你的邮件里加入了补丁，要确保它们是纯文本，就象Do*****entation/SubmittingPatches里所说的。内核开发者不希望处理附件或者压缩的补丁；他们会希望评价你的补丁的每一行，只有纯文本才符合这个要求。
确保你使用的邮件客户端软件不会破坏空格和制表符。你可以发一个邮件给你自己，然后应用你自己的补丁来先做个测试。
如果不行，修复你的邮件程序或者换一个。
最重要的一点，请记得显示出你对其他订阅者的尊重。

和社区一起工作
内核社区的目标是提供可能提供的最好的内核。当你提交了一个补丁等待被收录时，它会被且仅被该领域的技术权威所检

查。所以，你应该期待什么呢？
— 批评
— 评论
— 被要求改动
— 被要求解释
— 沉默
记住，这是让你的补丁进入内核的一部分。你必须能够接受对你的补丁的批评和评论，在技术的层面上评估它，然后要么对你的补丁作出修改，要么提供清晰而言简意赅的理由解释为什么不应该做改动。如果没有对你的补丁的回复，等几天再试一次，有时候在流量很大的时候信件可能丢失，或被人忽略遗忘了。

你不应该做什么：
— 期待你的补丁没有任何疑问的被接受
— 不接受批评，不承认缺点
— 忽略评论
— 在不做任何修改的情况下再次提交补丁
在一个寻找可能存在的最好的技术解决方案的社区里，关于一个补丁怎样的有用必定会存在不同的意见。你应该采取合作的态度，愿意改变你的意见以适应内核的需要。或者至少愿意证明的你的观点有价值。记住，犯错误是可以接受的，只要你愿意朝着正确的解决方案努力。
如果对于你的第一个补丁的回应只是一些要求你改正的意见，这很正常。这并不意味着你的补丁将不会被接受，这些意见也不是针对你本人的。你要做的只是改正你的补丁然后重新发送。

内核社区和企业架构的区别

内核社区和大多数传统的企业开发环境工作形式不一样。这里有一个列表你可以尝试遵照执行以避免出现问题：
关于你提交的补丁的好的说法：
— “这个可以解决多个问题。”
— “这个删除了2000行代码。”
— “这个补丁解释了我尝试想描述的东西。”
— “我在5个不同的架构上测试了它……”
— “这里是一系列的小补丁，它们可以……”
— “这个在典型的机器上可以提高表现……”

你应该避免的坏的说法：
— “我们在AIX/ptx/Solaris上都是这样做的，所以它一定是好的……”
— “我已经这样做20年了，所以……”
— “我的公司需要这样做来挣钱”
— “这是我的一千页的设计文档，它解释了我的想法”
— “我已经在它上面花了6个月的心血……”
— “这里是一个5000行的补丁，它……”
— “我重新写了所有现有的垃圾，在这里……”
— “我有完成期限，这个补丁必须现在被应用。”
内核社区和大多数传统软件工程工作环境的另一个不同是没有面对面的交流。使用email和irc作为主要交流形式的好处之一是不会存在基于性别或者种族的歧视。Linux内核工作环境接受女士和少数民族，因为你的存在只是一个email地址。国际化也帮助我们实现了平等的工作环境，因为你无法根据一个人的名字来判断一个人的性别。一个男人可以叫Andrea，一个女人可以叫Pat。大多数为Linux内核做过工作或者发表过观点的女性对此都深有体会。
对于不习惯英语的人来说语言障碍可能会导致一些问题。对于语言的良好的掌握可以令思想在邮件列表上交流的更畅通，所以建议你在发送邮件之前检查一下你的邮件内容在英语里是否有意义。

打散你的补丁
Linux内核社区不乐意接受大段的代码，一般会在收到时立刻丢弃。你的补丁需要适当的被介绍，讨论，并打散为细小的，独立的片段。这几乎和公司里经常做的完全背道而驰。你的提议必须在开发流程的早期提出，这样你才可以收到足够的关于你的工作的回馈。这也会令社区感觉到你是在和他们一起工作，而不是利用他们作为倾倒你的补丁的场所。但是，请不要一次向邮件列表发送超过50封email，你的一系列补丁的个数应该永远小于这个数字。
把补丁打散的理由如下：
1) 小的补丁增加了你的补丁被应用的机会，因为它不需要花太多的时间和精力来检查它的正确性。一个5行的补丁，一个维护者只要花一秒钟瞟一眼然后就可以应用了。不过，一个500行的补丁需要一个小时来检查是否有错误（所需的时间跟补丁的大小差不多成指数级别增长）。小补丁也使得在出错的时候很容易debug。如果出了问题，小补丁可以一个一个的取消，大补丁就比较麻烦了。
2) 除了要把补丁打散之外，在提交之前还要重写和简化（或者只是简单的重排序）你的补丁。这一点也是很重要的。
这里有一个内核开发者Al Viro所做的类比： “想一下一个老师为一个数学系学生批改作业的情景。老师不希望看到学生在回答出正确答案之前的尝试和失败。他们想看到最清楚的，最优美的答案。一个好学生了解这一点，并不会在得到最终答案

之前把他们的中间结果提交上去。对于内核开发来说同样是这样。维护者和评论者不希望看到问题的解决方案背后的思考过程。他们想看到一个简单和优美的解决方案。”
提供一个优美的解决方案和同社区一起工作讨论你未完成的作品，要维持此二者之间的平衡可能是一个很有挑战性的事情。所以应及早的参与一个开发流程以获得回馈来改进你的作品，但仍要保证你的补丁的小块头，这样它们可能提早被接受，哪怕是在你的整个作品还为完成时。
也请注意，如果你的补丁尚未完成而且还需要修改，请不要提交。

证明你的改动
除了打散你的补丁之外，让Linux社区理解为什么他们要加入这项改动也是很重要的。新的功能必须要被证实它们需要而且有用。

为你的改动写文档
在你提交补丁时，要格外留心你在email里说的话。这些信息将会成为这个补丁的ChangeLog信息，将会被保留从而使每个人任何时候都可以看到。它必须完整的描述这个补丁，包括：
— 为什么这个改动是需要的
— 这个补丁的整体设计方案
— 实现细节
— 测试结果
欲知这个过程到底看起来是什么样子的，请看这个文档的ChangeLog部分： “The Perfect Patch”
http://www.zip.com.au/~akpm/linux/patches/stuff/tpp.txt
所有这些事情有时候很难做到。要想完美做到这些要求可能需要几年的时间。这是一个持续的发展过程，需要很多耐心和决心。但是不要放弃，这是可以实现的。很多人已经做到了这一点，每个人都经历过你现在这个阶段。

上一篇 关于Linux内核学习(经典)
下一篇 关于LEA指令

我的同类文章

没有读的文章（85） linux内核（68）

- [集]erlang常用命令收集 2011-11-05 阅读 1242
- 欢迎来到Ubuntu部落 2011-08-01 阅读 693
- erlang分布式节点通讯方式 2011-03-19 阅读 3415
- Erlang简介 2011-03-19 阅读 815
- VOA慢速英语练听力需坚持 2010-11-07 阅读 469
- P2P之NAT类型检测方法 2010-10-31 阅读 724
- ** WARNING ** Mnesia is o... 2011-11-05 阅读 1852
- 轻量级线程和erlang 2011-03-19 阅读 2949
- erlang应用 2011-03-19 阅读 1117
- 英语听力的训练方法 2010-11-07 阅读 887
- NAT分类介绍及其打洞的思考 2010-10-31 阅读 1528

更多文章

参考知识库

猜你在找

查看评论

1楼 zhang蜗牛 2014-06-27 18:07发表

mark

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap