Chinese translated version of Documentation/stable_api_nonsense.txt

If you have any comment or update to the content, please contact the original document maintainer directly. However, if you have problem communicating in English you can also ask the Chinese maintainer for help. Contact the Chinese maintainer, if this translation is outdated or there is problem with translation.

Maintainer: Greg Kroah-Hartman <greg@kroah.com>
Chinese maintainer: TripleX Chung <zhongyu@18mail.cn>

Documentation/stable_api_nonsense.txt 的中文翻译

如果想评论或更新本文的内容，请直接联系原文档的维护者。如果你使用英文交流有困难的话，也可以向中文版维护者求助。如果本翻译更新不及时或者翻译存在问题，请联系中文版维护者。

英文版维护者： Greg Kroah-Hartman <greg@kroah.com>
中文版维护者： 钟宇 TripleX Chung <zhongyu@18mail.cn>
中文版翻译者： 钟宇 TripleX Chung <zhongyu@18mail.cn>
中文版校译者： 李阳 Li Yang <leoli@freescale.com>
以下为正文

写作本文档的目的，是为了解释为什么Linux既没有二进制内核接口，也没有稳定的内核接口。这里所说的内核接口，是指内核里的接口，而不是内核和用户空间的接口。内核到用户空间的接口，是提供给应用程序使用的系统调用，系统调用在历史上几乎没有过变化，将来也不会有变化。我有一些老应用程序是在0.9版本或者更早版本的内核上编译的，在使用2.6版本内核的Linux发布上依然用得很好。用户和应用程序作者可以将这个接口看成是稳定的。

执行纲要

你也许以为自己想要稳定的内核接口，但是你不清楚你要的实际上不是它。你需要的其实是稳定的驱动程序，而你只有将驱动程序放到公版内核的源代码树里，才有可能达到这个目的。而且这样做还有很多其它好处，正是因为这些好处使得Linux能成为强壮，稳定，成熟的操作系统，这也是你最开始选择Linux的原因。

入门

只有那些写驱动程序的“怪人”才会担心内核接口的改变，对广大用户来说，既看不到内核接口，也不需要去关心它。

首先，我不打算讨论关于任何非GPL许可的内核驱动的法律问题，这些非GPL许可的驱动程序包括不公开源代码，隐藏源代码，二进制或者是用源代码包装，或者是其它任何形式的不能以GPL许可公开源代码的驱动程序。如果有法律问题，请咨询律师，我只是一个程序员，所以我只打算探讨技术问题（不是小看法律问题，法律问题很实际，并且需要一直关注）。

既然只谈技术问题，我们就有了下面两个主题：二进制内核接口和稳定的内核源代码接口。这两个问题是互相关联的，让我们先解决掉二进制接口的问题。

二进制内核接口

假如我们有一个稳定的内核源代码接口，那么自然而然的，我们就拥有了稳定的二进制接口，是这样的吗？错。让我们看看关于Linux内核的几点事实：

- 取决于所用的C编译器的版本，不同的内核数据结构里的结构体的对齐方式会有差别，代码中不同函数的表现形式也不一样（函数是不是被inline编译取决于编译器行为）。不同的函数的表现形式并不重要，但是数据结构内部的对齐方式很关键。
- 取决于内核的配置选项，不同的选项会让内核的很多东西发生改变：
  - 同一个结构体可能包含不同的成员变量
  - 有的函数可能根本不会被实现（比如编译的时候没有选择SMP支持，一些锁函数就会被定义成空函数）。
  - 内核使用的内存会以不同的方式对齐，这取决于不同的内核配置选项。
- Linux可以在很多的不同体系结构的处理器上运行。在某个体系结构上编译好的二进制驱动程序，不可能在另外一个体系结构上正确的运行。

对于一个特定的内核，满足这些条件并不难，使用同一个C编译器和同样的内核配置选项来编译驱动程序模块就可以了。这对于给一个特定Linux发布的特定版本提供驱动程序，是完全可以满足需求的。但是如果你要给不同发布的不同版本都发布一个驱动程序，就需要在每个发布上用不同的内核设置参数都编译一次内核，这简直跟噩梦一样。而且还要注意到，每个Linux发布还提供不同的Linux内核，这些内核都针对不同的硬件类型进行了优化（有很多种不同的处理器，还有不同的内核设置选项）。所以每发布一次驱动程序，都需要提供很多不同版本的内核模块。

相信我，如果你真的要采取这种发布方式，一定会慢慢疯掉，我很久以前就有过深刻的教训...

稳定的内核源代码接口

如果有人不将他的内核驱动程序，放入公版内核的源代码树，而又想让驱动程序一直保持在最新的内核中可用，那么这个话题将会变得没完没了。

内核开发是持续而且快节奏的，从来都不会慢下来。内核开发人员在当前接口中找到bug，或者找到更好的实现方式。一旦发现这些，他们就很快会去修改当前的接口。修改接口意味着，函数名可能会改变，结构体可能被扩充或者删减，函数的参数也可能发生改变。一旦接口被修改，内核中使用这些接口的地方需要同时修正，这样才能保证所有的东西继续工作。

举一个例子，内核的USB驱动程序接口在USB子系统的整个生命周期中，至少经历了三次重写。这些重写解决以下问题：

- 把数据流从同步模式改成非同步模式，这个改动减少了一些驱动程序的复杂度，提高了所有USB驱动程序的吞吐率，这样几乎所有的USB设备都能以最大速率工作了。
- 修改了USB核心代码中为USB驱动分配数据包内存的方式，所有的驱动都需要提供更多的参数给USB核心，以修正了很多已经被记录在案的死锁。

这和一些封闭源代码的操作系统形成鲜明的对比，在那些操作系统上，不得不额外的维护旧的USB接口。这导致了一个可能性，新的开发者依然会不小心使用旧的接口，以不恰当的方式编写代码，进而影响到操作系统的稳定性。

在上面的例子中，所有的开发者都同意这些重要的改动，在这样的情况下修改代价很低。如果Linux保持一个稳定的内核源代码接口，那么就得创建一个新的接口；旧的，有问题的接口必须一直维护，给Linux USB开发者带来额外的工作。既然

所有的Linux USB驱动的作者都是利用自己的时间工作，那么要求他们去做毫无意义的免费额外工作，是不可能的。
安全问题对Linux来说十分重要。一个安全问题被发现，就会在短时间内得到修正。在很多情况下，这将导致Linux内核中的一些接口被重写，以从根本上避免安全问题。一旦接口被重写，所有使用这些接口的驱动程序，必须同时得到修正，以确定安全问题已经得到修复并且不可能在未来还有同样的安全问题。如果内核内部接口不允许改变，那么就不可能修复这样的安全问题，也不可能确认这样的安全问题以后不会发生。
开发者一直在清理内核接口。如果一个接口没有人在使用了，它就会被删除。这样可以确保内核尽可能的小，而且所有潜在的接口都会得到尽可能完整的测试（没有人使用的接口是不可能得到良好的测试的）。

要做什么

如果你写了一个Linux内核驱动，但是它还不在Linux源代码树里，作为一个开发者，你应该怎么做？为每个发布的每个版本提供一个二进制驱动，那简直是一个噩梦，要跟上永远处于变化之中的内核接口，也是一件辛苦活。
很简单，让你的驱动进入内核源代码树（要记得我们在谈论的是以GPL许可发行的驱动，如果你的代码不符合GPL，那么祝你好运，你只能自己解决这个问题了，你这个吸血鬼<把Andrew和Linus对吸血鬼的定义链接到这里>）。当你的代码加入公版内核源代码树之后，如果一个内核接口改变，你的驱动会直接被修改接口的那个人修改。保证你的驱动永远都可以编译通过，并且一直工作，你几乎不需要做什么事情。

把驱动放到内核源代码树里会有很多的好处：
- 驱动的质量会提升，而维护成本（对原始作者来说）会下降。
- 其他人会给驱动添加新特性。
- 其他人会找到驱动中的bug并修复。
- 其他人会在驱动中找到性能优化的机会。
- 当外部的接口的改变需要修改驱动程序的时候，其他人会修改驱动程序。
- 不需要联系任何发行商，这个驱动会自动的随着所有的Linux发布一起发布。

和别的操作系统相比，Linux为更多不同的设备提供现成的驱动，而且能在更多不同体系结构的处理器上支持这些设备。这个经过考验的开发模式，必然是错不了的 :)

感谢 Randy Dunlap, Andrew Morton, David Brownell, Hanna Linder, Robert Love, and Nishanth Aravamudan 对于本文档早期版本的评审和建议。

英文版维护者： Greg Kroah-Hartman <greg@kroah.com>