

## Re: Would I be violating the GPL?

**Source:** <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-11/0370.html>

---

**From:** Rob Landley (*rob\_at\_landley.net*)

**Date:** 11/01/05

To: "Jeff V. Merkey" <jmerkey@utah-nac.org>

Date: Tue, 1 Nov 2005 14:32:06 -0600

Ah, top posting. Not a techie then...

On Tuesday 01 November 2005 10:43, Jeff V. Merkey wrote:

> *Alan Cox and others have publicly stated that drivers, if complied stand  
> alone with NO DEPENDENCIES ON KERNEL HEADERS (i.e. they do not  
> incorporate in any way any kernel headers or source code tagged GPL) do  
> not violate the GPL when provided with Linux. DSFS, NVidia, and several  
> folks build kernel modules which are stand alone and are not objected to  
> by the majority of folks.*  
>  
> *If these drivers include kernel headers as part of the build, then the  
> drivers violate the GPL. Period. Check the code. If the vendor is  
> including **\*\*ANY\*\*** GPL kernel headers, then they are required to open  
> source the drivers. There are some zealots and GPL bigots that disagree  
> with this, but Linux folks seem to be reasonable on this point.*  
>  
> *Jeff*

I'd like to point out that the kernel headers are also part of the standard /usr/include headers for a Linux system, and that if you include something like <errno.h>, it eventually tunnels down to get its definition from asm-generic/errno.h. Thus it can be hard to build any linux binary without including linux-kernel headers, but no properly briefed judge would be likely to consider a strict posix binary a derived work of the Linux kernel. So the above advice is kinda stupid.

What the headers have that a module does not is that they implement a defined Application Programming Interface. (Posix, susv3, etc.) The code is not a derived work of that specific implementation of the API, but of the API itself. And since the API is an open and documented standard, life is good. Also, this API can be expressed in english rather than code: keep in mind that translating code into an english description and then having a complete separate team translate it back again with nothing else passing between them is what clean room reverse engineering does. It's pretty well established that copyright doesn't cross that kind of barrier. So a documented and standardized API is a strong barrier for copyright purposes. If challenged in court, you can point to the API as a defense, as well as being

## Linux-Kernel: Re: Would I be violating the GPL?

demonstrably derived from `_documentation_`, not from code. (Even when there are inevitable deviations from the documented API to use platform-specific features, it's a lot easier to sweep them under the rug of the API and claim they're insignificant when there IS an API, and it covers the vast majority of the code. A certain amount of optimization is understandable, and falls under "tweaking" rather than structural/central/fundamental.)

Now let's look at the module. The above legal theory of non-infringement of userspace programs is the existence of a barrier between userspace and the kernel, with a well-defined and intentionally exported API, that makes you definitely not a derived work. As a further defense, normal practice is to write userspace code from API documentation, not from an examination of the kernel source code. (General practice is to only look at the kernel or headers to figure out why something isn't working, not to figure out what it is you need to do in the first place. There are man pages, web pages, and books galore as nont just viable but preferable alternatives to deriving your code from any GPLed text.)

Does this apply to your module? No. What API is it conforming to? None. The only module API in the Linux kernel is the implementation of the linux kernel, and that's so fluid and changing that the same module is unlikely to work without changes even one year later. There are periodic attempts to document this stuff (such as Jonathan Corbet's Linux Device Drivers, third edition), but they don't even claim to be documenting a standard or stable API, they claim to be documenting the internal implementation details of a specific version of the Linux kernel. The best documentation for the Linux-kernel interface is reading the linux kernel source code, and the leading alternatives are also in the linux-kernel tarball: the kernel's own Documentation directory, and the documentation automatically generated from the linux-kernel source code. Unfortunately for would-be API sniffers, these are clearly intended as "aides to understanding the linux-kernel source", `_not_` as replacements for reading the source itself. They're not stable, they're not complete, and they're not even always accurate. You MUST read the code.

That means if your module's status is ever challenged in court, proving the module is not a derived work would kinda suck. There is no standard you're coding to. There's no other (non-linux) environment this module can run in (not without significant modification; and yes the difference between tweaking and structural changes can be important when determining what is and isn't a derived work). This module is completely useless except as an extension to a specific version of the Linux kernel. Without that linux kernel, the module cannot perform any of its functions. (And yes, this matters. What is the module's nature, why was it created, how is it normally used? Books are read, art is viewed, video is played, plays are performed, but modules are `_run_`.)

And then there's the killer: the module itself is highly specific to individual `_versions_` of the linux kernel. Your userspace program would run on Linux 2.2, 2.4, or 2.6, with at most a couple tweaks. Your module may require extensive modification to move from 2.6.9 to 2.6.14. That smells an

Re: Would I be violating the GPL?

## Linux-Kernel: Re: Would I be violating the GPL?

awful lot like "derived work". And a derived work of the GPL cannot be distributed except under the terms of the GPL.

Since the license the module is under is not compatible with the GPL, and this sounds like a module that was custom-designed and built to run in one and only one environment, that's setting you up for a license conflict leading to no ability to distribute, at all.

Now on a purely pragmatic level, what `_matters_` is what enforcement actions are you likely to face? The current (totally unofficial) attitude seems to be that if the module doesn't include any `GPL_ONLY` exports, people are much less motivated to bother with an enforcement action. (This is not a guarantee, just a rule of thumb.) The enforcer might be able to win a court battle against the enforcer, but there would be a battle, probably long and drawn out, and that's that's not very appealing. It's a court case that they'd have to work to prove, and we're generally a pretty lazy lot. (Mostly because we're all have to-do lists of doom and you're item #873 on it, but it comes across as lazy and we're too busy to argue.)

Now if the module uses any `GPL_ONLY` exports, it's clearly (and intentionally) violating something that is actually documented as NOT being an API barrier. There's a warning right there in the symbol: use this and you have crossed a bright line beyond which we consider you not just grey but definitely a derived work. Any module that includes one of those almost certainly `_will_` see an enforcement action, because it's not a whole lot of work on our part to point out a smoking gun.

So if your question is "are this module's license terms at all safe", the answer is pretty clearly "no". Not safe. If your question is "will anybody bother to do anything about it in the 2-3 years before this product is obsolete and forgotten", the answer is "probably not, especially if you're not very successful and never attract any attention to yourself or your product".

If you'd like to base business decisions on that risk analysis, that's your call. This isn't even getting into the fact that a binary-only module taints the kernel (so nobody on this list will ever support your product), it means upgrading your product to new kernel versions is a serious pain, so from a security standpoint you potentially open yourself to liability if the sucker is ever connected to the internet exactly `_because_` you knew you'd be crippling your customers' ability to upgrade, service, or even diagnose the product if they (or their IT support contractor) cared to do so. (Although this is an avenue nobody's been greedy enough to try to sue over and create a precedent about yet. "There was clearly a better way, you intentionally chose against it, and then a customer's machine got cracked and it cost them lots of money" does not `_automatically_` translate into liability for you. Tends to weigh against you in procurement decisions when your customers aren't stupid, though...)

Rob

Linux-Kernel: Re: Would I be violating the GPL?

P.S. In case anything I said made people feel too comfortable, IANAL. Just an educated laybeing who has followed this stuff as a hobby for years. I can spot the \_obvious\_ avenues of attack. Real Lawyers can spot the non-obvious, or jurisdiction-specific ones. If you care about the above potential liability issue of putting closed-source modules into a system widely advertised as open source, knowing that this will deny your customers the standard level of community technical support and upgrades associated with their expectations for "Linux" or "Open Source", by all means talk to your company's lawyer about it. :)

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to [majordomo@vger.kernel.org](mailto:majordomo@vger.kernel.org)

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>