

***Bionic Revisited:
What the Summary Judgment Ruling in
Oracle v. Google Means for Android and the GPL***

In September, Judge William Alsup, the federal judge presiding over the *Oracle v. Google* lawsuit, rejected Google's arguments that software application programming interface (API) specifications are *per se* unprotected by copyright. That [order](#), which is consistent with the legal precedent, will have significant implications for the dispute between Oracle and Google. The court has not yet decided that Google infringed Oracle's copyrights in the Java APIs and code – the trial has been delayed until next year – but [Google now has a tough hill to climb](#): at trial, it will need to walk through each of the 37 APIs and 12 code files one by one and demonstrate for each that there was no infringement.

The implications of Judge Alsup's decision go well beyond the Oracle claims. In particular, the ruling scuttles Google's arguments that it "cleaned" the Linux kernel headers of any copyright protection when it created Android's Bionic library – and that creates problems for the entire Android ecosystem.

How we got here: Google tries to "clean" Linux kernel headers to avoid the GPL

In the spring I issued a [white paper](#) that pointed out how Google had taken a unique and audacious approach to working with the GPL'd Linux kernel when it created the Android stack. Google needed to include a C library that would allow userspace applications to invoke kernel resources. Google could have done what all other embedded Linux projects have done: used one of the various existing libraries, including the GNU C library ("glibc"), or other libraries licensed under the Lesser General Public License ("LGPL") (such as uClibc or the Embedded Glibc). The LGPL'd libraries wouldn't work, because LGPLv2.1 requires that the code of the combined work must be subject to modification, and therefore an Android device could not be locked down, which is something that manufacturers and/or carriers would not accept. Google therefore created a new C library, which it called Bionic. It did this by taking 750 "raw" kernel header files, which are part of the kernel and licensed under the GPL, and running them through an automated "cleaning" process. That "cleaning" process supposedly removed all copyright protection from the headers, allowed Google to put them into Bionic, license Bionic under the BSD, and ["keep GPL out of user-space."](#)





After conducting an analysis that rested on a close reading of the code and applicable case law, I concluded that Google's actions were very likely a violation of GPLv2, both because Google had not "cleaned" all of the kernel headers of all copyright protection, but also because the overall structure of the API would be copyrightable even if each of the 750 header files was not. I also observed that Google's approach – if it worked – had serious negative implications for copyleft principles and the FOSS community and it potentially provided an easy bypass of the GPL for commercial exploitation.

Then, shortly after I issued my paper, it [came to light](#) that Google's "cleaning" approach went beyond the Linux kernel header files to at least two other GPL'd components of Android: the Bluez Bluetooth stack and the ext4 file system. Those components were not authored by the Linux kernel developers, and thus presented more complicated issues.

My initial paper generated a tremendous amount of discussion in the blogosphere. Many commentators recognized the problems that Google's approach created, but the ensuing discussion was heavy on emotion and light on analysis. Two salient arguments ultimately emerged: Google's "cleaning" of the Linux kernel headers was not much different than what others had done before, and it doesn't much matter because the Linux kernel copyright owners aren't likely to make an issue of it with Google. Surprisingly, very few commentators even addressed the cleaning of the Bluez and ext4 header files.

One commentator [acknowledged](#) that my initial paper identified an issue for further analysis, and noted that it "gives some examples that would make a good starting point for a complete analysis," but argued that more examples and analysis were necessary. As explained in my initial paper and below, the examples that I gave were more than enough to show that Google's approach didn't work, because it had not removed **all** copyrightable material from **all** of the headers. Nevertheless, this follow-up white paper provides some even further information and analysis, and it explains why Google's approach to Bionic, Bluez, and ext4 is very different from what had been done before. I start by discussing Google's approach to Bionic, because the creation of that library is the most sweeping implementation of Google's strategy to work around the GPL, and then respond to some of the arguments that have been offered in defense of Google's approach.

It's also important to step back and get a wider perspective on the implications of Google's approach to Bionic. The Android team purports to have developed an automated method for cleaning headers that allows proprietary programs to interface with GPL'd components without concern about any copyleft effect. As I explained in my initial analysis, I have significant legal concerns about this automated approach. Judge Alsup's ruling confirmed these concerns: the question of whether the header files are copyrightable can't be decided on a categorical basis, but instead requires a file by file analysis.

I have practical concerns as well: Google's approach, if it works, provides a roadmap for bypassing the GPL, as well as a relatively simple set of customizable scripts that could allow easier exploitation of GPL components by proprietary programs. An easy bypass of GPL protections runs contrary to what FOSS advocates stand for, and I certainly would not have expected such an uncritical defense of Google.



Why Bionic Is Not Just More Of The Same Old Thing

As I explained in detail in my first paper, Google’s approach works only if the cleaning process removes *all* of the copyrightable material from every one of the headers. Making that determination requires a careful review of the code in question. Many of those who commented on my initial whitepaper, including Linus Torvalds, acknowledged that they had not actually looked at any of the code in question; for others, it was [apparent from the substance](#) of their comments that they had conducted, at most, only a superficial review. One commentator [wondered](#), for instance, whether Bionic used “only parts of Linux headers that are required by specification to write POSIX programs,” but even the most cursory analysis shows that Bionic uses much, much more.

So let’s examine the files themselves.

Google ignored the standard way of making a C library using approved kernel headers

[The README file](#) that accompanies Bionic demonstrates that Google’s developers knew that there is a set of “official” header files that the kernel developers created and maintained specifically for use by userspace applications, a set that has been used by all other C libraries. The Google developers did not want to use that set of headers because they did not want to be dependent upon the kernel developers to curate them:

```
RATIONALE:  
=====
```

```
OVERVIEW OF THE CURRENT KERNEL HEADER MESS:  
-----
```

```
...
```

```
As a consequence, every Linux distribution provides a set of patched kernel headers to be used by userland applications (which installs in /usr/include/linux/, /usr/include/asm/, etc...). these are manually maintained by distribution packagers, and generated either manually or with various scripts. these headers are also tailored to GNU LibC and cannot be reused easily by Bionic.
```

```
* * *
```

```
the kernel authors have already stated that they don't want to fix the problem, even when someone proposed a patch to start cleaning the official headers. from their point of view this is purely a library author problem. yeah, right...
```

```
* * *
```

```
of course, this is only a race to the bottom. the kernel maintainers still feel free to randomly break the structure of their headers (e.g. moving the location of some files) occasionally, so we'll need to keep up with that by updating our build script/original headers as these cases happen.
```



Instead of following the industry standard practices, then, Google decided to go its own way, by creating an automated process that would create “cleaned” headers from the raw Linux header files:

WHAT WE DO:

so we're doomed to repeat the same effort than anyone else. the big difference here is that we want to automate as much as possible the generation of the clean headers to easily support additional architectures in the future, and keep current with upstream changes in the header definitions with the least possible hassle.

* * *

what we do is keep a set of "original" kernel headers, and process them automatically to generate a set of "clean" headers that can be used from userland and the C library.

However, userspace header files have traditionally been hand-curated precisely because pure automation does not permit the kind of line-by-line judgments that need to be made about copyrightability. Perhaps hand-curating was too much work for the Android team, or maybe they weren't too concerned with the copyright issues. Whatever the reason, Google settled on an automated process for cleaning the headers for Bionic. The problem is that they didn't do it right.

The description of the Bionic cleaning process shows that Google was concerned with convenience, not copyright

Each of the “cleaned” Bionic files includes the following boilerplate header:

```

/*****
*****
***
*** This header was automatically generated from a Linux kernel header
*** of the same name, to make information necessary for userspace to
*** call into the kernel available to libc. It contains only constants,
*** structures, and macros generated from the original header, and thus,
*** contains no copyrightable information.
***
*****/

```

The justification for this statement is set forth in [the README file](#) that accompanies the Bionic files. Here's the introduction:

Bionic comes with a set of 'clean' Linux kernel headers that can safely be included by userland applications and libraries without fear of hideous conflicts. for more information why this is needed, see the "RATIONALE" section at the end of this document.

these clean headers are automatically generated by several scripts located in the 'bionic/kernel/tools' directory, which process a set of original and unmodified kernel headers in order to get rid of many annoying declarations and constructs that usually result in compilation failure.

I'll have more to say about the cleaning scripts and their output below. For now, note that Google specifically designed these scripts to optimize for performance, not copyright compliance:

the 'clean headers' only contain type and macro definitions, with the exception of a couple static inline functions used for performance reason (e.g. optimized CPU-specific byte-swapping routines)

In other words, the cleaning scripts do not remove all copyrightable material; they remove some copyrightable material, and keep functions that the Android engineers wanted for improved performance. This is exemplified by Google's treatment of material guarded by an `#ifdef __KERNEL__ ... #endif` block. Those guards signify that the guarded code should be used only within kernel space. Sometimes the Android developers respected those guards, but other times they intentionally ignored them:

note that the "original" headers can be tweaked a little to avoid some subtle issues. for example:

- when the location of various USB-related headers changes in the kernel source tree, we want to keep them at the same location in our generated headers (there is no reason to break the userland API for something like that).

- sometimes, we prefer to take certain things out of blocks guarded by a `#ifdef __KERNEL__ .. #endif`. for example, on recent kernels `<linux/wireless.h>` only includes `<linux/if.h>` when in kernel mode. we make it available to userland as well since some code out there assumes that this is the case.

- sometimes, the header is simply incorrect (e.g. it uses a type without including the header that defines it before-hand)

Google's decision to remove some functions and variables but to retain others depending on how it affected performance shows that they were playing fast and loose with copyright.

Google seems to think that inline functions and variables are *per se* uncopyrightable and can, therefore, be taken as needed. As I pointed out in my earlier paper (points that have been echoed in some recent arguments in the *Oracle v. Google* case), that is a dangerous assumption. Often these inline functions and variables contain the kind of creative expression that is generally protected by copyright. In the next section, I will examine a specific example of where Google got it wrong.

Google kept static inline function and macros in Bionic

When the Android developers wrote their tool to remove copyrighted material, they designed it specifically to leave in certain static inline functions and macros that they found useful. This becomes apparent from an examination of the [script itself](#).

```
# this is the set of known static inline functions that we want to keep
# in the final ARM headers. this is only used to keep optimized byteswapping
# static functions and stuff like that.
```

This introductory comment says that the Android developers chose to keep material for the ARM headers and only byteswapping functions. In fact, an examination of the script itself shows that they chose to keep a variety of functions, and they kept them for the ARM headers, x86 headers, and generic Linux headers:

```
kernel_known_arm_statics = set(
    [ "__arch_swab32", # asm-arm/byteorder.h
    ]
)

kernel_known_x86_statics = set(
    [ "__arch_swab32", # asm-x86/byteorder.h
      "__arch_swab64", # asm-x86/byteorder.h
    ]
)

kernel_known_sh_statics = set(
    [ "__arch_swab16", # asm-sh/byteorder.h
      "__arch_swab32", # asm-sh/byteorder.h
      "__arch_swab64", # asm-sh/byteorder.h
      "__FD_ZERO", # asm-sh/posix_types_32/64.h
      "__FD_SET", # asm-sh/posix_types_32/64.h
    ]
)

kernel_known_generic_statics = set(
    [ "__invalid_size_argument_for_IOC", # asm-generic/ioctl.h
      "__cmsg_nxthdr", # linux/socket.h
      "cmsg_nxthdr", # linux/socket.h
      "ipt_get_target",
    ]
)

# this maps an architecture to the set of static inline functions that
# we want to keep in the final headers
#
kernel_known_statics = {
    "arm" : kernel_known_arm_statics,
    "x86" : kernel_known_x86_statics,
    "sh" : kernel_known_sh_statics
}
```

The Android team chose to leave these inline functions in Bionic, but they didn't make that choice because the functions were non-copyrightable. They chose to leave these inline functions in Bionic because they were useful.

The Android team also decided not to remove certain macro definitions, as the [README](#) file explains:

```
note that we do *not* remove macro definitions, including these macro that perform a call to one of these kernel-header functions, or even define other functions. we consider it safe since userland applications have no business using them anyway.
```

Thus, although the boilerplate headers in the Bionic file assert that there is “no copyrightable information,” the Android developers consciously chose not to remove copyrightable material.

*Bionic is not the same thing as glibc,
despite what Linus and others have suggested*

The end result of Google's automated “cleaning” process – a process that was designed to retain material that was useful, regardless of copyrightability – is a library that is very different from the glibc, or any other pre-existing C library. For instance:

- Bionic uses substantially different material from the kernel than glibc
- There is very little overlap between the header files that Bionic generates from the Linux kernel and those provided with glibc.
- A comparison of the Bionic header files with header files included in a recent release of glibc (v2.13) shows that **more than 600 of the Bionic header files have no equivalent header files in glibc**
- Recall that the Bionic header files are generated from a collection of approximately 750 kernel files. So approximately 80% of the kernel material that Bionic uses is simply not found in glibc.
- When they created the Bionic library, the Android developers deliberately chose to include copyrighted material that is **not** included in glibc.

(The history of glibc is complicated and there have been many versions; this comparison used the current version of glibc.)

When Linus Torvalds defended Google's approach, he said that he didn't think that the software giant had done “anything fundamentally different from glibc.” But Linus [admitted](#) that he had not reviewed the code at issue. A review of the code shows that his speculation was simply wrong. Bionic is very different from glibc in origin and content, and those differences are legally quite significant.



The Linux Kernel Headers Are Protected By Copyright

Let's come up out of the weeds to see why all of this technical detail matters. The substantive responses to my initial whitepaper fell into three broad categories.

First, quite a few commentators [argued](#) or [suggested](#) that header files are not copyrightable at all, such that Google could do whatever it wanted with the Linux kernel headers. Judge Alsup rejected the *per se* approach in *Oracle v. Google*, and not even Google relied on that argument here: it went to some trouble to try to remove the copyrightable information that it found in the header files. In support of the idea that the kernel header files are *per se* uncopyrightable, [some](#) quoted a [posting by Richard Stallman](#) in a kernel mailing list in which he noted that the inclusion of a header file in a program does not necessarily make the program a derivative work. But [others](#) observed – correctly – that Stallman's post should not be read too broadly. In fact, in that very same posting, Stallman acknowledged that taking more than “simple material,” and especially using “inline functions and macros with substantial bodies,” would create a derivative work. In other words, Stallman recognized that header files with inline functions and substantial macros are copyrightable.

And that was the point of my initial analysis: when the Android team created Bionic, they *did* include inline functions and substantial macros. I identified specific examples, including `<sys/socket.h>` and `<sys/byteorder.h>`. A further analysis of the 750 header files would likely identify more copyrightable material, since Google's README file describes how its “cleaning” scripts did not remove certain inline functions and macros. Given the copyrighted material that my analysis has already identified, however, it's not necessary to examine all of the other header files to conclude that Google's approach is legally problematic.

On the other hand, to defend Google's approach, and to be sure that all copyrightable material was removed, it is necessary to examine *every one* of the files. Judge Alsup recognized this in the *Oracle* case, and so, too, did some of those who criticized my analysis and defended Google; one [observed](#) that it would require a lot of work and a “careful analysis done jointly and in close collaboration with a lawyer.” **Yet no one undertook this analysis**; they simply took Google at its word, trusting the boilerplate header. In a blog post, that kind of hopeful wishing might be excusable. But in the real world, when a developer or a company has to make decision to invest money and energy, and where there are actual consequences to a wrong decision, hoping isn't enough. Certainty is vital.

The Header Files Taken As a Whole Are Protected By Copyright

Moreover, as discussed in my original paper, copyright protection isn't limited to the individual files. Regardless of whether any particular header file is copyrightable, the overarching structure of the API (or, to be technically precise, the application binary interface, or ABI) defined by the 750 header files is almost certainly copyrightable. Hardly any of the commentators considered this point. [One who did](#) claims that I made a “sweeping conclusion” without any analysis, but that's just incorrect. The point here is that it *isn't* necessary to do a line-by-line analysis of the content of each the 750 header files to conclude that, taken together as a whole, they define an interface that is protected by copyright. In my initial paper, I discussed the legal authority at fair length, and then I concluded that



the interface defined by the 750 kernel headers is copyrightable. See also *Engineering Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335 (5th Cir. 1994) (technical interface, including data format and structures and file formats, warranted copyright protection); *Control Data Sys., Inc. v. Infoware, Inc.*, 903 F. Supp. 1316 (D. Minn. 1995) (CDC CYBER Network Operating System API copyrightable and infringed); *CMAX/Cleveland, Inc. v. UCR, Inc.*, 804 F. Supp. 337 (M.D. Ga. 1992) (API and data structures contain protectable expression under copyright law); *Consul Tec, Inc. v. Interface Sys., Inc.*, 22 U.S.P.Q.2d (BNA) 1538 (E.D. Mich. 1991) (finding enforceable copyrights in program's "unique compilation of commands, its command line syntax, and its status message codes, all of which constitute[d] unique, creative expression, which [was] separable from the program's 'idea.'"). For his part, Judge Alsup recognized and emphasized that that the selection or arrangement of uncopyrightable elements is subject to copyright protection. See *Lamps Plus, Inc. v. Seattle Lighting Fixture Co.*, 345 F.3d 1140, 1147 (9th Cir. 2003) ("[A] combination of *unprotectable elements* is eligible for copyright protection only if those elements are numerous enough and their selection and arrangement original enough that their combination constitutes an original work of authorship.") (emphasis added).

Some who criticized my analysis relied on cases in which an abstraction-filtration-comparison analysis was used, but, as Judge Alsup's order recognized, that approach is used when the issue is the copying of non-literal elements. It doesn't generally apply to instances of literal copying like this one. See *Lotus Development Corp. v. Borland International Inc.*, 49 F.3d 807, 814-15 (1st Cir. 1995).

The "Normal System Calls" Exception in The COPYING File Doesn't Solve The Problem

Perhaps the most recurrent theme in the response to my white paper was that the [COPYING](#) file inserted in GPLv2 by the Linux kernel maintainers permitted Google to do what it did. That note says:

```
NOTE! This copyright does *not* cover user programs that use kernel
services by normal system calls - this is merely considered normal use of
the kernel, and does *not* fall under the heading of "derived work".
```

```
Also note that the GPL below is copyrighted by the Free Software
Foundation, but the instance of code that it refers to (the linux kernel)
is copyrighted by me and others who actually wrote it.
```

Linus Torvalds

This note says only that userspace programs using kernel services "by normal system calls" are not considered derivative works of the kernel. It does not define those "normal system calls," and it does not bless what the Android developers did: make literal copies of code from the kernel, i.e., the kernel headers, remove information from the kernel headers, and redistribute that code under a non-GPL license.

[Some](#) read the COPYING note broadly, as authorizing *any* userspace program to invoke the functionality of the kernel without becoming subject to the GPLv2. But that expansive interpretation ignores the important limitation that the program must use "normal system calls." Not every



communication from userspace to the kernel qualifies as a “normal system call.” Some communications from userspace to the kernel are regarded as too “intimate” to qualify as “normal system calls.” For instance, the kernel maintainers have taken the position that [userspace graphics drivers](#) do not communicate with the kernel through normal system calls. The same can be said of other hardware drivers, such as the camera driver exposed in Bionic.

So what are “normal system calls?” Typically these are understood by reference to the standard calls exposed by glibc and other standard, accepted C libraries. But Bionic exposes different calls and functionality; that’s precisely why Google created it. Some of the calls exposed by Bionic are ordinarily not available to userspace because they’re excluded by the use of the `#ifdef __KERNEL__ ... #endif` guards. If Google can define any call to the kernel from userspace as a “normal system call” (even those system calls ostensibly guarded by kernel maintainers) simply by including it in its new C library, then a “normal system call” becomes whatever Google (or Oracle or Microsoft) wants it to be. That surprising result would make the distinction meaningless, and it would have far-reaching consequences.

Bionic and Android Need More Certainty

The defense of Bionic, then, boils down to something like this: Even if Google did not remove all copyrightable expression, and even if Bionic is a derivative work of the kernel, it’s still permissible because it simply allows userspace programs to use the kernel and Linus Torvalds said that was ok. There seems, in other words, to be a surprising willingness to ignore the details of Google’s “cleaned” libraries in order to keep GPL out of userspace so that the Android ecosystem can work with proprietary applications – at least for now.

Quotes in blogs don't provide legal certainty for businesses

I say “at least for now” not to create FUD, but because it seems that it’s déjà vu all over again. Back in the early 2000s, my colleagues and I advised Mission Critical Linux (MCLX), a company that offered Linux-based high-availability cluster products. One of the challenges then faced by MCLX and other open source companies was the uncertainty surrounding loadable kernel modules (“LKMs”). For years, the FOSS community was quite fractured over the issue of whether LKMs could be offered under a non-GPL license. Many, including the [Free Software Foundation](#), took the view that LKMs were derivative works of the kernel that were subject to GPLv2. Viewed strictly from the perspective of copyright law, that [analysis](#) has some force, and one of the more persuasive pieces of evidence cited to support the argument was that a LKM includes literal copies of code from header files – just like Bionic does – such that it was a derivative work under the law.

In the case of LKMs, Torvalds [took the view](#) that LKMs were, almost by definition, derivative works of the kernel, but that he, using his own judgment, would grant exceptions case by case:

You just can't make a binary module for Linux, and claim that that module isn't derived from the kernel. Because it generally is – the binary module not only included header files, but more importantly it clearly is *not* a standalone work any more. So even if you made your own prototypes and tried hard to avoid kernel headers, it would *still* be connected and dependent on the kernel.



And note that I'm very much talking about just the *binary*. Your source code is still very much yours, and you have the right to distribute it separately any which way you want. You wrote it, you own the copyrights to it, and it is an independent work.

But when you distribute it in a way that is **clearly** tied to the GPLd kernel (and a binary module is just one such clear tie – a "patch" to build it or otherwise tie it to the kernel is also such a tie, even if you distribute it as source under some other license), you're **by definition** not an independent work, any more.

(But exactly because I'm not a black-and-white person, I reserve the right to make a balanced decision on any particular case. I have several times felt that the module author had a perfectly valid argument for why the "default assumption" of being derived wasn't the case. That's why things like the AFS module were accepted – but not liked – in the first place).

Torvalds' decisions were made on the basis of idiosyncratic considerations, however, and they do not bind the other holders of copyright in the Linux kernel:

I'd just like to finish off with a few comments, just to clarify my personal stand:

- I'm obviously not the only copyright holder of Linux, and I did so on purpose for several reasons. One reason is just because I hate the paperwork and other cr*p that goes along with copyright assignments.

Another is that I don't much like copyright assignments at all: the author is the author, and he may be bound by my requirement for GPL, but that doesn't mean that he should give his copyright to me.

A third reason, and the most relevant reason here, is that I want people to know that I cannot control the sources. I can write you a note to say that "for use XXX, I do not consider module YYY to be a derived work of my kernel", but that would not really matter that much. Any other Linux copyright holder might still sue you.

- I don't really care about copyright law itself. What I care about is my own morals. Whether I'd ever sue somebody or not (and quite frankly, it's the last thing I ever want to do – if I never end up talking to lawyers in a professional context, I'll be perfectly happy. No disrespect intended.) will be entirely up to whether I consider what people do to me "moral" or not. Which is why intent matters to me a lot – both the intent of the person/corporation doing the infringement, and the intent of me and others in issues like the module export interface.

Another way of putting this: I don't care about "legal loopholes" and word-wrangling.



It's this informal, ad hoc process that creates such legal uncertainty. It's not just lawyers for proprietary companies that say so; here's [Eben Moglen on the issue](#):

it would be very helpful if the kernel developers had decided to formalize the nature of their exceptions, and the Free Software Foundation and I have made a few attempts to discuss that matter with kernel developers. I had conversations with Ted Ts'o, I talked to Linus about it and I understood there were some reluctances to clarify, in a full and complete way, what was going on. There may have even been disagreements among kernel developers about that, I wouldn't know. **But I continue to think that it would be useful, for a whole variety of people who are trying in good faith to do the very best they can, and who may be navigating some dodgy legal territory, for them to be able to refer to something beyond the COPYING file which -- with all due respect -- I think probably doesn't contain all the terms that are relevant to the use of the kernel.**

That's absolutely correct.

Shifting attitudes have caused problems before

There was similar uncertainty in the embedded Linux space, which included companies like TiVo and one of my firm's clients, Embedded Support Tools Corporation (ESTC), which was later acquired by Wind River Systems. Companies that worked with embedded GPL'd code struggled with the apparent obligation to distribute the source and permit downstream modification of that code. TiVo was at first acclaimed by open source proponents because it brought Linux to widely popular consumer devices, but then the climate changed. The FSF said TiVo had only "[complied with GPL 2 by the skin of its teeth](#)," and "Tivoization" was declared one of the great evils that GPLv3 was intended to address. In December 2009, the Software Freedom Law Center [sued more than a dozen companies](#), including Best Buy, Samsung, JVC, Westinghouse, and Western Digital, among others, for distributing consumer devices that included embedded GPL'd code (in that case, not Linux, but BusyBox), and not providing the source code in a manner that permitted modification.

Android was specifically created to be embedded in consumer devices. Clients building on an Android platform want to know, in advance, whether they'll have to open code that reveals sensitive proprietary information about their applications and hardware. They're working in good faith in the face of considerable uncertainty, and they don't want to get sued because someone disagrees with their analysis.

The issue goes beyond the kernel headers

Even if Torvalds' comments could somehow provide comfort with regard to Bionic, they don't help at all with regard to Bluez and ext4. Those components don't come with a COPYING file or an archive of listserve postings that can be mined to excuse what the Android team did. And it seems plainly evident from the [history](#) that the Android team "cleaned" Bluez only shortly after it was [pointed out](#) by the community that the Bluez had been incorporated in Android in violation of the GPLv2 license. It's apparent, too, that Google is using this "cleaning" approach to working with other GPL'd code, and thus it wouldn't be surprising to find other examples.

There are two ways to try to resolve the legal uncertainty for the Android ecosystem: (1) Android device manufacturers will have to monitor how Google uses its “cleaning” tool and carefully examine each “cleaned” file to confirm that Google did indeed remove all copyrightable material and didn’t hijack a copyrightable interface; or (2) the FOSS community, and particularly its most vigilant enforcers (such as the FSF, the SFLC, and [gplviolations.org](#)), should address the issue definitively. The SFLC has done this before, when it gave [a written opinion](#) on whether the GPL applied to WordPress themes. Interestingly, in that opinion, the SFLC concluded that WordPress templates were derivative works of WordPress and, therefore, subject to WordPress’ GPLv2 license because the PHP elements of the templates

are combined with the WordPress code in memory to be processed by PHP along with (and completely indistinguishable from) the rest of WordPress. The PHP code consists largely of calls to WordPress functions and sparse, minimal logic to control which WordPress functions are accessed and how many times they will be called. They are derivative of WordPress because every part of them is determined by the content of the WordPress functions they call.

To be sure, the WordPress situation involved different technical details than the kernel headers, but the analysis that the SFLC used in that case shows how “derivative work” is usually understood expansively by FOSS proponents, to include interacting with GPL-licensed code by code that “consists largely of calls” and “sparse, minimal logic.”

Monitoring Google’s use of the cleaning script is a lot of work, and it doesn’t give a lot of certainty. It’s probably not a realistic option for Android device manufacturers who need to decide whether to bet many millions (or even billions) of dollars on a platform. A statement approving the use of the script would be great for many clients, although it is unclear who would have the power to give such a statement, since none of the relevant organizations can speak for all copyright holders in GPL code. Many software companies and developers would like very much to know if there is an “officially sanctioned” way to clean up GPL components for use by proprietary programs. It may not be so great for the FOSS community, however, to the extent that it could open up easy exploitation of other GPL’d programs. What’s to prevent someone else from taking a similar approach to “wash” the headers of the MySQL Connector, for instance, so that it can be embedded with proprietary code or relicensed under a more permissive open source license?

Conclusion

Nearly seven years ago, at the time that IBM was running [commercials](#) during the Super Bowl declaring that “The Future Is Open,” a colleague and I published several papers, including for the [Computer Law Association](#), that examined the transformation of the free software movement and ecosystem. Ieuan G. Mahony and Edward J. Naughton, “Open Source Software Monetized: Out of the Bazaar and into Big Business,” 21 *The Computer and Internet Lawyer* 1 (October 2004) (not available online). After tracing the history of the free software and its embrace by largest proprietary software companies, we observed that open source software was increasingly used simply as a marketing theme, to make customers feel good about their essentially proprietary products. Richard Stallman [made the point](#) even more powerfully in his essay, “Why Free Software Is Better than Open Source”:



In effect, these [commercial software] companies seek to gain the favorable cachet of “open source” for their proprietary software products—even though those are not “open source” software—because they have some relationship to free software or because the same company also maintains some free software . . . These companies actively try to lead the public to lump all their activities together; they want us to regard their non-free software as favorably as we would regard a real contribution, although it is not one. They present themselves as “open source companies,” hoping that we will get a warm fuzzy feeling about them, and that we will be fuzzy-minded in applying it. This [is a] manipulative practice.

In creating Android, Google took an approach to GPL'd code that warrants a rigorous analysis. I've analyzed it without being fuzzy-minded, and I don't think that it works legally, and Judge Alsup's recent order confirms my view. But if Google's approach does work – if the header files of the Linux kernel and **any** GPL'd component can be easily “washed” of their copyleft – that fact and its implications for the FOSS community should be made clear, as I have done here, so that the fuzziness is removed and all developers and device manufacturers can have that certainty.

New York
Seven Times Square
New York, NY 10036
+1.212.209.4800
+1.212.209.4801 [fax]

Boston
One Financial Center
Boston, MA 02111
+1.617.856.8200
+1.617.856.8201 [fax]

Washington, DC
601 Thirteenth Street NW
Suite 600
Washington, DC 20005
+1.202.536.1700
+1.202.536.1701 [fax]

Hartford
185 Asylum Street
Hartford, CT 06103
+1.860.509.6500
+1.860.509.6501 [fax]

Providence
10 Memorial Boulevard
Providence, RI 02903
+1.401.276.2600
+1.401.276.2601 [fax]

London
8 Clifford Street
London, W1S 2LQ
United Kingdom
+44.20.7851.6000
+44.20.7851.6100 [fax]

Dublin
Alexandra House
The Sweepstakes
Ballsbridge, Dublin 4
Ireland
+353.1.664.1738
+353.1.664.1838 [fax]

www.brownrudnick.com

BROWN RUDNICK is an international law firm with offices in the United States and Europe. Our 200 attorneys provide assistance across key areas of the law, including complex litigation and arbitration, intellectual property, real estate, bankruptcy and finance, corporate and securities, health care, energy, and government law and strategies.

For further information on this topic, please contact:

Edward J. Naughton
+1.617.856.8567
enaughton@brownrudnick.com

Information contained in this Alert is not intended to constitute legal advice by the author or the attorneys at Brown Rudnick LLP, and they expressly disclaim any such interpretation by any party. Specific legal advice depends on the facts of each situation and may vary from situation to situation.

Distribution of this Alert to interested parties does not establish an attorney-client relationship. The views expressed herein are solely the views of the authors and do not represent the views of Brown Rudnick LLP, those parties represented by the authors, or those parties represented by Brown Rudnick LLP.

