brownrudnick ADVISORY
www.brownrudnick.com

THE BIONIC LIBRARY: DID GOOGLE WORK AROUND THE GPL?

There's been much ado about the Android operating system lately. A good bit of the buzz has been the marketing hype and early adopter reviews that inevitably accompany the release of new products. But it's not this buzz that's so interesting; it's the discussion about the open source licensing and copyright issues that Android has generated. Android presents some fascinating lessons and some thorny legal questions for those who develop applications for the Android platform and the software lawyers who advise them.

Some of these questions have been playing out very publicly. Oracle sued Google, for instance, alleging that Android includes line-for-line copies of at least eight Oracle Java source code files, and that roughly one-third of Android's application programming interface (API) packages – more than fifty – are infringing derivatives of Oracle's Java API packages. Google denies that Android infringes Oracle's copyrights, but Google doesn't deny that Android contains copies of multiple portions of Oracle's code. Google argues, rather, that the copied portions of Oracle's code are not copyrightable, and/or that the copying was fair use.

Google's position is a bold assault on copyright protection for software and source code. There are cases, to be sure, that have permitted some copying of very small snippets of code when that is necessary to achieve interoperability. E.g., *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522 (6th Cir. 2004) (finding no copyrightable expression in a 55-byte program that acted as lockout code for printer toner cartridges); *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992) (finding no copyrightable expression in 20-byte initialization code for gaming system). Those cases do not provide much support for Google's argument that copyright law allows it to copy entire source code files, and even less for its suggestion that entire APIs are not copyrightable.

The battle between Oracle and Google certainly attracts many spectators (an Engadget article on the case generated more than 750 comments http://www.engadget.com/2011/01/21/android-source-code-java-and-copyright-infringement-whats-go/), but it is largely a private dispute that the parties will likely resolve between themselves someday. It raises

issues that are critically important to the open source community, however, and especially to those who develop for the Android platform. Just as Google has taken the position that Oracle's Java APIs do not contain expression that can be protected by copyright, Google has taken the same position with regard to the API for the Linux kernel. Relying on this rationale, Google has distributed Linux kernel header files under licenses other than the General Public License version 2 (GPLv2).

Google's unorthodox approach to the Linux kernel header files and GPLv2 has not generated nearly as much public attention as its lawsuit with Oracle, but it seems to conflict with public statements made by Linus Torvalds, the original developer of Linux and one of the primary maintainers of the Linux kernel, around how Linux header files may be used. Moreover, as Professor Ray Nimmer recently observed, it presents grave legal risks – and interesting opportunities – for developers and the open source community as a whole. See http://www.ipinfoblog.com/archives/licensing-law-issues-infringement-and-disclosure-risk-in-development-on-copyleft-platforms.html.

How Google Created Bionic

Some background is necessary here. Google built the Android operating system around the Linux kernel, specifically version 2.6. The Linux kernel is licensed under GPLv2, and thus carries with it all of the rights, obligations, and encumbrances of that license, including the obligation to provide source code to the original program and any derivative works, the freedom to modify and copy, and a potential waiver of patent rights, among others.

Like any UNIX system, the Linux kernel includes system header files. A header file is a file that contains declarations of variable types and definitions of macros. http://en.wikipedia.org/wiki/Header_file; http://gcc.gnu.org/onlinedocs/gcc-4.3.2//cpp/Header-Files.html. The Linux kernel header files define the interface between the Linux kernel and programs that use it. When an application needs to call on operating system resources, it does so through the interfaces defined by the kernel header files.

There are two ways for applications to use the header files to call the kernel. The application can be built directly against the GPL-licensed header files, which are called "raw" header files in the industry. This approach directly incorporates the "raw" header files into the application, and thus, when the application is subsequently distributed, it will be subject to the GPLv2. This approach is discouraged for technical reasons, and it's generally avoided so that applications do not become subject to GPLv2 simply because they link to and use the Linux kernel.

The preferred option in the industry is to build the application against a different set of kernel header files that accompany the GNU C library ("glibc"). The headers in glibc are different from the "raw" header files, because they've been created, with the apparent blessing of the Linux kernel maintainers, from a subset of the raw files by means of a standard process. Within the industry, these header files are referred to as "sanitized" header files. Linus Torvalds and

the kernel maintainers have publicly declared that applications can use these sanitized header files without becoming subject to GPLv2 because these sanitized header files are "normal system calls" http://lkml.org/lkml/2003/12/4/239. Thus, Linux distributions (such as RedHat, SuSE, Ubuntu, Mandriva, and others) generally use the sanitized header files so that developers can write applications to run on Linux and use the kernel without becoming subject to GPLv2.

Google took a different approach when it created Android. (The following description is taken from the Android open source project site, especially the README file available at http://android.git.kernel.org/?p=platform/bionic.git;a=blob_plain;f=libc/kernel/README.TXT;hb=froyo-release.) Google wanted to use a C library that was not licensed under GPLv2 in order to "keep GPL out of user-space." Instead of using glibc, therefore, it created a new C library called Bionic. (See http://www.zdnet.com/blog/burnette/patrick-brady-dissects-android/584; http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf at slides 34-39.)

Google thus did not use the sanitized header files that accompany glibc. Rather, the Android team created its own set of approximately 750 Linux kernel header files by running Google-created scripts against the raw, GPL-licensed Linux kernel header files. These scripts were designed to "clean up" the kernel headers by removing certain information. The output, which Google calls "the 'clean headers'" still contain type definitions, macro definitions, and static inline functions.

> 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.
>
> 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)

http://android.git.kernel.org/?p=platform/bionic.git;a=blob_plain;f=libc/kernel/README.TXT;hb=froyo-release

As the Android project site describes it, the scripts do not remove type or macro definitions from the raw header files, and indeed the scripts optimize the existing macros. http://android.git.kernel.org/?p=platform/bionic.git;a=blob_plain;f=libc/kernel/README.TXT;hb=froyo-release ("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.") The scripts also keep certain inline functions for performance reasons. What the scripts do remove are the comments and whitespace from the raw header files.

According to the Android project site, Google takes the position that removing this information "cleans" the raw Linux kernel header files of any copyrightable information, such that Google is free to re-distribute the files under a license other than the GPL:

> This directory contains the original kernel headers that are used to generate Bionic's "cleaned-up" user-land headers.
>
> They are mostly covered by the GPLv2 + exception, and thus cannot be distributed as part of the platform itself.
>
> (NOTE: The cleaned up headers do not contain copyrightable information and are distributed with Bionic)

http://android.git.kernel.org/?p=platform/external/kernel-headers.git;a=blob;f=original/README.TXT;h=d817b531c79b0e5a483eeabb43f0aca877218f17;hb=refs/heads/froyo-release.

Google thus distributes Bionic under the BSD license, a license that has no copyleft effect. What Google claims to have done, then, is to "wash" the Linux kernel headers, so that the "cleaned-up" headers are no longer subject to the GPLv2 and can be used in a proprietary software system without fear of becoming subject to the GPLv2 obligations (which many application developers seek to avoid).

Did Google Work Around GPLv2?

Google has taken an unusually audacious approach to working with the GPL-licensed Linux kernel header files: it has attempted to strip out all copyrightable expression, so that the obligations of the GPL will no longer apply, and re-license the code under a non-copyleft, less restrictive license. It's a fascinating strategy, but does it work?

Google's approach rests on two key assumptions, one micro and one macro, about the copyrightability of the Linux kernel header files. First, on the level of individual files, it assumes that the only copyrightable information in a Linux kernel header files is the "nondirective" text, such as comments. It assumes that header files are not copyrightable if they contain only declarations of data types, or a macro, or simple inline functions. Second, Google's approach assumes that if the individual header files are not copyrightable, then the overall structure – the combination of 750 header files that define the Linux kernel API – is not copyrightable. Neither of these assumptions is certain, and, in fact, they are very likely mistaken, both as a matter of law and as a matter of industry understanding and practice.

Copyright protection at the file level

Let's start at the individual file level. There has been much debate about the copyrightability of header files, and it can be a difficult question. Copyright law protects original expression, not ideas or methods of operation, but software code straddles this dividing line: code is both highly creative and functional. When a work contains both expressive and functional elements, "copyright is limited to those aspects of the work – termed 'expression' – that display the stamp of the author's originality." *Harper & Row, Publishers, Inc. v. Nation Enterprises,* 471 U.S. 539, 547-48 (1985). Header files can run the gamut of expression: a header file that is merely a list of declarations of variables or names of functions might not be copyrightable, but one that includes macros and inline functions probably is, because writing those macros and functions demands creativity and originality, and they can be written in different ways.

The appeals court decision in *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522 (6th Cir. 2004), makes this point quite clearly. Lexmark, a printer manufacturer, used a 55-byte program to create an authentication sequence "handshake" between the printer and a toner cartridge. Static Control Components (SCC) copied that program so that its toner cartridges would work in Lexmark printers. After the trial court entered an injunction against SCC, finding infringement, the Sixth Circuit Court of Appeals reversed. The appeals court found that there was no copyrightable expression in Lexmark's tiny program because there was, as a practical and computational matter, only one way to write a program that would provide a valid authentication sequence. *Lexmark*, 387 F.3d at 535 ("To the extent compatibility requires that a particular code sequence be included in the component device to permit its use, the merger and scènes à faire doctrines generally preclude the code sequence from obtaining copyright protection.") (citing *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524 (9th Cir. 1992) ("When specific instructions, even though previously copyrighted, are the *only and essential means of accomplishing a given task*, their later use by another will not amount to infringement.") (quoting National Commission on New Technological Uses of Copyrighted Works, Final Report 20 (1979)); *Atari Games Corp. v. Nintendo of Am., Inc.,* Nos. 88-4805 & 89-0027, 1993 WL 207548, at *1 (N.D. Cal. May 18, 1993) ("Program code that is strictly necessary to achieve current compatibility presents a merger problem, almost by definition, and is thus excluded from the scope of any copyright.")). For a larger, more complex program, by contrast, "it would have been exceedingly difficult to say that practical alternative means of expression did not exist." *Lexmark,* 387 F.3d at 539.

Determining copyrightability is thus a fact-specific, case-by-case exercise. In the case of the Linux kernel header files, most would likely be copyrightable (even with comments removed) because the headers include many macros, inline functions, and other logic that clearly qualifies as expression. In fact, there are several Linux kernel header files that consist almost entirely of static inline function, which is generally regarded as copyrightable expression. For instance, the Linux kernel header file <sys/byteorder.h> consists of an optimized byteswapping function that was included by the Linux kernel developers because it was clever and well-written – the kind of creativity that constitutes copyright

protection. (The initial thread from the Linux kernel mailing list that contributes this particular inline function and discusses the performance benefit it supplies is available at http://lkml.org/lkml/2002/11/29/68.) When it created the "clean" Bionic headers, Google kept this inline function. See http://android.git.kernel.org/?p=platform/bionic.git;a=blob;f=libc/kernel/archx86/asm/byteorder.h;h=a839798c8a476876968e6c435069460a55d9252d;hb=froyo-release.

To pick another example, <sys/socket.h> is a complex header file that defines socket functions and data structures in UNIX systems. There is a standard specification that defines the functionality that <sys/socket.h> should provide. See http://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/socket.h.html. Like all specifications, this one can be implemented in many different ways. The Linux 2.6 implementation (available at http://fxr.watson.org/fxr/source/include/linux/socket.h?v=linux-2.6) includes an inline function for creating data structures, including the cmsghdr data structure. The Bionic implementation (available at http://android.git.kernel.org/?p=platform/bionic.git;a=blob;f=libc/kernel/common/linux/socket.h;h=b578df94328dd78ef7e7bc57791bdae8e0192db7;hb=froyo-release) was generated automatically from the Linux 2.6 implementation, and it is substantially identical, even including the inline functions.

Other flavors of UNIX, by contrast, implement the <sys/socket.h> specification in different ways. Compare FreeBSD (http://fxr.watson.org/fxr/source/sys/socket.h), NetBSD (http://fxr.watson.org/fxr/source/sys/socket.h?v=NETBSD), OpenSolaris (http://fxr.watson.org/fxr/source/common/sys/socket.h?v=OPENSOLARIS), and XNU (Apple's Darwin kernel) (http://fxr.watson.org/fxr/source/bsd/sys/socket.h?v=xnu-1456.1.26). When there are so many different ways to express an idea, here the <sys/socket.h> specification, those different expressions are copyrightable. *Lexmark*, 387 F.3d at 535 (citing *Atari Games Corp. v. Nintendo of Am., Inc.,* 975 F.2d 832, 840(Fed. Cir. 1992 ("The unique arrangement of computer program expression which generates [the] data stream does not merge with the process so long as alternate expressions are available."). Consequently, the portions of the Linux 2.6 <sys/socket.h> header that are included in the Bionic version are almost certainly protected by copyright.

Although I haven't examined every one of the 750 "clean" header files in Bionic, the sheer number of files and the fact that the "cleaning" process was designed not to remove macros and static inline function leads me to believe that the Bionic headers likely include many other examples of copyrightable expression from the raw kernel headers. Certainly, sorting out what is and isn't subject to GPLv2 in Bionic would require at least a file-by-file, and most likely line-by-line, analysis of Bionic – a daunting task for even a very sophisticated intellectual property lawyer. But even if you were to undertake this analysis and segregate the copyrightable expression from the non-copyrightable, it doesn't change the fact that GPLv2 covers the entirety of the kernel code and any derivatives of it. Picking and choosing pieces of the kernel code to include in a new library is creating a derivative work, and, accordingly, GPLv2 would still apply.

Copyright protection for the overall API

Even if not all of the 750 header files that comprise Bionic contain as much copyrightable expression as the two discussed above, the overall structure, organization, and combination of noncopyrightable elements can still be copyrightable. This isn't a revolutionary idea: it's commonly understood that even if individual words and facts can't be protected by copyright, the creative selection and organization of those individual words and facts – into a novel, or a movie – is entitled to copyright protection. See, e.g., *CMM Cable Rep., Inc. v. Ocean Coast Props., Inc.,* 97 F.3d 1504 (1519) (1st Cir. 1996) (copyright protects a marketing brochure, taken as a whole, even though short phrases and fragmentary expressions in the brochure are not entitled to copyright protection). The case law makes clear, in fact, that only "some minimal degree of creativity" is needed for a work to be protected by copyright. *Feist Publications, Inc. v. Rural Telephone Service Co.,* 499 U.S. 340, 362 (1991); see *Assessment Technologies of WI LLC v. WIREdata, Inc.,* 350 F.3d 640, 643 (7th Cir. 2003) (copyright law "requires only enough originality to enable a work to be distinguished from similar works that are in the public domain"). The cases suggest that this "minimal degree of creativity" is minimal indeed. In one notable case, the Second Circuit Court of Appeals found that a form that set out nine different statistics for baseball pitchers, to predict their performance, was copyrightable. *Kregos v. Associated Press,* 937 F.2d 700 (2d Cir., 1991). Although the individual statistics themselves were commonly used (won-lost record, earned run average, innings pitched, etc.), the author's selection of those nine statistics was sufficiently creative because "the universe of known facts available only from inspection of box scores of prior games is considerably greater than nine." *Kregos*, 937 F.2d at 704. A leading treatise on copyright law sums it up nicely: "it must be recalled that even the most commonplace and banal results of independent effort may command copyright protection, provided such independent effort is more than minimal." 1 David A. Nimmer, Nimmer on Copyright § 2.01[B] at 2-17 (Matthew Bender, 2010).

Here, the 750 Linux kernel header files that were copied into Bionic are hardly banal or commonplace. They define a complex overarching structure, an application programming interface, that is thoughtfully and cleverly designed, and almost assuredly protected by copyright. The API provides functionality, insofar as it allows applications to call operating system resources, but that functional aspect does not deprive it of copyright protection because there are different ways to provide that functionality.

To the contrary, every software developer is familiar with the idea that there are an enormous variety of ways to express such a complex interface, some better and some worse. Indeed, one of Google's principal software engineers, Joshua Bloch (http://research.google.com/pubs/author32.html), has published an excellent document entitled "How To Design A Good API And Why It Matters" (http://research.google.com/pubs/archive/32713.pdf). As Mr. Bloch notes on the first page of his presentation, good API design is important because "APIs can be among a company's greatest assets." This is certainly well-accepted in the industry: software lawyers regularly handle transactions in which the

parties have negotiated over the ownership rights to an interface between systems. Public APIs, including as the APIs for Google Maps and Google Search, are governed by terms of use that declare that the intellectual property rights in the APIs belong to the developer. Google's argument that it has stripped all copyright protection from the API created by the 750 Linux kernel headers is almost certainly wrong under US copyright law – it would mean that virtually any API is unprotectable by copyright – and certainly contrary to industry understanding and practice.

What Does Bionic Mean for the Open Source Community?

It's especially difficult to reconcile Google's position on the copyrightability of the Linux kernel headers with the consensus views of the open source community. Leading open source attorneys, while acknowledging the difficult copyright issues presented by header files, note that the kernel headers likely are copyrightable because they are more complex than simple declarations and because they include macros and other logic. Heather Meeker, THE OPEN SOURCE ALTERNATIVE 169-70 (2008); Van Lindberg, INTELLECTUAL PROPERTY AND OPEN SOURCE 88 (2008).

For their part, the maintainers of the Linux kernel have consistently taken the position that any use of the raw kernel headers is subject to the obligations of GPLv2. Linus Torvalds himself has been quite clear on this:

> "… [Y]ou do _NOT_ have the right to use a kernel header file (or any other part of the kernel sources), unless that use results in a GPL'd program.
> …
> What you _do_ have the right is to _run_ the kernel any way you please . . . .
>
> BUT YOU CAN NOT USE THE KERNEL HEADER FILES TO CREATE NON-GPL'D BINARIES.
>
> Comprende?"

http://lkml.org/lkml/2003/12/5/13.

Indeed, it would seem that this position is required by the plain terms of GPLv2. GPLv2 permits copying, modifying, and distributing GPLv2 licensed code only under certain conditions. One of the most important conditions is that any modified versions of the code must be distributed under GPLv2 and not another license. GPLv2, Section 2(b) (http://www.gnu.org/licenses/gpl-2.0.html) ("You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.") The Android developers started with GPLv2 licensed code, the Linux kernel headers, and modified that code by removing copyrightable information to create the Bionic headers. The only permission to make those modifications came from GPLv2, and thus the modified code should also be subject to GPLv2.

Why Should Anyone Care About How Google Created Bionic?

The status of the Bionic headers may seem like an arcane technical and legal issue, but it has real world importance. It's an important issue for developers of Android applications to the extent that their software links to and incorporates the Bionic header files. If Google's assumptions are wrong, and if the Bionic header files remain subject to GPLv2, there is a considerable risk that applications using them become subject to GPLv2 as well. On Android, all native code must compile against Bionic, and at least in the view of the Free Software Foundation, compiling code against a GPLv2 licensed library like Bionic makes the code subject to GPLv2. http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MereAggregation. If GPLv2 applies to Android applications, developers' ability to differentiate on the Android platform would be seriously impaired, because they would be required to release the source code of their applications and would be precluded from limiting how anyone, including competitors, uses that code. At a minimum, developers are taking a significant business and legal risk, by joining in Google's bet that the Bionic headers don't contain any copyrightable material.

But these issues are even more important for the open source community as a whole, especially if Google is right. GPLv2 was designed to protect the free availability of code, and developers contributed to the Linux kernel expecting that their work would always remain free and could not be made proprietary. But if Google is right, if it has succeeded in removing all copyrightable material from the Linux kernel headers, then it has unlocked the Linux kernel from the restrictions of GPLv2. Google can now use the "clean" Bionic headers to create a non-GPL'd fork of the Linux kernel, one that can be extended under proprietary license terms. Even if Google does not do this itself, it has enabled others to do so. It also has provided a useful roadmap for those who might want to do the same thing with other GPLv2-licensed programs, such as databases.

Android has already changed the landscape of the mobile device marketplace. Its impact on free and open source software may be far greater still.

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

BROWNRUDNICK