Professor Robert W. Gomulkiewicz
Director, Intellectual Property Law & Policy Graduate Program
University of Washington School of Law

Professor Jane K. Winn
Shidler Center for Law, Commerce & Technology
Professor of Law
University of Washington School of Law

SOFTIC Symposium 2003
Issues for Discussion about Derivative Works in GPL

Introduction

The answers to the following questions will depend on several factors: whether the GPL is interpreted in light of commentary provided by its author, Richard Stallman, ("Form Drafter's view") or in light of how a US court would likely interpret its provisions if a lawsuit were brought involving the issue raised in the question ("Court's view").[1] The Court's view will likely further depend on whether the application program is being run by the author of the application program ("Programmer's derivative work") or by a licensee of a work created by a third party ("Passive Licensee's derivative work.")

In interpreting a contract, commentary explaining the intent of the drafter of a standard form contract used by the parties is one source a court might look to in interpreting the intentions of the parties. The court will need to establish the parties' intent in order to determine whether a contract has been formed, and if so, the meaning of the terms of the contract. A court interpreting the meaning of the GPL license is likely to look first to evidence of the intentions of the two parties to the dispute, and only as a secondary matter to the intent of the author of the GPL form contract.

If a contract is found to have been made, but the parties disagree over the interpretation of its terms, then a court will have to decide what the contract's terms are. If one party offers evidence to show that his or her interpretation of the terms contract is the same as the interpretation offered by the drafter of the standard form contract that the parties used, then that drafter's intent is relevant to the search for the meaning of the terms of the parties' contract. However, a court will not look only at evidence of what one party believes the contract's terms are. Rather, a court will go on to ask what whether the other party to the contract can reasonably be expected to agree with the form drafter's interpretation of what the contract's terms are.[2] If the party against whom enforcement is sought claims not to accept the form drafter's interpretation of the contract language, then the court will need to determine what the objective meaning of the language of the contract is, and whether the parties agreed to that objective meaning.[3]

---

[1] "The prophecies of what the courts will do in fact, and nothing more pretentious, are what I mean by the law." Oliver Wendell Holmes, The Path of the Law, 10 Harv. L. Rev. 461 (1897).

[2] This is known as the "objective theory of contract" under US law. See Restatement (Second) of Contracts, § 2(1) ("A promise is a manifestation of intention to act or to refrain from acting in a specified way, *so made as to justify a promisee in understanding that a commitment has been made*." (emphasis added)).

[3] As Judge Learned Hand explained, "A contract has, strictly speaking, nothing to do with the personal, or individual, intent of the parties. A contract is an obligation attached by the mere force of law to certain acts of the parties, usually

It is also important to distinguish between the consequences of a programmer possibly creating a GPL derivative work by combining both GPL and proprietary software at the same time, and a mere passive licensee creating the same work in the RAM of a computer by running the software. This is because while both a programmer and a licensee of proprietary software might be bound to the terms of the GPL, only the programmer will have the power to change the status of the source code of the proprietary program and thus the ability to comply with the terms of the GPL. A mere passive licensee of a program written by a third party and distributed under a standard proprietary software license will not be given the power to change the status of the source code of the non-GPL program from proprietary to open source under the terms of a standard end-user license.[4]

I. When a program (e.g. application program) statically links with a GPL program (e.g. a library program), should the whole executable program be covered by GPL?

If so, is this because a license is required because the whole executable program is deemed to be the derivative work under the US Copyright Act or the GPL program, and the copyright of the author of the GPL program extends to it?

Or, is it because you must observe the term of the GPL (Section 2) since you have agreed to it?

If it is assumed that user of the GPL program has a valid license to the GPL program before the application program statically links with the GPL program, then the question depends on whether a "derivative work" is created in the RAM of the computer running the two programs. US copyright law defines "derivative work" in the following terms: "[A] work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a "derivative work"."[5] However, the GPL uses the term "derivative work" in a slightly different way than the Copyright Act.

In *De-bugging Open Source Software Licensing*,[6] one of the authors of this piece described the GPL terms governing derivative works in the following terms:

---

words, which ordinarily accompany and represent a known intent." Hotchkiss v. National City Bank, 200 F. 287, 293-94 (S.D.N.Y. 1911).

[4] "Microsoft fears that if GPL code is combined with or incorporated into Microsoft proprietary code, this would make Microsoft code also subject to the open license and the person creating the derivative would then be required to open source the Microsoft code with the incorporated GPL code. In fact, however, because a licensee cannot grant greater rights than he received from the licensor, a Microsoft licensee could not by combining the Microsoft code with GPL code lawfully release the Microsoft code under the GPL. Were it otherwise, anyone could simply incorporate some GPL code in Windows, and then claim Windows had become GPL code against Microsoft's wishes. Thus, Microsoft's anti-copyleft license restriction is not necessary, but it nevertheless precludes the licensee from using Microsoft software or distributing it "in conjunction" with any open source or free software." Christian H. Nadan, Open Source Licensing: Virus or Virtue?, *10 Tex. Intell. Prop. L.J. 349, 373* (2002).

[5] 17 U.S.C. § 101.

[6] 64 U. Pitt. L. Rev. 75 at 88 (2002).

GPL Section 2's complex license grant and the conditions that follow it have vexed many a reader of the GPL.[7] One vexing aspect of the license grant[8] is ascertaining whether it grants a license only to modify software or whether it grants a broader license to create any sort of derivative work. Another vexing issue centers on the breadth of the license condition that requires licensees to provide their source code to anyone who wants it at no charge.[9]

(i) Scope of License. The license grant in Section 2 provides that: "You may modify your copy or any copies of the Program or any portion of it, **thus forming a work based on the Program** . . ."[10] Does it grant only the right to modify a program, or a broader right to create any derivative work?[11] Several things argue for the narrower grant.[12] Section 2's "thus creating" wording seems simply to be reminding the reader that a modification is a type of work[13] based on a program.[14] The GPL defines the term "work based on a Program" to mean "either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."[15] In addition, two of the license conditions in Section 2 speak in terms of "the modified program" and "the modified file" when describing what the condition applies to. Similarly, the wording that appears in the three paragraphs following Section 2's license conditions also speaks in terms of "the modified work."

However, there is wording that supports the interpretation that Section 2 is granting a broad right to make derivative works. The part of the license grant that describes distribution rights says that the licensee has the right to distribute "such modifications **or** work, "[16] suggesting the license grants the right to create both a modification and a work based on the program. Also, the license condition in Section 2(b) applies to more than modifications of the Program; it also applies to a program that "contains" or is "derived from" a licensed program. Further, the GPL's Preamble suggests that the GPL grants a broad license to create derivative works. The Preamble says that the GPL was designed so that programmers can "change[17] the software **or** use pieces of it in new programs."

---

[7] *See* Jeff C. Dodd & Brian Martin, *Building A Cathedral Over the Bazaar*, Doing Business Online (2000); Wendy C. Freedman, *Open-source vies with classic IP model*, NAT. L.J. March 13, 2000, at B14; *Some Questions Every Business Should Ask About the GNU General Public License GPL* (visited Jan. 17, 2002) <http://www.microsoft.com>. *See also* Bezroukov, *Dynamic Licensing, supra* note 9 (hacker expressing disappointment that Richard Stallman "chose to ignore" a list of questions Microsoft had prepared about the GPL).

[8] GPL §2, paragraph 1.

[9] GPL §2(b).

[10] GPL §2.

[11] One would expect the license grant to say something like: "You may create a work based on a Program or any portion of it."

[12] *See* 17 U.S.C. §101 ("modifications" are one type of derivative work).

[13] Section 2 is saying that a "modified program" is "a work based on a Program." However, the definition of a "work based on a Program" already encompasses modified programs--a modified work is a subset of all derivative works. *Id.*

[14] For instance, simply combining two programs would create a work based on the program, even though neither program is modified in the process. *See* GPL Q&A.

[15] *Id*. §0.

[16] The reference to "work" presumably refers to the defined term "work based on the Program."

[17] In other words, "modify."

(ii) Scope of 2(b) License Condition. Section 2(b) is at the heart of copyleft. This section requires a developer to publish his or her source code for free use—and in this case (with apologies to Mr. Stallman) "free" refers to both free speech and free beer. This section also makes the GPL a "viral" license because it allows one developer to impose GPL license terms on another developer's code. Section 2(b) says that a licensee "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."

What is the breadth of the condition? The Section 2(b) conditions apply to both works "derived from" other works and works that "contain" other works. A work that contains another work is usually characterized as a derivative work. Does the GPL's use of both words mean that it is treating "derived works" as something different and potentially broader than copyright derivative works? Does "derived from" mean derived from non-copyrightable aspects of the program, such as ideas or data? If so, Section 2(b)'s license condition may apply to programs derived from miniscule amounts of code or non-copyrightable code that would not otherwise make the host program a derivative work according to copyright law. It is possible that the GPL merely seeks to clarify and emphasize that this condition covers certain types of derivative work, namely works contained in other works. Unfortunately, the intent of the GPL is murky.

Following the license grant and conditions in GPL Section 2, three paragraphs ("Explanatory Paragraphs") attempt to clarify the license grant and conditions, especially the critical distinction in Section 2(b) between derivative and collective works on the one hand, and separate and independent works on the other. Reading these paragraphs is like attempting to tune in a distant radio station—moments of clarity followed by moments of distortion. The Explanatory Paragraphs say that if a programmer creates a modified work based on the licensed program and if identifiable parts of that modified work are not derived from the licensed program and if those part of the modified work can be "reasonably considered independent and separate," and if those parts of the modified work are distributed as a separate work that would not qualify as a derivative work of the licensed Program, then the GPL does not apply to those parts of the modified work in that context.

Attempting to clarify this provision further, the GPL states: "it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather the intent is to exercise the right to control the distribution of derivative or collective works based on the Program." It also states: "mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of storage or distribution medium does not bring the other work under the scope of this License."

The Explanatory Paragraphs are relatively helpful—at least as helpful as they can be under the circumstances. They track the intent of the Copyright Act closely. Even so, the distinction they try to make is complex and extremely fact intensive in the software context. FSF's GPL Question & Answer document illustrates this well when it concludes that often the distinction comes down to how "intimate" two programs are with one another. [18] The difficulty comes less from ill chosen

[18] *See* GPL Q&A discussion of what makes a mere aggregation.

words than from making such serious consequences turn on a distinction that is inherently uncertain.[19]

Even if one could draw the line with certainty, serious issues remain. Software engineering practices have evolved significantly since the GPL was last revised. Software has become more complex. This bears profoundly on the interpretation of the terms "independent work" and "separate work," terms which are terms that establish whether a work is subject to GPL Section 2(b). Use of object oriented programming and remote procedure calls make the concepts of independent and separate work more difficult to define. Technology is becoming more connected not more separate.[20]

In addition, programs that used to be "reasonably considered independent" are no longer so considered. Directories, TCP/IP stacks, fonts, spell checkers, clip art, file systems, compression software, and perhaps even operating system kernels[21] may or may not be independent works, depending upon how they are packaged or configured. Certain issues specific to the development of embedded software complicate the distinction between separate and combined works.[22] Interpreted programs, such as those written in Java, present unique GPL interpretation issues.[23] As one commentator noted: In an era of category-blurring inventions such as document macro-procedures, how are license-covered programs to be distinguished from non-licensed configurations and data files?"[24]

A. Form Drafter's View

Applying these insights to the question of whether a derivative work is created when an application program statically links to a GPL program such as a library program, it seems unclear what the answer would from Stallman's point of view.[25] This would depend on how "intimate" the

---

[19] The consequence is that a programmer's code becomes freely available to anyone free of charge whether the programmer desires that result or not. It is one thing to freely volunteer for the free software army, but forced conscription is another matter.

[20] One of the most troublesome aspects of the GPL is its failure to account clearly for linking and other forms of communication between applications software and systems software. FSF seems to take a broad view of the types of communication that trigger the copyleft aspects of the GPL. Contrast FSF's approach, however, to that of Linus Torvalds. Mr. Torvalds takes the position that making systems calls to the Linux kernel does not trigger the copyleft aspects of the GPL. Mr. Torvalds' clarifying note to the GPL as applied to Linux 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.' Linus Torvalds."

[21] *See id*.

[22] *See* Farhad Manjoo, *Open Source's Dot-Net Less Open*, WiredNews (Jan. 28, 2002) http://wired.com. Embedded software developers Wind River and Caldera have expressed concern with the uncertain viral effects of the GPL.

[23] See GPL Q&A.

[24] *See* Moen, *A public discussion of open source licensing*, *supra* note 37.

[25] Frequently Asked Questions about the GNU GPL, What is the difference between "mere aggregation" and "combining two modules into one program"? states:

Mere aggregation of two programs means putting them side by side on the same CD-ROM or hard disk. We use this term in the case where they are separate programs, not parts of a single program. In this case, if one of the programs is covered by the GPL, it has no effect on the other program.

Combining two modules means connecting them together so that they form a single larger program. If either part is covered by the GPL, the whole combination must also be released under the GPL--if you can't, or won't, do that, you may not combine them.

two programs are, or whether they can reasonably be considered to remain two independent programs. Stallman believes that the GPL does not apply to programs running in the RAM of a computer even though both GPL and proprietary software are running at the same time if, in light of their functional characteristics, they are "separate works." Therefore, Stallman might consider that the whole executable program created when an application program runs with a static link to a GPL program also running at the same time would not be covered by the GPL because the two programs together would not create a single "derivative work." However, the existence of the Lesser GPL, which permits the use of open source libraries with proprietary software, and the ordinary GNU GPL, which does not, tends to indicate that Stallman is more likely to come down on the side of finding that a derivative work subject to the GPL had been created.

B. Court's View

1. Programmer's derivative work

If a court were to look at a situation involving a programmer writing proprietary code that requires a static link to a GPL program in order to run, the court would ask whether the programmer had assented to the terms of the GPL. Manifestation of assent on the part of the programmer could be established by showing that an experienced programmer worked with object code well known in the software development community to be subject to the terms of the GPL.[26]

Assuming the programmer's assent could be shown, a court would next ask whether under the terms of the GPL, writing a program that required a static link to a GPL program would require that the programmer's own proprietary code program become subject to the GPL, notwithstanding the lack of her explicit agreement to subject her to the terms of the GPL. This would require the court to identify the objective meaning of the GPL in this context. If both the licensor of the GPL program and the programmer who wrote the application program believe that the combined executable program remain two separate works, not a derivative of the GPL program, this information might be highly persuasive. In the event the GPL licensor and the programmer disagreed, however, the court would have to decide what Section 2(b) of the GPL means in this context. The choice to use the ordinary GNU GPL instead of the Lesser GPL would indicate that the licensor may expect the proprietary software to become subject to the GPL. On the other hand, if Stallman's interpretation of the meaning of the ordinary GNU GPL is out of step with what the majority of software developers think, then a court would be justified in finding that a derivative

---

What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).

If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program.

By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as combined into a larger program.

Available at http://www.gnu.org/licenses/gpl-faq.html

[26] "[M]any experienced programmers would be hard-pressed to claim that they did not suspect that the code was subject to the GPL, or that their failure to look for the GPL notice was reasonable." Nadan at 367.

work based on the open source program was not created, or that even if it was, that its creation did not subject the proprietary program to the terms of the GPL.

However, if a court found that the whole executable program was a derivative work based on the GPL, then the programmer's rights to create this derivative work and also to run the GPL program are conditioned on the programmer meeting the terms of the GPL. This would create an obligation on the programmer to release the source code of the application program under the GPL. If the programmer failed to do so, the programmer would not have the right to create the derivative work or to run the GPL program, having not met the condition subsequent regarding distribution of derivative works set out in the GPL license.[27] One of the maxims of English law that is followed by US courts today is that "Equity abhors a forfeiture." In this context, this means that a court would try hard to avoid a reading of the GPL that would produce a forfeiture for the programmer, provided that the programmer's behavior was fair and reasonable under the circumstances.

2. Passive Licensee's derivative work

The problem of whether a mere passive licensee of a GPL work has manifested assent to the terms of GPL may be more difficult than the question of whether a developer has manifested assent, especially if the mere passive licensee is not sophisticated about how software is produced and distributed. In the case of a mere passive licensee of an application program distributed under a standard end-user license for proprietary software, a court could find that no derivative work had been created, or even if it had, that there was no requirement that the proprietary program be distributed subject to the GPL. However, if a court were to find that a mere licensee's use of both the GPL library and a proprietary program subjected the proprietary program to the terms of the GPL, the forfeiture problem reappears even more strongly than it did in the case of a developer. This is because the licensee cannot under any circumstances comply with the condition subsequent contained in the GPL that requires whoever creates a derivative work using a GPL program in turn to distribute any programs that combine with it under the GPL. On the other hand, if the mere passive licensee's use of the two programs together does not constitute "distributing" or "publishing" a work derived from the GPL program, then the mere passive licensee is not required to insure that the proprietary program is licensed under the GPL.

II. When a program dynamically links with a GPL program, should the program be covered by the GPL?

A dynamic link makes the relationship between the two programs more "intimate" than a static link. The answer from Stallman's perspective would clearly seem to be that the derivative work would be covered by the GPL, and so the proprietary software program that linked with the GPL program should also become subject to the GPL.

A court would consider the same factors listed above in the discussion of the static link between the programs, such as the fact that the licensor had a choice between both the ordinary GNU GPL and the Lesser GPL, and chose the former, which would indicate that the proprietary program should be covered by the GPL. However, if the licensee could show that the broad reading of the scope of the GPL is controversial and not generally accepted even in the software

[27] Nadan at 370.

development community, then it is less likely that the proprietary program would be subject to the GPL.

III. Should a dynamic loadable module of the Linux kernel which is licensed under the GPL be covered by the GPL?

A dynamic loadable module of the Linux kernel would be even more "intimately" linked with the GPL application than a library that is dynamically linked. Stallman would argue that such an application should be subject to the GPL; others in the software development community might disagree. Vendors of proprietary software might completely reject this analysis.

The problem is that as software programming becomes more modularized, then the distinction between what are separate applications becomes more blurred. Under Stallman's interpretation of the GPL, this trend in programming will tend to increase the scope of the GPL and the volume of software that must be distributed under the terms of the GPL. From the point of view of a court trying to determine the objective meaning of a GPL license in the context of a particular dispute, this creeping expansion in the scope of the GPL is a reason to interpret the GPL more narrowly if necessary in order to fulfill the expectations of the parties.

IV. Should a device driver, which often takes the form of dynamic loadable module, be covered by GPL?

The analysis is the same as III, above.

V. Should the application programs of Linux be covered by the GPL?

The analysis is the same as III, above.

VI. Mr. Linus Tovalds, copyright holder of the Linux kernel, has declared that application programs of Linux need not be covered by the GPL. Should they be covered by the GPL without his declaration?

Here the relevant form drafter is Linus Torvalds, not Richard Stallman, and he has made his interpretation explicit in the following note that is included with the Linux kernel source code: 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.' Since Torvalds' interpretation eliminates most of the confusion and controversy surrounding the application of the ordinary GNU GPL to Linux, it is very likely that a court would adopt his interpretation.

VII. It is supposed that an embedded system using Linux takes the form of one executable program including application programs. Should it be covered by the GPL as a whole?

Embedded system usually refers to the software embedded in "smart goods" such as mobile phones. In these systems, the distinction between an operating system and an application is even more blurred than in a conventional computer running software designed as modules. This

tendency for the operating system to merge with applications would tend to expand the scope of the GPL under Stallman's interpretation of it, and to encourage a court to read the terms of the GPL restrictively if necessary to avoid surprise and unfairness to the parties.

VIII. When using the C or C++ language to write programs, "header files" which define macros and data structures are needed. If header files distributed under the GPL are used, should the program be covered by the GPL?

The analysis is the same as III, above. In addition, since header files are used primarily to facilitate communication between applications written in different programming languages, Mr. Stallman might agree that the GPL does not cover the program.