Chapman Law Review

Volume 13 | Issue 1 Article 5

2009

Testing Open Source Waters: Derivative Works Under GPLv3

Joseph A. Chern

Testing Open Source Waters: Derivative Works Under GPLv3

Joseph A. Chern[*]

INTRODUCTION

Richard Stallman[1] has always believed that sharing software is a moral obligation.[2] As such, in 1985, Stallman pioneered a movement towards free software and created the non-profit Free Software Foundation ("FSF"), a tax-exempt charity for free software development.[3] The FSF, in turn, created the GNU General Public License ("GPL") as an instrument to enforce free software principles.[4] Although this license has become one of the most popular licenses in the world,[5] legal uncertainties continue to plague its enforceability.[6]

Among the debated issues of the license, the FSF maintains that dynamically linking to a GPL-ed library creates a derivative work under copyright principles subject to the original license.[7]

[*] J.D. Candidate 2010, Chapman University School of Law; B.S. Computer Engineering, University of California, San Diego. I would like to thank Ryan Roemer for all of his constructive guidance and insight; Pamela Frohreich for her comments and encouragement; Kyhm Penfil for her edits and suggestions; and the Senior Articles Editors of the Chapman Law Review for all of their hard work. I also would like to thank my family for their love and support which made this possible.

1 Software developer of GNU operating system and founder of Free Software Foundation. *See* Richard Stallman, *Richard Stallman's Personal Home Page*, http://www.stallman.org/#serious (last visited Oct. 7, 2009).

2 Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 445 (2005) (describing Richard Stallman's belief that proprietary software was incompatible with "the golden rule").

3 Free Software Foundation, *About Free Software and the GNU Operating System*, http://www.fsf.org/about/ (last visited Oct. 7, 2009).

4 Free Software Foundation, *What is free software and why is it so important for society?*, http://www.fsf.org/about/what-is-free-software (last visited Oct. 7, 2009).

5 Free Software Foundation, *Licenses*, http://www.gnu.org/licenses/licenses.html#GPL (last visited Oct. 7, 2009) (reporting that the "GNU General Public License . . . is used by most GNU programs, and by more than half of all free software packages.").

6 This article does not address all the legal issues with the GPL, but focuses on the issue of program linking. *See infra* Part IV.A; *see also* John Tsai, *For Better or Worse: Introducing the GNU General Public License Version 3*, 23 BERKELEY TECH. L.J. 547, 548 (2008) (analyzing other issues with GPLv3 including software patents and digital rights management).

7 David Turner, *LGPL and Java*, (2004), http://www.fsf.org/licensing/licenses/lgpljava.html.

There is, however, difficulty with this position because dynamic linking technically does not create a "copy" of any GPL-ed software.[8] This article analyzes perceived arguments both for and against the FSF position. The legal validity of these arguments will be evaluated in relation to the current approach courts have taken with software copyright. This article argues that the FSF most likely does not have legal support for its position that dynamic linking to a GPL-ed library creates any GPL obligation. The implications of this holding are also examined and alternative consequences are considered.

Part I of this article discusses the development of the FSF and explores the history of the GPL. Part II briefly describes the application of the GPL and introduces the FSF's interpretation of the extent of that application. Part III explains the technical complexity of computers, describes how copyright law has attempted to address the complex nature of software, and revisits the FSF's interpretations in depth. Finally, Part IV analyzes perceived legal issues with the FSF position regarding current copyright law and examines alternative outcomes.

I. BACKGROUND

Before examining the legal issues with dynamic linking, it is helpful to understand the rationale behind the GPL and the free software movement.

A. Beginning of Free Software

The motivation for the free software movement began in 1970 when Stallman worked as a programmer for the Massachusetts Institute of Technology's (MIT) Artificial Intelligence Lab.[9] Frustrated with a paper jam in one of the lab printers, he decided to take action.[10] Using the source code[11] freely provided with the centralized network printer, he modified the source code so that other users waiting would be alerted when the printer was jammed.[12] Eventually, the lab received a new Xerox printer, and Stallman tried to make the same

8 ABRAHAM SILBERSCHATZ ET AL., OPERATING SYSTEM CONCEPTS (8th ed. 2009).

9 SAM WILLIAMS, FREE AS IN FREEDOM: RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE 1–12 (2002), *available at* http://www.oreilly.com/openbook/freedom.

10 *Id.* at 3.

11 In computer programming, source code (also known as source or code singularly) is a collection of *human readable* statements and declarations that allow programmers to communicate with computers. Eventually, source code is compiled to object code, which makes up programs such as Microsoft Word, an Internet Browser, or a game. For more information on source code and an example of the first source code written in the C programming language, see BRIAN W. KERNIGHAN & DENNIS M. RITCHIE, THE C PROGRAMMING LANGUAGE 1–7 (2nd ed. 1988).

12 WILLIAMS, *supra* note 9, at 3–5.

improvements; however, Xerox did not provide the printer's source code.[13] Xerox supplied the hardware,[14] but denied the release of their source code under contractual obligations.[15]

Consequently, Stallman set out to challenge the direction of software culture. In 1984, Stallman left MIT and wrote a complete operating system[16] compatible with UNIX, the standard operating system at the time, and eventually called it "GNU."[17] In 1985, Stallman founded the FSF to promote the project's concept of software freedom.[18] The original focus of Stallman and the FSF was to bring a wholly free operating system into existence to give users an opportunity to inspect, share, and modify the source code of the software they used.[19] According to the FSF, open source software hinges on the four software freedoms: (1) to run a program for any use; (2) to study the code and modify it for individual needs; (3) to redistribute; and (4) to improve a program for society.[20] It is important to note that the

13 *Id.* at 4–6.

14 Hardware, as distinguished from software, is the general term to describe the physical components of technology such as the components that make up a personal computer, including a printer. Software is the general term to describe the collection or programs, which function to tell hardware how to act. *See* WALTER SAVITCH, JAVA: AN INTRODUCTION TO COMPUTER SCIENCE & PROGRAMMING 4–5 (3d ed. 2004).

15 WILLIAMS, *supra* note 9, at 6.

16 An operating system is the software component of a computer system, which acts as a "manager" of the resources in the computer to coordinate between the hardware and software applications. *See* SAVITCH, *supra* note 14, at 8. Common operating systems are Windows for PC's and Tiger for Apple computers.

17 Free Software Foundation, *Overview of the GNU Operating System*, http://www.gnu.org/gnu/gnu-history.html (last visited Oct. 7, 2009) ("The GNU operating system is a complete free software system, upward-compatible with Unix. GNU stands for 'GNU's Not Unix'.").

18 What is free software and why is it so important for society?, *supra* note 4 ("in 1985, Stallman started the Free Software Foundation, a nonprofit with the mission of advocating and educating on behalf of computer users around the world.").

19 Free Software Foundation, *Philosophy of the GNU Project*, http://www.gnu.org/philosophy/philosophy.html (last visited Oct. 7, 2009) (describing the "philosophy of the free software movement, which is the motivation for our development of the free software operating system GNU"); *see also* Eben Moglen & Richard Stallman, *GPL Version 3: Background to Adoption*, http://www.fsf.org/licensing/essays/gpl3-background.html (last visited Oct. 7, 2009).

20 The Free Software Foundation defines the four essential freedoms:

> Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it means that the program's users have the four essential freedoms: The freedom to run the program, for any purpose (freedom 0). The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbor (freedom 2). The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Free Software Foundation, *The Free Software Definition*, http://www.gnu.org/philosophy/free-sw.html (last visited Oct. 7, 2009).

movement's goals are "primarily social and political, not technical or economic."[21] The term "free" did not mean monetarily free, but rather "free" to study and modify.[22] FSF's often-quoted catchphrase is "'free' as in 'free speech,' not as in 'free beer.'"[23]

B. General Public License, Versions 1–3

When Congress modified the Copyright Act to explicitly protect software programs,[24] the FSF became concerned about its efforts to promote shared source code since battling a copyright holder required great time and resources.[25] As a result, "companies, not hackers, held the automatic advantage[,]"[26] and provider corporations effectively could hold a monopoly on their software. Software typically builds upon the source code from previous programs;[27] therefore, tipping the scales in favor of large corporations places an arduous burden on the FSF's goals. Eventually, the FSF recognized it could actually use copyright licensing to its advantage.[28] A programmer could use a copyright license to bind others, who modify or distribute the programmer's source code, to the license's terms.[29] Specifically, open source programmers,[30] such as Stallman, could license their code with specific conditions, such as to allow others to copy it so long as they also publish any modified versions in return.[31] This prompted the FSF to create a universal license making it possible for many projects to share the same code and keep programs

21 Moglen & Stallman, *supra* note 19.

22 GNU General Public License, version 2, http://www.gnu.org/licenses/oldlicenses/gpl-2.0.html (last visited Oct. 7, 2009) [hereinafter General Public License v2.0]; *see also* Carver, *supra* note 2, at 450–53 (using the term "open source" for software that refers to an open development process).

23 FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 41 (Joshua Gay ed., 2002).

24 Act to Amend the Patent and Trademark Laws, Pub. L. No. 96-517, 94 Stat. 3015, 3028 (1980) (codified as amended at 17 U.S.C. § 117 (2006)); *see also* discussion *infra* Part III.B.

25 WILLIAMS, *supra* note 9, at 123.

26 *Id.*

27 SAVITCH, *supra* note 14, at 12.

28 WILLIAMS, *supra* note 9, at 123–26.

29 *Id.*

30 Open source programmers are generally defined as computer software programmers who use and make software readily available to the public domain. Open source software is to be distinguished from proprietary software. *See* Open Source Initiative, The Open Source Definition, http://opensource.org/docs/osd (last visited Oct. 7, 2009) (describing ten criteria for open source software).

31 WILLIAMS, *supra* note 9, at 124.

free.[32] Thus, the first version of the GNU General Public License ("GPLv1") was born.[33]

GPLv1 attempted to solve two major problems that restricted the freedoms of "free software."[34] First, many software companies released only binary versions—executable, but not modifiable or understandable by humans—of their source code.[35] GPLv1 solved this problem by requiring the release of human readable source code under a GPLv1 agreement.[36] Second, like the FSF, programmers could also benefit from copyright protection when they modified source code and redistributed it with additional restrictions to the existing license.[37] GPLv1 did not allow for combinations with software that might impose additional restrictions, because this would create a conflict.[38]

Two years later, the FSF published the second version of the license called GNU GPL version 2 ("GPLv2").[39] The underlying idea behind the second version was what the FSF calls "copyleft."[40] A play on the phrase copyright, "copyleft" promoted rather than restricted the use of copyrighted materials.[41] This change was best embodied in Section 7 of the GPLv2, which states that if a user has restrictions—such as court orders, agreements, or otherwise—placed on GPL-covered software that may hinder the rights of others, the user cannot distribute that code at all.[42] Most importantly, "copyleft" principles subjected the author of a GPLv2 piece of software to two requirements: (1) to license the original and modified versions of the software under the same license; and (2) to provide the source code to any user of the license.[43] These provisions will be essential to the derivative work issue discussed below.[44]

32 Robert W. Gomulkiewicz, *General Public License 3.0: Hacking the Free Software Movement's Constitution*, 42 HOUS. L. REV. 1015, 1024 (2005).

33 The full text General Public License, version 1 is available at http://www.gnu.org/licenses/old-licenses/gpl-1.0.html (last visited Oct. 7, 2009) [hereinafter General Public License v1.0].

34 *Id.*

35 FREE SOFTWARE, FREE SOCIETY, *supra* note 23, at 4, 170–71.

36 General Public License v1.0, *supra* note 33.

37 *Id.*

38 *Id.*

39 Gomulkiewicz, *supra* note 32, at 1025–26 (reporting that Stallman created the second version to clarify misunderstandings and worries).

40 *See* Free Software Foundation, *What is Copyleft?*, http://www.gnu.org/copyleft/copyleft.html (last visited Oct. 14, 2009).

41 *Id.*

42 General Public License v2.0, *supra* note 22, at § 7.

43 *Id.* at §§ 2–3.

44 *See* discussion *infra* Part IV.A.

After fourteen years, GPLv2 began to show its age in the form of many legal problems.[45] As a result, the final version of GPLv3, released on June 29, 2007, attempted to address some of the legal issues from the previous version while maintaining the core "copyleft" obligations.[46] Among the significant issues, GPLv3 tried to clarify the scope of the license by introducing newly defined terms and consolidated requirements.[47] GPLv2 borrowed many terms from U.S. copyright law, such as "derivative works" and "collective works",[48] but this only led to ambiguities about how far the license should reach.[49] Furthermore, because the GPL is meant to be a global license that can apply in any country regardless of that country's copyright laws, the GPLv3 abandoned the term "derivative work" and redefined the license to meet global standards.[50] The new license also includes "propagate" and "convey" to define what constitutes distribution[51] and tries to clarify whether dynamic linking falls within the scope of the "copyleft" license.[52]

II. SOFTWARE AND GPL

The resolution of the legal issues with GPL greatly depends on the FSF's ability to advocate for definitions of legal terms as applied to software. Such definitions have yet to be evaluated by the courts. In addition, the unique nature of how a GPL is triggered can effectively limit its scope. This section first describes that unique nature and then previews the FSF's legal interpretations. A more in-depth discussion of the FSF's interpretation is revisited in Section IV.C, following a discussion of current case law.

45 Moglen & Stallman, *supra* note 19.

46 Not all of the legal issues are addressed here since the list could go on, but rather the most significant issues to this article will be discussed.

47 *Compare* Free Software Foundation, *GNU Project: GNU General Public License Version 3*, http://www.gnu.org/licenses/gpl-3.0.html (last visited Oct. 14, 2009) [hereinafter General Public License v3.0], *with* General Public License v2.0, *supra* note 22.

48 *See* General Public License v2.0, *supra* note 22, at §§ 0, 2(c); 17 U.S.C. § 101 (2006).

49 Carver, *supra* note 2, at 458–60.

50 Moglen & Stallman, *supra* note 19; *compare* GPLv3, *supra* note 47, at § 0, *with* GPLv2, *supra* note 22, at § 0.

51 General Public License v3.0, *supra* note 47, at § 0.

52 Tsai, *supra* note 6, at 566.

A. Triggering the GPL: "Distribution" and the "Viral" Effect

Pursuant to one of the "copyleft" provisions, modified versions[53] of GPL-ed software are subject to the same license as the original code[54] if released.[55] Consequently, critics of the license have described it as having a "viral effect."[56] The effect of the license is metaphorically similar to that of a virus because in order to trigger the effects of the GPL, the only requirement is that any portion of the software be covered by the GPL.[57] If any portion of the software is covered by a GPL license, the remaining portions of the code, even if it was intended to be proprietary, become "infected" with the license.[58]

Distributing any software under the license can cause this viral effect.[59] In order to use the license for a piece of software, the author only has to add two elements to his code: (1) "a copyright notice (such as 'Copyright 1999 Terry Jones');" and (2) "a statement of copying permission, saying that the program is distributed under the terms of the GNU General Public License."[60] When these two elements are present, any distribution of that code will be subject to the same terms of the GNU GPL.[61] For instance, assume software Program A contains the two elements and, as a result, is covered under the GPL. Anyone would be free to use Program A to create Program B. Accordingly, Program B is "infected" by the GPL of Program A

53 "Modified versions," as used here, is similar to the "derivative work" concept that was replaced from GPLv2. The term here assumes the work falls within the GPL definition of modified work.

54 *See* Free Software Foundation, *supra* note 40.

55 Among frequently asked questions, the Free Software Foundation provides:

> [t]he GPL does not require you to release your modified version, or any part of it. You are free to make modifications and use them privately, without ever releasing them . . . . But if you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the program's users, under the GPL. Thus, the GPL gives permission to release the modified program in certain ways, and not in other ways; but the decision of whether to release it is up to you.

Free Software Foundation, *Frequently Asked Questions About the GNU Licenses*, Does the GPL require that source code of modified versions be posted to the public?, http://www.fsf.org/licensing/licenses/gpl-faq.html#GPLRequireSourcePostedPublic (last visited Oct. 24, 2009) [hereinafter FSF Frequently Asked Questions].

56 Ron Phillips, *Deadly Combinations: A Framework for Analyzing the GPL's Viral Effect*, 25 J. MARSHALL J. COMPUTER & INFO. L. 487 (2008).

57 General Public License v3.0, *supra* note 47, at §§ 0, 10 (describing the viral effect by automating licensing to any downstream recipients for works that are covered, which means any unmodified programs or worked based on the program).

58 Phillips, *supra* note 56, at 492.

59 *Id.* at 492–93; General Public License v3.0, *supra* note 47, at §§ 0, 10 (describing the viral effect by automating licensing to any downstream recipients for works that are covered, which means any unmodified programs or worked based on the program).

60 Free Software Foundation, *How to use GNU licenses for your own software*, http://www.gnu.org/licenses/gpl-howto.html (last visited Oct. 24, 2009).

61 *Id.*

and must, at a minimum, have the same terms. In a simple situation, Program B typically contains a verbatim copy of Program A, which makes the license application a legal, rather than a technical, matter. In any case, the "viral effect" helps to achieve the goals of the free software movement by ensuring reciprocation of free software. It is important to note that neither private use nor modifications of a GPL code creates any licensing obligation, but obligations do apply if the code is publicly distributed.[62]

B. FSF's Interpretations of Derivative Works

The FSF has been less than clear on its definition of what creates a derivative work. However, in order to promote the free software movement and the goals of a GPL, it is important to draw the line between "infected"[63] derivative works and works that are not "infected" that carry no "copyleft" obligations.

One thing is clear: the FSF has abandoned the term "derivative work" and has adopted "modified works" essentially to cover the same principles.[64] Although "modified works" is defined in the latest GPL,[65] the scope of the definition remains ambiguous and is still debated.[66] One possible interpretation of the FSF definition of "modified works" is that the definition mirrors the scope of the U.S. copyright law's definition of derivative work.[67] However, the drafters of the GPL seemingly urge a broader interpretation.[68] When answering questions

62 FSF Frequently Asked Questions, *supra* note 55.

63 *See supra* Part II.A.

64 *See supra* Part I.B; *compare* General Public License v3.0, *supra* note 47, at § 0 ("The resulting work is called a 'modified version' of the earlier work or a work 'based on' the earlier work") *with* General Public License v2.0, *supra* note 22, at § 0 ("The 'Program,' below, refers to any such program or work, and a 'work based on the Program' means 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 . . . ").

65 The Free Software Foundation defines modify as:

> To 'modify' a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a 'modified version' of the earlier work or a work 'based on' the earlier work.

General Public License v3.0, *supra* note 47, at § 0.

66 Tsai, *supra* note 6, at 548, 566–68 (arguing that new language creates new ambiguities and the open source community is still split over several key issues as to what is covered under GPL licenses).

67 17 U.S.C. § 101 (2006) ("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.'"); *see also* copyright discussion *infra* Part III.B.1.

68 University of Washington School of Law, Derivative Works, http://www.law.washington.edu/lct/swp/Law/derivative.html (last visited Dec. 28, 2009) (arguing GPL drafters urge a broader definition of derivative works than US Copyright law); *see also infra* Part III.C; *see also* Tsai, *supra* note 6, at 555–56 (outlining a broader interpretation of the GPLv2 to include both collective works and compilations).

about whether programs are covered under the license, the FSF generally leans towards the answer that provides the broadest coverage.[69] Specifically, when dealing with a situation in which two separate programs are combined,[70] each with a different license, the FSF would argue that in most cases the programmer/user would be "combining them into one program" under the GPL.[71] However, the FSF has admitted that "[t]his is a legal question, which ultimately judges will decide."[72]

III. PROGRAM LINKING AND COPYRIGHT

The FSF's acknowledgment that courts ultimately will decide the scope of GPL licensing suggests the need for an analysis of the current legal interpretations of copyright protection by courts. Section III.A begins with a technical primer on software linking that may affect a court's treatment of copyright protection for software. Section III.B examines traditional copyright law and basic copyright principles and analyzes courts' approaches to derivative works. Section III.C revisits the FSF's interpretation of "modified works" and focuses on the role software linking might play in this analysis.

A. Program Linking (Static and Dynamic): Technical Background

Developers of software often use pre-written code[73] and data when writing their own programs to avoid "reinvent[ing] the wheel."[74] Software "libraries" are commonly used collections of reusable code and data that provide "common functions, such as printing, reading data from a disk, or opening a file."[75] A computer will at some point "link" the relevant code from the library with the developer's own code to create a larger functional

69 FSF Frequently Asked Questions, *supra* note 55 (asserting adding modules to GPL programs are also covered, output from GPL covered programs can be covered by GPL, linking to libraries is also covered); *see also infra* Part III.C.

70 *See* discussion *infra* Part III.A.

71 FSF, Frequently Asked Questions, *supra* note 55.

72 *Id.*

73 *See* SAVITCH, *supra* note 14, at 12, 18 (stating programs are not typically written as one piece of code and may consist of several different parts, written by different people, combining other components that already exist).

74 Alexandra J. Horne, *Shared Rights to Source Code: The Copyright Dilemma*, 32 SANTA CLARA L. REV. 497, 532 (1992) ("Furthermore, because of the complexity of the development process, the software developer can argue that the library routines must be afforded copyright protection because of the economic hardship the developer will face due to the prohibitive expense required to 'reinvent the wheel' for each new development project.").

75 Tsai, *supra* note 6, at 557.

program.[76] The timing of this linkage depends on whether the process is accomplished through *dynamic* or *static* linking.[77]

In computer science, "human readable source code" typically goes through several steps before it is executed by the compiler.[78] Generally, these steps are divided into three "times": compile time, load time, and execution time.[79] During compile time, the source code is translated into machine code to create an object module that is read by the computer.[80] What occurs during the load and execution times vary depending on hardware and software limitations.[81]

If an operating system supports only *static* linking, the libraries that the source code uses are combined with the object module during load time.[82] Effectively, this is akin to replacing footnotes in an essay with a copy of the referenced material. The libraries are treated like another object module that is physically combined together with the new code into which it was integrated.[83] Conversely, if the operating system allows *dynamic* linking, the linking is "postponed until execution time."[84] With dynamic linking, a "bookmark" is placed in the source code for each library reference, which indicates how to load the library code when it is actually being used.[85] Under this approach, there is no actual copy of the library code in the source code, but rather an address where the code can be accessed when needed.[86] The library and source code remain separate files on disk.[87]

B. Traditional Copyright Law

As copyright law has developed, it has dynamically embraced new technologies. For example, courts,[88] and eventually Congress,[89] agreed that lithographs and photographs constitute

76 SAVITCH, *supra* note 14, at 12 (describing the process of connecting several different pieces of code together as linking).

77 SILBERSCHATZ, *supra* note 8, at 321.

78 *Id.* at 318.

79 *Id.*

80 SAVITCH, *supra* note 14, at 9 (describing a compiler to translate the high-level language that humans can understand [source code] into low-level languages [machine code] that the computer can understand).

81 *See* SILBERSCHATZ, *supra* note 8, at 320.

82 *See* Tsai, *supra* note 6, at 557; SILBERSCHATZ, *supra* note 8, at 321.

83 *See* SILBERSCHATZ, *supra* note 8, at 828.

84 *Id.* at 321.

85 *Id.*

86 *See id.*

87 *See id.*

88 *See, e.g.*, Burrow-Giles Lithographic Co. v. Sarony, 111 U.S. 53, 58 (1883) (holding that there is "no doubt that the Constitution is broad enough to cover an act authorizing copyright of photographs, so far as they are representatives of original intellectual conceptions of the author").

89 17 U.S.C. § 102(a)(5) (2006).

copyrightable subject matter in the early years. More recently, Congress convened the National Commission on New Technological Uses ("CONTU")[90] to thoroughly explore the scope of copyright protection for software, a concept with which courts continue to struggle.[91] This section will introduce relevant principles of copyright law as applicable to software and briefly examine how the courts currently address the changes.

1. Brief Overview of Copyright and Software: Basic Application

Copyright is the right of an author to control the reproduction of his or her original expression.[92] Whether an author owns a valid copyright depends, *inter alia*, on whether the subject matter is copyrightable.[93] Copyrightable subject matter includes "original works of authorship fixed in any tangible medium of expression."[94] Works of authorship can include anything, from literary works such as novels, to musical works such as compositions and lyrics.[95] However, although copyright law protects the expression of ideas,[96] copyrights do not extend to either the functional elements or the ideas underlying those expressions; this is an area typically left to patent law.[97] Because of the functional nature of software, Congress has been reluctant to allow for copyright protection for software. In fact, it was not until 1980 that Congress approved a statutory amendment to define a "computer program" in the Copyright Act.[98] Following the decision in *Apple Computer, Inc. v. Franklin Computer Corp.*, courts have continued to "reflect Congress' receptivity to new

90 NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1978).

91 *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1247 (3d Cir. 1983); *see also* Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996); Ticketmaster Corp. v. Tickets.com, Inc., No. CV 99-7654 HLH (BQRX), 2000 WL 525390, (C.D. Cal. Mar. 27, 2000).

92 ROBERT A. GORMAN & JANE C. GINSBURG, COPYRIGHT CASES AND MATERIALS 12 (7th ed. 2006).

93 17 U.S.C. § 102 (2006).

94 § 102(a); *see also* Feist Publ'n, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 346 (1991) (providing "original works" has not meant novel or unique, but rather independent creations with a minimal modicum of creativity).

95 § 102(a).

96 *Id.*

97 § 102(b) ("In no case does copyright protection . . . extend to any idea, procedure, process, system, method of operation . . . "); *compare* 35 U.S.C. § 101 (2006) (discussing *patentable* inventions).

98 Act to Amend the Patent and Trademark Laws, Pub. L. No. 96-517, 94 Stat. 3015, 3028 (1980) (codified as amended at 17 U.S.C. § 117 (2006)); *see also* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1247 (describing that although computer programs are not expressly listed as a work of authorship, legislative history suggests that programs were considered copyrightable as literary works and software was not *per se* uncopyrightable purely based on functionality).

technology and its desire to encourage, through the copyright laws, continued imagination, and creativity in computer programming."[99]

Software copyrightability allowed programmers to become the beneficiaries of exclusive rights including the right to prepare derivative works.[100] The Copyright Act defines a derivative work as:

> [A] work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgement, 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.'[101]

Historically, "derivative works" have been simpler to identify in areas such as choreography,[102] visual art,[103] and literary sequels.[104] Defining a derivative work in the software context has not been as obvious and finds little clarity in the statutory definition or case law.[105] The recent trend in courts is that "copyright law today, in practice, affords narrow protection to software."[106] This may be because "as software patents gain increasingly broad protection, whatever reasons there once were for broad copyright protection of computer programs disappear."[107] Alternatively, copyright law may find difficulty affording broad protection because of the unique nature of software and the "inherent difficulty of fitting copyright[s] . . . to rapidly changing and variously described software technology approaches."[108]

---

99 *Apple Computer*, 714 F.2d at 1254.

100 17 U.S.C. § 106 (2006) (enumerating the exclusive rights that the owner of a valid copyright has and can authorize, including reproduction, preparation of derivative works, distribution, and public performance of their copyrighted work).

101 17 U.S.C. § 101 (2006).

102 *See, e.g.*, Horgan v. MacMillan, Inc., 789 F.2d 157, 162 (2d Cir. 1986).

103 *See, e.g.*, Lee v. A.R.T. Co., 125 F.3d 580, 581 (7th Cir. 1995).

104 Suntrust Bank v. Houghton Mifflin Co., 268 F.3d 1257, 1259 (11th Cir. 2001).

105 Carver, *supra* note 2, at 458 (noting the lack of case law and statutes to clarify derivative works for software).

106 Emanuela Arezzo, *Struggling Around the "Natural" Divide: The Protection of Tangible and Intangible Indigenous Property*, 25 CARDOZO ARTS & ENT. L.J. 367, 411 (2007).

107 Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 26 (1995).

108 Raymond T. Nimmer, *Legal Issues in Open Source and Free Software Distribution*, 885 PLI/P at 33, 91 (2006).

The difficulty in providing protection for software is further complicated by the "fair use" exception for copyright protection.[109] Essentially, fair use excuses "reasonable unauthorized appropriations" which is determined by evaluating four factors.[110] The four factors are: (1) the purpose and character of the use;[111] (2) the nature of the copyrighted work;[112] (3) the amount and substantiality of the portion used in relation to the copyrighted work;[113] and (4) the effect of the use upon the potential market.[114] Because copying software is often for the public benefit, to promote the sciences without substantially impairing the value of the copied work, "fair use" has provided a venerable defense against infringement in the software context.[115]

Despite the struggle to find steady copyright ground in the computer context, it may be useful to examine how courts have dealt with the issue thus far.

2. Reverse Engineering and Non-Literal Copying Software Cases

Recent court cases[116] have confirmed that the unique nature of software[117] yields different copyright results with similar fact patterns. Although there is currently scant case law specifically discussing whether dynamic linking of software programs can implicate a derivative work, there is an extensive history of reverse engineering and non-literal copying software cases that

109 17 U.S.C. § 107 (2006).

110 *Id.*; *see also* Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 578 (1994) ("All [factors] are to be explored, and the results weighed together, in light of the purposes of copyright.").

111 § 107(1); *see also* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514 (9th Cir. 1992) (holding intermediate software copying is allowed if it is the only means to get to the uncopyrighted functional material).

112 § 107(2); *see also* Harper & Row Publishers, Inc. v. Nation Enters., 471 U.S. 539, 561 (1985) (weighing sub-factors such as whether the copyrighted work has been published already since right to first publication is important under copyright).

113 § 107(3); *see also Harper & Row,* 471 U.S. at 601 (Brennan, J., dissenting) (weighing quality of portion used versus quantity).

114 § 107(4).

115 *See Sega,* 977 F.2d at 1523 (considering the public benefit that copying software for the functional requirements would increase creative expressions based on the unprotected ideas); *see also* Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 607 (9th Cir. 2000) (stating copyright law does not create monopolies over certain markets and legitimate software competitors may be welcomed so long as it does not merely supplant the original).

116 This is not an exhaustive list of all computer or software related cases, but rather a select few that reflect common principles within the scope of this article.

117 *See also* Andre R. Jaglom, *Internet Distribution, E-Commerce and Other Computer Related Issues: Current Developments in Liability On-Line, Business Methods Patents and Software Distribution, Licensing and Copyright Protection Questions*, SN019 A.L.I.-A.B.A. 535, 606 (stating that "[t]he difficulty is in establishing the scope of copyright protection, since unlike the traditional kinds of works protected by copyright, software has a significant functional component").

are instructive to the ultimate linking question. In a series of reverse engineering[118] cases, courts have held that copyright infringement existed even when there was no literal copying of any software, as long as the visuals created by the software were substantially similar in both idea and expression, and at least an exact description of the visuals existed in concrete form.[119] Some courts also have determined that there was no copyright infringement where the software distributed was not copied, even though there had been intermediate copying to create the distributed work.[120]

For example, in *Micro Star v. FormGen, Inc.*, the defendant owned the copyrights to a popular computer game.[121] The game included an editor, which allowed players to create their own levels for the game.[122] Without literally copying any of defendant's software, plaintiff packaged user-created levels for sale on a CD.[123] Plaintiff sought a declaratory judgment that there was no copyright infringement, while defendant counterclaimed for a preliminary injunction.[124] In reversing the district court, the Ninth Circuit granted the preliminary injunction and held that the defendant would likely succeed in showing that the audiovisual displays generated when running plaintiff's distributed levels would infringe on plaintiff's right to create derivative works.[125]

The Court reasoned that the displays would be similar in both idea and expression.[126] Even though the plaintiff did not literally copy any of the software for the games, the plaintiffs were infringing on the defendant's right to create sequels by creating new tales told by the plaintiff's new levels.[127] However, the Court clarified that the work the defendant was infringing upon was the *story* of the video game characters.[128] It conceded that the software files themselves did not create the derivative

118 Reverse engineering is essentially looking at a final product or output and working backwards to uncover the underlying elements that make the product or output work. *See Sega*, 977 F.2d at 1514.

119 Micro Star v. FormGen Inc., 154 F.3d 1107, 1111–12 (9th Cir. 1998); Worlds of Wonder, Inc. v. Veritel Learning Sys., Inc., 658 F. Supp. 351, 355 (N.D. Tex. 1986); Midway Mfg. Co. v. Arctic Int'l., Inc., 704 F.2d 1009, 1013 (7th Cir. 1983).

120 *See Sega,* 977 F.2d at 1514; *Sony*, 203 F.3d. at 600 (dismissing copying because it was only intermediate).

121 *Micro Star*, 154 F.3d at 1109.

122 *Id.*

123 *Id.*

124 *Id.*

125 *Id.* at 1114.

126 *Id.* at 1112.

127 *Id.*

128 *Id.*

works because they did "not, in fact, incorporate any of [plaintiff's] protected [software]."[129]

Similarly, in *Worlds of Wonder, Inc. v. Veritel Learning Systems, Inc.*, the defendant made cassette tapes compatible for use with a toy bear named Teddy Ruxpin.[130] The toy was animated by placing a cassette into the back of the animal, thus moving the eyes, nose, and mouth in synchronization with the story or song on the tape.[131] Although the defendant did not actually copy any of the plaintiff's tapes or software, the Northern District of Texas held that the "series of related images" that the tapes created were the copyrighted work in this case.[132] Granting a preliminary injunction, the Court held that the defendant infringed the plaintiff's right to prepare derivative works because defendant's tape cassettes created a modified work substantially similar to the plaintiff's "series of related images."[133] Again, because the defendant's tapes did not contain any copyrighted software, the Court emphasized that the defendant was infringing the *story* of the toy rather than the tapes.[134]

Furthermore, courts have found no copyright infringement where there was literal intermediate copying of plaintiff's software, done only to access uncopyrighted elements of the code. In *Sega Enterprises Ltd. v. Accolade Inc.*, the defendant copied the plaintiff's video game code in order to reverse engineer the program to discover the requirements for compatibility with plaintiff's video game console.[135] The defendant used the information to create its own games to sell in the market.[136] In the software code for these games, the defendant also included plaintiff's initialization code[137] into a standard header file[138] to allow games to interact with the console code.[139]

129 *Id.*

130 Worlds of Wonder, Inc. v. Veritel Learning Sys., Inc., 658 F. Supp. 351, 352 (N.D. Tex. 1986).

131 *Id.*

132 *Id.* at 355.

133 *Id.*

134 *Id.* at 356.

135 Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514 (9th Cir. 1992).

136 *Id.* at 1515.

137 *Id.* at 1516. Initialization code in the context of computer programming is code that gives variables a value prior to executing the code. For more discussion on initialization code, see KERNIGHAN & RITCHIE, *supra* note 11, at 40.

138 *Sega,* 977 F.2d at 1516. In computer programming, particularly in the C and C++ programming language, header files are separate files from the source code, which typically contain declarations for variables, classes, subroutines, and other identifiers. For more discussion on header files, see KERNIGHAN & RITCHIE, *supra* note 11, at 33.

139 *Sega*, 977 F.2d at 1514.

The Ninth Circuit ultimately held that the intermediate copying was fair use, and therefore was not infringement, because it was the only means to access the uncopyrightable elements, namely the compatibility procedure.[140] The Court reasoned that the defendant was only trying to "become a legitimate competitor in the field of Genesis-compatible video games" and that consumers would still continue to purchase multiple video games.[141] The Court also found the distribution of the header files was purely functional and held that "[t]he initialization code is a functional feature of a Genesis-compatible game and Accolade may not be barred from using it."[142] The Court focused on the purely functional aspect of the software copied in determining that the third party could distribute copies of the functional code.[143]

In a similar case, *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc.*, the plaintiff created a device called a Game Genie, which was sold for use with a Nintendo Entertainment Video Game system.[144] The device physically attached to the video games before they were placed into the console, in order to block data bytes sent to the console and replace the bytes with new values.[145] Effectively, this allowed users to modify aspects of the game by entering in "cheat codes."[146]

The Ninth Circuit held that because the Game Genie acted only as a window into the computer program, it did not create a derivative work and there was no infringement.[147] The Court focused on the fact that Game Genie, as distributed, did not contain any of the Nintendo code in "some concrete or permanent form" and therefore did not meet the statutory definition of a derivative work.[148] It also did not matter that the visual images the code generated could be reconstructed when the game was over by entering the same codes, because the images did not exist

140 *Id.* at 1518.

141 *Id.* at 1523; *see also* Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602 (9th Cir. 2000) (applying the analysis in *Sega* to determine that defendant's copying of plaintiff's code from a video game console was fair use because it was for purposes of a new video game, different in character, and that defendant was a legitimate competitor in the market for plaintiff's games).

142 *Sega*, 977 F.2d at 1531.

143 *See id.* at 1516.

144 Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc., 964 F.2d 965, 967 (9th Cir. 1992).

145 *Id.*

146 *Id.*

147 *Id.* at 968; *see also* Micro Star v. FormGen Inc., 154 F.3d 1107, 1111 (9th Cir. 1998) (distinguishing the case based on the software existing in some permanent or concrete form).

148 *Id.*

permanently *at distribution* of the Game Genie.[149] Despite copyright law's lack of protection for functional elements, the Court also mentioned that the "Game Genie is useless by itself, it can only enhance" and "such innovations rarely will constitute infringing derivative works under the Copyright Act."[150]

Whether or not one agrees with the holdings of these cases, they all provide insight into the various approaches of applying copyright principles to software. Courts first determine whether a work "incorporates" any portion of the copyrighted work.[151] If a work does not incorporate any protected expression, it cannot be a derivative.[152] Even when protected expressions have been incorporated into the new work, they must exist in some "concrete or permanent form" at distribution to be a derivative work.[153] Temporary reproductions following distribution are not enough to be "concrete or permanent."[154] Furthermore, even where there is intermediate copying, courts have suggested that this is permissible so long as it is used purely to gain access to the unprotected functional elements.[155] Finally, attempts to promote creative expression may also influence how a court ultimately rules, regardless of any adverse market effect.[156] Despite these opinions, the FSF appears to urge a different approach.[157]

149 *Id.* (characterizing the Nintendo case saying once the game was over, the audiovisual displays generated were gone and as a result, were not permanent).

150 *Lewis Galoob Toys,* 964 F.2d at 969; *see also* 17 U.S.C. § 117 (2006) (codifying the exception that it will not be infringement for the owner of a computer program to make a copy or adaptation provided the new copy or adaptation is created as an essential step in the *utilization* of the computer program).

151 *See Micro Star*, 154 F.3d at 1110–11.

152 *Id.* at 1110.

153 *Lewis Galoob Toys*, 964 F.2d at 967.

154 *Micro Star*, 154 F.3d at 1111 (characterizing the Nintendo case saying once the game was over, the audiovisual displays generated were gone and as a result, were not permanent).

155 Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1992); *see also* 17 U.S.C. § 117 (2006) (codifying the exception that it will not be infringement for the owner of a computer program to make a copy or adaptation provided the new copy or adaptation is created as an essential step in the *utilization* of the computer program).

156 *Sega*, 977 F.2d at 1523 (noting the court is "free to consider the public benefit resulting from a particular use notwithstanding the fact that the alleged infringer may gain commercially" and growth of creative expression can serve a public interest); *see also Lewis Galoob Toys*, 964 F.2d at 969 (recognizing that "technology often advances by improvement rather than replacement" as a reason to exclude works such as spell-checking or the Game Genie from infringement), *and* U.S. CONST. art. I, § 8 (granting Congress power to enact Copyright laws "to promote the progress of science and useful arts").

157 *See supra* Part II.B; *see also infra* Part III.C.

C. FSF / GPL Interpretation of Static / Dynamic Linking

The FSF, and the drafters of the GPL in particular, likely would not agree with courts' treatment of derivative works.[158] Terminology aside, whether or not a work is a "derivative" or "modified" version of a GPL-ed source is critical to the success of the free software movement.[159] Because of the "viral nature" of the GPL, the narrower the scope of coverage under the GPL, the fewer "infections" and the fewer GPL-species works there will be.[160] Accordingly, the FSF has attempted to tackle the problematic issue of what falls under the scope of the license by addressing program linking.[161]

Software that uses static linking to a GPL library warrants minimal discussion that it is a derivative work under any definition. With software utilizing static linking, an actual copy of the GPL code is transcribed verbatim into the larger program.[162] Because the software now contains the GPL code "or a portion of it, either verbatim or with modifications," the software is now subject to the provisions of the license.[163] Specifically, any future distribution and modification of that software must now include the source code and will be subject to the same license.[164] This much seems clear.

Dynamic linking has generated a different response to the question whether a derivative or modified work is subject to GPL's "copyleft" provisions.[165] This debate occurs in part because the GPL library is never physically combined with the larger program in which it is referenced.[166] As discussed above, the GPL library and the larger program exist on completely different disks, and the GPL software is never technically "modified."[167] Consequently, if no derivative work is created when dynamically linking to a GPL library, then there is no obligation to follow the license.[168] Under this analysis, software developers are free to retain proprietary software despite using GPL code.

158 *See* Derivative Works, *supra* note 68 (arguing GPL drafters urge a broader definition of derivative works than US Copyright law).

159 *See supra* Part II.B.

160 *See supra* Part II.A.

161 *See infra* Part IV.A.

162 *See supra* Part III.A.

163 General Public License v2.0, *supra* note 22, at § 0.

164 *Id.* at §§ 1–3.

165 Lothar Determann, *Dangerous Liasons – Software Combinations as Derivative Works? Distribution, Installation, and Execution of Linked Programs Under Copyright Law, Commercial Licenses, and the GPL*, 21 BERKELEY TECH. L.J. 1421, 1458–62, 1488 n.255 (2006) (stating that dynamic linking typically does not create a derivative work).

166 *See infra* Part IV.A.

167 *See supra* Part III.A.

168 *See* Phillips, *supra* note 56, at 500–02.

In spite of this, the FSF maintains that dynamic linking to a GPL library can create a single work that is subject to the GPL terms.[169] When asked about the specific issue of program linking and its relation to GPLv3, Stallman expressly addressed that:

> [I]t doesn't matter which kind of linking is being used. If there are two modules that are designed to be run linked together and it's clear from the design from one or the other that they are meant to be linked together then we say they are treated as one program and so I hope that will make it a little bit clearer although that's not really a change, it's a clarification. That's what we believe GPL version two means already.[170]

The "clarification"[171] to which Stallman was referring is a new express provision in GPLv3.[172] Under the new license, the "copyleft" obligation now extends specifically to "dynamically linked subprograms" that require an "intimate data communication or control flow between these subprograms."[173] The FSF appears to take a firm position that dynamic linking undeniably creates a "copyleft" obligation in a single work. However, it weakens its position with the self-imposed condition that the dynamic link must involve an "intimate data communication."[174] Thus, while the FSF clarifies rather than changes its view that dynamically linked programs always have been within the scope of the GPL, courts must now define terms such as "intimate" to determine the status of dynamically linked programs in addition to depending upon principles of copyright.

IV. CHALLENGES AND ISSUES WITH GPL AND DYNAMIC LINKING

This section analyzes the likelihood of success of the FSF's argument, perhaps simplified, that dynamic linking creates a derivative work. Infringing upon an exclusive right such as

169 Free Software Foundation, *supra* note 71 ("If the modules are included in the same executable file [static linking], they are definitely combined in one program. If modules are designed to run linked together in a shared address space [dynamic linking], that almost surely means combining them into one program.").

170 Richard Stallman, *Transcript of Richard Stallman in Torino*, (March 18, 2006), http://www.fsfeurope.org/projects/gplv3/torino-rms-transcript.en.html#q2-linking (last visited Nov. 11, 2009).

171 *Id.*

172 General Public License v3.0, *supra* note 47, at § 5.

173 *Id.* at § 6 (requiring one to supply corresponding source); *Id.* at § 1 (defining corresponding source to include "interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work").

174 *Id.* at § 1; *see also* Tsai, *supra* note 6, at 566 (arguing that the FSF takes a "tenuous position, stating first that dynamically linked programs are within the scope of the [license], then pulling back with a cryptic condition"); Free Software Foundation, *supra* note 71 (stating that one relevant condition should be what kind of information is exchanged).

preparing a derivative work would subject one to the restrictions of the corresponding license, namely the GPL in this case. Section IV.A considers the courts' likely approach to the FSF's argument based on current legal theories; Section IV.B examines the implications of the courts' likely trend; and Section IV.C asks whether that approach is the optimal solution.

A. Is There Any "Copying"?

Courts specifically have never addressed the issue of whether non-GPL libraries can dynamically link to GPL libraries with or without the same "viral effect" of the license.[175] The FSF leaves questions unanswered,[176] and courts struggle to set standards for software in general.[177] However, as case law on the issue is presently developing, a court likely would not find that a dynamically linked program falls within the scope of a GPL license.

With dynamic linking, the FSF would support a more technical analysis of the definition of "intimate."[178] If a linked program, whether dynamic or static, does not share the requisite "intimate data communication," there is no "copyleft" obligation.[179] However, as seen above, courts would likely focus on the abstract inquiry of whether the challenged software is a derivative work rather than on the underlying technology.[180] In any case, it is unlikely that a court would find an "intimate relationship" if the court did not also find that the linked program was a derivative work. When a work does not incorporate any protected expression, it cannot be a derivative.[181] Dynamic linking never incorporates any protected expression; it only makes references to it.[182] One may argue that the operating system physically incorporates the code somewhere in memory during runtime when needed; however, while having some merit, this argument should fail for two reasons.

First, recall that the GPL applies only if the work is released.[183] If a program can be released or distributed separately from a GPL work, the license obligation does not take

175 In a close case, MySQL AB v. Progress NuSphere, Plaintiff sued Defendant for copyright infringement by linking code, but the case settled. *See* Press Release, MySQL, FAQ on MySQL vs. NuSphere Dispute (Jul. 13, 2001) *available at* http://www.mysql.com/news-and-events/generate-article.php?id=75.

176 *See infra* Part IV.C.

177 *See supra* Part III.B.

178 *See* General Public License v3.0, *supra* note 47.

179 *Id.*

180 *See supra* Part III.B.2.

181 Micro Star v. FormGen Inc., 154 F.3d 1107, 1110 (9th Cir. 1998).

182 SILBERSCHATZ, *supra* note 8, at 268–71.

183 *See* Free Software Foundation FAQs, *supra* note 55.

effect.[184] For example, assume Program A is meant to be maintained as a proprietary word processing program, while Program B is a GPL-ed printing program. Program A can be distributed for use to word process without any use of printing capabilities of Program B. Program A used alone does not "incorporate" any of Program B's protected expression. If User X of Program A then wants to have printing functionality, he can independently acquire Program B and use it privately with no obligation. As a result, even if Program A and B are combined in memory when the computer is running both of them, there is no "copyleft" obligation for this private use.

Second, even if combined in memory when the user is running the program, copyright law requires the protected expression or, in the example above, Program B, to exist in some "concrete or permanent form."[185] Although virtual computer memory may be considered "concrete or permanent,"[186] temporary recreations are not enough to meet this requirement.[187] The combination will exist only while the programs run[188] and, as in the *Nintendo* case, as soon as the programs end, the combination does as well.

A second argument could be made that even references to GPL-ed code can be considered to "incorporate" the protected expression. This argument should also fail. Courts have suggested that even intermediate copying is permissible in order to take advantage of the unprotected functional elements.[189] References do not even rise to the level of intermediate copying, but are, as the name suggests, merely pointers to a particular function. Similar to copying an entire video game source code to find out compatibility functions,[190] referencing to GPL-ed code is a mechanism to use the unprotected function and would likely not be considered "incorporation" of any protected material.

184 *See* General Public License v3.0, *supra* note 47; Viral Code and Vaccination, *supra* note 58.

185 *See* Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc., 964 F.2d 965, 968 (9th Cir. 1992).

186 *See also* Micro Star v. FormGen Inc., 154 F.3d 1107, 1111 (9th Cir. 1998) (stating that concrete and permanent forms can exist on CD-ROMs, which are a type of computer memory).

187 *See Lewis Galoob Toys*, 964 F.2d at 967 (stating the effects of the toy were temporary).

188 *See* SAVITCH, *supra* note 14, at 5 (stating computer memory is used for computer's "intermediate calculations").

189 *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514 (9th Cir. 1992); Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d. 596, 600 (9th Cir. 2000) (dismissing copying because it was only intermediate).

190 *Sega*, 977 F.2d at 1526.

Finally, fair use factors may weigh in favor of no copyright infringement for dynamic linking. Even if there is copying in computer memory, the four fair use factors would likely not favor the FSF.[191] First, the nature and purpose of using GPL-ed code is typically to promote software efficiency and to create new programs with additional functionality.[192] Second, the nature of the GPL-ed work is that it already has been published because it is freely distributed and encourages its use. Third, the amount copied would be only that which is necessary for the program to function. Finally, courts have established fair use where the effect on the market from the copying is to establish legitimate competition.[193] Software vendors could make these arguments in their favor to establish fair use.

As a result, a court would likely not find that a derivative work is created simply by dynamic linking because the GPL-ed code is neither "incorporated" nor exists in a "concrete or permanent form." It is hard to imagine that the level of "intimate data communication" is met when one program is not a legal derivative of the other. Additionally, if a court finds "incorporation," it is likely that the incorporation would be protected by fair use.

B. Implications Assuming No Obligations

Assuming dynamic linking to GPL code does not impose any "copyleft" obligations,[194] proprietary software vendors can plan to selectively incorporate open software without further obligation. However, the free software community would not benefit from misappropriation of their work products. While pure believers in free software principles may continue their efforts in providing an open development model, others who once supported the FSF for its reciprocal guarantee that they would receive the same benefit may reconsider. This could result in a range of outcomes, from a floodgate of legal challenges to a decrease in free software development. The worst-case scenario might be an end of free software.[195]

191 *See supra* Part III.B.1.

192 Tsai, *supra* note 6, at 557; SAVITCH, *supra* note 14, at 12.

193 *Sega*, 977 F.2d at 1514, 1524.

194 *See* General Public License v3.0, *supra* note 47.

195 Free Software Foundation, *Copyleft and the GNU GPL*, http://www.gnu.org/gnu/thegnuproject.html (last visited Jan. 11, 2010) (admitting that "to permit such combinations would open a hole big enough to sink a ship").

C. Alternative Outcome and Implications

Although a court most likely would hold that the distribution of software that only dynamically links to a copyrighted work is not itself a derivative work, it is worth exploring legal scenarios in which a court *does* impose copyright obligations for such a distribution.

One possible scenario could use the same principle for which the FSF is fighting and use it offensively. If dynamic linking creates derivative works because the operating system combines the works into shared memory, could the creator of the operating system equally argue that loading the software into its operating system creates derivative works and thus subjects users to their own license? For instance, could Microsoft, owners of the common operating system Windows, assert that dynamic loading of libraries in Windows creates a Microsoft licensed product? Furthermore, the Microsoft license could impose conflicting restrictions with the FSF such as prohibiting any GPL-ed linking in its entirety.[196]

A second scenario may involve two dynamic libraries that both have the same functionality: lib_GPL, which is GPL-ed, and lib_BSD, which has no license obligations. Consider Program A that is distributed for use with either library, but will not function without one of them. Inevitably, users of Program A will choose to combine either library, but choosing one over the other may have entirely different consequences. Can it be possible that Program A is obligated by the GPL regardless of which library the user chooses? This likely would lead to yet another debate between the FSF, who would answer the question in the affirmative, and proprietary software vendors, who likely would disagree.

Although the preceding thought experiments also result in potential problems for the FSF, it is worth considering which scenario will yield greater benefits for the public.[197]

CONCLUSION

Software complexities have challenged the legal system to find workable standards in areas such as copyright law. While this area is still developing, the software industry has attempted to define its own responsibilities. However, these responsibilities may not always find legal support. The FSF, for instance,

196 General Public License v1.0, *supra* note 33 (stating that GPLv1 does not allow for combining code with conflicting licenses and thus would result in less possible code combinations in the world).

197 *See supra* Part III.B.2.

maintains that dynamic linking, just like static linking, to GPL-ed software can create a derivative work subject to a GPL license. Although dynamic and static linking perform the same functional goals from a programming perspective, their technical differences may produce different legal implications. To date, there is no case law addressing this specific issue; however, some opinions suggest that the FSF has questionable basis to assert that dynamic linking creates a derivative work. Unlike static linking, there is no requisite incorporation of protected material to create a derivative work. Furthermore, other copyright principles such as "fair use" likely weigh against the FSF's position. With potentially both positive and negative consequences for not only free software but also proprietary software proponents, this unresolved intersection of technology and copyright law will provide a rich and interesting source of future legal developments.