THE PENGUIN PARADOX: HOW THE SCOPE OF DERIVATIVE WORKS IN COPYRIGHT AFFECTS THE EFFECTIVENESS OF THE GNU GPL

Mitchell L. Stoltz[*]

INTRODUCTION 1440
I. WHAT THE GPL CLAIMS TO DO 1444
A. *The GPL's Goals* 1444
B. *The Legal Mechanism of the GPL* 1446
II. THE PROBLEM: COMBINING PARTIAL PROGRAMS 1447
A. *Modules and Linking* 1447
B. *Why Dynamic Linking Presents a Problem for the GPL* 1450
C. *The FOSS Community's Interpretation of Derivative Works* 1452
III. THE COPYRIGHT LAW OF LINKING 1453
A. *The Early Cases on Linking* 1453
B. *The Linking Exemption to the Derivative Work Right* 1456
1. Development of the Exemption 1456
2. Policy Rationales for the Exemption 1459
IV. BASIC RULES FOR LINKING WITH GPL SOFTWARE: A FIRST APPROXIMATION 1462
A. *Static Linking and the Pre-1992 Cases* 1462
B. *The Linking Exception Presents a Paradox for the GPL* 1463
1. Expressive Audiovisual Programs 1465
2. Creative or Arbitrary Interface Code 1466
3. Working Only with a Single Program 1467
4. The Paradox 1468
V. AN ARGUMENT FOR STRONGER DERIVATIVE WORKS PROTECTION FOR GPL CODE 1469
A. *An Alternate Balance for Copyright* 1469
B. *Limiting Fair Use for Linking to GPL Code* 1471
1. Fair Use and FOSS Have Parallel Goals 1471
2. "Exploitation" and Custom: Specific Fair Use Factors 1473
a. *FOSS Industry Custom* 1474
b. *Effect on Reputation* 1474
c. *Beneficial Network Externalities* 1475
C. *Broadening the Definition of a Derivative Work for GPL Code* 1476
D. *Potential Difficulties with this Approach* 1476
CONCLUSION 1477

[*] J.D. Candidate, Boston University, 2006.

INTRODUCTION

When computer companies began to use copyright law to prevent others from modifying software, MIT computer scientist Richard Stallman used copyright for the opposite purpose – to guarantee a right to study, tinker, improve, and redistribute.[1] The result was the GNU General Public License, or the GPL.[2] Part legal document and part manifesto, the GPL establishes the terms of distribution for a growing collection of free and open source software (FOSS)[3] and helps to preserve a cooperative ethic among its developers and users.[4] Stallman wrote the GPL as a response to what he saw as the harmful effects of proprietary software – produced by companies like IBM – on software development.[5] Ironically, the GPL is now the legal framework for

[1] Free Software Foundation, GNU General Public License Version 2, at Preamble (June 1991), http://www.gnu.org/licenses/gpl.txt [hereinafter GPL] (stating that "[this] License is intended to guarantee . . . [the] freedom to share and change free software"). The Free Software Foundation (the FSF) is currently drafting a revision to the GPL, to be called Version 3. The FSF's press release strongly suggests that the basic legal mechanism of the GPL will remain the same in Version 3. *See* Press Release, Free Software Foundation, GPL Version 3: Background to Adoption (June 9, 2005), http://www.fsf.org/news/gpl3.html (mentioning several new issues that GPL version 3 will address but suggesting that Stallman intends to "preserve its integrity").

[2] GNU is a recursive acronym for Gnu's Not Unix. Free Software Foundation, The GNU Manifesto, http://www.gnu.org/gnu/manifesto.html (last modified July 13, 2005) (discussing the goal of developing a free, UNIX-compatible operating system in order to facilitate software development). The GNU Project began with the goal of writing a complete "free software" operating system, similar to – but independent of – the Unix operating system. Free Software Foundation, Overview of the GNU Project, http://www.gnu.org/gnu/gnu-history.html (last modified Sept. 27, 2005) [hereinafter GNU Overview] (defining the "free" in free software as the freedom to copy a program and give it away, to change the program through access to the source code, and to distribute this new and improved version).

[3] The Free Software Foundation uses the term "free software" to emphasize the freedom to distribute and modify. Free Software Foundation, Why "Free Software" Is Better Than "Open Source", http://www.gnu.org/philosophy/free-software-for-freedom.html (last modified May 5, 2005) (stating that "[f]or the Open Source movement, non-free software is a suboptimal solution," but "[f]or the Free Software movement, non-free software is a social problem and free software is the solution"). Another group, the Open Source Initiative, coined the term "open source" as a politically neutral term for open, collaborative software development. Open Source, Open Source Initiative, http://www.opensource.org (last visited Nov. 17, 2005) (stating that the basic idea behind the Open Source Initiative was to facilitate the improvement of software development by allowing programmers to read, distribute, and modify software). The "penguin" in the title of this Note refers to the mascot of the GNU/Linux operating system, probably the best-known FOSS program. *See* Linux Online Home Page, http://www.linux.org.

[4] *See* discussion *infra* Part I.A (explaining that the GPL preserves collaborative software development by guaranteeing access to a program's source code and granting permission to modify it to a widespread community of programmers).

[5] GNU Overview, *supra* note 2 (describing the GNU Project as a way to bring back "the cooperative spirit that prevailed in the computing community in earlier days").

billion-dollar software businesses at some of the same companies, including IBM.[6] Where business goes, lawsuits inevitably follow.

The GPL allows anyone to copy, modify, and redistribute any program to which it applies, and requires that all distributions include source code, the program's recipe.[7] It also sets a second condition on redistributions: any program derived from a GPL-covered program must itself be distributed under the GPL, or else not distributed at all.[8] This condition, known as "copyleft," is a key legal innovation of the GPL.[9] It preserves and perpetuates the public's right to copy and modify both current and future versions of programs.[10] Copyleft is an application of copyright law, but one where the copyright owner has used her power to keep software effectively in the public domain, not out of it.[11] The limitations and exceptions of copyright also limit copyleft.[12] In particular, the boundaries of copyright law's definition of a derivative work determine to which versions and revisions the GPL applies.[13] For software, the definition of a derivative work is uncertain, because the boundaries separating one distinct "work of authorship" from another are hard to fix in a computer

[6] IBM claimed to have invested $1 billion in development and marketing for the GPL-licensed GNU/Linux operating system, and earned about the same amount in revenue from the project in 2002 alone. Stephen Shankland, *IBM: Linux Investment Nearly Recouped*, CNET NEWS.COM, Jan. 29, 2002, http://news.com.com/2100-1001_3-825723.html (discussing IBM's strategy to take advantage of the open-source movement to get ahead of competitors).

[7] *See* GPL, *supra* note 1, at §§ 1-3 (permitting user to copy, modify, redistribute, and have access to the source code of licensed software).

[8] GPL, *supra* note 1, at § 2(b) (requiring that derivatives of GPL software be distributed only under the terms of the GPL); *id.* at § 7 (prohibiting any distribution if the developer is unable to comply with the GPL).

[9] Free Software Foundation, What Is Copyleft?, http://www.gnu.org/copyleft/copyleft.html#WhatIsCopyleft (last modified May 5, 2005) (defining "copyleft" as "a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well").

[10] *See* GPL, *supra* note 1, at § 2(b) (stipulating that "[the licensee] must cause any work that [the licensee] distribute[s] or publish[es], 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").

[11] *See* GPL, *supra* note 1, at § 5 (granting rights to distribute and modify the program to those who accept the license, but warning that copyright law will otherwise prohibit these actions for those who do not accept).

[12] *See, e.g.*, Free Software Foundation, Frequently Asked Questions about the GNU GPL, http://www.gnu.org/licenses/gpl-faq.html#GPLFairUse (last modified Aug. 19, 2005) (responding affirmatively to the question "[d]o I have 'fair use' rights in using the source code of a GPL-covered program?"). Fair use is one of the most basic limitations or exceptions to copyright in the United States, avoiding rigid application of the copyright statute when to do so would "stifle the very creativity which that law is designed to foster." Campbell v. Acuff-Rose Music, 510 U.S. 569, 577 (1994).

[13] *See* discussion *infra* Part II.B (discussing issues raised by dynamically linked software when defining derivative works).

system of many tightly interwoven components.[14] Under what circumstances can two programs be said to combine into one, instead of simply being two programs that interact with each other?[15] When two programs interact closely enough to be considered a new, hybrid program (a derivative work of both programs), the GPL's terms dictate that it must apply to the whole.[16]

The legal question of when two interacting programs form a derivative work will determine how broadly the GPL applies,[17] and whether it can help preserve the cooperative values of FOSS development. If copyright law does not recognize a derivative work where two programs interact in common ways, the GPL copyleft regime may contain an enormous loophole for proprietary exploitation.[18] Ultimately, the effectiveness of the GPL may depend on whether courts are willing to apply copyright law to GPL software in an outcome-oriented way, acknowledging that the "freedoms" the GPL promotes are aligned with the economic and social policy behind U.S. copyright law.

Despite its popularity, the GPL has attracted criticism from legal scholars and practitioners. Professor Margaret Jane Radin has questioned whether a conditional copyright license like the GPL can bind anyone who uses the licensed software, whether they have voluntarily entered a contract with the copyright owner or not.[19] Others point to the inconsistent language of the

[14] The Copyright Act defines a derivative work as "consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship." 17 U.S.C. § 101 (2000 & Supp. 2004).

[15] For example, a protocol called CORBA allows programs running on different computers – potentially separated by great distances – to interact in complex ways. Object Management Group, CORBA Basics, http://www.omg.org/gettingstarted/corbafaq.htm (last modified May 26, 2005) (describing CORBA as an "open, vendor-independent architecture and infrastructure that computer applications use to work together over networks").

[16] *See* GPL, *supra* note 1, at § 2 (requiring that the GPL apply to any software distributed as "part of a whole" when the whole is a derivative work).

[17] *See* Free Software Foundation, Frequently Asked Questions about the GNU GPL, http://www.gnu.org/licenses/gpl-faq.html#LinkingOverControlledInterface (last modified Aug. 19, 2005) (explaining how copyright law determines whether the GPL applies to combinations of programs).

[18] *See infra* Part IV.B (concluding that most dynamically linked modules can be released under proprietary licenses and without disclosure of corresponding source code).

[19] Margaret Jane Radin, *Incomplete Commodification in the Computerized World*, *in* THE COMMODIFICATION OF INFORMATION 3, 15 (Niva Elkin-Koren & Neil Weinstock Netanel eds., 2000) (describing the idea of contractual terms that apply to a program no matter who uses it as "legally dubious"); *see also* Robert P. Merges, *The End Of Friction? Property Rights and Contract in the "Newtonian" World of On-Line Commerce*, 12 BERKELEY TECH. L.J. 115, 129 (1997) (arguing that the GPL is unenforceable against many users because of a lack of privity); Margaret Jane Radin, *Humans, Computers, and Binding Commitment*, 75 IND. L.J. 1125, 1132 (2000) (expressing doubt as to whether the GPL's terms can "run with the product"). Software licenses, in general, are legally uncertain. *See* Michael J. Madison, *Reconstructing the Software License*, 35 LOY. U. CHI. L.J. 275, 278 (2003) (concluding that many aspects of software licensing have weak legal support, and that licensing may be

license – it contains at least three different definitions of a derivative work – to illustrate the problems that may arise in any GPL litigation.[20]

To date, the GPL has not been litigated in a U.S. court. A well-established informal enforcement process combined with a strong norm within the FOSS community against violating the GPL have upheld the license so far.[21] As a result, no U.S. case has yet addressed the merits of a copyright claim involving the GPL.[22] However, this could soon change. With FOSS-related business generating billions of dollars[23] and major corporations aligning to either embrace or oppose the GPL,[24] high-stakes litigation looks more likely. In 2003, the SCO Group, a company selling the proprietary Unix operating system, launched a legal assault on the GPL-licensed GNU/Linux operating system, and by extension on the GPL itself.[25] Although the SCO Group

---

closer to a social norm than a legal regime).

[20] Phil Albert, *Sticks, Stones and the GPL*, ECT NEWS.COM, Nov. 27, 2004, http://www.ectnews.com/story/38089.html (acknowledging three possible definitions of "derivative work" within the GPL); Phil Albert, *A Consumer's Review of the General Public License*, LINUXINSIDER, July 20, 2004, http://www.linuxinsider.com/story/35193.html (indicating that the legal definition of "derivative work" within the GPL has been the subject of much case law).

[21] Eben Moglen, *Free Software Matters: Enforcing the GPL, II*, LINUXUSER, Oct. 2001, at 66, *available at* http://emoglen.law.columbia.edu/publications/lu-13.html (describing the FSF's informal enforcement practices).

[22] A German court upheld the GPL as a license under German copyright law. Welte v. Sitecom Deutschland GmbH, No. 21 O 6123/04 (Dist. Ct. of Munich 2004), *available at* http://www.jbb.de/judgment_dc_munich_gpl.pdf. The same court later granted a preliminary injunction against a different software vendor for alleged violation of the GPL. Press Release, GPL-Violations.org, GPL-Violations.org Project Was Granted a Preliminary Injunction Against Fortinet UK Ltd. (Apr. 14, 2005), *available at* http://www.gpl-violations.org (follow "Fortinet Injunction" hyperlink under "News"). In a much publicized United States case, MySQL AB alleged a violation of the GPL, but the parties settled out of court. ComputerWire, *MySQL, NuSphere Settle GPL Contract Dispute*, THE REGISTER, Nov. 21, 2002, http://www.theregister.co.uk/2002/11/21 (follow hyperlink to title of article under "Software").

[23] *See* Shankland, *supra* note 6 ("IBM nearly recouped the $1 billion it said it invested in the Linux operating system in 2001 . . . .").

[24] In addition to IBM, companies such as Novell, Apple, and Hewlett-Packard, among others, have made significant investments in GPL-covered software and other FOSS. Open Source Initiative, Products, http://www.opensource.org/docs/products.php (last visited Nov. 17, 2005) (listing companies that sell open-source based solutions). Microsoft, the world's largest mass-market software vendor, has criticized the GPL on policy grounds. Matthew Szulik, *On the Wrong Side of History*, WIRED, Feb. 26, 2001, http://www.wired.com/news/business/0,1367,42008,00.html?tw=wn_story_related (quoting Microsoft CEO Steve Ballmer's description of open source as an "intellectual property destroyer").

[25] SCO's Answer to IBM's Amended Counterclaims, Eighth Affirmative Defense, The SCO Group, Inc. v. Int'l Bus. Machs. Corp., No. 03-CV-0294 (D. Utah Oct. 24, 2003), *available at* http://www.groklaw.net/pdf/AnswerAmendCC.10-24-03.pdf (alleging that the

ultimately dropped its GPL-related claims,[26] the suits convinced many FOSS developers of the vital need to resolve legal uncertainties in their licensing schemes.[27]

This Note focuses on one ambiguity that affects the GPL: the scope of what constitutes a derivative work. Part I describes the goals of the GPL and how it uses copyright law. Part II explores software linking, including the common methods FOSS developers use to combine programs, and why linking presents a thorny problem for copyright law. Part III discusses courts' attempts to determine when unauthorized linking is permitted. These efforts have led to a broad but not unlimited permission to copy the elements of a program that are necessary to make linking work. Part IV applies the case law on linking to the issues of FOSS development discussed in Part II, concluding that the current pro-compatibility trend in software copyright law severely limits the effectiveness of the GPL in achieving its stated goals. Finally, Part V proposes that courts apply both a narrower application of fair use and a broader definition of derivative works in cases involving the GPL, in order to promote the goals of both the GPL and the copyright system as a whole. The Note compares the policy rationales of existing copyright law with the goals of the GPL and the practices of FOSS developers, and concludes that this solution can preserve copyright's effectiveness while avoiding the monopolistic practices the GPL was written to oppose.

I. WHAT THE GPL CLAIMS TO DO

A. *The GPL's Goals*

The GPL uses a legal mechanism to preserve the values of FOSS development that Stallman felt were threatened by the rise of proprietary software.[28] To its authors, free software involves reusing software components to solve common problems in a collaborative, academic-like effort.[29] FOSS

GPL is preempted by federal copyright law).

[26] *Compare id.* (claiming that the GPL is preempted by federal copyright law) *with* SCO's Answer to IBM's Second Amended Counterclaims, SCO Group, Inc. v. Int'l Bus. Machs. Corp., No. 03-CV-0294 (D. Utah Apr. 28, 2004), http://www.groklaw.net/pdf/IBM-141-1.pdf (omitting preemption claim against the GPL).

[27] *See* Matt Loney, *Open Source Leader: SCO Suits a Boon to Linux*, CNET NEWS.COM, Mar. 10, 2005, http://news.com.com/2100-7344_3-5608563.html (reporting that the SCO litigation resulted in due diligence and scrutiny directed at Linux code base).

[28] *See* Richard Stallman, *The GNU Operating System and the Free Software Movement*, *in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION (Chris DiBona et al. eds., 1999), *available at* http://www.oreilly.com/catalog/opensources/book/stallman.html (recalling the origins of the GNU Project and the GPL); STEVEN WEBER, THE SUCCESS OF OPEN SOURCE 179-80 (2004) (theorizing that FOSS licenses play a large role in shaping the social structure of FOSS development projects).

[29] Stallman likens programmers' freedom to share code with other programmers to chefs sharing recipes. Stallman, *supra* note 28 ("Sharing of software was not limited to our

development generally involves distributing frequent revisions to a widespread group of programmers and users, who can then test and improve the software.[30] For the FOSS development system to work, developers must have access to a program's source code, the human-readable list of instructions to a computer that tell the computer how to perform a task.[31] Modification is nearly impossible without source code, similar to modifying a cake recipe while having only the cake and not the recipe.[32] The would-be software tinkerer also needs legal permission to modify the source code, to avoid a copyright violation. The GPL both guarantees access to source code and grants permission to modify it.[33] In addition, because only GPL-covered software can be combined with other GPL-covered software,[34] the license may provide an incentive for more software developers to release their software under the GPL.[35] By releasing her programs under the GPL, a programmer

particular community; it is as old as computers, just as sharing of recipes is as old as cooking."). Reusing code for common functions reduces errors and increases programmers' efficiency. Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 STAN. L. REV. 255, 265 (1997) (discussing how systematic software reuse can improve the quality of components and increase the productivity of creators). FOSS programmers have a particularly strong incentive to reuse code and avoid reinventing existing functions, because they are often not paid for their work. WEBER, *supra* note 28, at 75.

[30] *See* ERIC S. RAYMOND, *Release Early, Release Often*, *in* THE CATHEDRAL AND THE BAZAAR (2000), *available at* http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html (theorizing how frequent releases and distributed development create robust software).

[31] *See* Free Software Foundation, The Free Software Definition, http://www.gnu.org/philosophy/free-sw.html (last modified July 22, 2005) (explaining that access to source code is required for any examination and improvement of a program).

[32] WEBER, *supra* note 28, at 4 (explaining that modification is very difficult without source code).

[33] GPL, *supra* note 1, at § 2 (permitting modifications to covered software); *id.* at § 3 (requiring subsequent developers to make source code available).

[34] Because a program combining significant amounts of GPL-covered code may be distributed only under the terms of the GPL, it follows that any code that must remain proprietary cannot be combined with GPL code, or at least, the resulting mix cannot be distributed. If distributed under a proprietary license, such a mix would violate the GPL and therefore infringe the copyright on the original GPL-covered code. If distributed under the GPL, the proprietary code would be unprotected from copying because the GPL authorizes anyone to make copies. The only lawful action in this case is to refrain from distribution. *See* GPL, *supra* note 1, at § 2 (establishing the GPL's intent to exercise a right of control over the distribution of derivative or collective works).

[35] Richard Stallman, Why You Shouldn't Use the Library GPL for Your Next Library, http://www.gnu.org/licenses/why-not-lgpl.html (last modified May 5, 2005) [hereinafter Why You Shouldn't Use the Library GPL] ("If we amass a collection of powerful GPL-covered libraries that have no [proprietary equivalent] . . . some projects will decide to make software free in order to use these libraries."); *see also* Matthias Strasser, *A New Paradigm in Intellectual Property Law? The Case Against Open Sources*, 2001 STAN. TECH. L. REV. 4, 62 (questioning the philosophy of the open source movement, but admitting that "[t]here are

gains legal access to a large collection of useful GPL code that she can use within her own programs.[36] Thus, the terms of the GPL help to define and perpetuate a particular method of software development, one which has given rise to some of the most important and fundamental software available today.[37]

B. *The Legal Mechanism of the GPL*

> 1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium . . . .
>
> 2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work . . . .
>
> (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.[38]

Copyright gives the author of a work the exclusive right to copy, distribute, and prepare derivatives of that work.[39] The GPL, as a copyright license, is a statement by the copyright holder that anyone may exercise some of these rights under specified conditions.[40] Specifically, the GPL gives anyone who uses a licensed program the copyright owner's permission to copy, modify, and redistribute the program. As with a license to use physical property, a copyright license can be conditional.[41] The GPL contains two important conditions on its grant of rights: the user must include the program's source code with any distribution of the program[42] and must distribute any derivative works, if at all, under the terms of the GPL.[43]

---

indeed various situations in which access to the source code of a piece of software benefits [other software developers]").

[36] *See* Why You Shouldn't Use the Library GPL, *supra* note 35.

[37] The basic functions of the Internet, including Web servers, e-mail transport, and domain name lookup, are predominantly handled by FOSS. Open Source Initiative, Products, *supra* note 24 (boasting that "[n]ot surprisingly, most of the software on top of the operating system that keeps the internet humming is also open source").

[38] GPL, *supra* note 1, at §§ 1-2.

[39] *See* 17 U.S.C. § 106 (2004).

[40] GPL, *supra* note 1, at §§ 1-3; *see* Eben Moglen, *Free Software Matters: Enforcing the GPL, I*, LINUXUSER, Sept. 2001, at 66, *available at* http://emoglen.law.columbia.edu/publications/lu-12.html [hereinafter Moglen, *Enforcing I*] (explaining that the GPL relaxes almost all the restrictions of the copyright system).

[41] *See* 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 10.15[A] (2005) [hereinafter NIMMER] (explaining that failure to satisfy a condition of a copyright license creates a cause of action for copyright infringement).

[42] GPL, *supra* note 1, at § 3 (requiring the distribution of the corresponding machine-readable source code of a program).

[43] *Id*. at § 2 (requiring that derivative works carry prominent notices informing others of

Proprietary software licenses, which supply the terms of distribution for most mass-market software, typically include restrictions entirely separate from the exclusive rights of the copyright owner, such as prohibitions on disassembling or reselling a program.[44] Because copyright law allows the disassembly of programs and the resale of a lawful copy,[45] the only legal basis for prohibiting those activities is contract law.[46] In contrast, the GPL governs only copying, modification, and distribution, all of which have their source in the Copyright Act.[47] This means that the GPL can derive its legal force exclusively from the Copyright Act, with no resort to contract law.[48]

II. THE PROBLEM: COMBINING PARTIAL PROGRAMS

A. *Modules and Linking*

Software subcomponents that combine into larger programs are a common

changes in the files). The GPL also contains some other conditions not relevant to this discussion, such as attribution and patent licensing requirements. *Id.* at §§ 2(a), 2(c) (mandating the preservation of that the author's name, date of changes, and copyright notice across modifications); *id.* at § 7 (requiring patent holders to license their patents royalty-free when including patented code in a GPL-covered program).

[44] *See* Moglen, *Enforcing I*, *supra* note 40 (describing the FSF's informal enforcement practices).

[45] Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1520 (9th Cir. 1992) (holding that copyright law's doctrine of "fair use" permits disassembly to study the operation of a program); 17 U.S.C. § 109(a) (2000) (permitting the owner of an authorized copy of a copyrighted work to sell or otherwise dispose of that copy).

[46] *See* Moglen, *Enforcing I*, *supra* note 40 (explaining that the prohibition against decompilation of software is governed by contract law, not copyright). Other common terms in proprietary software licenses, such as permission to make archival copies, can be understood as a license of some of the exclusive rights of copyright (in the case of archival copies, the right to reproduce). *See* 3 NIMMER, *supra* note 41, at § 10.1 (describing how the exclusive rights of § 106 can be licensed piecemeal).

[47] 17 U.S.C. § 106 (2000) (asserting that an owner of copyright has exclusive rights to reproduce, prepare derivative works of, and distribute the copyrighted work); *see* Moglen, *Enforcing I*, *supra* note 40 ("The copyright holder is legally empowered to exclude all others from copying, distributing, and making derivative works.").

[48] *See* 3 NIMMER, *supra* note 41, at § 10.15[A] (explaining that violating the terms of a license is generally a copyright infringement rather than a breach of contract). This view of the GPL as a conditional copyright license rather than a contract is the position taken by the FSF's general counsel, Eben Moglen. Moglen, *Enforcing I*, *supra* note 40 ("Licenses are not contracts: the work's user is obliged to remain within the bounds of the license not because she voluntarily promised, but because she doesn't have any right to act at all except as the license permits."). Others assume the GPL can only be enforced as a contract. *See* Radin, *supra* note 19, at 1132-1133 (questioning whether what she dubs "viral contracts" like the GPL are valid without the user's consent to be bound). This Note focuses on the limitations of the "pure copyright" view of the GPL; its contractual aspects have been analyzed in detail by others. *See, e.g.*, *id.*

aspect of all software development, including FOSS development.[49] Nearly all programs running on the GNU/Linux operating system – which is covered by the GPL – make use of subcomponents called libraries and kernel modules.[50] Libraries are collections of general-purpose code which perform tasks needed by various programs.[51] For example, a commonly used GNU/Linux library includes utilities which sort collections of data, a task that almost every program may need.[52] Libraries are vital for most programs, since it is impractical for each program to contain all of the code needed for these common functions.[53] Kernel modules, another kind of subcomponent, extend the functionality of the Linux kernel, which is the basic program at the heart of the operating system.[54] The typical kernel module is a device driver, which allows the computer to run a peripheral device such as a disk drive.[55]

Libraries and kernel modules[56] simplify the reuse of code, and reuse is a key part of FOSS development. They make adding functionality to a program easier, since a module can add new features to a program without the need to recompile the entire program.[57] In addition, modules simplify fixing errors in programs by isolating potential errors from the rest of the program.[58]

---

[49] *See* Ashish Bansal, *Shared Objects for the Object Disoriented*, IBM DEVELOPERWORKS, Apr. 1, 2001, http://www-128.ibm.com/developerworks/linux/library/l-shobj/ (acknowledging that "[a]t different times in our coding lives, all of us have used some sort of library, be it for a simple function like printf() in C or for a complex function like sort() in the C++ generic function library").

[50] *See id.* (extolling the virtues of shared libraries for Linux); Bryan Henderson, *Linux Loadable Kernel Module HOWTO*, LINUX DOCUMENTATION PROJECT, July 20, 2005, http://www.tldp.org/HOWTO/Module-HOWTO/index.html (summarizing the creation and use of Linux loadable kernel modules). The GNU/Linux operating system consists of the Linux kernel program, which was originated by Finnish programmer Linus Torvalds in 1992, and numerous other programs created by the FSF's GNU Project. Richard Stallman, *Linux and the GNU Project*, June 27, 2005, http://www.gnu.org/gnu/linux-and-gnu.html (providing background information on the development and components of the Linux kernel program).

[51] *See* Bansal, *supra* note 49 (listing common functions performed by libraries, such as receiving input from the user).

[52] FREE SOFTWARE FOUNDATION, THE GNU C LIBRARY REFERENCE MANUAL (July 6, 2001), http://www.gnu.org/software/libc/manual/html_mono/libc.html#Searching%20and%20Sorting (describing functions for sorting and searching arrays of arbitrary objects).

[53] *See* Bansal, *supra* note 49 (mentioning the common use of libraries for receiving user input).

[54] Henderson, *supra* note 50, at § 2.5 (describing typical uses of kernel modules).

[55] *Id.* at § 2 (listing device drivers as a common type of kernel module).

[56] This Note will refer to subcomponents collectively as modules.

[57] Henderson, *supra* note 50, at § 2.3 (pointing out that kernel modules allow the addition of new functionality to the kernel without restarting and/or rebuilding the system).

[58] *Id.* (remarking that modifying the kernel directly makes errors difficult to locate, while extending the kernel using modules allows the programmer to isolate potential errors).

There are several different mechanisms for linking modules with the programs that make use of them. Programs are written in source code, which is readable by humans.[59] To transform source code into a form that computers can understand and run, programmers use a tool called a compiler, which transforms source code into object code.[60] Object code can be run on a computer, but is very hard for a human to read and understand directly.[61] One way to make use of a module is to combine its source code with the source code for the program that will use the module and then compile the combination as one, creating a single object code file that contains both the original program and the module.[62] This process is known as static linking.[63] In the other common method, dynamic linking, the original program and the module occupy two separate object code files that can be sold and distributed separately.[64] When a user runs the program, her computer also loads the necessary object-code version of the module, and the program and module send commands and data to each other as they run.[65] Static linking resembles a gasoline-powered lawnmower since the tool (the lawnmower) and its power source (a gasoline motor) are attached inseparably to each other at the factory. Dynamic linking, on the other hand, is more like an electric lawnmower that allows the user to connect to any power source. Like a program with a dynamically linked module, the second lawnmower is sold separately from its power source, such as a wall socket. The two pieces, lawnmower and socket, are linked together only when they are used. Either part is interchangeable: the user can buy a new lawnmower or a new power source, such as a solar panel, and the system will continue to work as long as both parts are compatible.

Dynamic linking extends the benefits of modularity still further.[66] Because a dynamically linked module in object code form occupies a separate file from the program that uses it, multiple programs can use a single module at the same time, conserving memory.[67] Also, like the lawnmower's power supply, dynamically loaded modules can be replaced easily, sometimes even while a program is running.[68] Many widely-used modules on the GNU/Linux system

[59] *See* WEBER, *supra* note 28, at 4 (analogizing source code to a recipe for making the object code).

[60] *See id.* at 4 n.2 (describing use of compilers).

[61] *Id.* at 4 ("Most commercial software is released in machine language or what are called 'binaries' – a long string of ones and zeros that a computer can read and execute, but a human cannot read.")

[62] *See* Bansal, *supra* note 49 (explaining the process of static linking).

[63] *See id.* (describing how static linking works).

[64] *See id.* (discussing how to write dynamically linked libraries).

[65] *See id.* (elaborating on how dynamically linked libraries interact with the loader).

[66] *See id.* (describing how dynamically linked modules can reduce a program's memory footprint, and allow easy distribution, installation, and upgrading of a program).

[67] *See id.* (indicating that without dynamic linking, "the size of [a program] would become prohibitive").

[68] Henderson, *supra* note 50, at § 2 (explaining how kernel modules can be added or

are dynamically loaded.[69]

The mechanism by which a program communicates with a module is called an interface.[70] Continuing the electric lawnmower example, the interface between the mower and its power source is a simple power cord and socket. Depending on what a module does and on the programmer's engineering decisions, an interface may be simple or complex. An interface might be compared to the set of buttons on a pocket calculator and another interface to the bewildering array of controls and switches in the cockpit of an airplane. To use the functionality provided by a module, a program must communicate with the module using an interface that both parts comprehend.[71] Depending on the complexity of the interface, writing a program that uses a module may involve copying a small part of the module's source into the main program's source code, analogous to a pilot memorizing the location and function of the cockpit switches. It is this copying that brings copyright law into the picture.

B. *Why Dynamic Linking Presents a Problem for the GPL*

As discussed in Part I above, the GPL applies to any program that is a derivative work of another GPL-covered program.[72] If a programmer writes a new module and statically links it with an existing GPL-covered program, the result is almost certainly a derivative work of the existing GPL program.[73] For example, suppose a programmer writes a driver program to control a particular kind of printer, and statically links her driver with the Linux kernel. The resulting program would combine the Linux kernel and the new driver in one object code file. Because it contains the entire Linux kernel, the new program is a derivative work. According to the GPL, the programmer must distribute the new program under the terms of the GPL or refrain from distributing it.[74]

Dynamic linking is more complex. Because a dynamically linked module

removed while the kernel is running).

[69] *See* David A. Wheeler, *Program Library HOWTO* § 3.2, LINUX ONLINE, Apr. 2003, http://www.linux.org/docs/ldp/howto/Program-Library-HOWTO/shared-libraries.html#AEN70.

[70] Wikipedia: Computer Science, http://en.wikipedia.org/wiki/Interface_%28computer_science%29 (last visited Oct. 10, 2005) ("An interface defines the means of interaction between software components.").

[71] *See id.*

[72] *See supra* Part I.B (citing GPL, *supra* note 1, at § 2(b)).

[73] *See* Pickett v. Prince, 207 F.3d 402, 406-07 (7th Cir. 2000) (finding that a guitar that incorporates a copyrighted symbol is an infringing derivative); Anderson v. Stallone, 1989 WL 206431, at *6 (C.D. Cal. Apr. 25, 1989) (holding that a script that incorporates copyrighted characters and histories from a film is an infringing derivative); *see also* Free Software Foundation, Frequently Asked Questions about the GNU GPL, http://www.gnu.org/licenses/gpl-faq.html#GPLIncompatibleLibs (last modified Aug. 19, 2005) (claiming that all statically linked combinations of GPL-covered software are derivative works to which the GPL must apply).

[74] *See* GPL, *supra* note 1, at §§ 2, 5 (allowing modifications of the original program to be distributed if they are distributed under the GPL as well).

doesn't combine with a program until the user runs it, there is no reason to think the module, standing alone, is automatically a derivative work. The module might be a derivative by virtue of the small amount of interface information it must copy from the program in order to be compatible.[75] It might also be a derivative because the module's only possible use is to be combined with a particular program; therefore, it would make sense to consider the program and module to be a single work even before they are actually combined.[76]

The exact factors that make a dynamically loaded module a derivative work are unclear.[77] Ultimately, whether a given module that enhances a GPL program must itself be covered by the GPL will have to be decided in court.[78] If a module is not a derivative work, it is considered a separate, independent creation under copyright law,[79] and the module author is free to apply any distribution terms she desires, including terms that oppose the GPL's goals. This creates an enormous potential problem for the GPL. Suppose a programmer writes a dynamically linked module for the Linux kernel that controls a new brand of printer. If the programmer can write her module in a way that avoids being characterized as a derivative work, she can sell the module as proprietary software with all of the usual prohibitions against copying and distribution and, because the GPL will not apply, she can keep the source code secret.[80] Therefore, if many people can write improvements to GPL software in the form of modules without contributing the improvements' source code back to the GNU/Linux development community, the system of cooperation and reputation-based incentives that first led to the creation of GNU/Linux and other FOSS could begin to break down. As a result, the ability to link with GPL software will no longer be a strong incentive to license one's own software with the GPL. Because almost any improvement to a

[75] *Cf.* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) (rejecting plaintiff's claim that including four-letter compatibility code made defendant's program infringing). The opinion does not state whether Sega's claim regarding the four-letter compatibility code was for a violation of the reproduction right or the derivative work right; presumably either one could apply. *See id.*

[76] *See* Micro Star v. FormGen Inc., 154 F.3d 1107, 1112 n.5 (9th Cir. 1998) (reasoning that a set of files that can be used with more than one video game may not be a derivative work of that game).

[77] Free Software Foundation, Frequently Asked Questions about the GNU GPL, http://www.gnu.org/licenses/gpl-faq.html#MereAggregation [hereinafter GPL Aggregation FAQ] (last modified Aug. 19, 2005) (acknowledging that the legal definition of a derivative work is not conclusively established with regard to software).

[78] *Id.* ("This is a legal question, which ultimately judges will decide.").

[79] *See* 17 U.S.C. § 103(b) (2000) (establishing that the copyrights in separate parts of a work, by different authors, are independent of each other if and when the parts are separated from a compilation or derivative work).

[80] *See* GPL § 2, *supra* note 1, at ¶ 3 ("[I]t 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.").

program can be written in the form of a module, the loss of this incentive could mean that many future improvements to GPL software will no longer be freely modifiable by all.[81] Thus, correctly defining the scope of the derivative work as it applies to dynamically linked modules is crucial for the effectiveness of the GPL, as the ordering force of the FOSS community.

C. *The FOSS Community's Interpretation of Derivative Works*

Because the GPL is often enforced informally by the Free Software Foundation, the FSF's views on when a dynamically linked module should be considered a derivative work are an important force in shaping FOSS community norms. These informal rules also illustrate the technical elements that a court might consider in evaluating the derivative works question. The FSF and its founder, Richard Stallman, have expressed several views, some contradictory, on the derivative work test. Officially, the FSF acknowledges that the legal definition of a derivative work is not yet well defined as applied to software.[82] On the other hand, Stallman appears to believe that most or all dynamically linked modules are derivatives of the program to which they link.[83] In addition, the Lesser General Public License (LGPL), a variation on the GPL that is also published by the FSF, specifically allows dynamic linking with covered modules under license terms of the user's choice.[84] By writing a separate license that allows dynamic linking under any license terms, the FSF implied that it did not intend the GPL to grant that permission. The FSF website also gives its authors a more nuanced view of what the legal test should be:

- Is the technical method for linking the module and the program one that is ordinarily employed for communication between separate programs, or for communication within a single program?
- Alternatively, do the program and the module exchange simple data and commands, or do they share "complex internal data structures" across an interface? In other words, is the module's interface more like a pocket calculator or an airplane cockpit?[85]

Linus Torvalds, author of the Linux kernel, proposed a different test: "there

[81] *See Henderson*, *supra* note 50 ("These modules can do lots of things . . . .").

[82] *See* Free Software Foundation, GNU Lesser General Public License, at § 5 (Feb. 1999), http://www.fsf.org/licensing/licenses/lgpl.txt [hereinafter LGPL] (acknowledging that whether a particular work in the context of the LGPL is derivative is "not precisely defined by law"); GPL Aggregation FAQ, *supra* note 77 (describing the distinction between derivative and independent works as a "legal question, which ultimately judges will decide").

[83] E-mail from Richard Stallman to Steve Baur (June 15, 1998, 00:02:17 MST), http://list-archive.xemacs.org/xemacs-beta/199806/msg00523.html (asserting that any non-GPL, dynamically linked extension to the GPL-covered Xemacs program would be a violation of the GPL).

[84] LGPL § 6, *supra* note 82 (allowing linking "under terms of your choice").

[85] *See* GPL Aggregation FAQ, *supra* note 77 (responding to the question, "What is the difference between 'mere aggregation' and 'combining two modules into one program'?") (author's paraphrase).

are cases where something would be so obviously Linux-specific that it simply wouldn't make sense without the Linux kernel. In those cases it would also obviously be a derived work, and as such . . . it falls under the GPL license."[86]

The next part of this Note traces the courts' development of analogous tests for when to allow the unauthorized linking of modules. Part IV explores how well the courts' reasoning can be reconciled with the FSF and Torvalds tests.

III. THE COPYRIGHT LAW OF LINKING

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, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted."[87] Because unauthorized derivative works infringe the copyright in the original work, the owner of the original work gains effective control over copying and distribution of an unauthorized derivative.[88] Software makers have invoked the derivative work right to prevent others from linking modules to their software.[89] As discussed above, the GPL purports to restrict which modules can be linked with GPL-covered programs by asserting the copyright holder's exclusive right to prepare derivative works. However, the cases discussed in this section show a trend toward limiting the use of copyright law to prevent linking,[90] a trend that has ominous implications for the effectiveness of the GPL.

A. *The Early Cases on Linking*

Although it did not involve computer software, *Worlds of Wonder, Inc. v. Veritel Learning Systems, Inc.* illustrates the early approach to linking.[91] Worlds of Wonder created Teddy Ruxpin, a teddy bear that spoke and sang, directed by a cassette tape that provided both Teddy's voice and a signal that animated his mouth and eyes.[92] Veritel created its own tapes for Teddy Ruxpin – essentially, new software that allowed Teddy to tell new stories.[93] The U.S. District Court for the Northern District of Texas categorized Teddy

[86] Alessandro Rubini, *An Interview with Linus Torvalds*, 32 LINUX GAZETTE, Sept. 1998, http://www.linuxgazette.com/issue32/rubini.html.

[87] 17 U.S.C. § 101 (2000).

[88] *See* Sean Hogle, *Unauthorized Derivative Source Code*, 18 No. 5 COMPUTER & INTERNET LAW. 1, 6 (2001) (explaining that although 17 U.S.C. § 103(a) (2000) puts unauthorized derivative works in the public domain, the owner of the original work can prevent copying of any of the original material that appears in the derivative).

[89] *See infra* Part III.B.

[90] *See* Hogle, *supra* note 88, at 6 (observing that courts "have been increasingly solicitous of parties who copy only interfaces of copyrighted software . . . to achieve interoperability").

[91] 658 F. Supp. 351 (N.D. Tex. 1986).

[92] *Id.* at 352.

[93] *Id.* at 353, 356.

as an audiovisual work with a valid copyright.[94] Because Teddy performed a similar act when playing Veritel's tapes or Worlds of Wonder's own tapes, the court found Veritel's tapes to be an infringing derivative work, and entered a preliminary injunction ordering Veritel to stop selling its tapes.[95]

In another case from the same period, *Midway Manufacturing Co. v. Artic International, Inc.*, the Seventh Circuit upheld a preliminary injunction against a maker of add-on circuit boards for video arcade games.[96] The defendant's boards contained software that caused the plaintiff's game to run faster.[97] The court indicated that the copyrighted work was the series of images and sounds created by the game.[98] Because the defendant's add-on boards made this display run faster, the boards themselves were unauthorized derivative works.[99] Surprisingly, neither of the defendants' infringing products contained any copied material.[100] The actual products were simply modules that were compatible with plaintiffs' systems and altered their operation. These holdings seem to contradict the basic principle that "[a] work is not derivative unless it has substantially copied from a prior work."[101] The cases assume that if the products *in combination* form a derivative work, then the module *standing alone* can be enjoined as a derivative. Acknowledging that their holdings stretched the statutory definition of derivative works,[102] the *Worlds of Wonder* and *Midway* courts essentially defined the infringing work as the output generated by the combined products.[103] According to these cases, if the

[94] *Id.* at 355.

[95] *See Id.* at 356.

[96] 704 F.2d 1009, 1011 (7th Cir. 1983) (upholding an injunction against further infringement).

[97] *Id.* at 1010 (describing the effects of defendant's add-on circuit boards on plaintiff's video games).

[98] *Id.* at 1011-12 (identifying the copyrighted work that was allegedly infringed).

[99] *See id.* at 1013 (holding that because the add-on circuit board modified the game's audiovisual display, the board was an infringing derivative).

[100] Veritel's tapes for Teddy Ruxpin contained different songs and stories than Worlds of Wonder's own tapes. *Worlds of Wonder*, 658 F. Supp. at 353 (repeating the plaintiff's complaint that "the Veritel tapes alter Teddy Ruxpin's character"). Artic's video game add-on boards contained some code copied from Midway's own software, but the district court's infringement holding was focused on the modified visual display. *See* Midway Mfg. Co. v. Artic Int'l, Inc., 547 F. Supp. 999, 1004, 1013-14 (N.D. Ill. 1982). The Seventh Circuit's affirmance did not mention the copied code at all. *See Midway*, 704 F.2d at 1010 (observing only that the add-on board accelerates the game); *see also* Micro Star v. FormGen Inc., 154 F.3d 1107, 1112 (9th Cir. 1998) (finding that a collection of "MAP files," which generate new levels for the Duke Nukem video game, generated infringing derivative works because they "describe" a substantially similar audiovisual display).

[101] 1 NIMMER, *supra* note 41, at § 3.01 (defining a derivative work) (emphasis omitted).

[102] *See Midway*, 704 F.2d at 1014 (admitting that "[i]t is not obvious" whether the product in question is a derivative work).

[103] *Cf. id.*, 704 F.2d at 1011-12 (combination of speed-up cartridge and original game); *Worlds of Wonder*, 658 F. Supp. at 355 (combination of new song-tapes and original bear). Even if consumers actually combine the two products, the distributor of the add-on product may be liable for contributory infringement for "helping consumers create derivative works." *See* Micro Star v. FormGen Inc., 154 F.3d 1107, 1113 (9th Cir. 1998)

combined output of an original program and an add-on module has the same "total concept" as the output of the original program, then defendant's add-on module is a derivative work, even if the module itself incorporates little or no material from the original.[104] Of course, any compatible tape played in Teddy Ruxpin or any add-on to Midway's video game will almost certainly produce output very similar to the original, since the original component (the bear or the video game console) remains a major part of the combined system.[105] Under this test, any add-on software, no matter how different from the original software, will probably be an infringing derivative. Thus, these cases conclude that nearly any unauthorized linking is infringing, at least when the linked code alters the original program's audiovisual output.[106]

Market effects played a strong role in *Worlds of Wonder* and *Midway*. The district courts in both cases asserted that copyright should protect plaintiffs' investment in their reputations, and that defendants' add-on modules would damage the reputations of the originals.[107] More importantly, the courts considered the right to create add-on modules a type of merchandising right traditionally protected by copyright,[108] and implied that the market value of any add-on modules rightfully belonged to the developers of the original product.[109] Although the economic analysis in both cases technically pertained

(differentiating direct infringement from contributory infringement). Although the output of the programs in these cases was often an audiovisual display on a computer or television screen, the Federal Circuit has suggested that an arbitrary numerical data stream used internally and never displayed to the user was copyrightable material. *See* Atari Games Corp. v. Nintendo of Am., Inc. 975 F.2d 832, 840 (Fed. Cir. 1992) (finding that the "unique and creative" arrangement of programming instructions in the infringed product involved a creative element that merited copyright protection).

[104] *Worlds of Wonder*, 658 F. Supp. at 355 (quoting Sid & Marty Krofft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157, 1166-67 (9th Cir. 1977)).

[105] *See* Christian H. Nadan, *A Proposal To Recognize Component Works: How A Teddy Bears On The Competing Ends of Copyright Law*, 78 CAL. L. REV. 1633, 1652-53 (1990) (describing the reasoning in *Worlds of Wonder* as circular because the court "simply watched one bear perform twice and declared the displays the same"); *see also Micro Star*, 154 F.3d at 1112 (observing that plaintiff would almost certainly succeed in proving that defendant's "map files," which described additional levels of play for plaintiff's Duke Nukem video game, were infringing derivatives because the combined work of the game and map files used audiovisual information that came entirely from the original game).

[106] *See* Hogle, *supra* note 88, at 7 (concluding that despite the modern trend toward less copyright protection for software interfaces, linked code that enhances a program's visual display will generally be found to infringe).

[107] *Worlds of Wonder*, 658 F. Supp. at 357 (reasoning that "the popularity of the Teddy Ruxpin product is jeopardized when tapes altering the image of Teddy Ruxpin are played"); *Midway*, 547 F. Supp. at 1014 ("Plaintiff's reputation for high quality and distinctive video games is entitled to protection from copyright infringers, just as its potential sales are.").

[108] *See Worlds of Wonder*, 658 F. Supp. at 356.

[109] *See* Midway Mfg. Co. v. Artic Int'l, Inc., 704 F.2d 1009, 1013 (7th Cir. 1983) ("[C]opyright owners would undoubtedly like to lay their hands on some of that extra revenue. . . ."); *see also Micro Star*, 154 F.3d at 1113 ("[O]nly [plaintiff] has the right to enter that market; whether it chooses to do so is entirely its business.").

only to a preliminary injunction analysis,[110] the opinions suggest that the copyright in a technical work includes the right to control linking with that work to reap financial gain and to safeguard a commercial reputation.

Both courts explained their holdings by pointing out that the defendants' products were designed specifically to be combined with plaintiffs' products, and were useful only when combined.[111] This logic helps to justify the surprising result that an add-on module that incorporates no protected expression may still be a derivative work: if the module's only possible purpose is to link with and modify a specific product, it makes more sense to consider the combined programs or their output as the infringing "work." At the same time, this justification limits the scope of the holdings, because a module that can be used with even two different programs is more credibly a freestanding work.[112] This suggests that a module that is compatible with multiple programs will not be considered a derivative.

In considering the public interest, the only factor identified in *Worlds of Wonder* and *Midway* was copyright law's basic purpose to provide economic incentives for creative expression.[113] The courts found no public interest in encouraging the creation of add-on modules to plaintiffs' products,[114] even though the modules were arguably original creative works themselves.

B. *The Linking Exemption to the Derivative Work Right*

1. Development of the Exemption

By 1992, courts began to recognize that traditional copyright analysis did

[110] *Worlds of Wonder*, 658 F. Supp. at 357 (granting plaintiff's motion for a preliminary injunction); *Midway*, 704 F.2d at 1011 (describing the case as an appeal from a grant of preliminary injunction).

[111] *Worlds of Wonder*, 658 F. Supp. at 356 (emphasizing that defendants' infringing "tapes were designed exclusively for [plaintiffs'] Teddy Ruxpin"); *Midway*, 547 F. Supp. at 1014 (pointing out that defendants' "speed-up kit was designed solely to modify Midway's Galaxian game"); *see also Micro Star*, 154 F.3d at 1112 n.5 (indicating that infringing "MAP files" could only be used with plaintiff's video game, and suggesting that if another game could use the same files, the files would not infringe).

[112] *See Micro Star*, 154 F.3d at 1112 n.5 (limiting the holding by observing that if another game could use the map files to tell a completely different story, then the files would not infringe).

[113] *Worlds of Wonder*, 658 F. Supp. at 357 (asserting that an "injunction would serve the public interest by preserving the integrity of copyright laws which encourage individual effort and creativity by granting valuable enforceable rights"); *Midway*, 547 F. Supp. at 1015 (recognizing that "[t]he Copyright Act evidences a public interest in creativity by demonstrating an intent to provide an economic reward for creative expression," and contending that granting plaintiff's injunction would further that interest); *see also* U.S. CONST. art. I, § 8, cl. 8 (empowering Congress to "promote the Progress of Science and useful Arts" by granting "exclusive Right[s]" to "Authors and Inventors").

[114] *Midway*, 547 F. Supp. at 1015 ("The court can conceive of no public interest that is furthered by allowing defendant to continue to distribute and sell its infringing material.").

not apply well to computer software.[115] Although Congress has declared that computer programs are "literary works" protected by copyright,[116] programs also have many of the attributes of a machine or process, which traditionally have been protected by patent law.[117] Because computer software is functional and utilitarian rather than purely expressive, courts began to formulate different copyright rules for software, often narrowing the scope of protection.[118] Instead of comparing the "total concept" of programs, for example, courts began to compare programs at multiple levels of abstraction, from the literal lines of program code to the overall structure and function of the program.[119] Applying the principle that pure ideas are not protected by copyright,[120] the courts began denying protection to specific features of programs that are purely functional, dictated by efficiency, or necessary for compatibility with other programs.[121] Based on this change, courts became increasingly unwilling to let software makers use copyright to control linking, as discussed below.

A pair of video game cases from the Ninth Circuit illustrates the change and its limits. In *Sega Enterprises v. Accolade*, the Ninth Circuit found that "reverse engineering" Sega's video game system to create games that ran on the system was a permitted "fair use" even though it involved copying Sega's code.[122] Similarly, in *Sony Computer Entertainment, Inc. v. Connectix Corp.*,

[115] Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (Boudin, J., concurring) ("Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit."); Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992) (describing earlier attempts to apply copyright to software as trying "to fit the proverbial square peg in a round hole").

[116] *See* 17 U.S.C. § 106 (2000 & Supp. 2004) (including "literary works" among the enumerated categories of protected works of authorship); *Altai*, 982 F.2d at 712 (acknowledging that "Congress has made clear that computer programs are literary works entitled to copyright protection").

[117] *See* Patent Act, 35 U.S.C. § 101 (2000) (allowing patent protection for "any new and useful process, machine, manufacture, or composition of matter").

[118] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1993) ("[T]o those used to considering copyright issues in more traditional contexts, [denying copyright protection against unauthorized linking] may seem incongruous at first blush."); *Altai*, 982 F.2d at 712 (acknowledging that the court's new test for nonliteral copyright infringement in software may "narrow[] the scope of protection").

[119] *See Altai*, 982 F.2d at 707; *see also Sega*, 977 F.2d at 1525 (reasoning that because "each subroutine [a subpart of a program] is itself a program," determining infringement based only on the overall function of the program is inaccurate).

[120] 17 U.S.C. § 102(b) (2000) (denying copyright protection to "any idea, procedure, process, system, method of operation, concept, principle, or discovery"); *see also Altai*, 982 F.2d at 707-08 ("In order not to confer a monopoly of the idea upon the copyright owner, such expression should not be protected.").

[121] *Altai*, 982 F.2d at 707-10.

[122] 977 F.2d at 1527-28; *see also* N.Y. Times Co. v. Roxbury Data Interface, Inc., 434 F. Supp. 217, 2226-27 (D. N.J. 1977) (finding that defendants' index to plaintiffs' materials enhances the usefulness of plaintiffs' materials but is nonetheless a fair use). The Federal Circuit, however, reached the opposite result in a case with facts similar to *Connectix*. *See*

the court allowed Connectix to reverse engineer Sony's game console in order to write an "emulator" that allowed users to play Sony's games on a personal computer.[123] Both Accolade's new games and Connectix's emulator that played Sony's original games could be considered add-on modules.

But *Sega* and *Connectix* do not follow the result from the earlier cases, which held that an add-on module can be infringing even if it contains no copied material.[124] More recent cases seem to require some actual copying to find infringement, and this is true even if a program is useful *only* as an add-on module to another program.[125] A module that modifies the audiovisual output of a program is not automatically a derivative when the module itself contains no audiovisual information.[126] Pushing their holdings further, the cases strongly imply that any parts of a program that must necessarily be copied in order to create a compatible module are not protected by copyright, denying copyright holders one of their key tools for controlling unauthorized linking.[127] In addition, *Sega* gives software makers some affirmative right to create unauthorized modules for a program – even if in the process of developing a module they nominally infringe a copyright by making temporary copies of the original code.[128] This Note refers to this as the "linking exemption." The

Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 845 (Fed. Cir. 1992) (upholding an injunction barring Atari from selling its unauthorized, Nintendo-compatible games).

[123] 203 F.3d 596, 608 (9th Cir. 2000) (holding that reverse-engineering of a video game system in order to write emulation software that plays games designed for the system is a fair use).

[124] *See* Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc., 964 F.2d 965, 969 (9th Cir. 1992) (distinguishing *Midway* by pointing out that Galoob's add-on module for the Nintendo system, the Game Genie, "does not physically incorporate a portion of the copyrighted work [such as the Nintendo console or games]" as the *Midway* defendant's chip had). *But see* Micro Star v. FormGen Inc., 154 F.3d 1107, 1112 (9th Cir. 1998) (declaring "MAP files," which extend a video game by adding on more levels, to be infringing derivatives even though the files did not contain any of the game's code or artwork). *Micro Star* shows that the principle from *Midway* – that an add-on module containing no copied material can be infringing – may still carry some weight when an unauthorized module's only possible use is with a particular copyrighted program. *Id.*; *see also infra* Part IV.B.3.

[125] *Galoob*, 964 F.2d at 969 (acknowledging that the Game Genie worked only with the Nintendo system, but finding no infringement because the Game Genie "[did] not contain or produce a Nintendo game's output in some concrete or permanent form"). Even a small amount of copying may not be enough to find infringement. *See Sega*, 977 F.2d at 1515-16 (commenting that Accolade's add-on games "contained the standard header file that included the TMSS initialization code," but still not finding infringement).

[126] *See Galoob*, 964 F.2d at 968 (pointing out that the Game Genie could not produce an audiovisual display on its own, and for that reason, "no independent work [was] created").

[127] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1993) (determining that "functional requirements for compatibility with the Genesis console . . . are not protected by copyright"). Functional requirements certainly include interface specifications, *see supra* Part II.A, but whether this holding encompasses actual copied code of more than trivial length is an open question. *See infra* Part IV.B.2.

[128] *See Sega*, 977 F.2d at 1527-28 (holding that otherwise infringing copies made during the development of an add-on module are permitted when the copies are a necessary step toward making the module compatible).

affirmative right of the module author takes a small bite out of the exclusive rights of the original programmer, and the latter cannot "erect an artificial hurdle" to the module writer's own creative process.[129] The right to create add-on modules free of the original software maker's control seems to apply even when the software maker is willing to license the right to link.[130] Sega had offered Accolade a license agreement that would permit Accolade to create compatible games, on the condition that Accolade allow Sega to manufacture all of its games.[131] Nonetheless, the court allowed Accolade to build games without a license from Sega.[132]

2. Policy Rationales for the Exemption

Sega and *Connectix* rejected *Worlds of Wonder*'s underlying assumption that when a program contains an interface for add-on modules, the original program's copyright owner has the sole right to authorize the making and selling of those modules and to profit from their sale.[133] The Ninth Circuit proffered both economic and natural rights justifications for this change.

Much of the reasoning in *Sega* emphasizes the right to understand and learn from the ideas embedded in software.[134] By using the terms "understanding" and "learning," the court drew an analogy between developing original add-on modules and the sort of knowledge-increasing activities like scholarship that are sometimes allowed even when they involve copying.[135] Of course, the makers of unauthorized video games and other add-on modules are generally not in an academic setting, but the same rationale applies: unauthorized copying is sometimes permitted when the body of knowledge available to the public will increase.[136] In addition to spreading knowledge, another goal of copyright law is giving economic incentives to create new works by building

[129] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 605 (9th Cir. 2000) (declining to require defendants to minimize intermediate copying during development of their module, a requirement that would cause "wasted effort" in the use of unprotected ideas) (internal citation omitted).

[130] *See Sega*, 977 F.2d at 1514 (describing Sega's licensing of its copyrighted code to independent developers of computer game software).

[131] *Id.*

[132] *Id.* at 1527-28 (finding no copyright infringement).

[133] *See Connectix*, 203 F.3d at 602 (holding that fair use doctrine excuses the copying necessary to create a compatible module); *Sega*, 977 F.2d at 1522 (finding linking interface an unprotectable idea).

[134] *Sega*, 977 F.2d at 1514 (framing the issue around the ability to "gain an understanding" of functional elements of a program); *id.* at 1522 (allowing defendant Accolade to write software "based on what it had learned" from disassembling Sega's console); *accord* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 843 (Fed. Cir. 1992) (finding that "understand[ing] the ideas and processes in a copyrighted work" may be permitted).

[135] *See* 17 U.S.C. § 107 (2000) (listing "scholarship, or research" as some of the uses typically subject to the fair use defense).

[136] *See Connectix*, 203 F.3d at 4605 (finding no support for distinction between "studying" and "use" for fair use purposes).

on the ideas in existing works.[137] Software, like traditional literary works, is written using an iterative process that almost always builds on the work of others.[138] Allowing programmers to link their modules with other programs, without permission, creates a stronger incentive to write modules in the first place. Thus, because the public benefits from the existence of a greater selection of add-on modules,[139] module writers have a limited right both to link to existing programs and to profit from selling the linked modules.[140]

The courts use a variety of formulas in determining how to divide economic rewards between the original software maker and the maker of add-on modules. While an add-on module may legitimately compete in the market with the original software,[141] "diminishing potential sales, interfering with marketability, or usurping the market" for the original supports a finding of infringement.[142] These tests are nothing if not vague: at what point does competing in a market become "usurping" the original product?[143] When is using the information in a program to build a compatible program "misappropriat[ion]?"[144] The courts seem to resolve these uncertainties with implicit references to the law of unfair competition.[145] Unfair competition law gives content to the vague tests of "misappropriat[ion]," "exploitation," and

[137] *Sega*, 977 F.2d at 1523 ("It is precisely this growth in creative expression, based on the dissemination of other creative works and the unprotected ideas contained in those works, that the Copyright Act was intended to promote.").

[138] *See* Leeds Music, Ltd. v. Robin, 358 F. Supp. 650, 659 (S.D. Ohio 1973) (asserting that every copyrighted work builds on an earlier work in some way); WEBER, *supra* note 28, at 75 (explaining that programmers, especially but not exclusively in the FOSS paradigm, prefer to build on existing code instead of rewriting).

[139] *Sega*, 977 F.2d at 1523 ("[P]ublic benefit . . . may arise because the challenged use serves a public interest"); *cf.* Midway Mfg. Co. v. Artic Int'l, Inc., 547 F. Supp. 999, 1015 (N.D. Ill. 1982) (finding that permitting defendant's add-on modules would serve no public interest).

[140] *See Connectix*, 203 F.3d at 606 n.10 (rejecting argument that commercial use raises a "presumption of unfairness" that defeats a fair use defense); *Sega*, 977 F.2d at 1523 (holding that competing in the same market as a copyrighted work does not defeat fair use).

[141] *See Connectix*, 203 F.3d at 607 (stating that defendant's emulator for Sony's games "does not merely supplant" Sony's console); *Sega*, 977 F.2d at 1522 (commenting that defendant "did not seek to avoid paying a customarily charged fee for use of [Sega's interface]"); *id.* at 1523 (calling defendant's product a "legitimate competitor" in plaintiff's market even though defendant's product "undoubtedly affected" that market).

[142] *Sega*, 977 F.2d at 1523 (citing Hustler Magazine, Inc. v. Moral Majority, Inc., 796 F.2d 1148, 1155-56 (9th Cir. 1986)); *see also* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 843 (Fed. Cir. 1992) (stating that "extensive efforts to profit" from linking are not protected by fair use); *id.* at 844 (observing that fair use does not cover attempts to "exploit commercially or otherwise misappropriate protected expression").

[143] *See Sega*, 977 F.2d at 1523 (inquiring whether defendant's use would "usurp" the market for the copyrighted work).

[144] *See Atari*, 975 F.2d at 843 (warning that fair use is a "limited exception" and "not an invitation to misappropriate protectable expression").

[145] *See* Int'l News Serv. v. Associated Press, 248 U.S. 215, 236 (1918) (recognizing uncopyrightable news items as "quasi property" under "unfair competition" theory, because current news was the "stock in trade" for both of the competing parties).

avoiding "a customarily charged fee"[146] by looking to the standards of fair conduct in a given industry.[147] The holding in *Sega* was based not solely on copyright doctrine but at least partially on the court's conclusion that creating compatible, unauthorized video games is an acceptable practice in the software industry.[148]

In addition to an unfair competition rationale, the concept of network externalities seems to have informed the courts' opinions on linking. Network externalities cause a product to increase in value when it is compatible with other widely used products.[149] Customers will often choose a software product not simply because it is technically superior, but because it is compatible with software they already own.[150] Thus, network externalities increase the economic value of the right to link, potentially magnifying the reward that software makers can extract from their copyrights. When combined with control over the right to link, network externalities can allow a software maker to leverage legal control over one product into control over compatible products.[151]

Sega indicates that copyright does not allow software makers to capture the value derived from network externalities, because doing so is a form of monopoly control that the law disfavors.[152] The courts allow developers to copy code whenever copying is the only way to create compatible modules.[153]

---

[146] *Sega*, 977 F.2d at 1523 (characterizing defendant's "exploitation" as "indirect" and finding that defendant "did not seek to avoid paying a customarily charged fee"); *Atari*, 975 F.2d at 843 ("Atari could not . . . misappropriate protected expression.").

[147] *See Int'l News*, 248 U.S. at 236 ("[U]nfair competition in business must be determined with particular reference to the character and circumstances of the business."). Focusing on the customs of the particular market gives guidance in cases where copyright law does not clearly provide a remedy. *See* Leo J. Raskind, *The Misappropriation Doctrine as a Competitive Norm of Intellectual Property Law*, 75 MINN. L. REV. 875, 876-77 (1991) (arguing that where copyright protection is not clearly available, but conduct at issue seems like "piracy," courts should look to the "competitive market context").

[148] *Sega*, 977 F.2d at 1523 (describing defendant Accolade's activities as "legitimate").

[149] *See* Timothy S. Teter, *Merger and the Machines: An Analysis of the Pro-Compatibility Trend In Computer Software Copyright Cases*, 45 STAN. L. REV. 1061, 1066-67 (1993) (describing how the "satisfaction" a software user enjoys increases with the number of others who use the software).

[150] *See id.*

[151] *See id.* at 1067 (explaining that network externalities allow software producers to extend monopoly control).

[152] *Sega*, 977 F.2d at 1523-24 (holding that plaintiff's attempt to extend legal control by asserting copyright against module writers shifted equities in defendant's favor); *see also* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir. 1995) (Boudin, J., concurring) (acknowledging that widespread use of a particular interface for add-on modules raised the economic value of the interface, but denying control over that interface under copyright principles).

[153] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 608 (9th Cir. 2000) (permitting intermediate copying that is necessary to the process of writing a compatible module); *Sega*, 977 F.2d at 1527-28 (concluding that disassembly is a fair use of copyrighted work where it is the only means of access to ideas embedded in copyrighted software).

This test effectively separates the value created by network externalities – the value of compatibility – from the intrinsic value or technical merit of a program.[154]

Network externalities also give rise to a natural rights argument for affording protection to linking. As noted above, software users buy a program, in part, because it is compatible with other software they have invested in, not because it is technically superior or innovative.[155] Because the market value of software is derived in part from customers' prior investments in other compatible software rather than the software makers' technical or creative efforts, the linking cases imply that the commercial value of compatibility does not rightfully belong to the maker of the original software.[156]

IV. Basic Rules for Linking with GPL Software: A First Approximation

A. *Static Linking and the Pre-1992 Cases*

What do the cases on linking mean to the programmer from Part II, who wants to add an important feature to the Linux kernel program by writing a kernel module? Recall that she would prefer to release her module under a standard proprietary license rather than the GPL. To do so, she must establish that the module is not a derivative work of the Linux kernel.[157]

Under either the older or newer cases, a statically linked module is a derivative work once it is linked. After linking, the object code file contains both the GPL-covered Linux kernel and the module code, blended together and very difficult to separate. Like adding a chapter to a copyrighted novel and republishing it, this situation fits the canonical definition of a derivative work.[158] The only way our programmer could distribute such a work in object code form without infringing copyright is by adhering to the GPL.[159]

Dynamic linking can occur through many different technical mechanisms, either within a single computer or between multiple computers on a

[154] *See Sega*, 977 F.2d at 1523 (finding that copied material was not a part of the plaintiff's software that "determine[d] the program's commercial success").

[155] *See* Teter, *supra* note 149, at 1067 (attributing economic value of disputed command structure to users' investment in training, rather than the structure's inherent usefulness).

[156] *See id.* ("Standardization of user interfaces prevents user 'lock-in' because users do not have to learn a new user interface in order to switch application programs."); *Lotus*, 49 F.3d at 819 ("A new [interface] may be a creative work, but over time its importance may come to reside more in the investment that has been made by users in learning the [interface].").

[157] *See supra* Part II.B.

[158] *See* Anderson v. Stallone, No. 87-0592, 1989 U.S. Dist. LEXIS 11109, at **31-32 (C.D. Cal Apr. 25, 1989) (finding that original portions of a movie screenplay were not "severable" from the characters and plots taken from an earlier work, so the screenplay was an infringing derivative in its entirety).

[159] *See* GPL, *supra* note 1, at § 2(b).

network,[160] and the choice of linking method does not definitively determine whether the module is a derivative.[161] That said, nearly all dynamically linked modules are derivative works under the logic of *Worlds of Wonder* and *Midway*. Those cases are congruent with the first prong of the FSF's proposed derivative works test, which looks at whether the method of linking is one commonly used between separate programs or one employed between subcomponents of a single program.[162] Writing a new subcomponent linked to an existing program would appear to be part of the valuable "merchandising rights" reserved to the original program's copyright owner.[163] The strength of this prong of the test would make the FSF test's second prong – the type and complexity of data exchanged across the interface – superfluous.[164]

The *Worlds of Wonder* approach promotes the FSF's goals particularly well in one sense: subjecting nearly all add-on modules to the GPL would expand the body of available FOSS quickly and create a serious impediment to any proprietary (non-FOSS) extensions. This maintains the incentive for new FOSS development. Of course, *Worlds of Wonder*, *Midway,* and similar cases based in copyright presumably apply to any software regardless of license. As *Midway* and *Atari* illustrate, an expansive definition of derivative works gives proprietary software makers the ability to prevent others from extending their software with modules. Due to network externalities, this ability is especially powerful for a maker of popular and fundamental "platform" software such as an operating system, web browser, or networking protocol.[165] Many important and valuable programs are essentially "modules" of these fundamental programs. When applied to these programs, the *Worlds of Wonder* approach would severely limit programmers' freedom to write other software that links with proprietary software. This result is contrary to the FSF's goal of maximizing the "freedom to share and change" software.[166] Thus, an expansive definition of derivative works helps the FOSS community by strengthening the GPL, while at the same time strengthening proprietary licenses at the expense of the cooperative FOSS ethic. The next section explores this apparent paradox in more detail.

B. *The Linking Exception Presents a Paradox for the GPL*

Sega and other more recent cases suggest that a compatible module is not

[160] *See supra* Part II.A.

[161] The boundaries of a "work" under copyright law do not map cleanly onto the different mechanisms of linking. *See* GPL Aggregation FAQ, *supra* note 77.

[162] *See supra* Part II.C.

[163] *See Worlds of Wonder*, 658 F. Supp. 351, 356 (N.D. Tex. 1986) (stressing that "merchandise licenses are commercially valuable" and that defendant's derivative works "undermine the carefully tailored image of Teddy Ruxpin"); Midway Mfg. Co. v. Artic Int'l, Inc., 547 F. Supp. 999, 1014 (N.D. Ill. 1982) (giving the original copyright owners the sole merchandising right to prepare add-on products).

[164] *See supra* Part II.C.

[165] *See supra* note 149 and accompanying text (describing network externalities).

[166] GPL, *supra* note 1, at Preamble.

automatically a derivative work of the program with which it links.[167] Literal copying of interface information is not infringing, either because it is de minimis[168] or an unprotectable "idea."[169] A module that complements but does not replace a copyrighted program is outside the control of the original copyright owner.[170] This result applies even if the owner is willing to license the right to create modules subject to conditions.[171]

If the *Sega* line of cases applies in the same way to GPL-licensed software, then most dynamically linked modules, such as Linux kernel modules, will not be derivative works. These modules can be released under proprietary licenses and without disclosing the corresponding source code. For the reasons described above, this creates a substantial loophole in the GPL. Though the GPL will continue to ensure that the existing core of programs like Linux remain freely modifiable, the ability to add proprietary extensions on which users may come to depend makes the "openness" of the core program increasingly irrelevant.[172] If the proprietary extensions become widely used, the portions of GNU/Linux that are still GPL-covered may become obsolete when used without the proprietary modules.[173] Users would have to buy the modules and agree to their licensing terms, in order to have a useful and relevant GNU/Linux installation.[174]

Of course, the courts will not give a free pass to any kind of copying simply because it is done to create a compatible module.[175] The influence of traditional copyright principles is still strong, even for software. A few recent cases show how the linking exemption might correspond to the FOSS proponents' own derivative works tests, and also highlight the ambiguities that remain in the test.

[167] Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1528 (9th Cir. 1992) (allowing copying necessary to create compatible modules).

[168] *Id.* at 1524 n.7 (finding no infringement in use of short data sequence comprised of the letters S-E-G-A).

[169] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 603 (9th Cir. 2000) (applying fair use defense in order to protect access to "idea" embodied in compatibility code).

[170] 17 U.S.C. § 103(b) (2000) (granting independent copyright interest in portions of derivative work that can be separated from original work).

[171] *Sega*, 977 F.2d at 1514 (observing that defendant declined a license from Sega for the allegedly infringing activity, but finding defendant's activity a fair use regardless).

[172] Free Software Foundation, *supra* note 9, (last modified May 5, 2005).

[173] *See* Richard Stallman, The X Window's Trap, http://www.gnu.org/philosophy/x.html (last modified Apr. 26, 2004) (concluding that if the GPL does not apply, "anyone [can] make a non-free version dominant, if he . . . invest[s] sufficient resources to add significantly important features using proprietary code").

[174] *See id.* ("People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away.").

[175] *See* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 4843 (Fed. Cir. 1992) (refusing to excuse defendant's copying, even though it was done to create compatible video game modules).

1. Expressive Audiovisual Programs

Because many cases on derivative software have involved video games, the law in this area has undoubtedly been shaped by the customs and strategies of that industry. Although video games are software, their most important aspect is the visual experience they create for the user. The Copyright Office recognized this fact by allowing two copyright registrations on a game: for the program code as a "literary work" and for the output as "visual artwork."[176] Besides being primarily visual, and unlike most software, video games are highly creative. In many respects, they are similar to movies.[177] Because movie-like audiovisual works lie "closer to the core of intended copyright protection" than other software, courts afford stronger protection to video games.[178] In contrast, graphical elements created by functional programs, like the familiar windows and menus used by most application software, receive only weak copyright protection.[179] The layout of these elements is a form of interface, and anyone may write modules making use of that interface without permission.[180] The deciding factor seems to be whether that output is a work of pure creativity, not whether the program creates audiovisual output. Currently, most FOSS is highly functional software such as operating systems, server programs, and application programs.[181] The courts' tendency toward strong protection for creative audiovisual output may not apply to highly

[176] U.S. Copyright Office, Literary Works Registration, http://www.copyright.gov/register/literary.html (last visited Sept. 30, 2005) (including "computer programs" as a type of literary work); U.S. Copyright Office, Visual Art Works Registration, http://www.copyright.gov/register/visual.html (last visited Sept. 30, 2005) (follow "examples" hyperlink) (including "games" in the list of visual artwork).

[177] *See* Micro Star v. FormGen Inc., 154 F.3d 1107, 1112 (9th Cir. 1998) (referring to defendant's modules for plaintiff's video game as a "sequel" to the game); Matt Krantz, *Video Game College Is 'Boot Camp' for Designers*, USA TODAY, Dec. 3, 2002, *available at* http://www.usatoday.com/money/media/2002-12-03-video_x.htm (comparing movie and video game industries and describing visual experience of some games as similar to television).

[178] *See Micro Star*, 154 F.3d at 1113 ("The fair use defense will be much less likely to succeed when it is applied to fiction or fantasy creations, as opposed to factual works such as telephone listings.").

[179] *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 819 (1st Cir. 1995) (Boudin, J., concurring) (finding menu layout of spreadsheet program to be uncopyrightable, and that the "present case is an unattractive one for copyright protection of the menu"); Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1025-26 (N.D. Cal. 1992) (finding no infringement in Microsoft's use of menus, windows, and icons with a "look and feel" similar to Apple's user interface).

[180] The interface, in this case, is an interface between the human user and the software. *See Lotus*, 49 F.3d at 819 (Boudin, J., concurring) (offering no copyright protection to macro interface for add-on modules which automate common tasks).

[181] A search of the popular FOSS development site SourceForge.net shows that of the ten most active development projects, two are networking tools, two are graphics manipulation programs, three are network server programs, and three are utilities for manipulating text files. SourceFORGE.net, Statistics: Most Active, http://www.sourceforge.net (last visited Oct. 3, 2005).

functional software.

2. Creative or Arbitrary Interface Code

Although courts have recognized that allowing copyright law to control the linking of modules may stifle innovation, they are still wary of excusing significant literal copying, even where such copying is necessary to link a module to a program. In *Sega*, while the Ninth Circuit implied that interface information was not protected by copyright, the court explicitly did not rule on that question.[182] Instead, the court left open the possibility that a substantial, complicated, or creative interface specification might be protected by copyright.

As described above, to write a compatible module, a programmer often needs to copy a small amount of code from the original program.[183] In addressing Accolade's use of Sega's compatibility code (a file containing the letters S-E-G-A) in its games, the court emphasized that the file was trivially small.[184] In a similar situation, also in 1992, Atari Corporation copied the compatibility code from Nintendo's video game console in order to write games for that console.[185] The Federal Circuit, in holding Atari liable for infringement, pointed out that Nintendo's compatibility code used "creative organization and sequencing" to create a "purely arbitrary data stream."[186] The *Sega* court distinguished *Atari* by observing that Nintendo's compatibility code contained "creativity and originality" while *Sega*'s simple code did not.[187] Neither court considered it significant that Nintendo's compatibility code was intentionally complex, creative, and arbitrary to deter any unauthorized use.[188] Its "creativity" was similar to the arbitrary shape of notches in a key. Both the Ninth and Federal Circuits apparently considered creative programming for the purpose of preventing compatibility to be copyrightable despite the courts' policy that copyright should not restrict compatibility any more than necessary.[189]

---

[182] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1528 (9th Cir. 1992) (leaving open the issue of infringement in defendant Accolade's final products).

[183] *Id.* at 1515-16 (describing Accolade's need to put a small amount of data copied from Sega's console into its games in order for the games to run).

[184] *Id.* at 1524 n.7 (concluding that any infringement by use of the Sega compatibility code was de minimis).

[185] Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 836-37 (Fed. Cir. 1992) (describing Atari's use of Nintendo's compatibility code).

[186] *Id.* at 840.

[187] *Sega*, 977 F.2d at 1524 n.7 (distinguishing a complex, original compatibility code from a short, simple one).

[188] *Atari*, 975 F.2d at 836 (explaining that the compatibility code controls access to the Nintendo console).

[189] *See Sega*, 977 F.2d at 1524 n.7 (suggesting that a complex, original compatibility code might be copyrightable); *Atari*, 975 F.2d at 840, 846 (holding that Nintendo's "lock" program, which Atari needed to copy to create unauthorized game cartridges, was copyrightable, and determining that the record did "not demonstrate, as a matter of law, that

This conclusion fits the second element of FSF's proposed derivative works test: whether the communication between modules is "intimate" or involves "exchanging complex internal data structures."[190] Where the compatibility code is sufficiently complex, the policy of protecting the original copyright owner's creative investment may supersede the module writer's interest in creating a compatible module. Thus, one way FOSS programmers could ensure application of the GPL to new modules is to make module interfaces more complex or arbitrary. This would be an unfortunate result, as more complex interfaces seem likely to be less efficient and more prone to errors. Also, the degree of complexity necessary to overcome the module writer's fair use defense is unclear – the cases tell us only that it must be more than de minimis.[191]

3. Working Only with a Single Program

Another case decided long after *Altai* and *Sega* lends support to Linus Torvalds' proposal that a module is derivative when it can link only with one particular program.[192] In *Micro Star v. FormGen Inc.*, the Ninth Circuit used exactly that rationale (and others) to enjoin Micro Star's "map files," modules that described new levels for FormGen's Duke Nukem video game.[193] *Micro Star* shows that the *Worlds of Wonder* principle is still viable: a module that modifies the output of a program can be infringing even if the module contains no expression taken from the program.[194] The map files referred to and invoked the artwork contained in the Duke Nukem game, much as Veritel's tapes invoked the motors that animated Teddy Ruxpin.[195] The files, according to the court, were "sequels" to the game,[196] and thus derivative works. If the files could be used with any other game, however, they would not be infringing.[197]

Arguably, the logic of *Micro Star* could apply equally well to the facts of

such restrictions restrain the creativity of Nintendo licensees and thereby thwart the intent of the patent and copyright laws").

[190] GPL Aggregation FAQ, *supra* note 77 (explaining that intimate communication, in which complex internal data structures are exchanged, may serve as a basis for considering two modules a single program).

[191] *See Sega*, 977 F.2d at 1524 n.7 (classifying the letters S-E-G-A as de minimis for copyright purposes).

[192] *See supra* note 86 and accompanying text.

[193] 154 F.3d 1107, 1112 n.5 (9th Cir. 1998) (finding that Micro Star's files incorporated protected expression, and therefore constituted derivative works, because they could only be used with FormGen's video game). Micro Star did not create the map files, but merely compiled them into a collection and sold them. *Id.* at 1109.

[194] *See supra* notes 100-106 and accompanying text.

[195] *Id.* at 1110 (explaining that codes in the map files cause artwork from the game to appear at particular times and locations).

[196] *Id.* at 1112 (describing files as sequels because they used Duke Nukem to tell new stories).

[197] *See id.* at 1112 n.5 (explaining that if another game could use the files to tell a different story, the files "would not incorporate the protected expression").

Sega. Accolade's unauthorized game cartridges presumably worked only with the Sega console. However, the game program contained in the cartridge could run on other video game systems.[198] The cartridges, in effect, contained the game itself and some Sega-specific compatibility code.[199] In combination, these two elements worked only with the Sega console, but the game itself could be combined with different compatibility code and run on different companies' consoles. The map files in *Micro Star*, in contrast, could not be separated into original creative material and compatibility code. This suggests that separability plays a role in the derivative work analysis. In distinguishing original material from added material in derivative works, the Copyright Act supports this focus on separability.[200] Where elements of the original work "pervade" a derivative work and are inseparable, the derivative author can claim no copyright protection at all.[201]

Returning to the hypothetical raised in Part II, the archetypal Linux kernel module is a device driver, which allows a computer to connect to a particular external device.[202] Because a device driver written for a different operating system can be made into a Linux kernel module by "wrapping" it in compatibility code, it is a separable module and not Linux-exclusive – analogous to the game cartridges in *Sega*.[203] Many modules and libraries for FOSS programs are similarly separable, so they are probably not derivative works under the logic of *Micro Star* and *Sega*.

4. The Paradox

The cases on linking point to two categories of programs for which modules are most likely to be derivative works: programs whose main purpose is to generate creative audiovisual experiences and programs with complex or

[198] Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1515 (9th Cir. 1992) (recounting that Accolade adapted one game that it had previously released for personal computers so that the game would run on the Sega console).

[199] *See id.* at 1515-16 (describing how Accolade's engineers used the knowledge they had gained from reverse-engineering the Sega console to make the games they had already written compatible with the Sega console).

[200] *See* 17 U.S.C. § 103(b) (2000) (extending copyright in a derivative work only to "material contributed by the author" of the derivative).

[201] *See* Anderson v. Stallone, No. 87-0592, 1989 U.S. Dist. LEXIS 11109, at **28, 23-26, 31-32 (C.D. Cal. Apr. 25, 1989) (finding that the author of an unauthorized sequel to the *Rocky* films could claim no copyright protection at all because his work could not be separated from the original characters and plots that he copied).

[202] *See* Henderson, *supra* note 50, at § 2 (listing device drivers as a common type of kernel module).

[203] The court in *Sega* acknowledged that Accolade wrote its games originally for personal computers, and created compatibility code to allow the computers to run on the Sega console. *Sega*, 977 F.2d at 1515. By declaring the process of creating the compatibility code to be a fair use, the court implied that the game itself, independent of the compatibility code, was not a derivative work. *See id.* at 1520.

arbitrary module interfaces.[204] The module is particularly likely to be considered a derivative work if it can link only with one program.[205] Unfortunately for the defenders of FOSS, many if not most modules fall outside these categories. As a first approximation, the GPL's reliance on copyright law alone does not prevent programmers from making significant proprietary modifications to GPL software by adding proprietary libraries or modules. To some extent this is already happening: several GNU/Linux distributors derive substantial revenue from selling proprietary programs integrated fairly tightly with (and possibly dynamically linked to) the GPL-covered core operating system.[206]

If courts continue the trend of allowing unauthorized linking, proprietary software vendors will have less control of the market for modules that link to their programs. At the same time, the GPL's copyleft clause will apply to fewer works, reducing the incentive for programmers to use the GPL for their own works in return for permission to link. Conversely, in situations where courts adhere to *Midway*-style copyright analysis and forbid most unauthorized linking, the GPL becomes more effective, but proprietary software vendors will be able to further restrict the uses of their work.

Because the GPL relies on copyright law, it brings the courts' interpretations of the Copyright Act as a policy for promoting progress into conflict with the FOSS movement's own scheme for promoting software innovation. The goals are similar: both the Copyright Act and the FOSS movement seek to maximize innovation, promote educational and critical uses of software, and limit monopoly abuse. Yet strengthening one seems to weaken the other. The next section proposes a solution to this paradox and analyzes the solution's effectiveness.

V. AN ARGUMENT FOR STRONGER DERIVATIVE WORKS PROTECTION FOR GPL CODE

A. *An Alternate Balance for Copyright*

The Constitution's Intellectual Property Clause announces a goal (promoting the progress of science and the arts) and a means of achieving it (giving limited exclusive rights to authors and inventors).[207] The common

[204] *See Micro Star*, 154 F.3d at 1113 ("The fair use defense will be much less likely to succeed when it is applied to fiction or fantasy creations, as opposed to factual works . . . ."); Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 840 (Fed. Cir. 1992) (calling Nintendo's compatibility code, which contained "creative organization and sequencing," protectable expression).

[205] *See Micro Star*, 154 F.3d at 1112 n.5 (explaining that a module that works with only one program is very likely to "incorporate the protected expression" and thus be a derivative work).

WEBER, *supra* note 28, at 108, 195-196 (describing some businesses that package GNU/Linux systems together with proprietary software, using the GPL-covered portion as a "loss leader").

[207] U.S. CONST. art. 1, § 8, cl. 8.

understanding of this clause is that the right to control copying gives authors and inventors the financial incentive they need. Rights that are too strong, however, thwart the goal of progress by restricting the public's access to creative works and preventing others from building on them.[208] Courts seek a balance between exclusive rights and public access that will best promote progress. On many occasions, new technologies and new business models have required courts to modify their interpretations of copyright law principles in order to maintain this balance.[209]

FOSS development is a new business model that challenges the assumptions behind the Intellectual Property Clause. Proponents of FOSS maintain that creative people want to create and that their limiting factor is not financial incentives, but instead access to material on which to build.[210] The Framers' intuition that exclusive rights promote progress remains valid for FOSS, yet the connection between the means and the goal differs. With FOSS, the author's exclusive rights are not licensed in return for financial reward, but for a promise to keep the programmer's stock of raw material (source code) available for others to build on. The "right to distribute, [rather than] the right to exclude," promotes innovation.[211]

Because FOSS is a different method for using exclusive rights to promote progress, the equilibrium point between exclusive rights for programmers and access by others is also different. This is the paradox described in Part IV. Declaring more loosely linked software modules to be derivative works makes the GPL more effective, but also makes proprietary licenses more restrictive. This suggests that courts should apply a broader derivative works test to GPL software. Although using two different tests seems inconsistent, adjusting the derivative works test for FOSS would allow the courts to preserve the policies behind copyright law while accommodating FOSS's alternate mechanism for furthering the same policies. The linking cases discussed above present two doctrinal hooks for accomplishing this goal. The first, denying a fair use defense when proprietary software is linked with FOSS, would be a straightforward and useful approach in some cases. The second approach, actually broadening the definition of a derivative work when the original work is FOSS, would be harder to justify doctrinally but applicable to far more

[208] *See* Sony Corp. of Am. v. Universal City Studios, Inc. 464 U.S. 417, 429 (1984) (asserting the need to weigh the benefit of granting exclusive rights to individuals against the harm to the public that would result).

[209] *See id.* at 430 (observing that technological change is what drives the modification of copyright law).

[210] WEBER, *supra* note 28, at 84 ("The only times when innovation will be 'undersupplied' is when creative people are prevented from accessing the raw materials and tools that they need for work."). Courts have acknowledged that no creation of the mind is truly new; art, literature, and software are inevitably inspired by and grow out of earlier work by others. *See, e.g.*, Leeds Music Ltd. v. Robin, 358 F. Supp 650, 659 (arguing that "there are no truly new ideas under the sun" and overbroad copyright protection could cause "mankind's precious few core ideas [to] be removed from the marketplace of thought").

[211] *See* WEBER, *supra* note 28, at 1 (describing open source as relying on "the right to distribute, not the right to exclude").

cases, making a more effective solution to the paradox.

B. *Limiting Fair Use for Linking to GPL Code*

Sega and *Connectix* base their holdings on fair use, an affirmative defense to copyright infringement.[212] Fair use involves a "case-by-case, equitable" analysis,[213] taking into account any harm to the plaintiff's market, the "nature of the copyrighted work," and other practical factors.[214]

Because of its flexibility and focus on market factors, fair use has historically been the method by which courts adapt copyright law to new technologies, without the need for legislative changes.[215] For the same reason, fair use is the most sensible way to implement a special copyright policy for FOSS.

1. Fair Use and FOSS Have Parallel Goals

Fair use doctrine was originally created by judges in order to allow for uses that society historically viewed as more important than commercial exploitation.[216] The codification of the doctrine in § 107 of the Copyright Act lists these traditionally favored uses: "criticism, comment, news reporting, teaching . . . , scholarship, or research."[217] Because these uses are particularly important aspects of the freedom of speech, courts consider fair use to be the provision that harmonizes copyright law with the First Amendment.[218] The list of favored uses in the statute is not exclusive, and courts have recently allowed fair use defenses in commercial situations, as in the *Sega* and *Connectix* cases.[219] Even though they discounted the importance of commercial motivation in their fair use analysis, the courts in these cases justified their conclusions with a subtle reference to the original goals of fair use; the defendants' aim was to "gain an understanding" of the ideas expressed in

[212] 17 U.S.C. § 107 (2000); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1520 (9th Cir. 1992) (calling fair use the most "appropriate" way to resolve the case); Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602 (9th Cir. 2000) (invoking a fair use analysis).

[213] *Connectix*, 203 F.3d at 1520.

[214] 17 U.S.C. § 107 (listing the factors to be used in a fair use analysis).

[215] *See Sega*, 977 F.2d at 1520 ("[C]onsideration of the unique nature of computer object code thus is more appropriate as part of the case-by-case, equitable 'fair use' analysis . . . .").

[216] Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 569-70 (1994) (recognizing that statutory fair use "continues [a] common-law tradition," and that parody and criticism are historically fair uses even if they are "commercial" uses).

[217] 17 U.S.C. § 107.

[218] Eldred v. Ashcroft, 537 U.S. 186, 220 (2003) (calling fair use one of the "traditional First Amendment safeguards").

[219] *Sega*, 977 F.2d at 1522 (observing that the commercial nature of the use does not preclude a fair use defense); *see also* Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 605-07 (9th Cir. 2000) (finding that commercial purpose was only one factor to be weighed against others in its analysis of the "purpose and character of use" factor of the fair use defense).

plaintiffs' code.[220] Thus, the courts drew an analogy between writing a compatible module (i.e., a game cartridge) and studying code as an educational endeavor.[221]

Education is a strong component of FOSS development, and one of the FSF's stated goals is to allow people to "understand" the software they use by studying its source code.[222] The collaborative methods of open source software development grew out of the academic tradition of peer review.[223] Because the GPL requires free access to source code and grants general permission to modify and experiment, GPL software is a natural teaching tool. Linux, the best-known GPL software, began as a college student's hobby and was itself based on a kernel program written as a teaching tool for operating system design.[224] Safeguarding the availability of source code and the protections of the GPL maintains this source of educational material for students and also allows FOSS businesses such as Red Hat[225] and IBM to use students' work in their products.

For proprietary software, a broad fair use defense for linking modules promotes learning and "understanding" of the ideas embedded in software.[226] Because the GPL strongly promotes educational uses, and broad protection for derivative works enhances the GPL, the equitable fair use analysis pulls in the other direction for GPL software. For FOSS, the goal of promoting scholarship, as well as criticism, commentary, and teaching, justify *denying* a fair use defense to module writers who seek to add proprietary, "closed-

[220] *Sega*, 977 F.2d at 1522 ("[A]lthough Accolade's ultimate purpose was the release of Sega-compatible games for sale, its direct purpose in copying Sega's code, and thus its direct use of the copyrighted material, was simply to study the functional requirements for Genesis compatibility . . . ."); *see also Connectix*, 203 F.3d at 604 (allowing defendants to copy plaintiffs' code to aid in "understanding" and to develop a compatible program, even if less efficient development methods were available that could have reduced the amount of copying and use).

[221] *See Sega*, 977 F.2d at 1522-23 (stating that Accolade's use of Sega's code "to discover the functional requirements for compatibility with the Genesis console" was "legitimate" and "essentially non-exploitative").

[222] *See, e.g.*, Free Software Foundation, Frequently Asked Questions about the GNU GPL, http://www.gnu.org/licenses/gpl-faq#GPLInProprietarySystem (last modified Aug. 19, 2005) ("The goal of the GPL is to grant everyone the freedom to copy, redistribute, understand, and modify a program.").

[223] *See* WEBER, *supra* note 28, at 144-45 (explaining how many of the cultural norms of the programmers at MIT's Artificial Intelligence Lab – a research institution – were adopted by the FOSS movement).

[224] Andrew Tanenbaum, Some Notes on the "Who Wrote Linux" Kerfuffle, May 20, 2004, http://www.cs.vu.nl/~ast/brown (recalling that Linus Torvalds began writing the Linux kernel while a student, having been inspired by Professor Tanenbaum's model kernel program, called MINIX).

[225] Red Hat, Inc. is a major commercial distributor of the GNU/Linux operating system. *See* Red Hat: The Open Source Leader, http://www.redhat.com (last visited Dec. 5, 2005).

[226] *See Sega*, 977 F.2d at 1520 ("Where there is good reason for studying or examining the unprotected aspects of a copyrighted computer program, disassembly for purposes of such study or examination constitutes a fair use.").

source" extensions.

A related public benefit of FOSS (some might call it a subset of "criticism") is independent security analysis of critical software. Computer security researchers assert that widespread peer review is the most effective way to secure software against viruses, malicious hackers, and similar threats.[227] The ability to add proprietary modules to critical GPL-licensed software, such as the server and domain name lookup software that runs the Internet, hinders the public's ability to evaluate the security of important infrastructure software. If courts were to find that security is an important part of the "nature of the copyrighted work" under the fair use factors,[228] then security concerns may be another factor that weighs against a finding of fair use.

2. "Exploitation" and Custom: Specific Fair Use Factors

In software cases, courts deny fair use when an alleged derivative work "supplant[s]" the original, "usurp[s] the market," or is "exploitative."[229] Other statements the cases discussed above give more meaning to these terms. First, industry custom is relevant. The defendant in *Sega* "did not seek to avoid paying a customarily charged fee" to build its module, which could imply that the existence of such a custom would weigh against fair use.[230] This logic parallels the focus on acceptable industry practice in trade secret law, in which methods for discovering a competitor's trade secrets are allowed when they are customarily accepted as industry practice.[231] The second component of "exploitation" is the public interest in encouraging the creation of more software. A strong public interest can outweigh some degree of harm to the original copyright owner.[232] Finally, integrity concerns may still play a role. A use that harms the reputation of the original copyright owner impacts that

[227] *See* BRUCE SCHNEIER, SECRETS AND LIES: DIGITAL SECURITY IN A NETWORKED WORLD 344 (2004) (describing FOSS as the "best way to facilitate" widespread expert security review of software).

[228] 17 U.S.C. § 107(2) (2000) (listing "the nature of the copyrighted work" as a factor in fair use analysis).

[229] *See Sega*, 977 F.2d at 1523 (examining the fourth fair-use factor, the "effect on the potential market").

[230] *See id.* at 1522 (justifying a fair use defense by reference to the absence of a "customarily charged fee" paid by video game designers to console makers); William W. Fisher III, *Reconstructing the Fair Use Doctrine*, 101 HARV. L. REV. 1661, 1680-81 (1988) (acknowledging that "industry practice" may be a beneficial source of fair use standards so long as both parties belong to a particular industry with well-defined customs).

[231] Ronald L. Johnston & Allen R. Grogan, *Trade Secret Protection for Mass Distributed Software*, 11 COMPUTER LAW. 1, 11 (Nov. 1994) ("Whether circumstances will give rise to a duty to maintain secrecy or limit use of a trade secret may be supported by industry custom."). The use of industry custom in equitable legal analysis is also common in "many areas of property law" and may be "readily applicable to disputes over intellectual property." Fisher, *supra* note 230, at 1680.

[232] *See Sega*, 977 F.2d at 1523 (finding that public benefit from the availability of defendant's product is sufficient to overcome the presumption of unfairness associated with commercial exploitation).

owner's market, and this harm might weigh against fair use.[233]

As with educational uses, a policy of avoiding "exploitation" in the GPL context favors denying a fair use defense. An effective GPL promotes many of the same policies that motivated the courts' decisions on linking, including an adherence to industry customs, protection of reputation and the promotion of beneficial network effects.

a. *FOSS Industry Custom*

Unlike the video game market, GPL software developers have established a "customary fee" for the use of their interfaces. The fee is not direct compensation to the copyright owner, but a promise to release any modifications under the GPL only. This tradition of reciprocity comes not only from the terms of the GPL, but from customary practice among FOSS developers. At its heart, the custom is a straightforward rationale of fairness: FOSS developers give up their right to licensing royalties when they release software under the GPL. In return, anyone using the software as a basis for their own work by adding a module must license her module on the same terms. To the extent that "exploitation" involves violating the norms of a particular industry, the FOSS community's "share alike" ethic should be an important – although not decisive – factor in the fair use analysis.

The FOSS community also places a strong emphasis on programming as a creative, expressive activity.[234] Programmers on GPL projects strive not only for efficient solutions to problems, but also for a sort of artistry that displays the programmer's skill and distinctive style.[235] They seek an aesthetic dimension that goes beyond making the software perform its function correctly. A court could analogize this creative element to the creativity and expressiveness that played such an important role in the video game cases. Because the creativity in a work is closer to the "core of copyright protection" than the work's utilitarian aspects, the former receives stronger copyright protection and a broader definition of derivative works.[236] To the extent that FOSS 'programming as artistry' is independent of the software's utility and efficiency, the courts may grant a higher level of protection to FOSS by denying fair use.

b. *Effect on Reputation*

The "reputation" rationale of *Worlds of Wonder* and *Midway* may also justify denying fair use for linking to GPL code. In those cases, the courts used concern for damage to the plaintiffs' commercial reputations to justify

[233] *See* Worlds of Wonder, Inc. v. Veritel Learning Sys., Inc., 658 F. Supp. 351, 356-57 (N.D. Tex. 1986) (considering damage to the image of the plaintiff's product as part of the harm that copyright law protects against).

[234] *See* WEBER, *supra* note 28, at 145 (explaining how, for many FOSS programmers, a program can have intrinsic expressive value apart from its function).

[235] *Id.* ("Open source developers are certainly much like artists in the sense that they seek fun, challenge, and beauty in their work.").

[236] *See supra* notes 178-180 and accompanying text.

findings of infringement.[237] The GPL guarantees that source code will be perpetually available, and this guarantee is an important part of GPL software's commercial value.[238] Additionally, many people believe that programs developed through FOSS processes are more secure and reliable than proprietary software.[239] If anyone can link a GPL program with non-GPL modules, the presence of the GPL no longer serves to indicate the security and "openness" of the whole, and the guarantee of continuing permission to modify is lost. Just as permitting Veritel to produce technically inferior tapes for Teddy Ruxpin or allowing Artic to add unlicensed boards to Midway's video games could damage the reputation of the original manufacturers, allowing proprietary additions to GPL software could harm FOSS developers in a similar way.

Although copyright law is not concerned with protecting commercial images per se (that being the domain of trademark law), reputational harm may be a form of "usurping the market" for the original product. This is especially true for modules, such as Linux kernel modules, that become closely associated with the product they modify. As Richard Stallman points out, an operating system consists of many software components, which are often collectively referred to as 'Linux' in the GNU/Linux system.[240] If the collection of components that users know as 'Linux' comes to contain proprietary modules, and those modules add vital functionality, then this partly proprietary collection could replace the purely FOSS original in the market. This may be "exploitation."

c. *Beneficial Network Externalities*

One explanation for the courts' trend toward allowing fair use for linking is a recognition that network externalities increase copyright owners' control over their work and related works, to the detriment of later programmers.[241] The stronger the network externalities for a given program, the stronger the fair use argument for allowing linking with that program. In contrast, the GPL is an attempt to leverage network externalities to preserve later programmers' ability to interoperate with earlier work. For FOSS, the amplifying effect of

[237] *See supra* note 107.

[238] David Betz & Jon Edwards, *Richard Stallman Discusses His Public-Domain UNIX-compatible Software System with BYTE Editors*, BYTE, July 1986, http://www.gnu.org/gnu/byte-interview.html (commenting that a commercial advantage of the GPL-covered EMACS text editor program is that, because the source code is available, users can hire anyone they want to service the software, even if the original supplier goes out of business).

[239] Bruce Schneier, *Open Source and Security*, CRYPTO-GRAM NEWSLETTER, Sept. 15, 1999, http://www.schneier.com/crypto-gram-9909.html (explaining the widely held view that software can be made more secure if its source code is available to the public, because widespread testing and evaluation can occur); *see* also RAYMOND, *supra* note 30 (describing how FOSS processes allow problems in code to be fixed quickly).

[240] Stallman, *supra* note 50 (acknowledging that the GNU/Linux system as a whole is often called simply Linux).

[241] *See supra* notes 149-156 and accompanying text.

network externalities promotes rather than inhibits the public interest identified in *Sega*.[242] Denying fair use for otherwise infringing linking with GPL code would extend this beneficial network effect to more software.

In summary, several factors that led courts to grant a fair use defense in software cases actually support *denying* the defense in cases where the GPL applies. This approach resolves the GPL's paradox in cases that turn on fair use – in other words, cases such as *Micro Star* where linking creates a *prima facie* infringing derivative and the defendant raises fair use as a defense.[243] Of course, *Galoob* and other precedents remove many types of linking from this category. Where there is no *prima facie* infringement, fair use is irrelevant. For these cases, solving the paradox requires a more difficult doctrinal shift.

C. *Broadening the Definition of a Derivative Work for GPL Code*

If the hypothesis presented in this Note proves true – that is, if the courts' permissive stance toward unauthorized linking severely weakens the GPL – then preserving the license's potential will mean convincing a court to apply a broader definition of derivative works to GPL programs than is applied to other programs. This approach would close the proprietary module loophole in the GPL for cases where the fair use defense is never raised, cases where an unauthorized module is *prima facie* not a derivative work. Asking for a different derivative works test for FOSS would be much more difficult than asking for a denial of fair use, because the "equitable" doctrine of fair use leaves more discretion to judges than does the basic definition of a derivative work.[244] Arguing that the license applied to a particular piece of software should affect which other programs can be called derivative works of that software seems like a stretch. However, courts may be willing to apply the same policy arguments discussed above for denying fair use to the question of defining a derivative work.

D. *Potential Difficulties with this Approach*

While the approach laid out in this section would resolve the paradox caused by the GPL's unusual use of copyright law, it may encounter some obstacles in practice. From a practical perspective, many companies have built their businesses around a mixture of linked GPL and proprietary software.[245] The revenue these companies derive from selling the proprietary components may be an important source of funds for future FOSS development. FOSS-based businesses make money from ancillary products and services, such as software support, customization, documentation, and hardware tie-ins, but linked proprietary software remains an important part of the business model for many

[242] *See supra* Part III.A.2 (discussing how courts have considered network effects).

[243] *See* Micro Star v. FormGen Inc., 154 F.3d 1107, 1112 (9th Cir. 1998).

[244] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1520 (9th Cir. 1992) (describing fair use as a "case-by-case" inquiry).

[245] WEBER, *supra* note 28, at 108 (describing some businesses that packaged GNU/Linux systems together with proprietary software).

firms.[246] To the extent that the GPL's terms are grounded in ideology rather than an alternate economic paradigm, courts may allow more linking of GPL and proprietary work in deference to market realities.

Network externalities also cause a different problem under the GPL. When a GPL-based business such as Red Hat has invested heavily in marketing and distribution, it can appropriate and reap a large share of the rewards from any GPL-covered additions to its software, no matter who writes those additions.[247] The more a strong GPL compels smaller competitors to release their own software under the GPL, the more a larger outfit with a head start in marketing and distribution can appropriate the value of that work.[248] For FOSS, as for proprietary software, there is still a balance to be struck.

CONCLUSION

The GPL has been called "a hack on the copyright system" because it uses the exclusive rights of authors to guarantee rather than restrict public access to the inner workings of software.[249] Unfortunately, because it uses copyright law for such a radically different purpose, any change in the law that strengthens the GPL also promotes abuses in the proprietary system, and any change that directly addresses those abuses tends to weaken the GPL. Specifically, the current law on whether copyright can prevent unauthorized add-on modules in software represents the courts' solution to a problem of abusing copyright's statutory monopoly, yet it creates a serious obstacle to the FSF's efforts to combat the same abuse by different means. The solution proposed by this Note is that courts could broaden the definition of a derivative work in GPL-related cases in light of the GPL's purpose. This approach could help the courts remain true to the goals of the Copyright Act while allowing the FOSS community – an experiment in achieving the same goals – to thrive.

[246] *See* Shankland, *supra* note 6.

[247] WEBER, *supra* note 28, at 222 (describing how the GPL can give an advantage to established FOSS businesses that are better able to market and monetize new GPL-covered code regardless of who developed the program).

[248] *See id.*

[249] Li-Cheng Tai, The History of the GPL, July 4, 2001, http://www.free-soft.org/gpl_history (proclaiming that the GPL enables the development of "software by the people, of the people and for the people").