Chicago-Kent Journal of Intellectual Property

Volume 6 | Issue 2 Article 6

4-1-2007

A Gentlement's Agreement: Assessing the GNU General Public License and its Adaptation to Linux

Douglas A. Hass

A GENTLEMEN'S AGREEMENT
ASSESSING THE GNU GENERAL PUBLIC LICENSE AND ITS ADAPTATION TO LINUX

Douglas A. Hass[*]

Introduction

"Starting this Thanksgiving, I am going to write a complete Unix-compatible software system called GNU (for GNU's Not Unix), and give it away free to everyone who can use it."[1] With his post to the Usenet[2] newsgroup net.unix-wizards,[3] Richard Stallman launched a sea change in software development. In 1983, he could not have known that his lasting contribution would not be the GNU operating system, but instead the controversial software license that he would develop as its underpinning: the GNU General Public License (GPL).[4]

Today, the operating system most closely associated with the GPL is Linux, developed originally by Linus Torvalds, a Finnish university student.[5] Research group IDC's Quarterly Server Tracker marked Linux server revenue growth at three times Microsoft Windows server growth in the first quarter of 2006, its fifteenth consecutive quarter of double-digit revenue growth.[6] British research firm Netcraft's July 2006 Web Server Survey gives Linux-based Apache Web servers the largest market share among Web servers queried in its monthly survey.[7] With Linux gaining an increasingly larger position in these markets, the validity of the GPL takes on increasing importance as well.

The open source community's commercial and non-commercial members are conducting a robust debate on the intellectual property issues surrounding the GPL and Linux, its most

[*] Douglas A. Hass, Director of Business Development, ImageStream; J.D. Candidate, 2008, Indiana University School of Law—Bloomington. Special thanks to the University of Auckland Faculty of Law, Professor Louise Longdin, and Dean Rosemary Tobin for providing the author with a unique opportunity and the necessary accommodations to write this paper. Thank you also to Professors Fred Cate and Marshall Leaffer at Indiana and Lawrence Rosen at Stanford, and to those who provided insights and made suggestions. All errors and oversights are solely attributable to the author.
[1] Posting of Richard M. Stallman to net.unix-wizards, New UNIX Implementation (Sept. 27, 1983, 16:35:59 GMT), *available at* http://groups.google.com/group/net.unix-wizards/msg/4dadd63a976019d7?dmode=source.
[2] Usenet is one of the first distributed discussion systems on the Internet. An electronic mail-like system, Usenet enabled one-to-many communications. Conceived in 1979, Usenet predates the commercial Internet, and while message boards on the Web have largely supplanted it, many Usenet newsgroups remain active today. *See generally Usenet*, WIKIPEDIA: THE FREE ENCYCLOPEDIA, http://en.wikipedia.org/wiki/Usenet (last visited Dec. 31, 2006).
[3] Stallman, *supra* note 1.
[4] Since the Linux operating system, discussed at Section I.B., *infra*, uses version 2 of the GPL, references herein to the GNU General Public License (GPL) refer to version 2 of that license unless otherwise stated. *See generally* Free Software Foundation, GNU General Public License, version 2, http://www.fsf.org/licensing/licenses/gpl.txt (last visited Jan. 9, 2007) [hereinafter GPL].
[5] *See generally* Linux, WIKIPEDIA: THE FREE ENCYCLOPEDIA, http://en.wikipedia.org/wiki/Linux (last visited Dec. 31, 2006).
[6] IDC, Worldwide Server Market Experiences Lackluster Quarter Across Multiple Segments, According to IDC, (May 24, 2006), *available at* http://www.idc.com/getdoc.jsp?containerId=prUS20180706.
[7] Netcraft, July 2006 Web Server Survey, Jun. 28, 2006, http://news.netcraft.com/archives/2006/06/28/july_2006_web_server_survey.html (last visited Jan 9, 2007).

successful project to date. This paper argues that this process has led the Linux community to adjust its open source development model to accommodate realities of copyright law and the need to secure both significant commercial participation and widespread industry adoption. Erroneous ruminations about the GPL's legal effect threaten to undermine this important adjustment consensus. This paper outlines the GPL's legal shortcomings and boundaries to help bolster the community's practical, and undervalued, "gentlemen's agreement" that enables commercial participation in open source projects.

Part I of the paper examines the history of the free software movement founded by Stallman, including the development of the Linux operating system. Part I also highlights the stated objectives and terms commonly used by free software proponents. Next, the paper explains the GPL and discusses implications of recent litigation and possible attacks on the license's validity in Part II. Part III applies the United States' Copyright Act and relevant U.S. and international case law to the GPL-licensed Linux operating system to justify the "gentlemen's agreement" that the Linux community has reached independent of the GPL's drafters. Part IV discusses the reconciliation of commercial software interests with the free software objectives for which the GPL acts as a symbol.

I. History of the Free Software Movement and Linux Operating System

A. Richard Stallman, GNU, and the Free Software Foundation

Opponents of the free software movement founded by Richard Stallman can blame Xerox Corporation for starting Stallman on his path.[8] Working in MIT's Artificial Intelligence lab as a 27-year-old computer programmer in 1980, Stallman interrupted a programming session to retrieve a print job from the state-of-the-art network printer donated by Xerox.[9] In an experience still commonplace in computer labs today, Stallman arrived to find the printer jammed with his job still in the queue.[10] He immediately set out to implement a solution that he had used with the lab's old printer.[11] While he could not remedy paper jams with a computer program, Stallman wrote a program that enabled the printer to send a message to all of the user terminals with waiting print jobs notifying them that the printer had jammed.[12]

Unlike the previous printer, and against common practice at the time, the Xerox printer came without source code for the software used to operate the printer. Xerox provided only the

---

[8] This section follows Stallman and the development of the GNU project through the development of the GPL and Linux. The free software movement extends beyond these projects. A separate but related branch of the movement arose from the University of California-Berkeley and work on the UNIX variant called the Berkeley Software Distribution, or BSD. Development of BSD variants FreeBSD, NetBSD and OpenBSD continue today under a different free software license called the Modified BSD License. For more information on BSD, the BSD license, and the progression of development, see Marshall Kirk McCusick, *Twenty Years of Berkeley Unix: From AT&T to Freely Redistributable, in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION 31, 31-46 (Chris DiBona et al. eds., 1999).
[9] SAM WILLIAMS, FREE AS IN FREEDOM: RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE, Chapter 1 (2002), *available at* http://www.faifzilla.org/ch01.html.
[10] *Id.*
[11] *Id.*
[12] *Id.*

object code.[13] After weeks of fruitless searches for the source code, Stallman reportedly approached a former Xerox employee at Carnegie Mellon University who was one of the primary developers on the Xerox printer. Citing non-disclosure agreements, he rebuffed Stallman's request for source code.[14] According to biographer Sam Williams, Stallman viewed this as a rejection of "a system that, until then, had encouraged software programmers to regard programs as communal resources."[15] Stallman's experience with the Xerox printer, and subsequent encounters with non-disclosure agreements (NDAs) in the AI lab, galvanized and clarified the guiding concept of the GNU project he founded in 1983 that "software should be shared."[16]

As more and more programmers in the MIT AI lab left to join companies that required NDAs, Stallman recognized that the collegial programming culture to which he had become accustomed had changed.[17] Rather than follow the trend, he decided to create a new computing environment where source code would always remain free.[18] With the wide adoption of UNIX operating systems at the time, and given his extensive experience with UNIX and its variants, Stallman decided to create a UNIX compatible operating system called GNU, a recursive acronym for GNU's Not Unix.[19]

Concerned that MIT would attempt to claim ownership of any work he completed on GNU while the AI lab still employed him, Stallman quit his job in January 1984 to ensure that he could retain complete control of the source code for GNU.[20] As the body of GNU software - developed primarily by Stallman - grew, Stallman recognized a need for further organization and funding. In 1985, Stallman created the Free Software Foundation (FSF) as "a tax-exempt charity

---

[13] *Id.* Courts have defined object code as "the literal text of a computer program written in a binary language through which the computer directly receives its instructions." Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 835 (10th Cir. 1993) (citing Autoskill, Inc. v. Nat'l Educ. Support Sys., 994 F.2d 1476, 1492 n.18 (10th Cir. 1993)). Source code, on the other hand, is "the literal text of a program's instructions written in a particular programming language." *Id.* (citing Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 663 (4th Cir. 1993)). This paper and many of the sources cited herein also use the companion terms "closed source" and "open source." These common terms refer to the *availability* of source code. Closed source projects typically make only the object code freely available and limit or prevent distribution of source code. Open source projects typically make source code freely available under the GPL or another similar license.

[14] WILLIAMS, *supra* note 9.

[15] *Id.*

[16] Posting of Richard M. Stallman to gnu.cc, More Confusion on GNU Copying Conditions (Nov. 13, 1988, 14:22:34 GMT), *available at* http://groups.google.com/group/gnu.gcc/browse_thread/thread/3cd3f5a5c6f9457d/.

[17] WILLIAMS, *supra* note 9.

[18] Stallman, *supra* note 1

> I consider that the golden rule requires that if I like a program I must share it with other people who like it. I cannot in good conscience sign a nondisclosure agreement or a software license agreement.
>
> So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.

[19] *Id.*

[20] Richard M. Stallman, *The GNU Operating System and the Free Software Movement*, *in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 8, at 53, *available at* http://www.oreilly.com/catalog/opensources/book/stallman.html.

for free software development."[21] The FSF took over most of the business operations, selling tapes with GNU and other software, and soliciting donations. Stallman continued programming for the GNU project and evangelizing his view of free software.[22]

During the same period, Stallman continued to experiment with different software licenses. His first experience came with his Emacs software development during the late 1970s.[23] After developing the first Emacs implementation in 1975, Stallman's implementation attracted interest from outside MIT.[24] To ensure that the project remained open to the benefit of all programmers, Stallman told interested parties "that it belongs to the Emacs 'Commune,' that in order to use Emacs you had to be a member of the Commune and that meant that you had the responsibility to contribute all the improvements that you made."[25] This simple "license" met only limited success. By 1980, several programmers had released Emacs-based distributions, including Brian Reid's Scribe project.[26] Upon selling his project to commercial software manufacturer Unilogic, Reid inserted source code that disabled his program after a 90-day trial period to enable Unilogic to sell copies of the editor.[27] Stallman believed that "one after another, people would defect and stop cooperating with the rest of society, until only those of us with very strong consciences would still cooperate," and he tightened his control over Emacs.[28]

In 1984, Stallman continued to adapt GNU Emacs, borrowing heavily for new features from an existing project developed by a Carnegie Mellon researcher and subsequently sold to privately held software maker Unipress.[29] When Unipress learned of Stallman's appropriation of their copyrighted source code, the company threatened to take action.[30] Faced with completely rewriting this portion of GNU Emacs, Stallman recognized that direct control over the distribution of Emacs and other free software was limited solely to that code over which he had personal control.[31]

The incidents with Unilogic and Unipress forced Stallman to rethink the informal "commune" approach to software licensing. One of Stallman's main goals was to close the

[21] *Id.*
[22] *Id.*
[23] Emacs stands for Editor MACroS, originally a set of instructions for the TECO editor. Programmers have written numerous Emacs implementations over the years to control editing of text files (including source code), to compile programs, and even to browse the Web. Two main Emacs implementations exist today, including the GNU Emacs project that Stallman started in 1984 and still maintains. *See generally* Emacs, WIKIPEDIA: THE FREE ENCYCLOPEDIA, http://en.wikipedia.org/wiki/Emacs (last visited Dec. 31, 2006).
[24] Richard M. Stallman, *The GNU Operating System and the Free Software Movement*, *supra* note 8, at 53, *available at* http://www.oreilly.com/catalog/opensources/book/stallman.html.
[25] Richard M. Stallman & Bjørn Remseth, Partial Transcript of Lecture at KTH (Kungliga Tekniska Högskolan (Royal Institute of Technology) in Stockholm Sweden (Oct. 30, 1986), http://groups.google.com/group/soc.culture.argentina/msg/38e213511a3360ee (last visited December 31, 2006).
[26] Richard M. Stallman, *The GNU Operating System and the Free Software Movement*, *supra* note 8, at 53, *available at* http://www.oreilly.com/catalog/opensources/book/stallman.html.
[27] WILLIAMS, *supra* note 9, Chapter 6, *available at* http://www.faifzilla.org/ch06.html.
[28] Stallman & Remseth, *supra* note 25, at 4.
[29] WILLIAMS, *supra* note 9, Chapter 7, *available at* http://www.faifzilla.org/ch09.html.
[30] *Id.*
[31] *Id.*

loophole that led to the Unilogic and UniPress disputes.[32] With the release of the GNU Emacs project in 1985, Stallman also released two important documents: "The GNU Manifesto"[33] and the "GNU Emacs General Public License."[34] The GNU Manifesto expanded on Stallman's 1983 announcement of GNU and highlighted his reasoning for rejecting proprietary software.[35] It reiterated Stallman's "golden rule," that he must share programs with others who like those programs.[36] Rebutting what he saw as common attacks on the GNU position, Stallman outlined his position on copyright, asserting that societies that granted copyright licenses were worse for doing so.[37] Conversely, Stallman wrote, software programs did not lend themselves to copyright protection in the way that books had in the 18th and 19th centuries. He also argued, "a person who enforces a copyright is harming society as a whole both materially and spiritually; in which a person should not do so regardless of whether the law enables him to [sic]."[38]

Stallman reiterated his anathematic approach to copyright protection in the GNU Emacs General Public License. Consistent with the later GPL, discussed in Part II, the GNU Emacs General Public License opened with a preamble outlining two of Stallman's three main purposes for releasing code in the GNU project: (1) to keep software free, and (2) to ensure that licensees know that the Emacs software carried no warranty.[39] The license provisions first introduced Stallman's "copyleft" concept: subject to certain restrictions, licensees could modify the software or any part of it, but any derivative of GNU Emacs had to be licensed under the same terms.[40]

Stallman clarified the original 1985 GNU Emacs General Public License in 1988[41] and continued to write source code and release software for the GNU Project.[42] Each new piece of software required a new "General Public License." By 1989, after he discussed his plan with others at FSF and those within the programmer community,[43] Stallman created version 1.0 of the GNU General Public License (GPL).[44] The blanket GPL replaced the individual licenses Stallman had previously created, including the Emacs license, but maintained language that was virtually identical to those earlier licenses.[45] The GNU Project successfully developed three major software tools and several smaller ones by the 1989 release of the license.[46] Despite the simplified licensing and growing acceptance of GNU tools, Stallman's individual control of the

[32] *Id.*
[33] Richard M. Stallman, The GNU Manifesto, http://www.gnu.org/gnu/manifesto.html (last visited Dec. 31, 2006).
[34] Richard M. Stallman, The GNU Emacs General Public License, http://www.cogsci.indiana.edu/pub/COPYING (last visited Jan. 13, 2007).
[35] Stallman, *supra* note 33.
[36] *Id.* (Stallman continues by explaining, "Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way.")
[37] *Id.*
[38] *Id.*
[39] Stallman, *supra* note 34. The GNU Emacs General Public License, unlike the GPL, made no mention of patents or patent licensing. *Id.*
[40] *Id.* at §§ 1, 2(b). The Emacs license did not describe the nature of the "warranty" that it disclaimed. *Id.*
[41] *Id.*
[42] WILLIAMS, *supra* note 9, Chapter 9, *available at* http://www.faifzilla.org/ch09.html.
[43] *Id.*
[44] Richard M. Stallman, GNU General Public License Version 1 http://www.gnu.org/copyleft/copying-1.0.html (last visited Jan. 11, 2007).
[45] *Compare id. with* Stallman, *supra* note 34.
[46] WILLIAMS, *supra* note 9, Chapter 9, *available at* http://www.faifzilla.org/ch09.html.

Project slowed development considerably. Six years after his original Usenet announcement in net.unix-wizards, the GNU Project still had no operating system kernel, a glaring gap.[47]

B. Linus Torvalds and the Emergence of Linux

On a trip to Polytechnic University in Helsinki, Finland in late 1990, Stallman spoke to an audience including 21-year-old Linus Torvalds, a computer science student and programmer. In his 2001 biography, Torvalds recalled appreciating Stallman's free software approach, because it put the improvement of code ahead of monetary or legal concerns.[48] Torvalds had already developed UNIX code on Polytechnic's systems.[49] While trying to avoid a long walk across campus in the winter to use the computer lab, he looked to a UNIX variant, Minix, to provide connectivity from his apartment to the lab.[50]

Throughout 1991, Torvalds rewrote Minix and made his own Usenet posting announcing he was "doing a (free) operating system (just a hobby, won't be big and professional like gnu for 386 (486) AT clones)."[51] Torvalds originally released the source code under his own copyright terms, and did not use the GPL:

> - Full source must be available (and free), if not with the distribution then at least on asking for it.
> - Copyright notices must be intact. (In fact, if you distribute only parts of it you may have to add copyrights, as there aren't (C)'s in all files.) Small partial excerpts may be copied without bothering with copyrights.
> - You may not distibute [sic] this for a fee, not even "handling" costs.[52]

While rudimentary, Torvalds' copyright notice embodied the basic concept of the GPL: freely available source code at no charge. With the release of version 0.12 in December 1991, Torvalds integrated the GNU Project's C code compiler (gcc) and discarded his original license in favor of the GPL.[53]

Between the release of Linux version 0.01 and 0.12, Stallman and the FSF had revised the GPL.[54] Version 2 of the GPL made what Stallman termed "some fairly small changes" [55] with

---

[47] For readers unfamiliar with Linux or UNIX operating system environments, the Linux Information Project has the best plain English definition of a "kernel." Linux Information Project, Kernel Definition, http://www.bellevuelinux.org/kernel.html (last visited Jan. 5, 2007).
[48] LINUS TORVALDS & DAVID DIAMOND. JUST FOR FUN: THE STORY OF AN ACCIDENTAL REVOLUTIONARY 58-59 (2001).
[49] *Id.*
[50] *Id.*
[51] Posting of Linus Torvalds to comp.os.minix, What would you like to see most in minix? (Aug. 25, 1991, 20:57:08 GMT), *available at* http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b .
[52] Linus Torvalds, Notes for linux release 0.01, http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.01 (last visited Jan 1, 2007).
[53] Linus Torvalds, Release Notes for Linux v0.12, http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12 (last visited Jan 1, 2007) (adopting the GPL and discussing features that use gcc 2.0).
[54] GPL, *supra* note 4.

the exception of a new Section 7.[56] Affectionately nicknamed the "Liberty or Death" clause by Stallman, Section 7 represented the most significant change to the GPL and its predecessors.[57] For the first time, Stallman's license recognized software patents. The "Poison Pill" clause, described in detail in Part II, effectively eliminates the license in the event of court-imposed limitations due to patent infringement. Torvalds' Linux 0.12 and subsequent versions adopted this version 2 of the GPL.[58]

C. Linux and the GPL Today

Linux development has progressed significantly from its early days in Torvalds' dormitory room. The main kernel development has moved into its fifth generation.[59] Ported into 29 different languages from Catalan to Arabic,[60] programmers have created Linux distributions that operate on more than a dozen different hardware platforms, ranging from standard PCs and UNIX mainframes,[61] to real-time embedded systems often used in manufacturing and automation,[62] to more esoteric platforms such as the SEGA Dreamcast[63] and Sony PlayStation.[64] In 2002, NASA's Advanced Information System Technology group developed a version of the Linux operating system called FlightLinux to "address the unique problems of spacecraft onboard computers."[65]

The reach of Stallman's GPL extends beyond the Linux kernel. At SourceForge, one of the two most prominent on-line repositories of free software,[66] authors have licensed nearly 65% of over 81,000 licensed projects under the GPL.[67] The other main free software repository,

[55] Richard M. Stallman, Transcript of Richard Stallman at the 2nd international GPLv3 conference; 21st April 2006, http://fsfeurope.org/projects/gplv3/fisl-rms-transcript.en.html#liberty-or-death (last visited an Jan. 19, 2007).
[56] GPL, *supra* note 4, at § 7.
[57] Richard M. Stallman, Transcript of Richard Stallman at the 2nd international GPLv3 conference; 21st April 2006, http://fsfeurope.org/projects/gplv3/fisl-rms-transcript.en.html#liberty-or-death (last visited Jan 19, 2007).
[58] Torvalds, *supra* note 53.
[59] *See* The Linux Kernel Archives, http://www.kernel.org/pub/linux/kernel/ (last visited Jan. 4, 2007). The 1.3, 2.0, 2.2, 2.4 and 2.6 release trees represented stable platform releases. The current stable kernel revision as of this writing is 2.6.20.
[60] Linux Online – Distributions and FTP Sites, http://www.linux.org/dist/ (last visited Jan. 4, 2007).
[61] Linux Online – List of Distributions, http://www.linux.org/dist/list.html (last visited Jan 4, 2007) (selecting no criteria to generate the entire list of distributions). The list includes major commercial distributions such as Red Hat Linux, Debian and SuSE.
[62] LynuxWorks, Inc., Real-Time Operating System and Embedded Linux OS from Lynuxworks, http://www.lynuxworks.com/ (last visited Jan 4, 2007); MontaVista Software Inc., MontaVista Software – Platform to Innovate, http://www.mvista.com/ (last visited Jan. 4, 2007).
[63] Linux DC: Linux for the Sega Dreamcast, http://linuxdc.sourceforge.net/ (last visited Jan. 4, 2007).
[64] Linux for PlayStation 2 Community, http://playstation2-linux.com/ (last visited Jan. 4, 2007).
[65] NASA, FlightLinux – Open Source Linux Operating System for onboard spacecraft use, http://flightlinux.gsfc.nasa.gov/ (last modified May 10, 2005).
[66] Unlike the use of the term "free" at broader software repositories such as CNET's download.com, "free" refers not to cost, but to "freedom," as in the freedom to freely copy, distribute, and modify software. *See* Stallman, *supra* note 20.
[67] *Compare* Open Source Technology Group (OSTG), SourceForge.net: Software Map, http://sourceforge.net/softwaremap/trove_list.php?form_cat=14 (last visited Jan. 4, 2007) (listing all licensed projects at SourceForge) *with* OSTG, SourceForge.net: Software Map,

Freshmeat, lists nearly 67% of all projects under the GPL.[68] Stallman's free software philosophy reflected in the GPL's self-perpetuating mechanisms, along with the rapid growth of Internet connectivity, inexpensive bandwidth, and the availability of inexpensive, quality GPL-licensed software, helped ensure the spread of Linux and other GPL-licensed projects. The GPL's requirement that programmers, users, and companies distribute all modifications to GPL-licensed software under the GPL ensured the widespread adoption of those programs and the underlying GPL.

The good news about Linux and the GPL, however, masks a larger debate about the applicability and validity of the GPL itself. To date, no U.S. court has considered a case directly interpreting or enforcing the GPL.[69] The Linux community has largely policed any blatant violators, and the few major corporate holdouts regarding the release of GPL-licensed source code have folded under pressure from the FSF and the community at large. For example, Linksys, now a division of networking giant Cisco Systems, released Linux code used to operate its "WRT" model wireless routers in 2003.[70] The battle ultimately drove Linksys out of the Linux community, however. Although the company still offers Linux source code in its GPL Code Center,[71] and a Linux version of its WRT product line, the company shifted its flagship wireless router products to commercially licensed VxWorks in 2005.[72] While Linksys blamed its switch on a "larger memory footprint," Linksys officials noted, "a lot of companies in the networking space have already switched" from Linux to VxWorks.[73]

Some companies never embraced the GPL, preferring an interpretation that allowed proprietary code to interoperate with Linux, a much narrower interpretation of its terms than Stallman and the FSF advocate.[74] Video and networking card manufacturer NVIDIA releases only the object code for their products' Linux drivers under a proprietary license.[75] NVIDIA claims that its proprietary license and closed source software release protect its intellectual property and avoid potential patent infringement lawsuits, valid or not.[76] NVIDIA's creation of driver modules for the GPL-licensed Linux kernel sparks a heated debate in the broader community,[77] even though there is little agreement on exactly how the GPL applies, if at all, to

---

http://sourceforge.net/softwaremap/trove_list.php?form_cat=15 (last visited Jan. 4, 2007) (listing GNU GPL projects only).

[68] OSTG, freshmeat.net: Statistics and Top 20, http://freshmeat.net/stats/ (last visited Jan. 4, 2007). The next most-used license is the FSF's GNU Lesser GNU Public License (LGPL) at 6.33%.

[69] *See infra* Part II. B.

[70] Posting of Rob Flickenger, Is Linksys Shirking the GPL? (Maybe Not.), to O'Reilly Emerging Telephony, http://www.oreillynet.com/etel/blog/2003/07/is_linksys_shirking_the_gpl_ma.html (Jul. 10, 2003, 04:39 GMT).

[71] Cisco Systems, Inc., Linksys.com – GPL Code Center, http://www.linksys.com (search for "GPL Code Center" in the text box provided; then follow the GPL Code Center hyperlink on the results page) (last visited Jan. 5, 2007).

[72] LinuxDevices.com, Linksys courts Linux hackers with WRT54G"L," (Dec. 1, 2005), http://www.linuxdevices.com/news/NS4729641740.html.

[73] *Id.*

[74] *See supra* text accompanying notes 39-43.

[75] NVIDIA Corp., License for Customer Use of NVIDIA Software, http://www.nvidia.com/object/nv_swlicense.html (last visited Jan. 5, 2006).

[76] Stephen Shankland, *Nvidia updates closed-source drivers for open-source OSes*, CNET NEWS.COM, Jun. 25, 2005, http://news.com.com/2061-10795_3-5762319.html.

[77] *E.g.*, Joe Barr, *GPL concerns halt Kororaa live CD*, NEWSFORGE, May 15, 2006, http://trends.newsforge.com/article.pl?sid=06/05/15/1451229&from=rss. The terms "kernel" and "module" will

NVIDIA in this instance.[78] In a 2003 post to the Linux Kernel Mailing List, Linus Torvalds entered the debate about the GPL's scope when applied to modules in general. While Stallman and the FSF insist that all code that interacts with GPL code must be GPL-licensed,[79] Torvalds offered his view that not all modules fall within the definition of derivative works under the GPL.[80]

With the widespread adoption of Linux and its continued growth in mainstream computing,[81] the GPL occupies a central and precarious position as the enabling mechanism for continued growth of Linux and GPL-licensed projects. Without the GPL, the growth engine could suddenly stop, unleashing a morass of litigation as individual authors, companies, and end users rush to stake out territory in a new licensing landscape. Part II examines the GPL itself and discusses relevant statutory and case law in interpreting the license. Parts III and IV consider the application of the GPL to derivative works and discuss the policy implications, and a course of action, for the Linux and broader open source communities.

II. The GNU General Public License

A. Is the GPL a License or a Contract?

Richard Stallman divided the GPL into four main sections: a preamble, a set of numbered sections outlining various terms and conditions, a warranty disclaimer in sections 11 and 12, and a short section following the license text describing how to apply the GPL to software programs.[82] The terms and conditions section begins with definitions in section 0. The central concept of the GPL and other open source licenses is the allocation of intellectual property rights in software. Sections 1 through 4 describe the primary rights that the GPL purports to grant its licensees, and its other sections outline a range of other rights and obligations. The potential reach of these terms depends largely on whether the GPL acts as a license or contract. Therefore, this analysis begins with a discussion of judicial treatments of software licenses before turning to the specific terms and conditions outlined in the GPL.

appear frequently from this point forward. Readers unfamiliar with Linux or UNIX operating systems should see the Linux Information Project definition of a "kernel" *supra* note 47. A "module" in this context is software code that can perform one or many different tasks within the Linux operating system. Most modules do not function as separate programs, but are connected ("linked") to the main kernel either dynamically when their routines are needed by the operating system or statically when a systems administrator builds the Linux kernel from source code.

[78] For a sampling of the recurring debate surrounding the combination of closed source object code and GPL code, including NVIDIA, see, *e.g.*, Linus Puts Kibosh on Banning Binary Kernel Modules, http://linux.slashdot.org/article.pl?sid=06/12/14/1328249&from=rss (last visited Jan. 5, 2007).

[79] Free Software Foundation, Frequently Asked Questions about the GNU GPL – GNU Project - Free Software Foundation (FSF), http://www.gnu.org/licenses/gpl-faq.html (last visited Jan. 5, 2007).

[80] Posting of Linus Torvalds to the Linux Kernel Mailing List, Re: Linux GPL and binary module exception clause? (Dec. 4, 2003, 00:00:21 GMT), *available at* http://lkml.org/lkml/2003/12/3/228.

[81] *See supra* note 7 and accompanying text.

[82] GPL, *supra* note 4.

In the face of significant authority, the FSF and the GPL itself claim that the GPL acts as a license, and not a contract.[83] The license acceptance procedures in the GPL appear in Section 5:

> You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.[84]

Professor Rosen suggests "[T]his GPL reliance entirely on copyright law for license enforcement is legally sound."[85] Professor Eben Moglen, the FSF's General Counsel, essentially agrees, arguing

> The GPL, however, is a true copyright license: a unilateral permission, in which no obligations are reciprocally required by the licensor. Copyright holders of computer programs are given, by the [U.S.] Copyright Act, exclusive right to copy, modify and redistribute their programs. The GPL, reduced to its essence, says: "You may copy, modify and redistribute this software, whether modified or unmodified, freely. But if you redistribute it, in modified or unmodified form, your permission extends only to distribution under the terms of this license. If you violate the terms of this license, all permission is withdrawn."[86]

Professors Rosen and Moglen's analyses stand alone, since laws, treaties, and public policy in the U.S. and abroad contradict their opinions. Because the GPL's construction as either a bare license or a contract affects licensee and licensor rights and remedies, as well as the interpretation of the GPL itself, the legal status and worldwide effect of the GPL is an important question.

i. Contractual Interpretations of Software Licenses and the GPL Internationally

When licensors and licensees reside in different countries, the United Nations Convention on Contracts for the International Sale of Goods (CISG)[87] may apply. The United States and 66 other nations, with the notable exception of the United Kingdom, have ratified the treaty and

---

[83] The analysis of the GPL that follows in this section relies heavily on U.S. copyright law because the GPL authors drafted – and continue to interpret the license – from a U.S.-centric point of view.
[84] GPL, *supra* note 4, at § 5.
[85] LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW 138 (2004).
[86] Posting of Pamela Jones, to Groklaw, The GPL is a License, Not a Contract, Which is Why the Sky Isn't Falling, http://www.groklaw.net/article.php?story=20031214210634851 (Dec. 14, 2003, 15:06 PM GMT) (alteration added); *Accord* Eben Moglen, *Freeing the Mind: Free Software and the Death of Proprietary Culture*, 56 ME. L. REV. 1, 6 (2004) ("[The GPL] requires no acceptance. It requires no contractual obligation. It says you are permitted to do, just don't try to reduce anyone else's rights.").
[87] United Nations Convention on Contracts for the International Sale of Goods, Apr. 11, 1980, S. TREATY DOC. NO. 98-9, 19 I.L.M. 668 [hereinafter CISG].

entered it into force.[88] With software developers spread around the world, a GPL licensor and licensee will often be parties in different Contracting States under the CISG.[89]

The CISG does not require consideration for contract formation, and does not even define or use the term consideration. Part II of the CISG governs formation of contracts between parties in Contracting States in terms of offer and acceptance only.[90] Under the CISG, the GPL constitutes "an offer if it is sufficiently definite and indicates the intention of the offeror to be bound in case of acceptance" and "if it indicates the goods and expressly or implicitly fixes or makes provision for determining the quantity and the price."[91] The GPL terms apply to a piece of software, an indication of the goods required by CISG Article 14, and outline the terms by which the GPL binds the licensor.[92] As Professor Moglen notes, the GPL fixes the price: free.[93]

Section 5 of the GPL contemplates acceptance of the GPL's terms when "by modifying or distributing the Program . . . you indicate your acceptance of this License to do so, and all its terms and conditions . . . ."[94] This offer-plus-conduct approach meets the Convention's definition that "a statement made by or other conduct of the offeree indicating assent to an offer is an acceptance."[95] The CISG makes no mention of a separate class of licenses with different treatment from contracts under the treaty.

The GPL "license" meets the simple requirements for contract formation under the CISG. How the Convention treats software in its definition of goods is less clear. Reported CISG decisions support the inclusion of software in the CISG definition of goods.[96] The text of the Convention provides no clear definition of either goods or sale, other than the exclusion of certain goods in Article 2 and sales of services in Article 3.[97] The Convention does not mention sales of software, nor does it exclude the sale of intangible goods. Even though Article 2(f) explicitly excludes electricity, it does not exclude other intangibles such as the sale of gas.[98]

---

[88] UNCITRAL (United Nations Commission on International Trade Law), Status: 1980 United Nations Convention on Contracts for the International Sale of Goods, http://www.uncitral.org/uncitral/en/uncitral_texts/sale_goods/1980CISG_status.html (last visited Jan. 5, 2007).Ghana and Venezuela have signed the treaty, but not entered it into force. *Id.*
[89] *See* Free Software Foundation, Guide to Translating, http://www.gnu.org/server/standards/README.translations.html#TranslationsUnderway (last visited Jan. 5, 2007). A list of GNU translators alone reveals over three dozen different countries and languages. *Id.*
[90] The CISG never references the term consideration. Given the vast difference between contract formation concepts at common and civil law, Contracting States would undoubtedly face extreme difficulty in framing an acceptable definition.
[91] CISG, *supra* note 87, at art. 14.
[92] GPL, *supra* note 4.
[93] Jones, *supra* note 86.
[94] GPL, *supra* note 4, at § 5.
[95] CISG, *supra* note 87, at art. 18(1).
[96] *See infra* text accompanying notes 121-127.
[97] *Id.* at arts. 2(a)-2(f), 3(2).
[98] *See* J.W. Carter, *Article 2B: International Perspectives*, 14 J. CONTRACT LAW 54, 65 (1999) (Austl.) ("The reason for including electricity (and gas) is that electricity is capable of being consumed in the same way as, for example, food. However, this does not mean that we should treat all intangibles as goods . . ."); L. Scott Primak, *Computer Software: Should the U.N. Convention n Contracts for the International Sale of Goods Apply? A Contextual*

The argument for a contractual interpretation in light of the CISG generally makes sense, however. Issues surrounding the acceptance of GPL terms aside, if Programmer A downloads "GPL Ware" from Programmer B in consideration of the obligations contained in the GPL, the transaction between A and B bestows rights in the physical copy of the software on A. At a minimum, A buys both a physical copy of the software and a license to use the software, although those rights vary from jurisdiction to jurisdiction.

Although a software program delivered over an intangible medium does not exist in the way a car's catalytic converter, a bulb for a floodlight, or digital camera zoom lens exists, it functions much in the same way that these tangible machines function. A piece of software is not merely a packaged set of copyrighted expressions,[99] but a functional "machine" designed to accomplish a specific set of tasks. Rarely do software developers delight in the aesthetic beauty of a function call to a Linux kernel thread, reveling instead in its *functionality*. Viewed through the prism of a functional machine, software lends itself far more readily to CISG and domestic law prescriptions on contract formation, obligations, and remedies designed to protect and regulate sales of "things." Downloadable software fits the "sale of goods" concept outlined in CISG Article 30.[100] Article 31(c) allows the seller to "[place] the goods at the buyer's disposal at the place where the seller had his place of business at the time of the conclusion of the contract," such as in a download area of a Web site.[101]

Even if Programmer B includes a clickwrap agreement, specifically requiring Programmer A to treat the GPL as a bare copyright license that the parties will interpret using only the jurisdiction's relevant copyright law, the parties maintain the same seller/buyer relationship. The GPL recognizes intellectual property protections and purports to bind A to use B's software in accordance with those rights. This limitation on how A may use B's "GPL Ware" does not change the nature of the transaction. The restrictions do not magically transform a tangible software "good" into an amorphous glob of expressive letters and symbols ignored by the CISG or countries' other domestic sales laws. A fixation and license to use a copy of "GPL Ware" become A's property at the completion of the transaction, subject to intellectual property rights. Contract law, not copyright law, protects A's rights as a buyer in this transaction.[102]

In 1995, a German district court held that the CISG does apply to contracts for the sale of standard software.[103] In that case, a German company purchased software from a French

---

Approach To the Question, 11 COMPUTER/L.J. 197 (1991). *Contra* Franco Ferrari, *Specific Topics of the CISG in the Light of Judicial Application and Scholarly Writing*, 15 J. L. & COM. 1, 65, 65 n.430 (1995) (combining dictum from a non-software related German case with other scholarly definitions and the Convention relating to a Uniform Law on the International Sale of Goods to propose that only tangible objects qualify as goods under the CISG).

[99] Part III addresses the difficulties in finding blanket copyright protection in software.

[100] CISG, *supra* note 87, at art. 30 ("The seller must deliver the goods, hand over any documents relating to them and transfer the property in the goods, as required by the contract and this Convention.").

[101] *Id.* at art. 31.

[102] This may be true whether a court considers the physical copy of the software or the license as standalone items. The buyer and seller have established a contractual relationship. Contract law will necessarily govern at least a portion of the relationship between the parties, even if copyright law or licenses like the GPL supply additional intellectual property guidance.

[103] Landgericht [LG] München [Munich District Court], Feb. 8, 1995, Docket No. 8 HKO 24667/93, *available at* http://cisgw3.law.pace.edu/cisg/wais/db/cases2/950208g4.html.

company. The French seller delivered and installed the programs and sued in a German court to recover the purchase price when the buyer refused to pay.[104] Although this German case concerned software delivered on tangible media, no logical reasons exist to treat software delivered over the Internet, like the software in *Specht v. Netscape Communications Corp.*,[105] or on tangible media differently. Regardless of the mode of delivery, the software remains a "good" subject to an agreement between the parties regarding its transfer.

A 2000 case in the Zurich, Switzerland Commercial Court held that "the purchase of software as well as the joint purchase of software and hardware constitutes a sale of goods that falls within the ambit of the CISG."[106] Citing a German CISG treatise[107] and the text of the CISG itself,[108] the court applied the CISG to a contract for the sale of software and hardware between an Austrian seller and a Swiss defendant.[109] After the seller delivered the hardware and software, the parties negotiated a second contract for software maintenance.[110] Negotiations on the second contract failed, and the buyer refused to pay for the installed software, citing non-conformity.[111] The Zurich court found that the ancillary installation and customization services did not "prevail in the [ ] contract" and "the CISG has to be the governing law of the contract."[112] The seller provided the Oracle software under a license, and not as a sale.[113] The court did not draw a distinction between the license and the sale contract, referring to "the procurement of the user rights to the standard software" in calculations of the sale versus service component of the master contract.[114] The court ultimately found for the seller, holding that the buyer had failed to detail the non-conformity as required by Article 39(1) of the CISG.[115]

Applying the CISG to transfers of software under the GPL license conforms to the underlying goals of the CISG. Article 7(1) of the CISG contains two instructions for the Sphere of Application of the treaty: (1) promotion of uniformity in international law, and (2) observance of good faith.[116] Exempting software licenses, especially if only for intangible delivery, frustrates

---

[104] *Id.*
[105] 150 F.Supp. 2d 585 (S.D.N.Y. 2001).
[106] Handelsgericht [HG] Zurich [Commercial Court of the Canton of Zurich], Feb. 17, 2000, Docket No. HG 980472, *available at* http://cisgw3.law.pace.edu/cases/000217s1.html [hereinafter HG Zurich].
[107] VON E. CAEMMERER & PETER SCHLECHTRIEM, KOMMENTAR ZUM EINHEITLICHEN UN-KAUFRECHT-CISG [COMMENTARY ON THE UN CONVENTION ON THE INTERNATIONAL SALE OF GOODS (CISG)] (Peter Schlechtriem ed., Geoffrey Thomas trans., Oxford Clarendon Press 2d ed. 1998). The court cited Caemmerer & Schlechtriem to support the finding that software could be the object of a tangible goods purchase under the CISG. *HG Zurich* at § IV.1.c).
[108] *HG Zurich,* at § IV.1.c. ("see also Art. 3 and Art. 51 N 4 . . . . the typical element still remains as a sale of goods.").
[109] *Id.* at § II.1.
[110] *Id.* at § IV.1.c).
[111] *Id.* at § IV.5.a).
[112] *Id.* at § IV.1.c).
[113] *Id.* at § IV.5.a).
[114] *Id.* at § IV.1.c).
[115] *Id.* at § IV.5. *See also* note 118, *infra,* regarding the implications of applying CISG rules on contract formation, obligation and remedies.
[116] CISG, *supra* note 87, at art. 7(1) ("In the interpretation of this Convention, regard is to be had to its international character and to the need to promote uniformity in its application and the observance of good faith in international trade.")

the purpose of the CISG. While uniform interpretation of software licensing under the CISG is no more possible than in any other area of CISG jurisprudence,[117] application of the CISG to GPL license transactions has inherent value. By interpreting the GPL under the auspices of the CISG, courts place themselves in a better position to consider international legal uniformity and the international character of software development under the GPL.[118]

Decisions in foreign courts that find domestic contractual relationship in a software license also support the interpretation of a GPL as a contract. For example, German law defines a contract as a congruence of two or more persons' will to create a legal effect.[119] As with the CISG, the German Civil Code does not require consideration, only a mutual agreement. Section 311 of the German Civil Code requires that "for the creation of an obligation by legal transaction, and for any modification of the substance of an obligation, a contract between the parties is necessary, unless otherwise provided by law."[120] Superficially, the obligations under the GPL appear to require a contract between the parties, and German courts have adhered to this interpretation.

In *Welte v. Sitecom Deutschland GmbH*,[121] a district court in Munich held that it had "no doubts as to the validity of the conditions" set forth in the GPL.[122] The court agreed with "the prevailing view in German legal doctrine that the terms of the GPL are general terms and conditions (Allgemeine Geschäftsbedingungen) in the meaning of Section 305 et seq. of the German Civil Code (BGB)."[123] The opinion examined the license under application of German Civil Code Section 305, and stated "general business conditions" of the GPL were "effectively incorporated into a possible *contractual relationship between the defendant and plaintiff*."[124] The court then referred to the reversal of rights upon infringement contained in GPL Section 4 as non-discriminatory to "the contractual partner of the user."[125] Notably, the court interpreted the GPL under the German Copyright Act, reinforcing the dangers of the lack of a choice of law

[117] *See* Franco Ferrari, *Uniform Interpretation of the 1980 Uniform Sales Law*, 24 GA. J. INT'L & COMP. L. 183 (1999); PETER WINSHIP, *The Scope of the Vienna Convention, in* INTERNATIONAL SALES 1-1 at, 1-15 (Nina M. Galston & Hans Smit eds., 1984) (discussing the rejection of his proposal to create an official commentary to the Convention).
[118] The application of the CISG through the characterization of the GPL as a sale of goods may have other significant implications. Remedies in the CISG differ significantly from those available at common law or under the UCC. Courts may disallow the GPL's disclaimer of warranty or interpret copyright terms differently depending on the jurisdiction and the applicable local laws. Other treaties, declarations related to the CISG further affect the interpretation and validity of contracts in CISG Contracting States, and the resulting implications when only one party resides in a CISG Contracting State. A full discussion of the GPL's failure to include or consider choice of law provisions, while likely of significant interest to commercial and non-commercial software developers alike, extends beyond the scope of this paper.
[119] Bürgerliches Gestzbuch [BGB] [Civil Code] §145.
[120] *Id.* at §311 Abs. 1, *available at* http://www.iuscomp.org/gla/statutes/BGB.htm (English translation).
[121] Landgericht [LG] München I [Munich District Court I], May 19, 2004, Docket No. 21 O 6123/04, *available at* http://www.jbb.de/judgment_dc_munich_gpl.pdf [hereinafter Munich I].
[122] *Id.* at 12. *But see infra* note 322 and accompanying text (disclaimers such as those in GPL §§ 11, 12 "are generally considered to be invalid under German law").
[123] Martin Braun, *German Court Upholds Linux License*, 5 INT'L IT AND OUTSOURCING NEWSL., October 5, 2004, *available at* http://www.mayerbrown.com/broker.asp?id=1893&nid=6.
[124] *Munich I*, *supra* note 121, at 9 (emphasis added).
[125] *Id.*

provision and the dangers of inexact definitions of terms.[126] If the prevailing view persists, German courts will continue to draw no distinction between a license and a contract for purposes of GPL enforcement.[127]

ii. Contractual Interpretations of Software Licenses and GPL in the United States

In the United States, federal courts originally followed the CISG and German interpretation above, and commonly applied state contract law to licenses. In *Power Lift, Inc. v. Weatherford Nipple-Up Systems, Inc.*, the Court of Appeals for the Federal Circuit held that "a license agreement is a contract governed by ordinary principles of state contract law."[128] Similarly, the same circuit court held that licenses create an implied right to enforce contractual remedies.[129] As late as 1998, Professor Raymond Nimmer wrote, "Contract and intellectual property law have always co-existed, not only peacefully, but [also] in an aggressive interaction between mutually supportive fields."[130] However, by 2002, the climate had changed and the "very few lower court cases that hold to the contrary [were] outside the mainstream and inconsistent with commercial practice."[131] Although earlier decisions such as *Power Lift* were never explicitly overruled, of the few courts that held software licenses were traditional sales contracts governed by state contract law, "two were reversed on appeal . . . [and] two were vacated."[132]

By 2002, federal courts had moved largely to an interpretation that applied copyright law, albeit narrowly, in order to preempt conflicting contract provisions, and used state contract law to govern the remainder of software license agreements. In 2000, a California district court heard a case representative of this doctrinal change in *Adobe Systems, Inc. v. One Stop Micro, Inc.* [133] Plaintiff Adobe alleged that defendant One Stop acquired Adobe educational license software packages from an Adobe distributor, "adulterating [those] educational versions of Adobe

[126] *Id.* at 9-12; *supra* note 118 (discussing the effect of the CISG on the interpretation of the GPL). For a discussion of definitions in the GPL itself, *see infra* Part III E.
[127] For a discussion of software copyright interpretations and the GPL in the United Kingdom and other Commonwealth countries, *see generally* Douglas A. Hass, *Uneasy Détente: Strengthening the Market's Adaptation of the GNU General Public License in Common Law Jurisdictions*, 2 OXFORD J. INTELL. PROP. L. & PRAC. (2007) (U.K.).
[128] 871 F.2d 1082, 1085 (Fed. Cir. 1989); *See also* In re CFLC, Inc., 89 F.3d 673, 677 (9th Cir. 1996). *But see* Adobe Sys. Inc. v. One Stop Micro, Inc., 84 F. Supp. 2d 1086, 1092 (N.D. Cal. 2000) (holding that the distribution agreement was a license and not a sale, therefore, the 'first sale' doctrine was inapplicable); Peter C. Quittmeyer, *Software Licensing*, 2003 PRACTISING L. INST. 903, 909 *available at* WL 763 PLI/Pat 903 ("In general, a license does not grant or create a property interest at all, but merely a permission to act or, conversely, a covenant not to sue for such action.").
[129] McCoy v. Mitsuboshi Cutlery, Inc., 67 F.3d 917 (Fed. Cir. 1995) (holding that the defendant could resell goods under commercial contract law principles of mitigation after plaintiff breached a license agreement).
[130] Raymond T. Nimmer, *Breaking Barriers: The Relation Between Contract and Intellectual Property Law*, 13 BERKELEY TECH. L.J. 827, 829 (1998).
[131] Raymond T. Nimmer, THE LAW OF COMPUTER TECH.: COPYRIGHT & COMPUTER TECH. §1:97 (2002).
[132] *Id.* (citing DSC Commc'ns Corp. v. Pulse Commc'ns, Inc., 170 F.3d 1354 (Fed. Cir. 1999) (reversing District Court ruling that licensee was an owner and expressly rejecting the view that a single payment perpetual license means that the transfer of the copy was a sale).
[133] 84 F. Supp. 2d 1086 (N.D. Cal. 2000).

software and re-packaging them as full retail versions."[134] Adobe argued that an end user license agreement (the "On/Off Campus Educational Reseller Agreement" or "OCRA") covered all of its software products. That license granted "a nonexclusive license to use the Software and Documentation" subject to certain limitations.[135] Among the limitations in the license agreement were prohibitions on copying or commercial redistribution of the software, and a list of "Reseller Rights" that further restricted how the defendant could sell Adobe educational license software.[136]

The court overlooked any privity issues and applied the agreement to One Stop even though it had not signed the OCRA, but purchased the software on the open market.[137] Further, the court held that One Stop had infringed Adobe's copyright by violating Adobe's license agreement.[138] One Stop, however, contended that it had purchased the software, entitling the company to a first sale exemption under the U.S. Copyright Act.[139] The relevant portion of the Act provides that "the owner of a particular copy or phonorecord lawfully made under [the Act], or any person authorized by such owner, is entitled without the authority of the copyright owner, to sell or otherwise dispose of the possession of that copy or phonorecord."[140] This first sale doctrine "terminates the copyright holder's authority to interfere with subsequent sales or distribution of that particular copy."[141]

In 1990, Congress amended the Copyright Act with respect to software purchases.[142] As amended by Congress, the first sale doctrine permits only non-profit libraries and educational institutions to lend or lease copies of software and phonorecords.[143] The amendment corrected a deficiency in the Act that enabled buyers to purchase a copy of a computer program, and then lease it or lend it to another without infringing the program's copyright.[144] Because users could easily copy software, software companies feared that buyers would purchase copies of programs and lease them to consumers, who would then copy the software themselves.

By characterizing the original transaction as a license, rather than a sale, as Adobe had done, and by prohibiting transfers of the license, software companies hoped to avoid the first sale doctrine and establish contractual privity to sue directly any companies "renting" the software to

[134] *Id.* at 1093.
[135] *Id.* at 1091.
[136] *Id.* at 1091, n.2.
[137] *Id.* at 1088. The court explains "Adobe initially distributes the educational versions to an Adobe-authorized educational distributor, who then transfers the software to an Adobe-authorized educational reseller." *Id.* One Stop purchased the software, containing the shrink-wrapped OCRA, from a distributor. *Id.*
[138] *Id.* at 1093 ("One Stop has committed copyright infringement as a matter of law under § 501 (a): Anyone who violates any of the exclusive rights of the copyright owner as provided by sections 106 through 118 . . . is an infringer of the copyright.'"). The same court reaffirmed this analysis in a 2002 case that also involved Adobe: *Adobe Sys., Inc. v. Stargate Software, Inc.*, 216 F. Supp. 2d 1051 (N.D. Cal. 2002).
[139] Adobe Sys. Inc. v. One Stop Micro, Inc., 84 F. Supp. 2d at 1093.
[140] 17 U.S.C. §109(a) (2000).
[141] Parfums Givenchy, Inc. v. Drug Emporium, Inc., 38 F.3d 477, 480 (9th Cir. 1994).
[142] Computer Software Rental Amendments Act of 1990, 104 Stat. 5134 (1990) (codified at 17 U.S.C. § 109(b) (2000)).
[143] 17 U.S.C.A. § 109(b)(1)(A) (West Supp. 1991).
[144] *See generally* Bobbs-Merrill Co. v. Straus, 210 U.S. 339 (1908); 17 U.S.C.A. § 109(a) (West 1977).

others. Congress resolved the tension between the first sale doctrine and state contract law by passing the amendment.[145] This amendment eliminated the need to term sales as "licenses" for purposes of avoiding the first sale doctrine.[146] Unsurprisingly, despite provisions in the end user license agreement that denoted a sale,[147] the court agreed with Adobe, finding that the OCRA was a software licensing agreement and that the first sale doctrine was therefore inapplicable.[148]

Tellingly, though, the *Adobe* court continued to refer to the license agreement as a contract throughout its opinion.[149] The court cited California civil code in allowing parol evidence to construe the ambiguity between the plaintiff's and defendant's interpretations of the contract as either a license or sale.[150] Ruling that the OCRA was a license and not a sale, the court referred to "the express restrictive language of the contract."[151] The court refused to recognize Professor Moglen's concept of a license that rested outside the bounds of state contract law and was governed solely by U.S. copyright law. Instead, the court applied copyright law narrowly and used state contract law to define and interpret the relationship between licensor and licensee.[152]

The most frequently cited case supporting this principle, *ProCD, Inc. v. Zeidenberg*, appears in numerous first-year Contracts casebooks.[153] The court in *ProCD* held that a unilateral software license, such as the shrinkwrap end-user license in that case, or the GPL here, is enforceable as "a simple two-party contract." [154] Whether general or restrictive, the court found that a license "is not equivalent to any of the exclusive rights within the general scope of copyright . . . ."[155] The court's argument in *ProCD* also effectively rebuts any arguments asserting that federal copyright law preempts all state law claims in contract, tort, or other areas.[156] Section 301 of the Copyright Act provides, in relevant part, that federal laws preempt "all legal or equitable rights that are equivalent to any of the exclusive rights within the general

[145] *See generally* Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141 (1989); Sears, Roebuck & Co. v. Stiffel Co., 376 U.S. 225 (1964).
[146] *See* 104 Stat. 5134.
[147] Adobe Systems, Inc. v. One Stop Micro, Inc., 84 F. Supp. 2d 1086, 1090 (N.D. Cal. 2000) ("Reseller shall have the right to purchase"; "the respective Educational Software Products owned by Reseller"; "Adobe may, at its option, repurchase").
[148] *Id.* at 1092.
[149] *E.g.*, *id.* at 1090 (interpreting the agreement as a contract under California Civil Code).
[150] *Id.* ("The parties' intent is inferred exclusively from the language of the contract, assuming the language is 'clear and explicit.' See Cal. Civ. Code. § 1638.")
[151] *Id.* at 1092.
[152] For other cases supporting the combination of narrow copyright law application and broad contract law application, *see, e.g.*, Microsoft Corp. v. Harmony Computers & Electronics, Inc., 846 F. Supp. 208, 210 (E.D.N.Y. 1994); Adobe Systems, Inc. v. Stargate Software, Inc., 216 F. Supp. 2d 1051 (N.D. Cal. 2002); Softman Prod. Co. v. Adobe Systems, Inc., 171 F. Supp. 2d 1075, 1080 (C.D. Cal. 2001).
[153] 86 F.3d 1447 (7th Cir. 1996). For examples of casebooks, *see, e.g.*, THOMAS D. CRANDALL & DOUGLAS J. WHALEY, CASES, PROBLEMS, AND MATERIALS ON CONTRACTS (4th ed. 2004); ALLAN FARNSWORTH ET AL., CONTRACTS: CASES AND MATERIALS (6th ed. 2001).
[154] *ProCD* , 86 F.3d at 1455.
[155] *Id.*
[156] The converse is also true. The court did not go as far as *Power Lift*, 871 F.2d 1082, 1085 (Fed. Cir. 1989), finding it "prudent to refrain from adopting a rule that anything with the label 'contract' is necessarily outside the preemption clause: the variations and possibilities are too numerous to foresee." *ProCD,* 86 F.3d at 1455.

scope of copyright."[157] Therefore, in order for the Act to preempt claims under state law, "(1) the work at issue [must come] within the subject matter of copyright, and (2) the state law rights [must be] equivalent to any of the exclusive rights within the scope of copyright."[158] State law survives preemption when claims made under contract law contain an "extra element" that differs materially from those rights protected by copyright law.[159] The Ninth Circuit held in *Altera* that "[a] state law tort claim concerning the unauthorized use of the software's end-product is not within the rights protected by the federal Copyright Act."[160]

The *Altera* court also noted, "[m]ost courts have held that the Copyright Act does not preempt the enforcement of contractual rights."[161] However, the Ninth Circuit in *Del Madera Properties v. Rhodes & Gardner, Inc.*,[162] refused a plaintiff's claim for unjust enrichment because it lacked an extra element.[163] The court specifically cited the lack of expected consideration between the parties.[164] In *Del Madera*, however, the parties' dispute revolved around a copyrighted map prepared by the plaintiff and not a software license with terms requiring the licensee to undertake and avoid certain actions.[165]

Although neither case turned on contract validity issues, the only two U.S. cases as of this writing that involved the GPL accepted the license as a valid contract. In *Progress Software Corp. v. MySQL AB*,[166] neither party challenged the validity of the GPL. MySQL's counterclaim alleged a breach of contract by the plaintiff in violating the terms of the GPL.[167] The parties in the second case, *MontaVista Software, Inc. v. Lineo, Inc.*,[168] also accepted the validity of the GPL. The parties' complaints discussed contractual breaches of the GPL, among other alleged violations.[169]

---

[157] 17 U.S.C. § 301 (2000).
[158] Grosso v. Miramax Film Corp., 383 F.3d 965, 968 (9th Cir. 2004) (internal citations and quotation marks omitted) (alterations added), amended by, reh'g denied by 400 F. 3d 658 (2005).
[159] Altera Corp. v. Clear Logic, Inc., 424 F.3d 1079, 1089 (9th Cir. 2005) (citing Summit Mach. Tool Mfg. v. Victor CNC Sys., 7 F.3d 1434, 1439-40 (9th Cir. 1993)).
[160] *Altera Corp.*, 424 F.3d at 1090.
[161] *Id.* at 1089 (citing Bowers v. Baystate Techs. Inc., 320 F.3d 1317, 1323-24 (Fed. Cir. 2003); ProCD, Inc. v. Zeidenberg, 86 F.3d 1447 (7th Cir. 1996); Nat'l Car Rental Sys. Inc. v. Computer Assocs. Int'l, 991 F.2d 426, 431 (8th Cir. 1993)).
[162] 820 F.2d 973 (9th Cir. 1987).
[163] *Id.* at 978.
[164] *Id.*
[165] *Id.* at 975. Nonetheless, a U.S. court could refuse similar claims if it deems that the forbearances in the GPL do not constitute valid consideration. Claims for unjust enrichment may be unavailable in other common law jurisdictions. A debate about preemption and the sufficiency of consideration and what, if any, requirements various common law and civil law jurisdictions, the CISG, and state UCC enactments impose on both is beyond the scope of this paper. For purposes of discussion, this paper accepts the majority U.S. rule on preemption as outlined by the Ninth Circuit in *Altera*.
[166] 195 F. Supp. 2d 328 (D. Mass. 2002).
[167] Laura A. Majerus, *Court Evaluates Meaning of "Derivative Work" in an Open Source License*, FINDLAW, Jun. 16, 2003, http://library.findlaw.com/2003/Jun/16/132811.html (last visited Jan. 5, 2007).
[168] First Amended Complaint of Plaintiff, MontaVista Software, Inc. v. Lineo, Inc., No. 2-02 CV-00309J (D. Utah Jul. 23, 2002).
[169] *E.g.*, *id.* at ¶ 50 ("The aforesaid individual or joint acts of Defendants constitute a breach of the GPL."). The parties settled the case before trial in 2003.

Despite the presence of an offer, purported consideration, and an invitation to accept the GPL, Professor Moglen's argument still has merit, albeit for different reasons. Rather than a novel interpretation of law that attempts to attach the GPL to U.S. copyright law, the license argument finds more solid ground in the realm of property law. Accepting Professor Moglen's argument, the GPL unilaterally grants a property right in order to achieve Stallman's goal of a self-perpetuating software commons.[170] The GPL uses its grant of property rights, here the rights to copy, distribute, and modify copyright-protected material, to enforce its social contract.[171] Contrary to the court's view in *ProCD*,[172] under this interpretation the GPL does not create an agreement for the transfer of software with separate promises. Rather, the GPL creates a single, non-exclusive right to use the GPL-licensed source code subject to conditions spelled out in the license. Depending on local copyright law, this property right argument could potentially overcome the CISG and various civil code jurisdiction equations between licenses and sales of goods, while Professors Moglen and Rosen's interpretation immediately fails in many jurisdictional situations outside of U.S. courts.

Professor Moglen may provide the best argument against this approach, though, when he argues that "Free software . . . is a commons: no reciprocity ritual is enacted there. A few people give away code that others sell, use, change, or borrow wholesale to lift out parts for something else."[173] The copyright system rewards authors "by means of an incentive system,"[174] validating Ronald Coase's theory on production.[175] Coase recognized that transaction costs create disincentives for actors to participate in a market.[176] The disincentive also applies to production: market actors decide what to produce, instead of purchase, based on transaction costs inside and outside the firm.[177] Copyrights confer a property right—the right to exclude others—in an author's works. This exclusive property right overcomes the transaction costs of production inherent in a commons. Instead of facing a tragedy of the commons, copyright law allows authors to earn a return on their investment by charging what the market bears for access to the creation. Coase's theorem dictates that authors will charge a fee sufficient to induce them to create the work initially.[178] Economically, a market's buyers internalize the additional costs under the assumption that authors would not otherwise create those works.[179] The rights created by copyright, according to Coase's theorem, generate a marginal increase in authorship sufficient to overcome the costs of exercising the rights to exclude, (copyright licensing, patent licensing,

[170] Jones, *supra* note 86.
[171] *See* GPL, *supra* note 4, at §§ 1-4.
[172] *See supra* text accompanying note 154 (characterizing a license as a "simple two-party contract").
[173] Eben Moglen, Anarchism Triumphant: Free Software and the Death of Copyright, http://emoglen.law.columbia.edu/my_pubs/anarchism.html (last visited Jan. 6, 20076).
[174] Robert A. Kreiss, *Accessibility and Commercialization in Copyright Theory*, 43 UCLA L. REV. 1, 14 (1995).
[175] *See generally* Ronald H. Coase, *The Institutional Structure of Production*, *in* ESSAYS ON ECONOMISTS AND ECONOMICS 3, 9 (1994).
[176] *Id.*
[177] *Id.*
[178] *Id.*
[179] *See* Kreiss, *supra* note 174, at 8.

etc.) plus any transaction costs associated with the system itself.[180] If free software is truly a commons, it contradicts copyright law doctrine.

The survival of the vibrant free software commons, therefore, depends on the ability of community members to enforce the community's social norms. Enforcing the GPL as a contract comports with current U.S. and international case law. Additionally, the construction of the GPL as a contract carries other benefits to licensees and licensors that seek to enforce the free software community's values. For example, contractual interpretation opens up a range of options to licensors, including enforcement by distributors of code rather than just copyright holders, and the availability of state and local venues.[181] Part III discusses the danger of relying solely on copyright law for GPL enforcement given the limited protections that copyright affords certain types of software under Linux.

A. A Brief Primer on Assent to Software Licenses

Despite the *ProCD* decision that raised the validity of shrinkwrap licenses, software companies have increasingly turned to less traditional methods of contract formation better suited to an on-line environment. The GPL is no exception. Companies as varied as Ticketmaster,[182] CNET's download.com,[183] and United Airlines[184] rely on form contracts. Commentators have long seen the value in using these streamlined, standardized agreements, and an overwhelming majority of contracting parties today relies on standard forms.[185] In the realm of open source software, licensors cannot efficiently negotiate contracts with every licensee. The requirement of individual negotiations, or even product-by-product form contracts, helped prod Richard Stallman to create the GPL in the first place.[186]

In 1972, the Supreme Court's decision in *Gottschalk v. Benson* severely limited protections available to software, denying patentability for a mathematical formula out of a fear that such a patent "in practical effect would be a patent on the algorithm itself."[187] Although

[180] Sony Corp. of America v. Universal City Studios, Inc., 464 U.S. 417 (1984) ("The immediate effect of our copyright law is to secure a fair return for an 'author's' creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good.").
[181] *See* ROSEN, *supra* note 85, at 139.
[182] Ticketmaster, Purchase Policy, http://www.ticketmaster.com/h/purchase.html (last visited Jan. 6, 2007).
[183] CNET Networks, Site Terms of Use, http://www.cnetnetworks.com/editorial/terms.html (last visited Jan. 6, 2007).
[184] United Airlines, Contract of Carriage Guide, http://www.united.com/ual/asset/Contract_of_Carriage_070606.pdf (last visited Jan. 6, 2007).
[185] *See* Restatement (Second) of Contracts § 211 cmt. a (1981) ("Standardization of agreements serves many of the same functions as standardization of goods and services; both are essential to a system of mass production and distribution."); W. David Slawson, *Standard Form Contracts and Democratic Control of Lawmaking Power*, 84 HARV. L. REV. 529, 529-30 (1971) (standard form contracts "probably account for more than ninety-nine percent of all the contracts now made"); Randy E. Barnett, *Consenting to Form Contracts*, 71 FORDHAM L. REV. 627, 627 (2002) ("There is a remarkable dissonance between contract theory and practice on the subject of form contracts. In practice, form contracts are ubiquitous. From video rentals to the sale of automobiles, form contracts are everywhere. Yet contract theorists are nothing if not suspicious of such contracts, having long ago dubbed them pejoratively 'contracts of adhesion.'").
[186] *See* Stallman, *supra* note 57 (discussing the process that led Stallman to create a single GPL).
[187] 409 U.S. 63, 71 (1972).

courts and amendments to the Copyright Act have changed interpretations of patent and copyright law in the thirty-plus years since *Gottschalk*, the decision and lack of clarity created considerable uncertainty in the market at the time.[188] Given the uncertainty, software developers like ProCD created licenses that protected their products from unauthorized copying and distribution to others, blocked efforts to "reverse engineer" the software to alter it or create a similar competing product, and limited warranty claims.[189] With mass distribution of software increasing dramatically through the 1990s as businesses and individuals increasingly adopted personal computers and networks, "individually negotiated contracts [were] not feasible."[190] The solution, shrinkwrap licenses, had little legal foundation in contract law. Buyers received a sealed package with notification that a license governed the software inside.[191] The seller, however, typically placed the license inside the sealed package or physically on the media itself.[192] Unlike a traditional contract, the parties to shrinkwrap agreements do not negotiate and the buyer makes no manifestation of assent to the unseen terms. Retailers, then and now, dissuade customers from returning items by refusing to accept opened boxes or charging exorbitant restocking fees.[193] Such terms and the lack of a simple, or any, remedy create the possibility of significant abuse by software companies.

Despite uncertainty about the validity of shrinkwrap or "terms later" licenses, standardized licensing became the enabling mechanism for growth. The holding in *ProCD*, despite significant criticism in legal academia, cemented this method of standardized, electronic commerce.[194] As delivery options expanded and software capabilities evolved, the shrinkwrap license expanded to include intangible media such as Internet downloads, and gave rise to two other common licensing processes.

As software capabilities improved, software companies built programs designed for permanent installation on personal computers and servers. Shrinkwrap licenses morphed into "clickwrap" licenses that required users to review and accept a license agreement presented automatically during software installation.[195] Since software no longer ran solely from the enclosed media and could be copied and distributed easily, software companies had an additional

[188] Steven Fraser, *Canada-United States Trade Issues: Back from Purgatory? Why Computer Software "Shrink-Wrap" Licenses Should Be Laid to Rest*, 6 TUL. J. INT'L & COMP. L. 183, 189. Part III, *infra*, argues that the uncertainty is no less pronounced today.
[189] *See*, *infra*, Part III.
[190] Robert Gomulkiewicz & Mary Williamson, *A Brief Defense of Mass Market Software License Agreements*, 22 RUTGERS COMPUTER & TECH. L.J. 335, 342 (1996).
[191] *See, e.g.*, ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1450 (7th Cir. 1996) (discussing shrinkwrap license used by ProCD).
[192] *Id.* at 339-41 (explaining that buyers can enter into a shrinkwrap license agreement by, for example, "tearing open the plastic wrapper covering the box" or "installing or using the software").
[193] *See, e.g.*, Amazon.com, Refunds, http://www.amazon.com/gp/help/customer/display.html/102-6299803-1267335?nodeId=901926 (last visited Jan 6, 2007) (discussing the restocking fee for "[a]ny . . . software . . . that has been opened (taken out of its plastic wrap) [is] 50% of [the] item's price.") (alterations added).
[194] For criticisms of Judge Easterbrook's holding in *ProCD* and shrinkwrap licenses as a whole, *see, e.g.*, Barnett, *supra* note 185 (criticizing the difference between form contract theory and practice); Roger C. Bern, *"Terms Later" Contracting: Bad Economics, Bad Morals, and a Bad Idea for a Uniform Law, Judge Easterbrook Notwithstanding*, 12 J.L. & POL'Y 641, 641-43 (2004); Fraser, *supra* note 188 (advocating the elimination of shrinkwrap licenses).
[195] Gomulkiewicz & Williamson, *supra* note 190, at 340 (describing transitions in end user licensing as technological improvements provided additional options for software companies).

incentive to secure valid license agreements from end users. Given these new concerns and the academic criticism of the "terms later" model, software installation programs routinely required users to check a box or click a button labeled "I agree," "I accept" or a similar manifestation of assent.[196] Without accepting the license agreement, the software installation typically aborts.[197] Many Microsoft products now require users to scroll to the bottom of the on-screen license before the installation process enables the "Accept" button.[198] This added clickwrap step avoids criticism that licensors encourage users to bypass or skip complicated, lengthy license agreements.[199] Ultimately, consumers face the same choice as with shrinkwrap contracts: rejecting the license and returning unopened software could be expensive and difficult, if not infeasible.

As software distribution moved on-line, some companies made their license agreements available pre-purchase by putting the document on-line and linking their Web site to it.[200] The GPL relies primarily on this "browsewrap" method. Despite language in Section 2(c) describing situations where "you must cause [the Program], when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice," few GPL programs do so.[201] Theoretically, users have the ability to review the documentation before acceptance, but links to legal terms are rarely obvious.[202] In *Specht v. Netscape Communications Corp.*, the court refused to enforce Netscape's browsewrap license for its SmartDownload software because the customer was neither required to view the license nor

---

[196] Gary Hamilton & Jeffrey Hood, *The Shrink-Wrap License - Is it Really Necessary?*, 10 COMPUTER & INTERNET LAWYER, 16, 16 (1993) (describing "clickwrap" contracts and user assent during the installation process).
[197] *Id.* (describing the clickwrap installation process).
[198] Although this added scroll requirement ostensibly "forces" users to read the license agreement, it also can cause considerable frustration for users, *e.g.*, Posting of Hugh Thomas to microsoft.public.windowsupdate, *available at* http://groups.google.com/group/microsoft.public.windowsupdate/browse_thread/thread/7e9b87da3e40661f/3f8a4d26bc31f52b%233f8a4d26bc31f52b (Jan. 7, 2003, 15:16:52 GMT).
[199] *See* Scarcella v. America Online, No. 1168/04, 798 N.Y.S.2d 348 (Civ. Ct. Sept. 8, 2004), *aff'd*, 811 N.Y.S.2d 858 (App. Div. 2004). In *Scarcella,* a Civil Court refused to enforce a forum clause in part because "the sign-up process involved viewing over 80 computer 'screens' (91, to be exact . . .), two of which invited Claimant to consent to the terms of the Member Agreement." The court noted that customers can easily avoid reviewing the agreement, since "[t]he customer can bypass all that bother by simply pressing the 'OK, I Agree' button. If the customer nonetheless . . . presses the 'Read Now' button, Defendant affords him or her a second opportunity to skip over what will become the contract."
[200] *See* Specht v. Netscape Comm'cns Corp., 150 F. Supp. 2d 585, 594 (S.D.N.Y. 2001) (defining a browsewrap license). Some companies follow a hybrid approach, using a shrinkwrap license accompanied by posting the full terms and conditions on-line in a "browsewrap" format. *E.g.*, Dell, Inc., U.S. Terms and Conditions of Sale, http://www.dell.com/content/topics/global.aspx/policy/en/policy?c=us&l=en&s=gen&~section=012 (last visited Jan. 9, 2007).
[201] GPL, *supra* note 4, at § 2(c). As a test of Section 2(c)'s provisions, the author ran several common interactive, GPL-licensed Linux programs on a Web server running Linux version 2.6.9 (on file with author). Even interactive programs written by the GNU Project such as "find", "gcc", "bash", "gzip" and "grep" failed to display copyright notices during interactive operation or when passing the commands a "help" option. A few of the programs had a "version" option that displayed a copyright and a reference to "free software" but not the GPL, its license terms or where to find them.
[202] *E.g.*, Dell, *supra* note 200. The Dell link to terms and conditions of sale appears in small print at the bottom of each page amidst seven or eight other links about Dell and Dell's Web site. *Id.* However, buyers must agree to Dell's terms and conditions of sale through a clickwrap agreement prior to completing a purchase, addressing the concerns expressed by the court in *Specht*. *Id.*

manifest assent to it prior to downloading the software.[203] The court, while noting the acceptance of clickwrap licenses, required that the customer would (a) actually see the license agreement and (b) manifest actual assent.[204]

Browsewrap licenses present fewer challenges for the GPL in the context of acceptance by software developers than with unsophisticated users.[205] Professor Moglen argues that the "GPL only obliges you if you distribute software made from GPL'd code, and only needs to be accepted when redistribution occurs. And because no one can ever redistribute without a license, we can safely presume that anyone redistributing GPL'd software intended to accept the GPL."[206] The second sentence may overstate the motives of software developers in some contexts, especially since software developers may interpret the GPL differently than Professor Moglen and the FSF. A Linux software developer, though, would face considerable difficulty convincing a court that the developer was simply *unaware* of the existence of the license and its general terms, one of the central issues in *Specht*, given the significant publicity surrounding the GPL, Linux, and open source software generally.[207]

B. Forming a Contract Under the GPL

Of the three methods of software license acceptance described above, the GPL most closely resembles a browsewrap license. Although the GPL claims to require notification akin to a shrinkwrap license,[208] this does not happen in practice. A visit to SourceForge.net, a leading on-line repository for GPL-licensed and other open source licensed programs, shows that visitors can download every project without assent to, or even a review of, the accompanying license.[209]

[203] *Specht*, 150 F. Supp. 2d at 595-96.
[204] *Id.* at 595.
[205] This paper is primarily concerned with developers and the application of the GPL to works developed for the Linux operating system. Therefore, the more common public policy arguments against the enforcement of adhesion contracts against unsophisticated users carry less weight in this analysis. The concerns remain vitally important, if not somewhat muddled, though. For an in-depth discussion of adhesion contracts and the public policy problems presented by shrinkwrap, clickwrap, and browsewrap licenses for end users, *see generally* Andrew Burgess, *Consumer Adhesion Contracts and Unfair Terms: A Critique of Current Theory and a Suggestion*, 15 ANGLO-AM. L. REV. 255 (1986) (critiquing classical contract theory on contracts of adhesion, and proposing that adhesion contracts are only unfair when they contradict public policy); Robert Oakley, *Fairness in Electronic Contracting: Minimum Standards for Non-Negotiated Contracts*, 42 HOUS. L. REV. 1041 (2005).
[206] Eben Moglen, Enforcing the GNU GPL, Sept. 10, 2001, http://www.gnu.org/philosophy/enforcing-gpl.html (last visited Jan. 9, 2007).
[207] *Specht*, 150 F. Supp. 2d at 591 ("More specifically, [the court] must consider whether the web site gave Plaintiffs sufficient notice of the existence and terms of the License Agreement"). A Google search for "GPL AND enforceability" generated over 539,000 results. Google, http://www.google.com/search?hl=en&lr=&rls=com.microsoft%3Aen-us%3AIE-Address&q=GPL+AND+enforceability&btnG=Search (last visited onJan. 8, 2007). Indeed, the controversy about the GPL and Linux led directly to the author's decision to research and write about this topic in the first place.
[208] *See supra* note 201 and accompanying text (discussing and empirically testing the provisions of GPL § 2(c)).
[209] *E.g.*, SourceForge.net: Openads (aka Max Media Manager):Ad Server, http://sourceforge.net/projects/max (last visited on Jan. 8, 2007). Underneath a large green "Download" link box, the page lists the name of the license in small text along with other categories. *Id.* Following the large green "Download" link on the page takes visitors to SourceForge.net: Files – Max Media Manager: Open Source Ad Server, http://sourceforge.net/project/showfiles.php?group_id=103644 (last visited on Jan. 8, 2007). This page contains multiple direct download links, but no references to the GPL.

Even with conduct (distribution of GPL-licensed programs), notice appears lacking when applying the notice rule in *Specht* strictly.

Although U.S. courts are split on the validity of shrinkwrap, clickwrap, and browsewrap licenses, a few general rules have emerged that appear to reform the GPL's notice deficiency. The UCC and Restatement Second of Contracts also provide some relief for GPL enforcement against developers. The GPL should fall under the "usage of trade" definitions in both the Restatement and the UCC.[210] The UCC clearly endorses the conduct portion of the GPL's notice-plus-conduct model. Parties may form a contract "in any manner sufficient to show agreement, including conduct by parties recognizing the existence of such a contract."[211] Under a contractual interpretation of the GPL, courts consider the "regularity of observance in a place, vocation, or trade as to justify an expectation that it will be observed with respect to a particular agreement."[212] Courts will "liberally construe and apply to promote [the UCC's] underlying purposes and policies," which include "the continued expansion of commercial practices through custom, usage and agreement of the parties."[213]

The specific terms that a court will enforce in any single case depends on the facts surrounding the case, the course of performance of any prior agreements, and, most importantly, the usage of trade in the Linux community as it relates to the specific dispute. The notice-plus-conduct model may form a contract between parties who both know or reasonably should know some of the terms of the GPL beforehand. Courts may also find a contract where the parties both know or reasonably should know that the license exists. For example, the notice on SourceForge that a particular piece of software uses the GPL may be sufficient for a court to enforce those terms as an agreement.

Existing case law surrounding shrinkwrap and browsewrap licenses that use a notice-plus-conduct model suggests that courts would find that the GPL creates an enforceable contract, if a party challenged this point directly. The earliest significant case surrounding notice-plus-conduct licenses is *Step-Saver Data Systems, Inc. v. Wyse Technology*.[214] Although often criticized as rejecting shrinkwrap licenses,[215] the court did not reject the concept. The holding in the case turned instead on two peculiar facts. The plaintiff, a software reseller, sued for breach of warranty when the defendant's software failed.[216] The defendant's shrinkwrapped terms disclaimed all warranties, and the court considered whether the parties' agreement included that

[210] This paper concerns the modification, distribution, and derivation of GPL code by individual and corporate software developers and applies the UCC to any transactions between the parties. While some individual, non-merchant users may fall into this paper's scope of analysis, the vast majority of parties distributing GPL software are "merchants" dealing "in goods of the kind" in accordance with the UCC definition. U.C.C. § 2-104(1) (1995). Even an individual user writing for his own enjoyment may be one who "holds himself out as having knowledge or skill peculiar to the practices or goods involved in the transaction" and, consequently, falls under the "merchant" definition.
[211] U.C.C. § 2-204(1) (1995).
[212] Restatement (Second) of Contracts § 222(1) (1981). The revised UCC § 1-205 (2004) contains virtually identical language.
[213] U.C.C. § 1-102 (1995).
[214] 939 F.2d 91 (3d Cir. 1991).
[215] *E.g.*, I.Lan Sys. v. Netscout Serv. Level Corp., 183 F. Supp. 2d 328, 337 (D. Ma. 2002).
[216] *Step-Saver*, 939 F.2d at 91-92.

disclaimer.[217] The court accepted the notice-plus-conduct model, but rejected the disclaimer, finding that the parties had formed their contract before the plaintiff received the shrinkwrap license.[218]

The dispositive issue in the case was not the shrinkwrap license, which otherwise could have supplied the terms of the contract between the parties. The *Step-Saver* court did not dispute the potential validity of the license or "the existence of a contract, but the nature of its terms."[219] The court relied on two facts unique to this case. First, the court noted that the parties had agreed before making the order that Step-Saver could "transfer its copies [of the software] to the purchasers of the Step-Saver multi-user system" despite a contrary term on the shrinkwrap license.[220] The court also relied on evidence that, twice before, the defendant had attempted unsuccessfully to get Step-Saver to sign formal agreements containing warranty disclaimers.[221] Both facts illustrated that the parties expressly did not intend to agree on the shrinkwrap license terms.[222]

Returning to *ProCD, Inc. v. Zeidenberg*,[223] the Seventh Circuit held that the parties formed a contract despite the fact that the terms were contained inside a sealed box.[224] Purchasers had notice that ProCD placed additional terms in the package.[225] Although the court focused on the terms on the outside of the box, which the defendant agreed were part of the agreement, courts can find an analogy between the outside of the box in ProCD and the Linux environment. Developers working within the Linux community would face a high burden of proof to show that they had no notice that the GPL existed and applied to Linux. In any GPL contract formation issue, the usage of trade would supply the notice of additional terms, as it did in *ProCD*.[226]

---

[217] *Id.* at 96-97. Wyse Technology was one of two named defendants. *Id.* at 93-94. Wyse sold the software to Step-Saver for use on Wyse's WY-60 terminal. *Id.* The other defendant, The Software Link, Inc., manufactured the failed software. *Id. Accord* Arizona Retail Systems, Inc. v. Software Link, Inc., 831 F. Supp. 759, 766 (D. Ariz. 1993) (adopting the same holding as the *Step-Saver* court in a subsequent case alleging breach of warranty by The Software Link).

[218] *Step-Saver*, 939 F.2d at 95-96. Describing the process, the court states:

> Step-Saver would typically purchase copies of the program in the following manner. First, Step-Saver would telephone TSL and place an order. (Step-Saver would typically order twenty copies of the program at a time.) TSL would accept the order and promise, while on the telephone, to ship the goods promptly. After the telephone order, Step-Saver would send a purchase order, detailing the items to be purchased, their price, and shipping and payment terms. TSL would ship the order promptly, along with an invoice. The invoice would contain terms essentially identical with those on Step-Saver's purchase order: price, quantity, and shipping and payment terms. No reference was made during the telephone calls, or on either the purchase orders or the invoices with regard to a disclaimer of any warranties.

[219] *Id.* at 98.

[220] *Id.* at 103.

[221] *Id.* at 102-03.

[222] *Id.*

[223] 86 F.3d 1447 (7th Cir. 1996).

[224] *Id.* at 1450-51.

[225] *Id.*

[226] *Id.*

The Seventh Circuit affirmed its holding in *ProCD* only a year later in *Hill v. Gateway 2000, Inc.*[227] Unlike the package in *ProCD*, the computer that the plaintiffs, the Hills, purchased did not have a warning on its packaging about additional terms to follow delivery.[228] Although the Hills asserted at oral argument that the failure to include terms on the outside of the box distinguished the case from *ProCD*, the court termed the difference "functional, not legal."[229] The court dismissed the difference because ProCD used its box for retail display in a store; Gateway's box, "by contrast, [was] just a shipping carton . . . not on display anywhere. Its function is to protect the product during transit, and the information on its sides is for the use of handlers ("Fragile!" "This Side Up!") rather than would-be purchasers."[230] The court refused to require Gateway to place the notice on the outside of a box that a purchaser would immediately open and discard, thus implying that the Hills had notice after such actions. The court noted that the Hills would have "had a better argument if they were first alerted to the bundling of hardware and legal-ware after opening the box and wanted to return the computer in order to avoid disagreeable terms, but were dissuaded by the expense of shipping."[231]

The court also based its conclusion in part on Gateway's public statements. Gateway's advertisements did not specify the terms of their warranties or support, instead stating only "[Gateway's] products come with limited warranties and lifetime support."[232] The court consequently concluded that customers should know that computers from Gateway would arrive with additional legal terms.[233]

This portion of the court's holding is important for contract formation under the GPL. In the Linux community, the GPL acts both as the primary license of the Linux kernel and enjoys wide acceptance in related software projects.[234] Both Linux and its relationship to the GPL garner significant publicity. Advertisements, news stories, and notices on Web sites and in the source code itself sufficiently notify developers that additional terms exist.[235] As with the Hills, who chose not to ask for the additional terms in advance but to review them when the computer arrived, the GPL imposes a conduct requirement. "By keeping the computer beyond 30 days [the time specified in the shrinkwrap] the Hills accepted Gateway's offer, including the arbitration clause" at issue in the case.[236] Similarly, a developer by "modifying or distributing" GPL code indicates "acceptance of [the] License."[237]

[227] 105 F.3d 1147 (7th Cir. 1997).
[228] *Id.* at 1150.
[229] *Id.*
[230] *Id.*
[231] *Id.*
[232] *Id.*
[233] *Id.*
[234] *See supra* notes 67-68 and accompanying text.
[235] *See, e.g.*, Open Source Technology Group (OSTG), SourceForge.net: VRRP - Virtual Router Redundancy Protocol, https://sourceforge.net/projects/vrrpd/ (last visited Feb. 16, 2007) (Web site listing the GPL as the governing license for the VRRP project); Linux Cross Reference, Linux-2.6.17/drivers/net/eepro.c, http://www.gelato.unsw.edu.au/lxr/source/drivers/net/eepro.c (last visited Feb. 16, 2007) (source code file for a common network hardware driver listing the GPL as the governing license).
[236] *Hill*, 105 F.3d at 1150.
[237] GPL, *supra* note 4, at § 5.

In light of the wide usage of the GPL by Linux developers, distribution of GPL software, whether modified or unmodified, standalone or as part of a compilation, allows courts to readily apply this common rule. A licensor can usually establish conduct, since parties often learn of infringement only after actual distribution or publication of potentially infringing code. The key is notice. If a licensee understands (or reasonably should understand) that a GPL licensor will deal only on terms in the GPL, regardless of whether the license is provided at the time of agreement, then the portions of the GPL that form the common understanding of the parties govern their agreement. The definitions of terms contained in the GPL and the interpretation of copyright law are less clear.[238] Ambiguous terms will likely force courts to state contract law for interpretations.

However, as long as the requirement of the GPL is clear to both licensor and licensee before contract formation, then the notice-plus-conduct model contemplated by the GPL operates successfully despite the lack of formal notice in practice. Courts will likely hold Linux developers to the same standard as parties who receive printed forms and choose not to read them. The notice-plus-conduct model binds the Linux developer to the GPL, regardless of the interpretation of its terms, just as form contracts bind those agreeing to the substance of a purchase of concert tickets at Ticketmaster,[239] a visit to CNET's download.com,[240] or a purchase of flight itineraries from United Airlines.[241] Although none of the parties know the exact, relevant terms, they all agree to proceed on that basis.

Where the GPL may not result in contract formation with end users, the failure does not thwart its goals. The GPL intends to restrict licensees from expropriating GPL-licensed code and distributing it. For developers undertaking the conduct described by the GPL, however, courts will not enforce all of the terms of the GPL automatically. While conduct indicates that the developer intends to agree to the general basis of the GPL, developers may interpret GPL provisions differently, leaving agreement perhaps only on the broad type of transaction (one involving GPL code), but nothing else. With a contract formed, courts must next turn to the obligations and definitions in the GPL text.

C. Rights and Obligations in the GPL

Regardless of the interpretation of the GPL as a license or as a contract, the text of the GPL helps to determine its enforceability. As described in Part I, Stallman rewrote the GPL from the GNU Emacs General Public License as a template license. Rather than creating individual licenses for each new GNU program, the GPL can apply to any program, provided a licensor incorporates the terms of the GPL by notice in the software.[242] The GPL opens with a Preamble designed to highlight its intended audience. [243] While Stallman intended the GPL to act as a legal

[238] This paper deals with these subjects in Parts II.E. and III, *infra*, respectively.
[239] Ticketmaster, *supra* note 182.
[240] CNET Networks, *supra* note 183.
[241] United Airlines, *supra* note 184. *See also* Carnival Cruise Lines, Inc. v. Shute, 499 U.S. 585 (1991) (enforcing a forum selection clause included in three pages of terms attached to a cruise ship ticket).
[242] *See* GPL, *supra* note 4 (describing how to apply the GPL to programs in a short explanation that accompanies the GPL text).
[243] *Id.*

document, the absence of legalese suggest that he and the FSF wrote the Preamble with a lay audience in mind. Without using Stallman's "copyleft" term,[244] the Preamble describes its concept and the primary purposes of the GPL.[245] Interestingly, the copyright notice preceding the Preamble forbids modifications of any type to the license: "Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed."[246] While the GPL encourages the free distribution of GPL-licensed code and derivative works, the license prohibits authors from creating derivative works from the text of the GPL itself.

First, and most importantly, the GPL is "designed to make sure that you have the freedom to distribute copies of free software . . . that you receive source code or can get it . . . [and] that you can change the software or use pieces of it in new free programs."[247] While the GPL guarantees these rights for the licensee, it imposes the same restrictions on parties when they, in turn, become licensors instead of licensees. Stallman's "copyleft" idea, retains copyright in the software, but requires licensees and licensors to share GPL-licensed code freely. The Preamble specifically refers to Stallman and the FSF's purposes, noting, "[t]o protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender these rights."[248] The Preamble informs licensees that the purpose of the GPL is, in part, to ensure "that you know you can do these things" described in the license.[249]

The second purpose of the GPL outlined in its Preamble specifically concerns warranty and patent issues. Couched in terms of protecting the original author's reputation, the Preamble "make[s] certain that everyone understands that there is no warranty for this free software."[250] The last full paragraph of the Preamble repeats the concerns Stallman had for software patents, referring to GPL Section 7's termination provisions for patented software.[251] The disclaimer of warranty, mentioned in the Preamble and discussed in Part II.F herein, appears in many proprietary licenses and other open source licenses, but the license termination in GPL Section 7 is a unique creation.

Sections 1 through 3 of the GPL grant to licensees rights reserved for the copyright holder under the Copyright Act and address rights not contained in the Act. The Act reserves for copyright holders "the exclusive rights to do and to authorize"[252] any party to "reproduce the

---

[244] WILLIAMS, *supra* note 9, at http://www.faifzilla.org/ch09.html (last visited Jan 11, 2007). Williams describes the origin of the term:

> [A] California hacker named Don Hopkins mailed him a manual for the 68000 microprocessor. Hopkins, a Unix hacker and fellow science-fiction buff, had borrowed the manual from Stallman a while earlier. As a display of gratitude, Hopkins decorated the return envelope with a number of stickers obtained at a local science-fiction convention. One sticker in particular caught Stallman's eye. It read, "Copyleft (L), All Rights Reversed." Following the release of the first version of GPL, Stallman paid tribute to the sticker, nicknaming the free software license "Copyleft."

[245] GPL, *supra* note 4, at Preamble.
[246] GPL, *supra* note 4.
[247] *Id.* at Preamble.
[248] *Id.*
[249] *Id.*
[250] *Id.*
[251] *Id.*
[252] 17 U.S.C. § 106 (2000).

copyrighted work,"[253] to "distribute copies,"[254] and to "prepare derivative works."[255] The GPL reserves these same rights: "You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium . . . ,"[256] "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"[257] and "copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above."[258]

Section 2 embodies Stallman's "copyleft" idea, granting licensees the licensor's statutory rights to copy, distribute, and reproduce works or derivative works, provided downstream licensees receive the same exact rights when licensees reproduce and redistribute works.[259] The GPL not only attempts to grant rights to licensees, but also eliminates the potential for opportunistic developers to expropriate source code, deploy it in their own works, and license those works under non-GPL licenses. Section 4 reinforces the grant under Section 2 by ensuring that only the GPL applies to licensed software:

> You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.[260]

Section 4 adds an enforcement mechanism to the earlier sections' grants of rights. Licensees that distribute infringing works, commandeer licensed code for use in proprietary-licensed works, or infringe the GPL in other ways automatically lose their rights under the GPL.[261] Presumably, such a violation would leave a putative licensee with only those rights granted under the Act (essentially none, when compared with those that the GPL provides).

Section 4 also contains a protection for parties downstream of infringing licensees. When an upstream licensee loses a license grant under the GPL, Section 4 protects licensees who received GPL-licensed works from any infringement actions.[262] This grant accompanies Section 6 of the GPL, which attempts to address any lack of privity issues. Section 6 creates "a license from the original licensor to copy, distribute or modify the Program" and grants it to each downstream licensee.[263]

---

[253] *Id.* at § 106(1).
[254] *Id.* at § 106(3).
[255] *Id.* at § 106(2).
[256] GPL, *supra* note 4, at § 1.
[257] *Id.* at § 2.
[258] *Id.* at § 3.
[259] *See id.* at § 2.
[260] *Id.* at § 4.
[261] *Id.*
[262] *Id.*
[263] *Id.* at § 6.

Sections 1 through 3 also grant access to "verbatim copies of the Program's source code."[264] Section 3 defines

> source code for a work [to] mean the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable.[265]

The reference in Section 1 to "verbatim copies" in a "preferred form" and the extended definition of source code to include "associated interface definition files, plus . . . scripts" attempts to guard against potentially abusive practices by licensees. A narrower definition of source code would enable licensees to obfuscate source code and evade the purposes of the GPL while remaining in technical compliance. Without these specific inclusions, a licensee could release source code conglomerated into a single file, or exclude header definitions, makefiles, and other configurations necessary to compile a binary version of the Program.[266] When Linksys originally released source code used on their wireless routers, the company did not include its modifications to the GPL code used on the routers until after the Linux and open source communities applied significant pressure on the company to do so.[267] Absent the safeguards in Sections 1 and 3, Linksys could have justifiably excluded their modifications.[268]

Section 3, though, does not require licensees to distribute all related GPL source code in every situation:

> However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.[269]

This logical exclusion allows licensees to distribute source code for the entire Linux operating system along with a single program solely because the licensee wrote the program to work with Linux. The section continues by obligating distributors of derivative works to accompany the work "with a written offer . . . to give any third party" the source code to the

[264] *Id.* at § 1.
[265] *Id.* at § 3.
[266] Programmers use "headers" or "header files" (common names referring to the same type of file, often ending in ".h") to describe elements of the main source code, including variables, functions or other commonly referenced items. A "makefile" is a set of instructions for the operating systems' automated compiler. The makefile provides detailed information on sets of files, headers, libraries, and source code and how to combine these files and programs together to create executable programs.
[267] *See supra* note 70 and accompanying text.
[268] Arguably, the protections discussed here may not go far enough to provide functional source code to users. As a commenter to the blog post in note 70, *supra*, noted, Linksys could release GPL code that only compiles when using a non-GPL licensed compiler. Without the compiler, the source code would be useless, since users could not use it to create functional binaries. Linksys' original source code release also may not have violated the spirit of the GPL, even if it violated its terms. The company's initial reluctance to release proprietary modifications to GPL code did nothing to prevent the open source community from obtaining the original, unmodified code from either Linksys or its original authors or maintainers.
[269] GPL, *supra* note 4, at § 3.

derivative work, but only for a minimum of three years, though parties may make a longer offer.[270]

Section 2 modifies this obligation to release source code to conform to Stallman's view that software should be free.[271] Section 2 requires "any work that you distribute or publish" that is covered by the GPL must "be licensed as a whole at no charge to all third parties under the terms of [the GPL]."[272] In addition to requiring parties to license any works covered by the GPL solely under the GPL, closing another potential loophole for opportunistic developers, this section effectively limits what parties can charge to the costs of physical media, as allowed by Section 1. Creative accountants could assign labor costs, portions of fixed overhead expenses, amortization of goodwill and numerous other charges to the "costs" of producing physical media with GPL software. However, the requirement to provide "verbatim copies of the Program's source code" allows interested users to provide their own physical media and eliminate any assigned "costs." Without any pricing power in the market for GPL software, developers cannot sustain prices above the marginal cost of producing physical media.

Finally, the GPL recognizes the importance of awareness and notification. Throughout the license, the GPL reminds those who copy and distribute Programs, or derivative works, to make others aware of copyrights and the GPL. Section 1 grants rights, but only when licensees "conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program."[273]

Section 2 applies this principle to "works based on the Program," requiring licensees to "copy and distribute such modifications or work under the terms of Section 1 above."[274] Section 2 extends this notification provision by requiring that "modified files . . . carry prominent notices stating that you changed the files and the date of any change."[275] This non-attribution provision mirrors those found in myriad other open source licenses.[276] Section 2(a) protects downstream licensees by ensuring they know that the Program received is not the original work, and that the Program provides information about both the original authors and the authors responsible for the derivations.

Section 2(c) attempts to implement an automated notification as a bulwark against an author's failure to notify users. The section requires licensees to "cause [the Program], when

---

[270] *Id.*
[271] *See supra* text accompanying note 18.
[272] GPL, *supra* note 4, at § 2.
[273] *Id.* at § 1.
[274] *Id.* at § 2.
[275] *Id.* at § 2(a).
[276] *E.g.*, Open Source Initiative (OSI) - The MIT License, http://www.opensource.org/licenses/mit-license.php (last visited Jan. 9, 2007) ("The above copyright notice and this permission notice shall be included . . . ."); OSI, BSD License, http://www.opensource.org/licenses/bsd-license.php (last visited Jan. 9, 2007) ("Redistributions of source code must retain the above copyright notice . . . ." and "Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software . . . .").

started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice,"[277] Stallman and the FSF were prescient to include a provision requiring a Program to output the notice, because download sites and authors routinely fail to provide adequate notification.[278] Again, this provision illustrates the GPL authors' intent to ensure that downstream licensees receive adequate notice of the GPL and an opportunity to review its terms. As with the licensor-to-licensee notice requirements detailed in Section 1 and Section 2, authors, including the FSF itself, routinely ignore the provision in Section 2(c) requiring automated notice.[279]

D. Divining Intent: Defining "works based on the Program" and Other Terms

In addition to general rights and obligations, acceptance provisions, and statements of Stallman and the FSF's philosophies about free software, the GPL contains a number of important, albeit ambiguous, definitions of its terms. Section 0 of the GPL defines key terms and outlines the scope of the license itself (in relevant part):

> 0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) . . . .[280]

The FSF's principles contained in its Free Software Definition animate the GPL.[281] The Free Software Definition relies primarily on the term "software."[282] The GPL, however, uses the term "program" more closely associated with the Copyright Act's "computer program." 17 U.S.C. §101 defines computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[283] The term "program" in the GPL falls within this definition. The GPL defines the term "Program" in part as "any program or other work."[284] The GPL never defines Section 0's phrase "other work," presumably referring to the Act's definitions. The Act defines more than a dozen variations on the term "work."[285] Given

[277] GPL, *supra* note 4, at § 2(c).
[278] *See supra* note 209.
[279] *See supra* note 201 and accompanying text.
[280] GPL, *supra* note 4, at § 0.
[281] Free Software Foundation, The Free Software Definition – GNU Project, http://www.gnu.org/philosophy/free-sw.html (last visited Jan.9, 2007).
[282] *Id.* The FSF definition does use the term "program" in a few places interchangeably with the term "software," for example: "The freedom to run the program . . . The freedom to study how the program works . . . The freedom to improve the program . . . ." *Id.* The use is sparse and seemingly used to delineate the difference between "free software" as a subset of the larger "programs."
[283] 17 U.S.C. § 101 (2000). References herein to "the Act" refer to the Copyright Act of 1976, as amended to include computer programs in the category of literary works by the Copyright Act of 1980.
[284] GPL, *supra* note 4, at § 0.
[285] 17 U.S.C. § 101 (2000). Among the examples relevant to a discussion of software, the Act lists a "translation . . . abridgement, condensation" and "editorial revisions, annotations, elaborations" in addition to the broader "other modifications" definition.

that the GPL licenses software and purports to control future works related in some way to that software, the definition of "other work" under the GPL would likely include only the Act's definitions of "collective work," "compilation," "derivative work," and "joint work." To this point, the GPL's definitions align nicely with the Copyright Act definitions.

Section 0 continues, though, by defining a "work based on the Program" as "either the Program or any derivative work under copyright law: that is to say, a work containing the Program or another portion of it . . . ."[286] This definition muddles the Copyright Act definitions of collective and derivative works. Under the Copyright Act, a work merely containing the Program would likely fall into the concept of aggregation embodied in the definition of "collective work."[287] The Act defines a collective work as ". . . a work . . . in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole."[288] A work that incorporates and modifies the entire Program or some portion of it would fall under the definition of a "derivative work." A derivative work according to the Act is ". . . a work based upon one or more preexisting works, such as a translation . . . or any other form in which a work may be recast, transformed, or adapted."[289] The GPL definition incorrectly tries to incorporate both concepts into the single phrase "based on the program."

Section 0 elaborates on its definition of a "work based on the Program" by including verbatim copies, modified versions, or translated versions of the Program.[290] Here, the GPL again fails to make the distinction between a work containing the Program and a work based on the Program, or collective and derivative works as Congress defined them under the Act. To combine these terms into a single all-encompassing definition is illogical, especially given the GPL's reference in the same sentence to copyright law, and the importance of those terms of art under the Act. The definition, however, is critical in defining the reach of the GPL to programs that run under or work with Linux without incorporating GPL code.

The first paragraph of Section 0 is one of several places where the GPL refers to a "work based on the Program." In the second paragraph of Section 0, the GPL clarifies the definition slightly in response to concerns from developers using GPL-licensed editors and compilers.[291] The GPL applies to "Output from the Program is covered only if its contents constitute a work based on the Program . . . ."[292] In short, merely using an editor or program compiler licensed under the GPL does not trigger GPL coverage of the created work by itself. This reference does not help to explain the GPL's apparent combination of collective and derivative works in Section 0.

---

[286] GPL, *supra* note 4, at § 0.
[287] The Debian Linux distribution's repository of GPL and non-GPL software is one example of a collective work. The originality embodied in the decisions about which programs to include in the collection qualifies for copyright protection. The collection does not affect the underlying copyrights in the collected works. *See infra* notes 301-302 and accompanying text.
[288] 17 U.S.C. § 101.
[289] *Id.*
[290] GPL, *supra* note 4, at § 0.
[291] Free Software Foundation, *supra* note 79.
[292] GPL, *supra* note 4, at § 0.

The GPL turns to a "work based on the Program" again in Section 2. The second full paragraph attempts to clarify how the GPL applies to different types of works:

> These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.[293]

The first sentence applies the GPL to a "modified work as a whole." The Act refers to "work[s] consisting of . . . modifications" representing an "original work of authorship" in the context of derivative works.[294] The second sentence above, therefore, applies to non-derivative works and echoes the Act's intent in granting rights to creators of works. The third sentence addresses the independent works when distributed "as part of a whole" and, alternately, to "each and every part regardless of who wrote it." This sentence does not clarify what a "work based on the Program" is, but establishes that the distribution of a derivative work requires a party to have a license "to the entire whole . . . regardless of who wrote it."[295] This sentence also follows the Act's grant of exclusive rights of reproduction, distribution, and the creation of derivative works to copyright holders.[296]

The next paragraph in Section 2 asserts another intention directly contradictory to the Act by expressing that its "intent is to exercise the right to control the distribution of derivative or collective works based on the Program."[297] The Act does not grant rights to works as broadly as the GPL does. The Act restricts the right to control its grant of rights "to the material contributed by the author of such work," noting that a copyright in derivative works or a compilation "does not affect or enlarge the scope, duration, ownership, or subsistence of, any copyright protection in the preexisting material."[298] While the GPL may intend to exercise certain rights, the Act does not support textually such a broad grant or interpretation.

Section 2 concludes by exempting collective works from the GPL, stating "mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License."[299] This last sentence restricts the GPL's application solely to "works based on the Program," without reconciling the differences between this GPL definition and the Act's definition of derivative works.

[293] *Id.* at § 2.
[294] 17 U.S.C. § 101.
[295] GPL, *supra* note 4, at § 2.
[296] § 106.
[297] GPL, *supra* note 4, at § 2.
[298] 17 U.S.C. § 103(b).
[299] GPL, *supra* note 4, at § 2.

Commentators have noted that, notwithstanding any arguments against construing the GPL according to its stated intent, licensees can easily avoid the requirements of Section 2 by separating the distribution of GPL and non-GPL tools.[300] Debian, one of the major Linux versions, maintains both "free" (fully GPL-licensed) and "contrib" and "non-free" (non-GPL licensed) archives.[301] Debian maintains that the "non-free works are not a part of Debian."[302]

The GPL could redefine derivative works from the narrower definition in the Act as part of a mutual contract between licensor and licensee. The GPL's vague but apparently broader definition and its conflict with the Act creates potential problems for the open source community and licensees. Acknowledging the split in the definitions could potentially leave the Linux community in full-scale war over which software users can license, which they cannot, and how to enforce the GPL regardless of either side's position on the definitions. This paper will return to the problem of derivative works in Part III.

E. The GPL's Limits: Enforceability, Patent Protection, and Warranties

Not every provision of the GPL grants rights or delineates obligations. Stallman and the FSF devote nearly half of the GPL's sections to areas that the license does not address: enforceability (Sections 7 and 8), patent protection (Section 7) and warranties (Section 11 and 12).[303] The GPL's focus on the policy of "free"[304] software requires the inclusion of sections protecting that policy from revision by courts, licensees, or patent holders.

The longest of the five limiting sections, Section 7, attempts to address most of these issues at once. Understanding Section 7 requires breaking down each paragraph into its several parts. Section 7 opens by stating its ultimate purpose. The section attempts to prevent outside acts, including contracts between individual parties, litigation settlements, claims of infringement, court decisions, and any other reason, from altering the GPL's provisions:

> If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.[305]

This passage leaves much of its interpretation up to licensees and, ultimately, courts. The GPL does not attempt to define consequences of a court judgment, or the scope of pertinent obligations. Professor Moglen explains that this provision "ensures that other rules outside of the

[300] ROSEN, *supra* note 85, at 118.
[301] Debian Social Contract – Social Contract with the Free Software Community, http://www.debian.org/social_contract (last visited Jan. 9, 2007).
[302] *Id.*
[303] *See* GPL, *supra* note 4.
[304] Again, the word "free" refers to "freedom" rather than price. *See supra* note 66.
[305] GPL, *supra* note 4, at § 7.

direct software license cannot take rights away from users, distributors, and modifiers of GPL'd software."[306]

Section 7 then provides strong incentive for licensees to consider the GPL's provisions carefully before proceeding. The section warns, "if you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all."[307] This provision, construed in the way the FSF intends, could make it impossible for a GPL licensee to retain rights under the GPL while maintaining rights under other agreements or compliance with other obligations. For example, consider a licensee that wants to incorporate GPL code into its software product that also contains licensed code from a chipset vendor whose license prohibits disclosure of the chipset code. Since the licensee cannot comply simultaneously with the GPL's obligation to release source and the chipset vendor's non-disclosure obligation, Section 7 would preclude licensees from distributing the Program at all. Section 7 restricts its effects to distribution of the code, leaving the licensee free to modify the code for its own internal use. For companies that produce products for sale, though, the loss of distribution rights essentially amounts to a loss of all significant rights.

The section also contains a limited severability clause that attempts to save the section should courts rewrite or invalidate portions of it: "If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances."[308]

Although no court has addressed this specific section of the GPL, neither sentence would have much, if any, effect. The first sentence uses a simplified version of common contract language, however, if a court strikes the critical first paragraph, then not much of the section remains to be enforced. Courts ultimately have discretion to interpret agreements and fashion remedies, regardless of the intent of the drafters.[309] While courts would hear arguments from licensors, licensees, the FSF, and others in the open source community about preserving the integrity of the license, courts would hesitate to allow the GPL to unilaterally limit its available options for remedy. The second sentence goes further in trying to avoid any precedential effects of partial or entire invalidations of Section 7. The GPL leaves "other circumstances" undefined, but courts could distinguish cases or establish precedent without guidance from the GPL.

Section 7 finishes with a paragraph explaining that the section's provisions are "a consequence of the rest of this License" and an attempt at "protecting the integrity of the free software distribution system, which is implemented by public license practices."[310] Although stated broadly, the provision likely refers to protecting the GPL and its model. Other "public

[306] Eben Moglen, Remarks at the Free Software Foundation Seminar: Detailed Study and Analysis of the GPL and LGPL (Jan. 20, 2004) (summary available at http://gnucvs.vlsm.org/licenses/200104_seminar.html) (last visited Jan. 9, 2007).
[307] GPL, *supra* note 4, at § 7.
[308] *Id.*
[309] Restatement (Second) of Contracts § 206 (1981) states the common law rule that ambiguity should be construed against the drafter, a potentially problematic rule for enforcement of the GPL's ambiguous terms.
[310] GPL, *supra* note 4, at § 7.

licenses" such as the BSD License,[311] the Mozilla Public License,[312] and the Apache License[313] do not contain these clauses. The Apache License, for example, contains an entirely different patent protection clause that limits the effect on derivative works and downstream licensees without completely invalidating the license as the GPL does in Section 7.[314] The GPL model zealously protects free software, and Stallman and the FSF intend the section to preserve the GPL, even if the result is to remove software from the open source community that licensees would happily pay to license. Even if a court chose to refashion the GPL to include paid patent licensing options, the community would reject using the code and would rewrite offending portions to protect the integrity of the GPL.[315] Linux International President John "maddog" Hall summarizes the community's position on infringing code nicely: "Take your code, please! We don't want it."[316]

Sections 8 and 10 act as companions to Section 7. In the GPL's only reference to non-U.S. law and one of two sections that allows licensors to modify the GPL, Section 8 allows licensors to grant a license with "an explicit geographical distribution limitation excluding" countries where "distribution and/or use of the Program is restricted . . . by patents or by copyrighted interfaces."[317] Given the goal of wide, free distribution of GPL software, the GPL drafters made a sensible concession. In Section 10, the license again uses explanatory language rather than a binding provision. Here, the section instructs licensees on remedies for the GPL's incompatibility with other licenses, directing them to "write to the author to ask for permission" to incorporate "the Program into other free programs whose distribution conditions are different."[318] The section notes that the FSF "sometimes make[s] exceptions" for its own software in the interests of promoting free software development and distribution.[319]

Sections 11 and 12 contain warranty disclaimer language common in many contracts. However, the blanket disclaimers may conflict with some non-U.S. jurisdictions' proscriptions on warranty disclaimers. The sections, written in all capital letters and headed with a label "NO WARRANTY," attempt to disclaim any warranties for the software. Section 11 disclaims warranties "of any kind, either expressed or implied, including, but not limited to, the implied

---

[311] OSI, BSD License, *supra* note 276.
[312] Mozilla Foundation, Mozilla Public License version 1.1, http://www.mozilla.org/MPL/MPL-1.1.html (last visitedJan. 9, 2007).
[313] Apache Software Foundation, Apache License Version 2.0, http://www.apache.org/licenses/LICENSE-2.0.html (last visited Jan. 9, 2007).
[314] *Id.* at § 3. For a full discussion of the Apache Software Foundation's approach to patent infringement, how the Apache Software Foundation has interpreted the GPL and a detailed treatment of the dispute between the Apache Software Foundation and the Free Software Foundation that, as of this writing, renders the Apache License, v2.0, as incompatible with the GPL, *see generally* Apache Software Foundation, Apache License v2.0 and GPL Compatibility, http://www.apache.org/licenses/GPL-compatibility.html (last visited Jan. 9, 2007).
[315] *See* Pamela Jones et al, *Digging for Truth - Research in support of an Open Source/Free Software Community's Reply to Darl McBride*, THE INQUIRER, Sept. 19, 2003, http://www.theinquirer.net/default.aspx?article=11649 (last visited Jan. 9, 2007) (discussing SCO's suit against IBM for copyright infringement and offers by Torvalds, Stallman, Moglen, and others to remove infringing code).
[316] Jones, *supra*, note 86 (citing Remarks of John Hall at 2003 Usenix Conference, Jun. 12, 2003).
[317] GPL, *supra* note 4, at § 8.
[318] *Id.* at § 10.
[319] *Id.*

warranties of merchantability and fitness for a particular purpose."[320] Section 12 disclaims liability "unless required by applicable law" for any "general, special, incidental or consequential damages."[321]

The GPL fails to include a choice of law provision in any section and does not attempt to state positively what warranty and remedial rights a licensee retains under the license. As a result, courts in non-U.S. jurisdictions may hesitate to apply these sections and attempt to hold developers and distributors of GPL licensed programs to standards of liability in their local jurisdictions. In Germany, the disclaimers in Sections 11 and 12 "are generally considered to be invalid under German law."[322] Under the French Code de la Consommation and Civil Code, merchantability disclaimers are invalid when asserted against non-merchants.[323] Under Australian law, "an exclusion of implied conditions and warranties may not be effective in Australia, even when the governing law of the open source license is a State of the USA or some other place."[324] To the apparently great extent that provisions in foreign codes prevent warranty and liability disclaimers only as applied to consumers and not merchants, the status of Sections 11 and 12 are irrelevant here.[325]

In its only other official section, Section 9, the GPL provides notice that the FSF may update or change the license. As of this writing, the FSF has developed several drafts of version 3 of the GPL, due for release in early 2007 or whenever the FSF finalizes the terms of the revised license.[326] Public statements by Linux kernel developers and other GPL licensors, and the text of GPL version 2, make application of future versions and notice requirements more difficult.[327] The proposed GPL version 3 and its effect both on GPL version 2 and the issues raised here

[320] *Id.* at § 11.
[321] *Id.* at § 12. Author and lawyer Andrew St. Laurent notes that Section 12 fails to disclaim liability for direct damages. Laurent suggests that this could force distributors to remedy defective media, or it could simply be an oversight. Courts could read the list of damages as "illustrative, not definitive." ANDREW ST. LAURENT, UNDERSTANDING OPEN SOURCE & FREE SOFTWARE LICENSING 48 (2004).
[322] Julian Höppner, *The GPL prevails: An analysis of the first-ever Court decision on the validity and effectivity of the GPL*, 1 SCRIPT-ED 628 (2004), *available at* http://www.law.ed.ac.uk/ahrb/script-ed/issue4/GPL-case.pdf (last visited Jan 9, 2007) (citing an expert report prepared for VSI, the German software industry association). The German Institute for Legal Issues on Free and Open Source Software created the German Free Software License in 2004, http://www.dipp.nrw.de/d-fsl/index_html/lizenzen/en/D-FSL-1_0_en.txt (last visited Jan. 9, 2007), as a GPL-compatible license that conformed to German law.
[323] C. CONS. art. L.132-1 (Fr.) (forbidding warranty disclaimers against consumers and non-merchants); C. CIV. art. 1386-15 (Fr.) (forbidding warranty disclaimers against consumers and non-merchants in suits for damages caused by a defective products). To address the shortcomings of the GPL, three public research foundations in France have released the free software licensing agreement called "CeCILL" (an acronym combining the names of the three organizations), http://www.cecill.info/licences/Licence_CeCILL_V1.1-US.txt (last visited Jan. 9, 2007), to address "conformity with French law" along with the free software principles of the FSF. CeCILL claims to be compliant with French tort and intellectual property law.
[324] Peter C.J. James, *Open Source Software: An Australian Perspective*, LEGAL ISSUES RELATING TO FREE AND OPEN SOURCE SOFTWARE 63, 76-80 (Brian Fitzgerald & Graham Bassett eds., 2003).
[325] However, even under the UCC, Sections 11 and 12 may present problems. U.C.C. §2-316 (1995) allows the exclusion or modification of warranties, but does not specifically address whether this grant applies only to contracts or to both contracts and licenses. Software licenses may further complicate any interpretation.
[326] Free Software Foundation, GPLv3 2nd Discussion Draft, Jul. 27, 2006, http://gplv3.fsf.org/draft (last visited Jan. 9, 2007).
[327] *See infra* Part IV.C.

merit more significant treatment than this overview of terms. The paper undertakes a discussion of the potential effects of GPL version 3 in Part IV.

II. The Linux Kernel Module Problem

A. The Unavoidable Technical Discussion

For purposes of examining the GPL and Copyright Act, Linux has three different types of software: standalone applications running in kernel space or user space,[328] the Linux kernel itself, and kernel modules that use the "system call interface" described later in this section. Each of these types of programs presents a potentially different treatment under the GPL and Copyright Act. Of the three, the GPL and Copyright Act speak clearly about the first two.

Standalone applications do not fall under the definition of a "derivative work," regardless of whether the applications run in user or kernel space.[329] Provided the developer did not use GPL-licensed code to create it, no application merely running on an operating system can constitute a derivative work of that operating system. An entire industry has grown up around proprietary applications that run on hundreds of operating systems, including Linux.

The Linux kernel developers have licensed the Linux kernel code under the GPL.[330] Developers that directly modify this source code fall on the opposite end of the spectrum from developers of standalone applications. Developers that acquire the Linux source code, modify it, and compile it to form a new kernel, have created a derivative work of Linux. The GPL requires that such modifications also use the GPL.[331]

The third category creates significant debate. With the ambiguous definitions in the GPL,[332] and rather paltry protections provided to software under the Act,[333] this kernel module code likely falls outside of the definition of "derivative works." To this point, this paper has carefully avoided technical discussions about kernel modules, program compilation, static and dynamic linking, and a host of other Linux and programming-specific topics. In order to understand fully the debate about the applicability of the GPL under the Copyright Act (and to

[328] For background information on the difference between the kernel and user space, *see generally* User Space, WIKIPEDIA: THE FREE ENCYCLOPEDIA, http://en.wikipedia.org/wiki/User_space (last visited Jan. 9, 2007). The article describes the difference between "kernel space" and "user space" this way: An operating system usually segregates the available system memory into kernel space and user space. The kernel space is strictly reserved for running the kernel, device drivers and any kernel extensions. This kernel space is usually never swapped out to the disk. On the other hand user space is the memory area used by all user mode applications and this memory can be swapped (paged) out at any time depending upon the current system requirement.
[329] *See supra* text accompanying note 283 (discussing the Copyright Act definition of derivative works).
[330] *See supra* note 53.
[331] GPL, *supra* note 4, at § 2.
[332] See Part II E, *infra.*
[333] See Part III, *infra.*

highlight the relative simplicity of the proposed solution in Part IV.), this section proceeds first with an unavoidable, but brief, technical discussion.[334]

Linux, like other UNIX and UNIX-like operating systems and even Microsoft Windows, runs concurrent "processes," programs and sub-programs, handling different tasks under the operating system.[335] All of the processes use system resources: system memory (RAM), processing power measured by the amount and percentage of the processor's available processing time used, input/output to physical media such as hard drives or CD-ROMs, network connections (bandwidth), etc.[336] The Linux kernel is the core that allocates these resources to individual processes as requested.[337] Rubini & Corbet[338] use a helpful diagram to illustrate the basic structure of the Linux kernel:

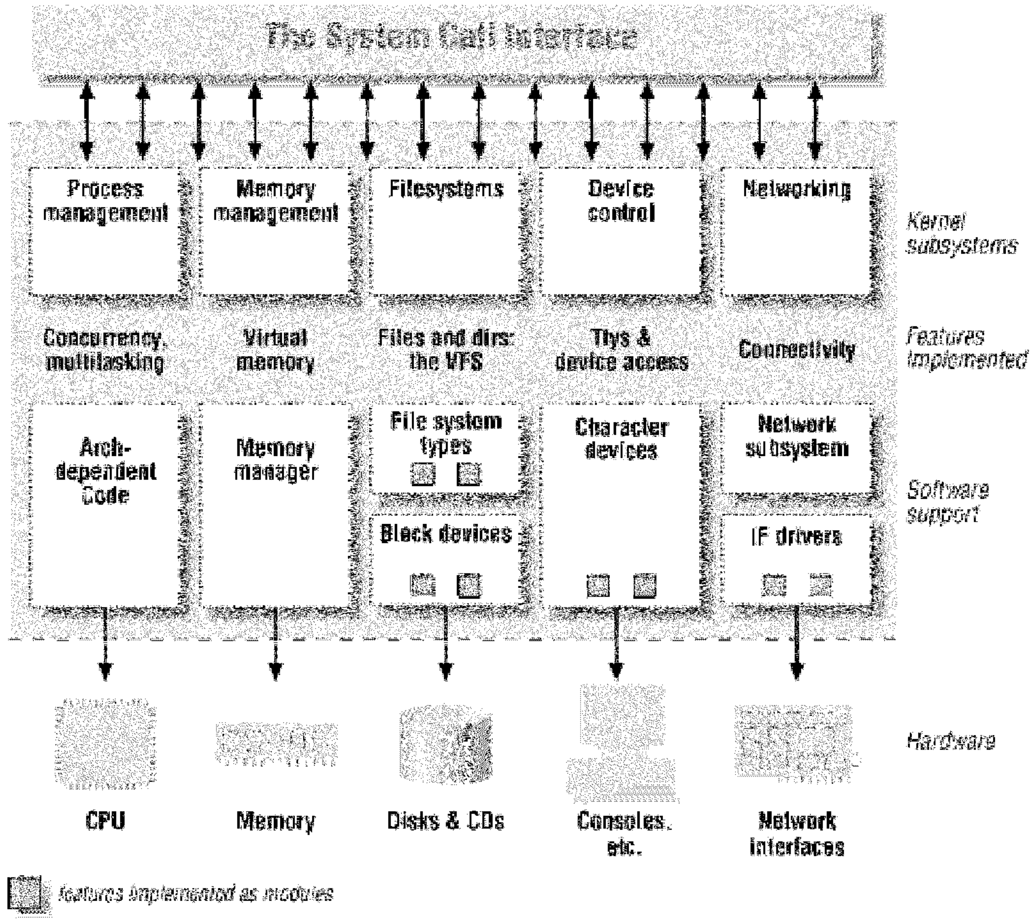
[334] This section includes only the barest details necessary to understand the issues. While some technical language is unavoidable, this section attempts to present the concepts in as non-technical a manner as possible. As such, some of the actual concepts described in this section may have more technical nuance than this paper conveys. For a more in-depth explanation of kernel modules and device drivers, *see* Linux Information Project *supra* note 77; *see generally* ALLESSANDRO RUBINI & JONATHAN CORBET, LINUX DEVICE DRIVERS (2d ed. 2001), *available at* http://www.xml.com/ldd/chapter/book/index.html (last visited Jan. 9, 2007).

[335] Non-Linux users can get a sense of this hierarchy by pressing Ctrl-Alt-Del on any Windows 2000, Windows XP, or Windows Vista release. The "Windows Task Manager" appears, displaying the applications running on the system. One of the available tabs in this dialog box is "Processes." This tab provides Windows' version of the "top" program under Linux, showing each process (Image Name), which user started the process, what percentage of time the CPU spends servicing the process, and the amount of system memory allocated to the process.

[336] RUBINI & CORBET, *supra* note 334, at Chapter 1, *available at* http://www.xml.com/ldd/chapter/book/ch01.html (last visited Jan 8, 2007).

[337] *Id.*

[338] *Id.*



The "system call interface" in the diagram represents the "mechanism used by an application program to request service from the operating system."[339] The kernel flow diagram shades the "application programs" of particular interest under the GPL: kernel modules.[340] The kernel module, as the name suggests, is a routine or set of routines that does not constitute a standalone program by itself.[341] On an as needed basis, each "module" of code can add itself to the core code run by the Linux kernel.[342] As the diagram illustrates, the Linux kernel supports numerous types of modules, including device drivers, the type of module on which the subsequent sections focus as an example.[343]

Developers can create the simplest of kernel modules easily, as in this variation on the familiar "Hello, world" program that grade school students learn in compulsory secondary school computing courses:

[339] *See generally* System Call, WIKIPEDIA: THE FREE ENCYCLOPEDIA, http://en.wikipedia.org/wiki/System_call (last visited Jan. 8, 2007).
[340] RUBINI & CORBET, *supra* note 334, at Chapter 1, *available at* http://www.xml.com/ldd/chapter/book/ch01.html (last visited Jan 8, 2007).
[341] *See* Linux Information Project, *supra* note 77.
[342] *Id.*
[343] RUBINI & CORBET, *supra* note 334, at Chapter 1, *available at* http://www.xml.com/ldd/chapter/book/ch01.html (last visited Jan 8, 2007).

```
#include <linux/module.h>
#include <linux/kernel.h>
int init_module(void) {
     printk("Hello, world.\n");
return 0;
}
void cleanup_module(void) {
     printk("Goodbye, world.\n");
}
```

As with the more familiar BASIC language program, this simple module prints two simple messages. The first two lines invoke the system call interface from the diagram in order to assist the compiler program in creating a binary version of this source code for the Linux system to execute. The inclusion of the module.h and kernel.h header files is mandatory, as are the names of the functions "init_module" and "cleanup_module."[344] Without these standard files and function names specified by the kernel, the module will not load (run) in the Linux operating system.[345] For this interaction with the kernel, developers must use this convention, or API.[346] After the Linux kernel starts (the machine "boots up" and loads the operating system into memory), a simple program, "insmod," can instruct the kernel to load the additional functionality in the "Hello, world" module.[347] The "Hello, world" module above has virtually no functionality and can only print a message. However, in the case of device drivers for video cards or network drivers for Ethernet cards, for example, the loaded code adds significant functionality. Linux kernel modules can also modify the behavior of the kernel itself, such as changing the way that the kernel allocates its resources.[348]

When the insmod program inserts a loadable module into the kernel, the process includes a linking step.[349] The kernel must resolve function names that the module uses, such as "init_module" and "cleanup_module" in the "Hello, world" example, as well as any data structures that the module uses or populates with data.[350] The Linux kernel also supports adding the module code directly when a compiler builds the kernel from source code. Programmers can combine module source code with the source code for the kernel (or other program) that will use

[344] *See* Vans Info., Ltd., Introduction to Linux kernel modules, http://www.freeos.com/articles/4051/ (last visited Feb. 15, 2007).
[345] *Id.*
[346] *Id.* The acronym API stands for Application Program[ming] Interface, "routines or protocols that perform certain widely-used functions." *U.S. v. Microsoft Corp.*, 253 F.3d 34, 53 (D.C. Cir. 2001).
[347] Vans Info., Ltd., *supra* note 344.
[348] *See, e.g.*, MontaVista Software, MontaVista Linux Products, http://www.mvista.com/products/ (last visited Jan. 9, 2007). MontaVista designed its "real time" version of Linux for embedded devices. Among other contributions, "real time" operating system developers change some of the schedulers and timers used within the Linux kernel. The changes guarantee that the kernel can complete certain tasks in a predictable, definable timeframe, regardless of the number of processes running on the system. Real time operating systems have obvious uses in manufacturing or automation.
[349] Vans Info., Ltd., *supra* note 344.
[350] In exceedingly non-technical terms, the "resolution" involves the kernel looking up the name of each function that the module calls. The kernel ensures that it has a copy of that function and its actual instructions before loading the module. Modules using unknown functions or data structures will generate an error when loaded.

the module.[351] The compilation process creates a single object code file containing the compiled versions of both the module and the program it extends. Instead of loading and unloading the modules as needed, the kernel contains a permanent copy of the module's compiled code.

The Linux community has debated the applicability of the GPL to code that links statically or dynamically to the Linux kernel for many years.[352] Both the dynamic and static linking approaches have their respective merits technically. As discussed in the following sections, for legal purposes in determining the status of the module's source code under the Copyright Act, the dynamic or static nature of the interaction between a module and the Linux kernel has no functional distinction.

B. Early Treatment: The "Broad" Construction of Derivative Works

The component architecture in Linux is not unique. The concept of "object-oriented programming" and programming tools like Microsoft's Visual FoxPro or Visual C++ rely on the practice of reusing code and combining smaller routines and functions into a larger program.[353] Libraries, subroutines, and kernel modules simplify the programming process, and the application interfaces that enable these bolt-on components make functionality more robust without adding additional, complex designs and structures.[354]

Depending on what a Linux kernel module does and how the programmer and the Linux kernel developers designed a particular interface, a module could have a simple or complex interface to the Linux kernel. The "Hello, world" is no more complex than plugging a cord into a wall socket. An interface to provide communication between a high-speed data networking card and the Linux kernel's networking substructure would look more like the equipment in the power company's substation. Anywhere along the complexity spectrum, the functionality provided by a kernel module requires that the module communicate with Linux through a common interface. In the "Hello, world" example, the interface copied from the Linux kernel is limited to two function calls. In a network card driver, the module may copy required data structures and other function names, similar to the way that Xbox or PlayStation players master the myriad of moves in a football game or a first-person action title. Regardless of complexity, the location of the module's compiled code is immaterial. The interaction, not the combining of the "boxes" that comprise the code repositories, implicates copyright law.

The Copyright Act, without specifically mentioning software, 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

[351] *See* Vans Info., Ltd., *supra* note 344.
[352] *See, e.g.*, Posting of Jeremy Andrews to KernelTrap, Linux: Reverse Engineering Wireless Drivers, http://kerneltrap.org/node/6692 (Jun. 7, 2006, 11:01 GMT) (discussion about a Linux Kernel Mailing List thread concerning the legality of adding non-GPL licensed wireless driver code to the Linux kernel); Posting of Alan Sanderol to Slashdot, Kororaa Accused of Violating GPL, http://linux.slashdot.org/article.pl?sid=06/05/14/2059242 (May 14, 2006, 20:19 GMT).
[353] *See generally* HAROLD ABELSON & GERALD JAY SUSSMAN, STRUCTURE AND INTERPRETATION OF COMPUTER PROGRAMS 187-358 (1996).
[354] *Id.*

reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted."[355] Two other provisions of the Act affect this provision.

Section 103(a) eliminates protection for "any part" of the derivative work that contains unauthorized portions of the original work.[356] Unfortunately, this provision is silent on who *does* hold the copyright in the infringing work, though the most "plausible possibility is that no one has copyright ownership . . . ." of an unauthorized derivative work.[357] Ultimately, this restriction has little practical significance for the GPL, since 103(a) leaves intact the original copyright. Although an unauthorized derivative work may not enjoy copyright protection, copyright protection for the original GPL-licensed work would prevent any downstream licensees from exploiting the GPL code.

Section 103(b) of the Act extends copyright protection only "to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work, and . . . is independent of, and does not affect or enlarge the scope, duration, ownership, or subsistence of, any copyright protection in the preexisting material."[358] Read together with Section 103(a), the owner of the original work "is effectively the only party that can exploit such a derivative work."[359] As discussed in Part II, *supra*, the GPL purports to restrict non-GPL software from linking with GPL-licensed programs by asserting the copyright holder's exclusive right to prepare derivative works granted by the Act. However, courts have increasingly sought to narrow the definition of derivative works, especially for code written to be compatible with underlying original works.[360] This trend toward stricter tests for derivative works contradicts the broad assertions in the GPL.

Earlier cases, however, interpreted Section 101 broadly. The Seventh Circuit found a derivative work when a product altered or changed the output of an original work in *Midway Mfg. Co. v. Artic Int'l, Inc.*[361] The court held that the defendant's video acceleration boards created a derivative audiovisual work by speeding up game play in the plaintiff's games.[362] Commentators have decried this type of broad definition of Section 101. By 1984, articles warned that the derivative works definition "[was] not only expansive, it [was] still expanding."[363] Later commentators lamented that a broadened definition would lead to situations

[355] 17 U.S.C. § 101 (2000).
[356] *Id.* at § 103(a).
[357] Sean Hogle, *Unauthorized Derivative Source Code*, 18 NO. 5 COMPUTER & INTERNET LAWYER 1, 5 (2001).
[358] 17 U.S.C. § 103(b).
[359] Hogle, *supra* note 357, at 5.
[360] *Id.* at 6 ("Courts, however, have been increasingly solicitous of parties who copy only interfaces of copyrighted software, where the purpose of doing so is to achieve interoperability."); Part III, *infra*.
[361] 704 F.2d 1009 (7th Cir. 1983).
[362] *Id.* at 1013-14. *See also* Addison-Wesley Pub. Co. v. Brown, 223 F. Supp. 219 (D.N.Y. 1963) (holding that a problem solutions manual for a physics textbook constituted an infringing derivative work of that text).
[363] Ralph S. Brown, *The Widening Gyre: Are Derivative Works Getting Out of Hand?*, 3 CARDOZO ARTS & ENT. L.J. 1, 2-3 (1984).

where "any work that [did] not violate the prohibition against copying would at least violate the prohibition against derivatives . . . ."[364]

An early pair of cases illustrating what the Ninth Circuit subsequently called a "hopelessly overbroad" definition of derivative works[365] will be familiar to most adults who grew up during the 1980s. *Worlds of Wonder, Inc. v. Veritel Learning Systems, Inc.*[366] and *Worlds of Wonder, Inc. v. Vector Intercontinental, Inc.*[367] did not involve software, but closely mirrored the debate about software linked to GPL code. Worlds of Wonder created the animated teddy bear Teddy Ruxpin. The bear told stories and sang based on sounds and a signal from a cassette tape player in his back.[368] The defendants in each case created their own tapes containing new stories.[369] These tapes emitted the same voice and signal and "interoperated" with Worlds of Wonder's product.[370] In essence, the defendants had written "software" (or what today would be software, rather than a cassette tape) to interoperate with Teddy Ruxpin and allow him to tell new stories. The court apparently enjoyed the third-party tapes far less than the children who listened to them, however. Despite the fact that the new tapes contained no Worlds of Wonder content, the courts found them to constitute infringing derivative works.[371]

Commentator and judicial criticism of the principles behind the holdings in *Midway* and the two *Worlds of Wonder* cases is unsurprising. As discussed below, subsequent courts have had difficulty reconciling these decisions with the Act. Even the *Midway* court acknowledged the difficulty, admitting that "[i]t is not obvious from this language [of Section 101 of the Act that] a speeded-up video game is a derivative work."[372] Essentially, the early court decisions looked at the output of the combination of works rather than at the works themselves. Under this broad definition, courts would struggle to find any add-on components or software that did not create an infringing derivative. Any board added to a Midway video game or tape playable in Teddy Ruxpin necessarily would have created a similar output, since the underlying game and toy were unchanged. Under this older definition, reading a copy of this paper in Adobe Acrobat Reader would create an infringing work of Microsoft Windows, since the add-on Adobe program creates output nearly identical to the output of other programs for Windows.

[364] Christian H. Nadan, *A Proposal to Recognize Component Works*, 78 CAL. L. REV. 1633, 1640 (1990); *See also* Glynn Lunney, Jr., *Reexamining Copyright's Incentives-Access Paradigm*, 49 VAND. L. REV. 483, 547-48 (1996) (positing that the "systematic expansion of the derivative right in the twentieth century" narrowed the range of lawful fair uses); Michael Gemignani, *Copyright Protection: Computer-Related Dependent Works*, 15 RUTGERS COMP. & TECH. L.J. 383, 410 (1989) (urging a narrow interpretation of derivative works in software contexts).
[365] Micro Star v. FormGen Inc., 154 F.3d 1107, 1110 (9th Cir. 1998).
[366] 658 F. Supp. 351 (N.D. Tex. 1986). The court cited *Midway* and its standard with approval at 354-56.
[367] 653 F. Supp. 135 (N.D. Ohio 1986). The court cited *Midway* and its standard with approval at 139-40.
[368] *Veritel Learning Systems*, 658 F. Supp. at 352-53.
[369] *Id.* at 353; *Vector Intercontinental*, 653 F. Supp. at 137.
[370] *Veritel Learning Systems*, 658 F. Supp. at 356; *Vector Intercontinental*, 653 F. Supp. at 139-140.
[371] *Veritel Learning Systems*, 658 F. Supp. at 356; *Vector Intercontinental*, 653 F. Supp. at 139-140. The Texas court called Teddy Ruxpin an "audiovisual work comprising animated plush toy bear with unique voice." *Veritel Learning Systems*, 658 F. Supp. at 356. Readers in need of a non-mentally taxing diversion after the short technical discussion that led off this section can judge for themselves if Teddy Ruxpin and his friend Grubby have unique voices by visiting YouTube, 1980's Teddy Ruxpin & Grubby Commercial, http://www.youtube.com/watch?v=h0sgFxFeuqM (last visited Jan. 9, 2007), or learn more about the new, all-digital Teddy Ruxpin and friends at BackPack Toys Int'l, Ltd.,The world of Teddy Ruxpin, http://www.teddyruxpin.com/ (last visited Jan. 9, 2007).
[372] Midway Mfg. Co. v. Artic Int'l, Inc., 704 F.2d 1009, 1014 (7th Cir. 1983).

Both courts explained their holdings by pointing out that the defendants had designed their products specifically to combine with the plaintiffs' products, and that the products were useful only in combination.[373] Linus Torvalds repeated this same broad construction argument to support his conclusion that Linux kernel modules would constitute a derivative work, even though the code did not incorporate protected expression. Torvalds argued, "You just can't make a binary module for Linux, and claim that that module isn't derived from the kernel . . . [E]ven if you made your own prototypes and tried hard to avoid kernel headers, it would <u>still</u> be connected and dependent on the kernel."[374]

The definitive 1980s-era broad construction case is the Third Circuit's *Whelan Associates, Inc. v Jaslow Dental Lab., Inc.*[375] To avoid an interpretation that would allow nearly verbatim copying, yet still offer a simple rule to apply, the *Whelan* court took *Midway* to its extreme as it applied to software. In *Whelan*, the plaintiff alleged infringement of its copyright in a custom program for dental laboratory record keeping.[376] Whelan developed the custom program based on specifications its customer, Jaslow designed. Jaslow later developed and started marketing its own competing version of the Whelan program, and Whelan sued.[377] Although Jaslow did not copy Whelan's source code directly, the structure, sequence, and organization (SSO) of the two programs were extremely similar.[378]

The Third Circuit started its analysis by citing the general rule that non-literal expressive elements of a program were copyrightable. Noting that its reading of the Act ran opposite holdings in cases before the 1980 revision, the court justified its construction by looking at the Act itself and finding "no statutory basis for treating computer programs differently from other literary works in this regard."[379] To establish a rule, the court looked back to *Baker v. Selden* and its separation of idea from expression. Although programs are utilitarian, the *Whelan* court held that the central holding in *Baker* was a general idea that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea." [380]

With this general rule in mind, the Third Circuit set out to devise a test that would separate the protectable expression from the unprotectable idea. The court applied a simple test, describing the "idea" in the case as "the efficient management of a dental laboratory (which

---

[373] *Veritel Learning Systems*, 658 F. Supp. at 356 (the defendants' infringing "tapes were designed exclusively for [plaintiffs'] Teddy Ruxpin"); Midway Mfg. Co. v. Artic Int'l, Inc., 547 F. Supp. 999, 1014 (N.D.Ill. 1982) (the defendants' "speed-up kit was designed solely to modify Midway's Galaxian game"), *aff'd* 704 F.2d 1009 (7th Cir. 1993). In *Midway* and the two *Worlds of Wonder* cases, none of the courts found a public interest in encouraging the creation of add-on modules to plaintiffs' products, even though the modules arguably constituted protected creative works themselves. *E.g.*, *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.").
[374] Posting of Linus Torvalds to Linux Kernel Mailing List, Re: Linux GPL and binary module exception clause?, *available at* http://lkml.org/lkml/2003/12/5/116 (Dec. 5, 2003, 17:19:52 GMT).
[375] 797 F.2d 1222 (3d Cir. 1986).
[376] *Id.* at 1225-27.
[377] *Id.*
[378] *Id.* at 1228.
[379] *Id.* at 1240.
[380] *Id.* at 1236 (citing Baker v. Selden, 101 U.S. 99, 102 (1879)) (emphasis omitted).

presumably has significantly different requirements from those of other businesses)."[381] This simplistic reading assumed that a computer program has just *one* idea. This approach makes separating idea from expression a simple endeavor. The court expressed the difference this way: "Because that idea could be accomplished in a number of different ways with a number of different structures, the structure of [Whelan's] program is part of the program's expression, not its idea."[382] Because the court could separate the "idea" of managing a dental lab from the myriad of ways that parties could write software to accomplish that broad rule, they found Jaslow's program an infringing derivative work because of the similar SSO.[383] The court felt that its simple dichotomy test best "provide[d] the proper incentive for programmers by protecting their most valuable efforts," citing research that programmers spent most of their time, and cost of labor, developing the structure and logic of a program.[384]

The broad interpretation in *Midway*, *Whelan*, and other cases may have worked well for video games or cassette tapes, but not for software. *Whelan*'s primary weakness was its definition of "idea." The definition was at once both too broad and too narrow. In an overall sense, Section 102(b) of the Act excludes facts, processes, procedures, ideas that have "merged" with expressions, and other considerations from copyright protection.[385] The *Whelan* court acknowledged facts and "scenes a faire,"[386] but its exclusion of *only* the idea of the program and other elements necessary to that idea was too narrow. At the same time, defining the idea as managing a dental lab was too broad. Judge Learned Hand wrote long ago that courts could define ideas at different levels of abstraction.[387] A court's choice of level of abstraction singularly affects the analysis of any possible derivative work infringement. In *Whelan*, the court's definition of the idea as "efficiently managing a dental laboratory" could have easily fallen under a broader expression umbrella. A computer program, while undoubtedly helpful, is merely one possible expression of an idea of efficient management of a dental laboratory. By choosing a low level of abstraction, the *Whelan* court extended broad protection to the plaintiff's program.[388]

Neither the pre-1980 approach that necessitated Congressional action, nor the broad tests applied in *Midway* and *Whelan,* addressed software copyrights effectively. Judge Hand recognized this difficulty in separating idea from expression with a general rule, since "no

[381] *Id.* at 1236 n.28.
[382] *Id.* at 1236.
[383] *Id.* at 1239-40.
[384] *Id.* at 1237.
[385] *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 836-38 (10th Cir. 1993).
[386] *Whelan*, 797 F.2d at 1236.
[387] *E.g.*, Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).
[388] *See also* DAVID NIMMER, 4-13 NIMMER ON COPYRIGHT § 13.03(F) ("[t]he crucial flaw in [*Whelan*'s] reasoning is that it assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression."). None of this discussion suggests the *Whelan* court might have reached a different decision if it had applied either a higher level of abstraction or the *Altai* test devised later by the Second Circuit. The court may have found protected expression even if it had completed a deeper analysis.

principle can be stated as to when an imitator has gone beyond copying the 'idea', and has borrowed its 'expression'. Decisions must therefore inevitably be *ad hoc*."[389]

C. The Modern "Narrow" Construction of Derivative Works in Software

By the early 1990s, courts began discarding the traditional infringement analysis.[390] While Congress' revision to the Copyright Act declared software a literary work,[391] programs clearly contain elements of a machine.[392] Because of this hybrid purpose, courts abandoned the more general analysis in *Whelan*. Instead, the circuit courts formulated different rules that narrowed the scope of protection—and consequently the realm of derivative works—in favor of greater copyright exclusions. As discussed below, courts began comparing programs at more fundamental levels in addition to the *Whelan* approach of considering the work's output or general functionality.

Since the Second Circuit decided *Computer Associates International, Inc. v. Altai, Inc.* in 1992, no court has adopted *Whelan*'s "broad" construction, opting instead for a narrower approach.[393] The first blow to *Whelan* came from the Supreme Court in *Feist Publications, Inc. v. Rural Telephone Service Co.*[394] The Court rejected the "sweat of the brow" test generally and noted that the Second Circuit, who had originally developed the test, had since fully repudiated it.[395] The Court made clear that despite the plaintiff's significant labor cost investment in developing a telephone directory, labor investment did not automatically transform uncopyrightable ideas and facts into protected, copyrightable elements.[396]

A third critical passage of the Act unravels Torvalds' and the FSF's contention that software modules that link for interoperability create a derivative, and undermines the broad construction in *Whelan*. Section 102(b) of the Act limits protection for certain types of works: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, *regardless of the form* in which it is described, explained, illustrated, or embodied in [an otherwise protected]

[389] Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) (emphasis in original).
[390] *E.g.*, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir. 1995) (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 traditional copyright analysis to software as "fit[ting] the proverbial square peg in a round hole").
[391] *Altai*, 982 F.2d at 712 ("Congress has made clear that computer programs are literary works entitled to copyright protection.").
[392] *See supra* text accompanying notes 99–101; Patent Act, 35 U.S.C. § 101 (2000) (offering patent protection for "any new and useful process, machine, manufacture, or composition of matter").
[393] 982 F.2d 693 (2d Cir. 1992). The only circuit to reject the narrow construction in *Altai* did so because it felt that the *Altai* test was too broad, not too narrow. *Lotus Dev. Corp.*, 49 F.3d at 815. The Third Circuit has never expressly overruled *Whelan*, but declined to follow it. The court opted to use the *Altai* test in *Dun & Bradstreet Software Servs. v. Grace Consulting, Inc.*, 307 F.3d 197, 214-15 (3d Cir. 2002). *See also infra* note 423 and accompanying text.
[394] 499 U.S. 340 (1991).
[395] *Id.* at 360.
[396] *Id.*

work."[397] Regardless of the amount of time that a programmer spends developing the SSO of a program, under *Feist* and Section 102(b), such functional aspects and ideas do not fall within the scope of copyright protection.

The Second Circuit took the lead in narrowing the *Whelan* test, first in *Computer Associates International, Inc. v. Altai Inc.*[398] Applying Section 102(b), the Ninth Circuit began to deny protection to specific elements of programs: purely functional features, features dictated by efficiency, and features necessary for compatibility with other programs.[399] The *Altai* court criticized the *Whelan* approach for focusing on the work's idea rather than specific ideas *within* the work. Those internal ideas may merge with expression, and because a program's "ultimate function or purpose is the composite result of interacting subroutines . . . [and] each subroutine is itself a program, and thus, may be said to have its own 'idea', *Whelan*'s general formulation that a program's overall purpose equates with the program's idea is descriptively inadequate."[400]

Instead of using the inadequate *Whelan* approach, the court recognized that a work might employ multiple ideas, requiring a court to examine the program from multiple levels of abstraction.[401] In *Altai*, Computer Associates ("CA") sued Altai for copying portions of CA's scheduling program to make the Altai program compatible with CA's. Altai's original version copied code directly from CA's program, but Altai rewrote those portions when it received CA's complaint.[402] The resulting code, while no longer a literal copy of CA's code, depended heavily on it and used non-literal expressions from CA's program.[403]

The Second Circuit agreed with the *Whelan* analysis,[404] although the court did "not end [its] analysis . . . [but] determine[d] the scope of copyright protection that extend[ed] to a computer program's non-literal structure."[405] In the continued analysis, the *Altai* court departed from *Whelan* as a "somewhat outdated appreciation of computer science."[406]

---

[397] 17 U.S.C. § 102(b) (2000) (emphasis added). Professor Rosen omits mention of subsection (b) from his book. ROSEN, *supra* note 85. For example, on page 19 of the book, *supra*, Rosen cites a portion of 17 U.S.C. § 102, but omits the "(a)" from the citation without notation. A reader who never researches the text of the Act would never know that subsection (b) exists. Subsection (b) is critical to *Altai* and other cases discussed in Section III, *infra*. Without mentioning the subsection directly, Rosen does summarize its intent on page 39. ROSEN, *supra* note 85, at 39. In an e-mail to the author explaining this less legalistic approach, Rosen noted that he "think[s] it is generally not helpful to confuse technical people with the idea/expression distinction when they are deciding whether or not to copy a published work or to modify it for commercial purposes." E-mail from Lawrence Rosen, Lecturer in Law, Stanford University, to Douglas A. Hass, Author (Sept. 6, 2006, 00:09:42 GMT) (on file with Journal of Intellectual Property at Chicago-Kent).
[398] 982 F.2d 693 (2d Cir. 1992).
[399] *Id.* at 707-10.
[400] *Id.* at 705.
[401] *Id.* at 705-06.
[402] *Id.* at 700.
[403] In the case before the Ninth Circuit, Altai did not appeal the district court's finding that the original literal copying constituted a copyright infringement. *Id.* at 701.
[404] *Id.* at 702-03.
[405] *Id.* at 703.
[406] *Id.* at 706.

Instead of the simple idea/expression test, the *Altai* court adopted a three-stage "abstraction, filtration, comparison" test[407] based on the Ninth Circuit's endorsement of the "analytic dissection" of computer programs in *Brown Bag Software v. Symantec Corp.*,[408] and Judge Hand's abstraction test in *Nichols v. Universal Pictures Corp.*[409] The first step, abstraction, involves dividing the program into its component parts.[410] Returning to the diagram of the Linux system call interface,[411] the high-level function of Linux as an operating system breaks down into the five kernel subsystems. Within each subsystem, different routines implement features (concurrency, multitasking, connectivity, etc.) or support hardware (network drivers, video drivers, etc.). Some subsystems break down further, such as in the network subsystem or architectures to support a particular range of network interfaces. Each subsystem breaks into component routines, functions, subroutines, and, ultimately, mathematical instructions.

Unlike *Whelan*, the court did not choose any one level of abstraction as the *best* level. Instead, the *Altai* test proceeded to examine *each* level of abstraction to determine what, if any, protected expression existed at that level.[412] Here, the court borrowed reasoning from Judge Hand in *Nichols*, who noted "there is a point in this series of abstractions where they are no longer protected, since otherwise the [author] could prevent use of his 'ideas'."[413]

Once the court had abstracted it, *Altai* proceeded to filter the unprotectable elements out of the program.[414] The *Altai* court agreed with *Whelan* that the Act does not protect an underlying idea or any elements necessary to implement that idea.[415] In addition to this general maxim, *Altai* supplied several additional considerations for this second stage. First, the court excluded any code structure dictated by efficiency, because "efficiency concerns . . . so narrow the practical range of choice as to make only one or two forms of expression workable options."[416]

Second, and more important for analysis of the GPL, the court adopted a form of "scenes a faire" for software. The *Altai* court identified several standard practices and techniques common to programming applicable in a specific industry or programming in general. The court named examples of

> extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer

[407] *Id.* at 706-11.
[408] 960 F.2d 1465, 1475 (9th Cir. 1992).
[409] 45 F.2d 119, 121 (2d Cir. 1930) (discussing that courts could test copyright infringement cases at different levels of abstraction).
[410] *Altai*, 982 F.2d 706-07.
[411] *See supra* diagram accompanying note 338.
[412] *See Altai*, 982 F.2d 706-07.
[413] *Nichols*, 45 F.2d at 121.
[414] *Altai*, 982 F.2d at 707-10.
[415] *Id.*
[416] *Id.* at 707-08.

> manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.[417]

The third stage involved analyzing the remaining portions of the original work and allegedly infringing work. This step compared only the "core of protectable expression."[418] The court then applied a more standard infringement test to the cores in each work to determine if the works were substantially similar, and, if so, whether any copying was substantial enough to constitute infringement.[419] At each level of abstraction, the plaintiff had to prove substantial similarity to, and substantial copying of, a protectable expression.[420] The *Altai* court rejected CA's counterargument that the abstraction, filtration, comparison test would strip the industry of protection for the fruits of research and development, providing a disincentive to market participants. In dismissing this argument, the court relied on *Feist*'s rejection of the "sweat of the brow" defense, stating "[t]he interest of the copyright law is not in simply conferring a monopoly on industrious persons, but in advancing the public welfare through rewarding artistic creativity, in a manner that permits the free use and development of non-protectable ideas and processes."[421]

Altai gained wide acceptance across a number of jurisdictions. All but three federal circuit courts follow the Second Circuit's approach[422] including the Federal Circuit and Federal Claims Court.[423] The Seventh Circuit has never considered the *Altai* decision directly, but has cited it with approval[424] and several of the Seventh Circuit district courts have followed the decision.[425] The Eighth Circuit and D.C. Circuit have never considered *Altai*, either, but district

---

[417] *Id.* at 709-10 (citing NIMMER, *supra* note 388).
[418] *Id.* Later in the same passage, the court also refers to each stage's protectable expression as the "golden nugget."
[419] *Id.* at 710-11.
[420] *Id.*
[421] *Id.* at 711.
[422] For other Second Circuit decisions following this approach, *see, e.g.*, MyWebGrocer, LLC v. Hometown Info, Inc., 375 F.3d 190 (2d Cir. 2004); Briarpatch Ltd., L.P. v. Phoenix Pictures, Inc., 373 F.3d 296 (2d Cir. 2004); Samara Bros. Inc. v. Wal-Mart Stores, 165 F.3d 120 (2d Cir. 1998); Softel, Inc. v. Dragon Med. & Sci. Commc'ns, 118 F.3d 955 (2d Cir. 1997).
[423] *E.g.*, Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 839 (Fed. Cir. 1992); Trek Leasing, Inc. v. United States, 66 Fed. Cl. 8 (2005); Dun & Bradstreet Software Servs. v. Grace Consulting, Inc., 307 F.3d 197, 214-15 (3rd Cir. 2002); Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 658-660 (4th Cir. 1993); Kepner-Tregoe, Inc. v. Leadership Software, 12 F.3d 527, 534-36 (5th Cir. 1994); Kohus v. Mariol, 328 F.3d 848, 855-857 (6th Cir. 1993); Apple Computer v. Microsoft Corp., 35 F.3d 1435, 1442-43 (9th Cir. 1994) (using the Ninth Circuit's modified version of *Altai* called "analytic dissection"); Autoskill, Inc. v. Nat'l Educ. Support Sys., 994 F.2d 1476, 1489-93 (10th Cir. 1993); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1543-47 (11th Cir. 1996). The Tenth Circuit has extended the *Altai* test beyond software to a potentially wide range of other cases in Country Kids 'N City Slicks, Inc. v. Sheen, 77 F.3d 1280, 1285 n.5 (10th Cir. 1996) ("The 'abstraction-filtration-comparison' test, or the 'successive filtration' test, was developed for use in the context of alleged infringement of computer software . . . . However, we see no reason to limit the abstraction-filtration-comparison approach to cases involving computer programs.") (internal citations omitted).
[424] Micro Data Base Sys., Inc. v. Dharma Sys., Inc., 148 F.3d 649, 652 (7th Cir. 1998).
[425] *E.g.* Computer Assocs. Int'l v. Quest Software, Inc., 333 F. Supp. 2d 688, 694 (N.D. Ill. 2004); Micro Data Base Sys. v. Nellcor Puritan-Bennett, Inc., 20 F. Supp. 2d 1258, 1262 (N.D. Ind. 1998).

courts in those jurisdictions have used the test.[426] The First Circuit criticized and declined to follow the *Altai* test, but only because the court felt that the test was too broad, not too narrow.[427] Internationally, *Altai* also quickly gained acceptance.[428]

Not all of the approaches follow *Altai* identically. Notably, the Tenth Circuit clarified the filtration test in *Gates Rubber*, because any "[a]pplication of the abstractions test will necessarily vary from case-to-case and program-to-program. Given the complexity and ever-changing nature of computer technology, we decline to set forth any strict methodology for the abstraction of computer programs."[429] The Tenth Circuit adopted six levels of "generally declining abstraction" to facilitate its *Altai*-style analysis in the case,[430] strongly suggesting that the court would apply more levels of abstraction than those offered in *Altai*. The Fifth Circuit explicitly adopted a more detailed approach in *Engineering Dynamics, Inc. v. Structural Software, Inc.*[431] Despite the differences in the exact filtration test that the circuit courts use, *Whelan* is no longer good law now that the circuit courts have adopted *Altai* and its progeny.[432]

Using *Altai*, courts increasingly have prevented software makers from using copyright law to control linked software. A pair of video game cases from the Ninth Circuit illustrates the shift. For example, in *Sega Enterprises, Ltd. v. Accolade, Inc.*, the Ninth Circuit held that Accolade could "reverse engineer" Sega's video game system to create games that ran on the system, even though it involved copying Sega's code.[433] Similarly, in *Sony Computer Entertainment, Inc. v. Connectix Corp.*, the court permitted the defendant 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.[434]

D. Applying Altai and its Progeny to Linux Kernel Modules and the GPL

As Judge Hand's discussion of abstraction in *Nichols* suggests, the highest levels of abstraction provide very little copyright protection for Linux kernel modules. Most

[426] Control Data Sys. v. Infoware, Inc., 903 F. Supp. 1316, 1322 (D. Minn. 1995); United States v. Microsoft Corp., 1998-2 Trade Cas. (CCH) P 72,261 15 (D.D.C. 1998). The D.C. Circuit Court has used the *Altai* test in conjunction with architectural work, however. Sturdza v. United Arab Emirates, 281 F.3d 1287, 1295 (D.C. Cir. 2002).
[427] Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995).
[428] *E.g.*, Chua Puay Kiang v. Singapore Telecomm. Ltd. (1999), 3 S.L.R. 640 (Sing.); Coogi Austl. Pty. Ltd. v. Hysport Int'l Pty. Ltd. (1998), 157 A.L.R. 247 (Austl.); Admar Computers Pty. Ltd. v. Ezy Sys. Pty. Ltd. (1997), 38 I.P.R. 659 (Austl.); Data Access Corp. v. Powerflex Svcs. Pty. Ltd. (1996), 33 I.P.R. 194 (Austl.); Ibcos Computers, Ltd. v. Barclays Mercantile Highland Finance, Ltd., (1994) 29 I.P.R. 25 (Ct. Chanc.) (U.K.); John Richardson Computers, Ltd. v. Flanders, (1993), 26 I.P.R. 367 (Ct. Chanc.) (U.K.); Delrina Corp. v. Triolet Sys., [1993] 4 C.P.R. 3d 1 (Ont. Ct. Just.), *aff'd*, [2002] 17 C.P.R. 4th 289 (Ont. Ct. App.) (Can.); Matrox Elec. Sys. v. Gaudreau, [1993] R.J.Q. 2449, 2457-58 (Mont. Ct. Just.) (Can.).
[429] Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 834 (10th Cir. 1993).
[430] *Id.* at 835. The levels that the *Gates Rubber* court used were "(i) the main purpose, (ii) the program structure or architecture, (iii) modules, (iv) algorithms and data structures, (v) source code, and (vi) object code." *Id.*
[431] 26 F.3d 1335, 1342-43 (5th Cir. 1994).
[432] Even the Third Circuit no longer follows *Whelan*. *See supra* note 423 and accompanying text (noting that the Third Circuit has used the *Altai* test).
[433] 977 F.2d 1510, 1527-28 (9th Cir. 1992).
[434] 203 F.3d 596, 608 (9th Cir. 2000) (holding that reverse engineering a video game system in order to write emulation software that plays the video game system's games is a fair use).

programmers, both inside and outside the Linux and open source communities, write for efficiency, meaning that highly abstracted structures often have only one or two forms of expression. The "Hello, world" module sample earlier in this section provides one simple example. The only way to instruct the Linux kernel to load or unload the module is to call a specific function by name.[435] The *Altai* test would eliminate the efficient functions used to load and unload Linux modules, for example.

Linus Torvalds, posting on the Linux Kernel Mailing List, suggested "when you have the GPL, and you have documented for years and years that [the kernel module interface] is NOT a stable API, and that it is NOT a boundary for the license and that you do NOT get an automatic waiver when you compile against this boundary, then things are different [than with a stable API]."[436] At the same time, Torvalds focuses on today's kernel modules "that are MUCH more extensive than they were back in '95 or so. These days modules are used for pretty much everything, including stuff that is very much 'internal kernel' stuff and as a result the kind of historic 'implied barrier' part of modules really has weakened."[437] Torvalds argues "there 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."[438] The level of integration required, in essence, could render a kernel module's source code useless for other software platforms. The *Sega* court provided a clear rationale for rejecting the *Whelan*-style approach suggested by Torvalds.[439]

In *Sega*, the court denied copyright protection for "functional requirements for compatibility with the Genesis console . . . ."[440] Under this approach, the right of the kernel module author to create a compatible module overrides any nominal copyright infringement created by the static or dynamic linking process.[441] Sega's Genesis console had no public API whatsoever,[442] stable or otherwise, yet the court still carved the functional elements out of the Act. Regardless of the status of the API or system interface required for compatibility, any parts of a program that a developer must copy—such as kernel headers, definition files, variables, or mandatory Linux kernel function calls—in order to create a Linux-compatible kernel module would not receive copyright protection.

---

[435] *See supra* text accompanying note 346.
[436] Posting of Linus Torvalds to Linux Kernel Mailing List, *available at* http://lkml.org/lkml/2003/12/10/101 (Dec. 10, 2003 00:23:33 GMT) (emphasis in original).
[437] Posting of Linus Torvalds to Linux Kernel Mailing List, *available at* http://lkml.org/lkml/2003/12/3/234 (Dec. 10, 2003 00:23:33 GMT) (emphasis in original).
[438] Alessandro Rubini, *An Interview with Linus Torvalds*, 32 LINUX GAZETTE, Sept. 1998, *available at* http://linuxgazette.net/issue32/rubini.html (last visited Sept. 7, 2006).
[439] *See supra* text accompanying note 374.
[440] Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1993). *Accord* Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1536 (11th Cir. 1996) ("The interfaces between the application program, operating system, and hardware are internal, and thus are invisible to the user . . . . For the entire system to function, the components must be 'compatible'- i.e., communication must be unimpeded throughout the system.") (internal citation omitted).
[441] *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).
[442] *Id.* at 1526 ("The interface procedures for the Genesis console are distributed for public use only in object code form . . . .").

The Linux source code is publicly available, unlike Sega's code for the Genesis.[443] Any Linux copyright holder would have a far weaker argument for protection given the public availability of code under the GPL (unlike the closed source Sega Genesis code). No court from *Midway* to *Sega* has ever discussed stability, relative usefulness, platform specificity, or advancements in functionality as acceptable standards. However functionally extensive or poorly documented the API is, the source code provides the tools necessary for developers to write Linux-compatible modules. The portions of Linux kernel source code necessary to make a compatible module fall under the *Sega* exclusion.

To rebut *Sega*, Torvalds and other commentators[444] rely on a misreading of the Ninth Circuit's 1998 decision in *Micro Star v. FormGen, Inc.*[445] The Ninth Circuit opened its analysis with an indictment of the broad *Whelan*-style interpretation of derivative works under the Act, and called it "hopelessly overbroad."[446] Micro Star had created additional levels for FormGen's Duke Nukem game using FormGen's MAP file format.[447] The MAP files did not contain any game code or art from the game's libraries.[448] The files merely acted "as a paint-by-numbers kit" that instructed the game where to place different items available in game play.[449] To determine if the plaintiff's MAP files infringed the defendant's copyrights, the court used a dual-prong test: "a derivative work must exist in a concrete or permanent form . . . and must substantially incorporate protected material from the preexisting work."[450]

The court found infringement, but not because of "linking" from the MAP files to the game art libraries.[451] Micro Star's argument used precisely the same arguments in this paper against finding infringement in making compatible works. The court noted that Micro Star claimed, "the MAP files are not derivative works because they do not, in fact, incorporate any of [Duke Nukem's] protected expression."[452] The court implicitly accepted this argument, and held that Micro Star "misconstrue[d] the protected work. The work that Micro Star infringes is the [Duke Nukem] story itself" by retelling the story of the original game.[453] The court drew a further distinction between the unprotected elements used for interoperability and the protected story in its footnote explaining why "[a] book about Duke Nukem would infringe for the same reason, even if it contained no pictures:"[454]

[443] *See id.* at 1514-15 (discussing Accolade's reverse engineering of the object code into source code, an unnecessary step if source code was publicly available).
[444] *E.g.* Patrick K. Bobko, *Linux and General Public Licenses: Can copyright keep "open source" software free?*, 28 AIPLA Q.J. 81 (2000) ("Linux-derived works are analogous to new tapes for 'Teddy Ruxpin' [in the *Worlds of Wonder* cases] or additional levels for the 'Duke Nukem' computer game [in *Micro Star*]").
[445] 154 F.3d 1107 (9th Cir. 1998).
[446] *Id.* at 1110.
[447] *Id.*
[448] *Id.*
[449] *Id.*
[450] *Id.* at 1109-10 (internal quotation marks and citation omitted).
[451] *Id.* at 1112-13.
[452] *Id.* at 1112.
[453] *Id.*
[454] *Id.*

> We note that the [Micro Star] MAP files can only be used with [Duke Nukem]. If another game could use the MAP files to tell the story of a mousy fellow who travels through a beige maze, killing vicious saltshakers with paperclips, then the MAP files would not incorporate the protected expression of [Duke Nukem] because they would not be telling a [Duke Nukem] story.[455]

Notably, the court did *not* say that merely telling a second story would avoid infringement. The court focused on the incorporated expression of the Duke Nukem story, disregarding the argument contained in FormGen's brief that "the MAP file structure . . . embodies substantial protected expression improperly copied and distributed for commercial gain by Micro Star."[456] FormGen raised this argument in its brief as a ground for appeal after the district court had rejected this same argument at trial, finding that the structure of the MAP files themselves were ineligible for copyright protection.[457]

The Ninth Circuit explicitly drew the distinction between the protected story and unprotected functional elements required to create a MAP file to distinguish the violation in *Micro Star* from the non-infringing uses in *Sega* and *Connectix*.[458] In *Sega* and *Connectix*, the alleged infringers created compatible games for gaming consoles. The consoles had no protected "story" to extend. Similarly, the holding in *Micro Star* cannot apply to Torvalds' linking argument Linux kernel modules because Linux has no "story" and kernel modules create no "sequel" in the way that Micro Star's MAP files did. On non-protection for those functional SSO elements required for compatibility, *Micro Star* is in complete accord with the earlier Ninth Circuit rulings.[459]

Torvalds also discussed the introduction of the "EXPORT_SYMBOL_GPL" kernel symbol. He noted that this symbol serves as "documentation" and a "big cluehint" to developers that if they use functions containing this symbol, then they must license the resulting GPL-reliant code under the GPL.[460] The Ninth Circuit refined the *Sega* analysis further in *Sony Computer Entertainment, Inc. v. Connectix Corp.* to address this type of barrier to the creation of works.[461] The court rejected Sony's argument that the type or amount of intermediate copying necessary affected the infringement analysis.[462] Applying the *Connectix* analysis to Linux, a choice between code with or without EXPORT_SYMBOL_GPL would present a developer "with two engineering solutions that each require intermediate copying of protected and unprotected

[455] *Id.* at 1112 n.5.
[456] Appellees' and Cross-Appellants' Opening and Answering Brief at 26, Micro Star v. FormGen, Inc., 154 F.3d 1107 (9th Cir. 1998) (Nos. 96-56426, 96-56433).
[457] Micro Star v. FormGen, Inc. 942 F. Supp. 1312, 1316 (S.D. Cal. 1996) ("Therefore, the court finds that movants have not shown a valid copyright in the data sequence contained in the Nuke It MAP files . . . ."), *aff'd in part, rev'd in part by* 154 F. 3d 1107 (9th Cir. Cal. 1998).
[458] *See supra* notes 433-434 and accompanying text.
[459] *Accord* The United States Copyright Office "Copyright Registration for Computer Programs (Circular 61), Revised July 2006" *available at* http://www.copyright.gov/circs/circ61.pdf ("Note: The description of authorship on the application should not refer to elements such as 'menu screens,' 'structure, sequence and organization,' 'layout,' 'format,' or the like.").
[460] Torvalds, *supra* note 436.
[461] 203 F.3d 596 (9th Cir. 2000).
[462] *Id.* at 605.

material, [requiring that the developer] often follow the *least efficient solution*."[463] The Ninth Circuit rejected this approach to infringement analysis as "erect[ing] an artificial hurdle" and creating "precisely the kind of wasted effort that the proscription against the copyright of ideas and facts . . . [is] designed to prevent.'"[464] The Ninth Circuit in *Sega* and *Connectix* explicitly eliminated the need, for example, to avoid automatically EXPORT_SYMBOL_GPL, to create "wrappers" that separate closed source code from the Linux kernel, or to use "clean room" development procedures to insulate developers from the "taint" of the GPL-licensed kernel code and allow them to redevelop necessary code from scratch.[465] All of these approaches erect artificial hurdles rejected in *Feist* and as applied to software in *Sega* and *Connectix*.

Torvalds goes further than advocating adopting a *Whelan*-style test. He suggests considering the reason for development, since "these days it would be hard to argue that a new driver or filesystem was developed without any thought of Linux."[466] This test would divine the *intent* of the developer, and attempt to determine whether the developer wrote the software with the intent to make it Linux compatible. In addition to the existing EXPORT_SYMBOL_GPL "artificial hurdle," this added intent test would return courts to *Whelan* and transform any software that works with Linux into a derivative work, much like the Teddy Ruxpin tapes. As discussed earlier in this section, courts have consistently declined to follow this broad interpretation since *Altai*.

In the preamble to the GPL, the FSF essentially argues that the license does not erect an artificial hurdle.[467] The GPL carries no fees and expressly makes source code freely available for the public. At first glance, this free software approach would support "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."[468] The *Sega* court, however, rejected this favorable license argument. The court extended the right of developers to create compatible modules free from the control of the original copyright holder even if the original copyright holder was willing to license that right under other terms.[469] Sega offered Accolade a license agreement that would have allowed Accolade to create Sega-compatible games with the condition that Sega manufacture those games.[470] Despite this license offer, the court declined to find copyright infringement in Accolade's creation of Sega-compatible games without a license.[471] The commercial nature of the use did "not alter [their] judgment in this

---

[463] *Id.* (emphasis in original).
[464] *Id.* (citing Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 354 (1991) (internal quotation marks omitted).
[465] *See* Torvalds, *supra* note 436. Torvalds contributed his post to an existing mailing list thread that discussed NVIDIA's use of "wrapper" code for this purpose. The citation at note 436 includes all of the messages in the thread, including Torvalds' post. *See also* Free Software Foundation, GNU SASL Library – Libgsasl, http://www.gnu.org/software/gsasl/ (last visited Jan. 11, 2007) (discussing one goal of the project as a "clean room implementation" that "means the copyright and license conditions are clear.").
[466] Torvalds, *supra* note 436.
[467] GPL, *supra* note 4, at Preamble (contrasting licenses "designed to take away your freedom" with the GPL, claiming that the GPL is "intended to guarantee your freedom to share and change free software").
[468] Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1523 (9th Cir. 1993).
[469] *Id.* at 1514 (discussing Sega's available license for independent developers of computer game software).
[470] *Id.*
[471] *Id.* at 1522-23.

regard."[472] The court found that Accolade's creation of Sega-compatible games had led to "an increase in the number of independently designed video game programs offered for use with the Genesis console," precisely what the Act "was intended to promote."[473] Like Sega's license, the GPL would only apply the Act's principle of growth to some works (those carrying its license).

E. Policy Rationales Behind Sega *and its Progeny*

The Ninth Circuit carefully rejected Torvalds' postulation that the intimately connected, but undocumented and unstable, kernel module API granted Linux copyright holders the sole right to authorize the creation of compatible modules.[474] Unlike *Sega* and *Connectix*, authors of GPL programs do not stand to profit directly from the sale of the software or development licenses, but the *Sega* court found no distinction between commercial and non-commercial uses.[475] While Stallman, the FSF, and the GPL itself highlight economic and philosophical justifications for the open source software movement, the Ninth Circuit offers its own compelling economic and natural law justifications for its narrowed approach.

The holdings in *Sega* and *Connectix* rest on a scholarly foundation. In *Sega*, the court looked at Accolade's disassembly of Sega software as a scholarly pursuit, despite its commercial underpinning. The *Connectix* court found no support for a distinction between "studying" and "use" in its analysis.[476] The *Sega* court emphasized that Accolade "discover[ed] the functional requirements for compatibility" and "wrote its own procedures based on what it had learned."[477] The court found that Accolade was not trying to "avoid performing its own creative work" but was permissibly using Sega's code to "study [its] functional requirements."[478] The court repeatedly drew an analogy between Accolade's module development and the "scholarship, or research" fair use defenses contained in the Act.[479] Where the GPL encourages certain developers by making source code available, the copyright exemptions and fair use exceptions encourage developers who would not otherwise develop GPL code. The GPL and the exceptions for functional elements and fair use exceptions that enable closed source code to link to GPL-licensed code further the Act's purposes.

[472] *Id.*
[473] *Id.* at 1523.
[474] Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602 (9th Cir. 2000) (the Act's fair use provisions excuse the copying required to create a compatible module); *Sega*, 977 F.2d at 1522 (holding that a linking interface is an unprotectable idea under the Act) [hereinafter *Connectix*].
[475] *Sega*, 977 F.2d at 1523.
[476] *Connectix*, 203 F.3d at 604.
[477] *Sega*, 977 F.2d at 1522.
[478] *Id. Accord* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 843 (Fed. Cir. 1992) (holding that the Act permits uses to "understand the ideas and processes in a copyrighted work.").
[479] 17 U.S.C. § 107 (2000). Although the exceptions to copyright protection in 17 U.S.C. § 102(b) that include challenges for compatibility and the fair use exceptions in 17 U.S.C. § 107 are undoubtedly related, they function differently. The provisions in § 102(b) serve to deny copyright protection to a work or portions of it. Section 107 applies the doctrine of fair use after a court determines that a work qualifies for copyright protection. The two tests frequently work toward the same end, though. As other courts have noted "while [nonprotectable ideas and functional elements] may be unoriginal and not worthy of copyright protection, [original expression], though original and hence copyrightable, may also be denied protection where its use is found to be 'fair' under [the fair use statute]." Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1540 n.18 (11th Cir. 1996).

In addition to scholarship and research, the Act contains an economic component that grants incentives to create new works that build on existing works.[480] Developers write software by building on processes and procedures in the work of others, similar to the broader literary works category itself.[481] As in the expanded market for Sega Genesis-compatible games in *Sega*, the expanded availability of software and supported hardware for Linux creates a public benefit and a further incentive to create Linux-compatible works.[482] Accordingly, the Ninth Circuit in *Sega* and *Connectix* found that module writers not only could link to existing programs for compatibility and interoperability, but also profit from selling those modules. The *Connectix* court explicitly rejected Sony's argument that commercial use raised a "presumption of unfairness" that defeated any assertions of fair use or non-infringement.[483] The earlier *Sega* decision rejected Sega's argument that Accolade could not claim fair use or non-infringement if it competed in Sega's market.[484]

Like the generic abstraction standard applied by the circuit courts under *Altai* and its progeny, courts have used flexible standards to draw lines between permissible use for interoperability and impermissible infringement when interpreting the non-exclusive statutory requirements of Section 107.[485] Neither Congress nor courts chose to draw these lines along the boundaries of intimate links or stable APIs, but instead along the boundaries of market competition. While closed source kernel modules could compete with GPL code in the Linux kernel or in other open source modules,[486] courts must consider if "[the challenged use] should become widespread, [whether] it would adversely affect the potential market for the copyrighted

---

[480] *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.").
[481] *See, e.g.*, Mazer v. Stein, 347 U.S. 201, 217 (1954) ("Unlike a patent, a copyright gives no exclusive right to the art disclosed; protection is given only to the expression of the idea—not the idea itself.") (internal citations omitted); *Accord* Leeds Music Ltd. v. Robin, 358 F. Supp. 650 (S.D. Ohio 1973) ("For unlike the business of building better mousetraps (where the continuing possibilities of achieving dramatic technological breakthroughs may still exist) it might well be said that there are no truly new ideas under the sun.").
[482] *Sega*, 977 F.2d at 1523 ("[P]ublic benefit . . . may arise because the challenged use serves a public interest"). Open source proponents obviously support this proposition as well. Stallman and other open source proponents based the open source movement on this communal theory. *E.g.* Stallman, *supra* note 20 (discussing the common origins of his GNU project and the GPL); STEVEN WEBER, THE SUCCESS OF OPEN SOURCE 179-80 (2004) (postulating that communal open source licenses encourage open source development projects and their unique social structures).
[483] *Connectix*, 203 F.3d at 606 n.10.
[484] *Sega*, 977 F.2d at 1523.
[485] 17 U.S.C. § 107 (2000). "[T]he factors to be considered shall include--

> (1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
> (2) the nature of the copyrighted work;
> (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
> (4) the effect of the use upon the potential market for or value of the copyrighted work.

[486] *See Connectix*, 203 F.3d at 607 (Connectix's emulator for Sony's games "does not merely supplant" Sony's gaming console); *Sega*, 977 F.2d at 1523 (terming Accolade's products a "legitimate competitor" in Sega's market, despite the fact that Accolade's games "undoubtedly affected" that market) (internal quotation marks omitted).

work."[487] "Diminishing potential sales, interfering with marketability, or usurping the market" for the original work can support a finding of infringement.[488]

These tests leave courts with wide discretion to determine when a competing software product "usurps" the original product. If anything, in the open source software market, open source products would tend to usurp closed source ones and not the reverse.[489] Similarly, the Act and the test outlined in *Sega*, do not clearly define when information from an original program used to create a compatible one rises to the level of "misappropriation." In situations where the Act does not offer guidance, the courts "invoke the misappropriation doctrine to remedy allegations of 'piracy' or 'dirty tricks'," in light of the "competitive market context."[490]

In *Sega* and *Bateman*,[491] the courts relied on the benefit of "network externalities" derived from increasing numbers of users and owners of software.[492] Software gains value when it is compatible with other popular software.[493] As a simple illustration, consider the number of programs available for Microsoft Windows as opposed to those available for Sun's Solaris operating system. The benefits of a broad network of users and compatibility with existing software often causes customers to choose otherwise technically inferior products over less widely supported, but superior, ones.[494] Anyone who has experienced the Windows "Blue Screen of Death" or who has moved either to Linux or to Apple's Macintosh platform would probably agree. While the Windows platform does not always dominate from a technical standpoint, it nonetheless commands a dominant market share. Network externalities, therefore, present a compelling reason for courts to take a broad view of developers' use rights for compatibility and interoperability.

A narrow view of exceptions for compatibility and interoperability would not only allow GPL licensors to leverage control over their own code, but to exert monopoly control over

---

[487] Sony Corp. of Am. v. Universal City Studios, 464 U.S. 417, 451 (1984).

[488] *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 supra* text accompanying note 331 (noting that obvious infringement occurs when developers directly modify GPL licensed kernel code to create a derivative work). *Compare* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 843 (Fed. Cir. 1992) (finding that fair use does not protect "extensive efforts to profit" from creating a compatible copyrighted work); *Sony Corp.*, 464 U.S. at 451 (holding that, unlike noncommercial uses that may be eligible for fair use protections, "every commercial use of copyrighted material is presumptively an unfair exploitation" of copyright). Unlike *Sega*, the Federal Circuit in *Atari* found that the uses in question exceeded the necessary requirements for compatibility and ventured into outright misappropriation.

[489] *E.g.*, *supra* Part I A. (discussion of Stallman's creation of an open source alternative to UNIX); WILLIAMS, *supra* note 9 at http://www.faifzilla.org/ch07.html (last visited Jan 11, 2007) (discussing "the so-called 'Symbolics War' of 1982-1983" where Stallman usurped Symbolics' proprietary LISP code with his own open source code).

[490] Leo J. Raskind, *The Misappropriation Doctrine as a Competitive Norm of Intellectual Property Law*, 75 MINN. L. REV. 875, 876-77 (1991).

[491] *Sega*, 977 F.2d at 1523; Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1537 n.11 (11th Cir. 1996).

[492] *See* Timothy S. Teter, Note, *Merger and the Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases*, 45 STAN. L. REV. 1061, 1066 (1993). The Eleventh Circuit in *Bateman* relied extensively on Teter's analysis in its opinion. Teter's note predates the *Sega* case, but the Ninth Circuit's references to public benefit illustrate a reliance on the same network externalities principle.

[493] *Id.*

[494] *Id.* at 1067.

compatible products as well.[495] *Sega* and *Bateman* indicate that copyright protection does not extend to these network externalities, otherwise copyright owners would gain monopoly control over entire ideas.[496]

The natural law arguments follow from the same network externalities. By allowing developers to copy code for compatibility or interoperability purposes, courts separate the value of creating new works from the value inherent in existing ones.[497] As discussed earlier, markets place a value on software based in part on its compatibility and interoperability and its adoption by previous consumers, and not just the aesthetic beauty of its source code expression or the relative effort of its developers. *Sega*, *Connectix*, and *Bateman* suggest that as a matter of natural law, the commercial value of compatible software belongs to the subsequent developer and the community of software users, and not the original copyright holder.[498]

III. Solving the Dilemma by Embracing the Status Quo

A. Identifying the Problem

Richard Stallman and others promoting the GPL have a thoroughly reasonable attraction to copyright law. Copyright is simple: even a moment of poetic inspiration jotted down on the back of cocktail napkin automatically gains copyright protection. The innovative, if legally dubious, construct of the GPL enables at least a partial return to the collaborative commons that Stallman experienced in his early days at MIT. At the same time, though, copyright is weak: it covers only narrowly defined expressions. Both independent authorship and compatibility provide subsequent developers with defenses to infringement. Unless a copyright holder can prove plagiarism of source code and convince a court to ignore the ambiguities and holes in the GPL, the Act provides little coverage.

The current amorphous exception for kernel modules is dangerous. Linux and the GPL face the same problem today that Stallman faced in the late 1970s, though with far more players,

[495] *Id.* ("Where copying to achieve compatibility is not permitted, the forces of network externalities may enable a software producer to achieve a far-reaching monopoly."); Joseph Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35, 36 (1989) ("[T]he more people use a given computer operating system, the more software is likely to be written for [it] . . . . [E]ntry, competition, and innovation may be easier if a competitor need only produce a single better component . . . [rather] than develop an entire 'system.'").
[496] *Sega*, 977 F.2d at 1523-24 ("[A]n attempt to monopolize the market by making it impossible for others to compete runs counter to the statutory purpose of promoting creative expression and cannot constitute a strong equitable basis for resisting the invocation of the fair use doctrine."); *See Bateman*, 79 F.3d at 1543 n.23 (noting that copyright owners own copyrights, not their works, evidenced by the work entering the public domain after the expiration of the copyright).
[497] *Connectix*, 203 F.3d 596, 608 (9th Cir. 2000) (allowing intermediate copying necessary to the development of compatible modules); *See Sega*, 977 F.2d at 1523 (holding that the material copied by Accolade was for compatibility and not a part of Accolade's software that "determine[d] the program's commercial success").
[498] *See* Teter, *supra* note 492 ("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."); *See also* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 819 (1st Cir. 1995) (Boudin, J., concurring) ("A new [user 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 [user interface] . . . .").

developer and commercial, than before. A single developer could easily upset the delicate balance between the GPL and its software commons ideal on one hand and legal realities created by the *Altai* line of cases on the other. The Emacs balancing act unraveled over this same debate.[499] In an interview with Linux Weekly News, Professor Moglen illustrated the problem clearly:

> If the kernel is pure GPL, then I think we would all agree that non-GPL, non-free loadable kernel modules represent GPL violations. Nonetheless, we all know that there are a large number of such modules and their existence is tolerated or even to some degree encouraged by the kernel maintainers, and I take that to mean that as an indication that there is some exception for those modules.[500]

Moglen correctly argues that closed source kernel modules violate the spirit of the GPL, yet acknowledges that Linux still tolerates an undefined exception for code, which also violates that spirit. Competing interests compound the problem. Moglen and the FSF must divide loyalties between promoting the vision of a software commons and encouraging the wide acceptance of Linux and other GPL-licensed projects.

Moglen focused on the locus of the problem as he continued:

> The kernel also maintains a technical mechanism, namely the GPL-only symbols and tainting structure, which seems to suggest an API for the connection of non-GPL'd code to the kernel, which also seems to me a strong indication of the presence of an exception. The difficulty as a lawyer, even a lawyer that is reasonably knowledgeable about these matters, is that I don't understand what the terms of that exception are.[501]

Here, Professor Moglen hints at the legal reasoning behind the exception explained in Part III. Closed-source modules that do not copy code beyond necessary compatibility elements do not violate copyright law under the *Altai* interpretation. At best, proving infringement would require a lengthy, detailed expert analysis of source code in front of courts unlikely to reverse course on copyright protection just to fit the ideological goals of the GPL's drafters.

IBM and SCO have spent millions battling each other in court[502] ostensibly over source code. SCO's suit, whatever its merit, represents another field of battle in a long competitive struggle. Linux is merely the latest proxy. With every commercial open or closed source entry into the Linux market, the incentives to litigate increase. The incentive to siphon money or market share from competitors or to enforce ideology may lead to a case that confuses rather than resolves the kernel module ambiguity.

---

[499] *See supra* text accompanying notes 23–28.
[500] Joe Brockmeier, *Interview: Eben Moglen*, LINUX WEEKLY NEWS, http://lwn.net/Articles/147070/ (Aug. 10, 2005).
[501] *Id.*
[502] *See* Jones, *supra* note 315.

B. Two Potential Solutions

Richard Stallman, the GPL's chief architect, made his intentions very clear in his GNU Manifesto. His utopian goal was to reject all commercial or proprietary modifications and eliminate money from software development.[503] Stallman believed "this can give us a feeling of harmony which is impossible if we use software that is not free. For about half the programmers I talk to, this is an important happiness that money cannot replace."[504]

However, as with all purported Utopias, not all is well. Stallman's plans left software developers and commercial companies looking to leverage open source software with several problems to face. The FSF and Linux community could choose a course that would all but eliminate closed source code from coexisting with GPL-licensed code. Eliminating major drivers of growth and destroying a collaborative system that has made Linux a leading operating system,[505] while certainly detrimental to the community, has the sole, but significant, benefit of ideological consistency. Instead of focusing on copyright law, the FSF and other GPL-licensed code developers could turn to patent law.

The open source community has traditionally demonized software patents,[506] and the GPL employs a termination clause in an attempt both to avoid patent problems and to penalize patent holders.[507] Open source proponents could argue that the decentralized, collaborative development environment eliminates the need for patent licensing necessary in commercial software development. The problem, however, is that the open source community, Linux, and the GPL exist in a commercial world.

The GPL and Stallman's "copyleft" idea allow open source developers to try to operate outside of the intellectual property framework that dominates commercially licensed software. Many open source developers have no patent portfolio of their own that could aid them in litigation over the GPL, either in prosecuting GPL violations or in defending patent infringement actions. Additionally, while attempting to opt out of the copyright framework, developers have often written code with little attention paid to potential patent infringement. The community's attitude focuses on cleaning up *after* infringement happens.[508]

Since developers and copyright holders cannot opt out of intellectual property law and cannot force closed source code from the market wielding only copyrights, open source developers must either acquire patents of their own, or rely on patent portfolios held by parties that also participate in the proprietary software market. For years, commercial software vendors and predatory patent holders have built up large portfolios of intellectual property that could pose threats to the open source community. If a court finds that portions of the Linux kernel, for example, infringe a patent, then the entire community, commercial and non-commercial, could

[503] Stallman, *supra* note 33.
[504] *Id.*
[505] *See supra* text accompanying notes 6-7.
[506] *See*, *e.g.*, League for Programming Freedom, Software Patents, http://lpf.ai.mit.edu/Patents/patents.html (last visited Jan. 11, 2007).
[507] GPL, *supra* note 4, at § 7.
[508] *See supra* notes 315-316 and accompanying text.

face potential liability. By setting itself largely outside of U.S. intellectual property law, open source developers have traditionally had few patents of their own to use for protection in any litigation.[509] Here, the open source community faces two paths on the intellectual property road: work to eliminate software patents, or use the existing intellectual property system to achieve the goals of the FSF and the rest of the open source community.

Despite anti-patent advocates concerns, the open source community has chosen the latter option with its patent commons project. Commercial and open source software Linux developers can agree on one critical point: "Software patents are a huge potential threat to the ability of people to work together on open source."[510] Patent holders can contribute or license patents to the Open Source Development Labs (OSDL) project, now part of The Linux Foundation,[511] assuring that the patents will help "accelerat[e] the development and use of open source software" and defend against predatory use of patents against open source projects.[512] Alongside The Linux Foundation, IBM, Novell, Red Hat, Phillips, and Sony have formed the Open Invention Network (OIN).[513]

The major corporate members of The Linux Foundation and OIN groups, including IBM and HP, continue to invest heavily in patented software technology. In 2004, IBM obtained 3,248 patents from the U.S. Patent and Trademark Office (USPTO), the most obtained by a single company for the twelfth year running.[514] The USPTO granted HP nearly 1,800 patents.[515] Even more traditionally open source companies such as Red Hat have begun building patent portfolios.[516] However, IBM, HP, Red Hat and others still must deliver returns to shareholders. Their pledges not to assert patents against the open source community rely on economics, not benevolence. IBM effectively "licensed" 500 patents to the open source community through its non-assertion pledge, but holds over 10,000 patents in total.[517]

---

[509] The creation of projects like the Open Source Development Labs to promote patents and patent licensing for the open source community illustrates the difficulty that open source developers have had registering and licensing intellectual property.
[510] OSDL, OSDL Announces Patent Commons Project (Aug. 9, 2005), *available at* http://old.linux-foundation.org/newsroom/press_releases/2005/2005_08_09_beaverton.html/newsitem_view. In early 2007, the OSDL merged with the Free Standards Group to create The Linux Foundation. The Linux Foundation, New Linux Foundation Launches (Jan. 22, 2007), *available at* http://old.linux-foundation.org/newsroom/press_releases/2007/2007_january_22_beaverton.html.
[511] *See id.*
[512] *Id.*
[513] OIN, Open Invention Network Formed to Promote Linux and Spur Innovation Globally through Access to Key Patents (Nov. 10, 2005), *available at* http://www.openinventionnetwork.com/press_release11_05.php.
[514] Stephen Shankland, *IBM Offers 500 Patents for Open Source Use*, CNET NEWS.COM, Jan. 10, 2005, http://news.com.com/2100-7344_3-5524680.html (last visited Jan. 11, 2007).
[515] *Id.*
[516] USPTO, USPTO Full-Text and Image Database, http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-adv.htm&r=0&f=S&l=50&d=PTXT&Query=AN%2F%22Red+Hat%22 (last visited Jan. 11, 2007). Red Hat has also made a pledge to use its patents only to further open source efforts. Red Hat, Inc., Statement of Position and Our Promise on Software Patents, http://www.redhat.com/legal/patent_policy.html (last visited Jan. 11, 2007).
[517] Shankland, *supra* note 512.

IBM accepts commoditization of the markets covered by its 500 pledged patents because it can extract profits in other areas of its products and services portfolio. This has two implications for the open source community: (1) patents assume even greater importance in these non-pledged areas, and (2) IBM's pledge not to assert does not prevent closed source code from creeping into coexistence with GPL-licensed code. IBM may actually be giving up very little in its pledges, since the patents in the pledge may or may not have value as revenue generators. IBM does not provide non-assertion guarantees for its ostensibly profitable closed source products or patent holdings.[518]

As commercial development blossoms further in the maturing Linux market, the pressures on the GPL from closed source, interoperable software will only increase. If the community wants to avoid embracing the code entirely, it must extend its patent portfolio to include the patentable processes in the core Linux kernel itself. Groups such as The Linux Foundation and OIN not only must obtain patents from the USPTO and acquire patents from other holders, but also must aggressively pursue closed source companies whose software relies on those patents. To this point, these groups and the Linux community as a whole have focused on defense, rather than offense. If closed source code does represent a scourge, then the traditional tool of patent litigation represents the most effective weapon against it.

Starting a patent war would require significant funding and a strong desire to carry out Stallman's utopian ideals to a potentially bitter end. Patent acquisition by The Linux Foundation and OIN creates a standoff that enables Linux to avoid infringement confrontations, or at least deter them. Patent litigation, while ideologically consistent with the goals of complete free software, could have disastrous consequences for the adoption of Linux.

Ultimately, the push to patent by the open source community may justify a wholesale rethinking of software patents in general. Unfortunately, determining the effects of eradicating software patents is impossible without evidence that would allow Congress and the courts to weigh the comparative intellectual property benefits to open source and commercial software communities. That process extends far beyond the scope of this paper.

Unlike the days when software was a vehicle to sell hardware, and despite utopian ideals of an open source software world entirely free of patents and copyrights, today's open source community must coexist with the commercial software industry. Viral propagation of the GPL license is unnecessary. Closed source kernel modules that incorporate GPL code from the Linux kernel for compatibility do nothing to prevent others from obtaining the original, unmodified Linux kernel code from its original authors or maintainers.[519] The animosity toward closed source Linux kernel modules and other code makes little sense given the rapid convergence of

---

[518] IBM, IBM Statement of Non-Assertion of Named Patents Against OSS, http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf (last visited Jan. 10, 2007.

[519] Putting aside the legal issues, this same argument applies to modifying GPL-licensed code directly to create a derivative work. Proprietary extensions or modifications to GPL-licensed code do not prevent users from obtaining the original, unmodified source code. Users can choose between using the proprietary source code, the open source code (forgoing the proprietary extensions), or developing open source versions of the proprietary code under the same shelter from infringing expressions described in Part III, *supra*. Richard Stallman himself chose the last option when faced with proprietary code from Symbolics in 1982-83. *See supra* note 489.

closed and open source licensing models. Commercial companies often provide access to source code under favorable terms.[520]

This does not mean that the GPL has no place in today's integrated open source/closed source software industry. In the commercial software world, copyright strikes a perfect balance between protection and innovation. Under *Altai*, copyright holders get protection from those who effortlessly plagiarize their code. At the same time, developers retain a strong incentive to author new works and compete, since *Altai* provides broad protection for both independent invention and compatibility.[521] The GPL, under the analysis in this paper, strikes that same perfect balance. The Debian project's use of "free" (fully GPL-licensed) and "contrib" and "non-free" (non-GPL licensed) archives provides an excellent example of the ability of closed and GPL code to coexist to the benefit of Linux users and developers worldwide.[522]

The nebulous "exception" for closed source code encourages participation from a broader base of developers and companies. The exception's existence does not represent a fundamental problem; its uncertain nature does. The FSF and the Linux community should embrace closed source code, instead of ignoring or sidestepping the problem as Stallman did when the Emacs project ran into similar problems with its ill-defined license.[523] Rather than continuing an API that is "NOT a stable API,"[524] Linux should follow the example of its UNIX-variant cousin, FreeBSD. FreeBSD uses the BSD license,[525] allowing both open source and closed source modules to coexist without ambiguity.

For example, FreeBSD uses a networking architecture called netgraph.[526] Along with the rest of FreeBSD, this modular architecture accepts modules under virtually any license.[527] Unlike Linux, Netgraph's API integrates tightly with the FreeBSD kernel, using a well-documented set of standard function calls, data structures, and memory management schemes.[528] Regardless of the underlying licensing structure, modules written for netgraph compliance must interact with netgraph's structure in a predictable, predefined manner.[529]

---

[520] *E.g.*, Microsoft Corporation, Shared Source Initiative, http://www.microsoft.com/resources/sharedsource/default.mspx (last visited Jan. 11, 2007); ImageStream Internet Solutions, Inc., Inetics Technology, http://www.imagestream.com/Inetics.html (last visited Jan. 11, 2007).
[521] In practice, the plagiarism protections in copyright law have their own special problems when applied to software, though. While there is no chance that an author would accidentally create a near-verbatim version of this paper, the structure of programming languages dictates a different result in the software realm. A developer creating a Linux kernel module that prints the text message "Hello, world" when loaded would undoubtedly create something nearly identical to the "Hello, world" module in the text accompanying note 346, *supra.*
[522] *See supra* text accompanying note 302.
[523] *See supra* text accompanying note 23–28.
[524] Torvalds, *supra* note 436.
[525] *See supra* note 276.
[526] Dru Lavigne, FreeBSD: An Open Source Alternative to Linux, http://www.freebsd.org/doc/en_US.ISO8859-1/articles/linux-comparison/article.html (last visited Jan. 11, 2007).
[527] *Id.*
[528] *See generally* Archie Cobbs, *All About Netgraph*, DAEMON NEWS, March 2003, http://ezine.daemonnews.org/200003/netgraph.html (last visited Jan11, 2007).
[529] *Id.*

Open source code can contain (and constrain) closed source code inside the strict parameters of a stable, documented, well-defined API. The GPL, despite its flaws, serves a useful purpose in guiding the direction of Linux development. If the FSF, Torvalds, Stallman, Moglen, or Linux developers generally find closed source code odious, devising open source APIs and architectures to control closed source code's interaction with the Linux kernel accomplishes as much of the GPL's stated goals as legally possible. Torvalds hinted at the efficacy of this approach when discussing the API in newer Linux kernels, suggesting, "historically . . . you could load a module using nothing but . . . standard interfaces" and that this approach created an "implied barrier" between modules and the kernel.[530] While a functionality-crippling barrier is unnecessary, a stricter focus on standardized interfaces and architectures, such as the one used by netgraph, would balance the community's concerns about closed source code with the realities of copyright law.

This approach preserves the long-term success of Linux by recognizing the limitations of the Act and embracing the actual practice of positive symbiotic commercial/open source Linux software development. The Linux community has an opportunity to adopt such an approach as it debates a new version of the GPL that may or may not cover subsequent Linux distributions.

C. Reversioning: The Move to GPLv3

At the time of this writing, the FSF has undertaken an effort to create a new version of the GPL to replace the aging GPLv2.[531] Prior to the release of the first draft of the GPL version 3, Stallman and Moglen identified several issues that the new version needed to address:[532]

1. Internationalization: GPLv2 relied solely on United States law and must adopt a more international flavor.
2. Standardization: Stallman and Moglen argue that the GPL acts as an international standard for open source licenses and developers, and any update must consider this reliance.
3. Adaptation: Digital rights management and patents represent two of a number of new threats to open source software, and a new version must respond to these changed circumstances.[533]

Unfortunately, Stallman and Moglen have largely avoided the debate over kernel modules. Instead, much of the discussion of the new version of the GPL has focused on digital rights management.[534] Despite the hope that competing factions can find common ground and result in creating "everyone a license that reflects their own view of freedom,"[535] Linus Torvalds does not plan to adopt the new version of the GPL for Linux.[536] A failure to resolve the

[530] Torvalds, *supra* note 436.
[531] Eben Moglen and Richard M. Stallman, GPL Version 3: Background to Adoption, Jun 9, 2005, http://www.fsf.org/news/gpl3.html (last visited Jan 11, 2006).
[532] *Id.*
[533] *Id.*
[534] Victor Loh, *Embedded Linux and GPLv3*, EXTREMETECH, Sept. 7, 2006, http://www.extremetech.com/article2/0,1697,2013471,00.asp (last visited Jan. 11, 2007).
[535] *Id.*
[536] Posting of Linus Torvalds to the Linux Kernel Mailing List (Jan. 25, 2006, 22:39:16 GMT), *available at* http://lkml.org/lkml/2006/1/25/273 ("The Linux kernel is under the GPL version 2 . . . . And quite frankly, I don't see that changing . . . . Conversion isn't going to happen.").

outstanding issues with the GPL could leave them for courts, and not the open source community, to resolve.

Until the community completes the reversioning process, an assessment of the GPLv3 is premature. The second draft addresses a few of the issues that this paper raises, ignores several others, and raises a host of new questions. Of course, unless Torvalds and the Linux community accept the GPLv3 as a whole (an unlikely outcome, because the GPLv2 does not require GPL licensors to adopt new versions automatically), the reversioning work may not resolve any of the issues raised in this paper.

Conclusion

Courts and legal practitioners worldwide have only recently joined the open source and Linux communities' intellectual property debate en masse. Despite the often inaccurate guidance some in the industry offer, the Linux community has adapted its development model to U.S. and international copyright and contract law. At the same time, the Linux community has protected vital commercial participation and encouraged widespread industry adoption. The legal issues underlying this transformation have not undergone a robust analysis. This paper sheds light on those issues and tests some of their limits in an attempt to cement the community's legally prudent consensus on closed source participation in open source projects.

The GPL fails to define fundamental terms adequately, including the inconsistent use of "based on" (derivative works), the lack of a choice of law provision, and the ambiguous treatment of patents. The GPL holds itself out as a "viral" license, purporting to foist itself on any software developer who has incorporated GPL code into a project. These and other factors discussed in Part II, combined with the Linux community's outdated views on copyright protection for kernel modules in Part III, make it unlikely that a court could give full effect to the GPL or protect open source code from closed source intrusions in the manner envisioned by the FSF and Linus Torvalds.

However, the GPL does act as the most important beacon for Linux and the rest of the open source world. The GPL's most significant contribution may differ greatly from the one envisioned by its creators: collaborative, decentralized development rather than free software. Contrary to some non-legal analysis, the "gentlemen's agreement" model employed by Linux is a common sense adaptation of the GPL. This arrangement successfully accommodates both closed source and open source software. The gentlemen's agreement is also consistent with U.S. copyright law. Rather than push to change this functional approach, the community should work to embrace and solidify it. Courts and market inertia will force both the open source and closed source models to coexist. Neither purely open source nor purely closed source models for Linux will succeed without the other.