http://www.groklaw.net/article.php?story=20031231092027900

Understanding Open Source Software - by Red Hat's Mark Webbink, Esq.

Wednesday, December 31 2003 @ 09:20 AM EST

Here is a good article to share with your boss.

Mark Webbink, Senior Vice President and General Counsel of Red Hat, Inc., wrote this article for corporate attorneys, explaining free and open source software and comparing various open source licenses, detailing how the GPL really works, explaining US copyright law, and listing some corporate law office best practices for software, from the standpoint of what policies are prudent for the corporate environment.

He also explains how derivative works are defined, touches on the indemnification issue and the difference between open source and "shared source", and highlights some of the main myths and misconceptions about the GPL and open source.

I get email about this subject, so I know some of you are very interested in this subject, so I hope you enjoy finding answers in this thorough and accessible information.

This article was originally published in the March 2003 Journal of the New South Wales Society for Computers and the Law, and we republish with Mr. Webbink's kind permission.

****

What is Open Source Software?

The Open Source Initiative ("OSI") defines Open Source as software providing the following rights and obligations:

1. No royalty or other fee imposed upon redistribution.
2. Availability of the source code.
3. Right to create modifications and derivative works.
4. May require modified versions to be distributed as the original version plus patches.
5. No discrimination against persons or groups.
6. No discrimination against fields of endeavour.
7. All rights granted must flow through to/with redistributed versions.
8. The license applies to the program as a whole and each of its components.
9. The license must not restrict other software, thus permitting the distribution of open source and closed source software together.

This definition clearly leaves room for a wide variety of licenses, and we will examine a number of those license types shortly. Although it is this OSI definition of Open Source to which the remainder of this paper relates, it is worthwhile to also examine the definition of Free Software, for often times the terms Free Software and Open Source are used interchangeably. While they are similar, there are differences worth appreciating.

When we speak of Free Software, we are not talking about freeware, i.e., software that is essentially in the public domain. Rather, we are talking about software that is licensed under the precepts of the Free Software Foundation ("FSF") and its flagship GNU General Public

License.

According to the FSF definition:

"Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

1. The freedom to run the program, for any purpose (freedom 0).
2. The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
3. The freedom to redistribute copies so you can help your neighbour (freedom 2).
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms."

Contrasting the Open Source and Free Software definitions, one finds that all Free Software is Open Source, but as administered by the Free Software Foundation, not all Open Source is Free Software. The difference principally arises from so-called license compatibility, but in large measure the differences are principally philosophical and not substantial.

Fundamentals of Copyright Law

To better appreciate Open Source software, we need a basic understanding of copyright law. Open source software is fundamentally grounded in copyright law[1] . In order to appreciate the rights granted under Open Source licenses, one must first be familiar with the basic bundle of rights granted to the holder of a copyright. Under U.S. copyright law, those rights are:

1. The exclusive right to copy the work;
2. The exclusive right to make derivative works;
3. The exclusive right to distribute the work;
4. The exclusive right to perform the work; and
5. The exclusive right to display the work.[2]

These rights, in turn, are subject to certain limitations, such as rights of "fair use." Fair use includes the use of a work for purposes of criticism, comment, news reporting, teaching, scholarship or research and does not constitute infringement of the work. Whether a specific use is fair use is determined by a number of factors, including:

(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.[3]

Works, such as software, may be placed in the public domain and exist outside of the scope of copyright law.[4] However, with changes in the copyright law in the 1970's and 1980's, including the automatic application of copyright under the Berne Convention, it is no longer an easy task to contribute software to the public domain.[5] Software (or any other body of

work) that is in the public domain cannot, by definition, assert any restrictions on who or how it can be used, modified or distributed (though other laws, such as export controls, may still restrict some software's use or distribution). If Open Source software were in the public domain (that is, not subject to copyright because the author has disclaimed copyright in the work), any business or individual could use the software for any purpose without any copyright restriction, and there would be no requirements for legal review above and beyond ensuring compliance with other statutes (which apply equally to all other software, public domain, or not). Because Open Source software is not in the public domain, but instead protected by copyright law and licensed for use under certain, perhaps unconventional, terms, those terms must be understood.

A valid copyright license applies to a body of work and must assert at least one restriction. A copyright license that states no restrictions implicitly grants all rights, including rights to use, modify, distribute, etc. Most proprietary software copyright licenses assert restrictions on use (including definitions of "fair use", which, according to such licenses, usually does not include decompiling, reverse engineering, or other such uses), copying (usually only for the purposes of backup), and redistribution (usually only when acting as an authorized agent for the copyright owner).

Types of Open Source Licenses

Open source licenses may be broadly categorized into the following types: (1) those that apply no restrictions on the distribution of derivative works (we will call these Non-Protective Licenses because they do not protect the code from being used in non-Open Source applications); and (2) those that do apply such restrictions (we will call these Protective Licenses because they ensure that the code will always remain open/free).

To better appreciate the nature of these licenses, it is helpful to picture software licenses on a continuum based on the rights in copyright extended to the licensee. See Diagram 1 at the conclusion of this article.

Software that has been placed in the public domain is free of all restrictions, all rights under copyright having been granted to the public at large. Licensors of Non-Protective Open Source licenses retain their copyright, but they grant all rights under copyright to the licensee. Licensors of Protective Open Source licenses retain their copyright, grant all rights under copyright to the licensee, but apply at least one restriction, typically that the redistribution of the software, whether modified or unmodified, must be under the same license. Licensors of propriety licenses retain their copyright and only grant a few rights under copyright, typically only the rights to perform and display. The following table, where the BSD license is used as an example of a Non-Protective Open Source license and the GNU General Public License as an example of a Protective Open Source license, displays these contrasts - see Diagram 2 at the conclusion of this article.

Non-Protective Open Source licenses include: Academic Free License v.1.2; Apache Software License v.1.1; Artistic; Attribution Assurance license; BSD License; Eiffel Forum License; Intel Open Source License for CDSA/CSSM Implementation; MIT License; Open Group Test Suite License; Q Public License v.1.0; Sleepycat License; Sun Industry Standards Source License; University of Illinois/NCSA Open Source License; Vovida Software License v.1.0; W3C Software Notice and License; X.Net, Inc. License; zlib/libpng License; and Zope Public License v.2.0.

Protective Open Source licenses include: Apple Public Source License v.1.2; Artistic License; Common Public License v.1.0; GNU General Public License v.2.0; GNU Lesser General Public License v.2.1; IBM Public License v.1.0; Jabber Open Source License v.1.0; MITRE Collaborative Virtual Workspace License; Motosoto Open Source License v.0.9.1; Mozilla Public License v.1.0 and v.1.1; Nethack General Public License; Noika Open Source License v.1.0a; OCLC Research Public License v.1.0; Open Software License v.1.1; Python License; Python

Software Foundation License v.2.1.1; Ricoh Source Code Public License v.1.0; and Sun Public License v.1.0.

All of these, and additional new licenses, can be found on the Open Source Initiative website.

Some Open Source licenses of both types include other provisions, such as restrictions on the use of trademarks, express grants of license with respect to applicable patents, disclaimers of warranties, indemnification of copyright holders in commercial distributions, and disclaimers of liability. However, none of these provisions are as fundamentally important as the obligations/restrictions that are imposed on redistribution rights under the Protective Open Source licenses, and it is with those restrictions on redistribution that we next focus.

The GNU General Public License

As of this writing, the GNU General Public License ("GPL") is the most pervasive license of Open Source software. Of all the software to which it has been applied, none is better known than the Linux® kernel. In fact, the GPL has been applied to a majority of those software modules that are included in the best known of the Linux® distributions, such as Red Hat® Linux®. Its wide appeal among the Open Source community stems from the fact that it falls into that category of Open Source licenses which obligate parties who wish to redistribute such software, either in original or modified (derivative) form, to do so under the terms of the license agreement under which such software was received (all of which we refer to as Protective licenses). That is, having been granted the right to use, modify and redistribute the software under the GPL, the GPL requires you to extend those same privileges under the same terms to others who receive the software from you. This is the common thread that governs Protective licenses, and for that reason, we will focus on the GPL as the standard for Protective licenses.

The GPL provides certain rights to anyone receiving a license to software governed by the GPL. At the same time, it imposes very few obligations except on those who wish to redistribute the software: Those rights and obligations are:

1. The right to copy and redistribute so long as you include a copyright notice and a disclaimer of warranties. You may charge for the cost of distribution and you may offer warranty protection for a fee.
2. The right to make derivative works for your own use.
3. The right to distribute derivative works so long as you:
    1. Identify the work as modified;
    2. License it under the GPL; and
    3. Provide the license information interactively if the program normally runs interactively.

    This section, and the obligation to license under the GPL, does not apply to works which are independent works distributed with the GPL'd work and which run on the GPL'd works.

4. You may distribute the work only in executable form so long as the source code is:
    1. distributed with the object code;
    2. offered by a written offer, valid for a period of at least three years, to make the source code available for no more than the cost of distribution; and
    3. for non-commercial distributions, accompanied with the offer the redistributing party received as to the availability of the source code.
5. You may not impose restrictions on any of these rights.

This is a simple, yet elegant approach. Basically, the licensor is permitting any licensee to exercise virtually all of the rights available under copyright, i.e., the right to copy, the right to

make derivative works, the right to distribute, the right to perform, the right to display. The only obligation imposed is, if the licensee, in turn, wishes to distribute the software to other parties, they agree to do so only under the GPL. The sole purpose of these restrictions is to preserve the integrity of the original grant of freedom through any path of redistribution and to make it impossible for anybody to create a version of the software that offers less freedom to any recipient than the original version would have granted. To paraphrase, the GPL states "once free, always free."

Note that the GPL has no relevance to the case where a party licenses the software and chooses not to redistribute it. This is true whether the party is an individual, a corporation, a corporate conglomerate, or the government. As noted by the FSF, when the GPL refers to "You" in the context of a corporation, it means the parent company and all of the controlled subsidiaries of that parent. Similarly, when "You" is addressing a unit of government, it means that unit of government and all of the subdivisions of that government that are under the direct control of that government. In that context, "You" can readily mean the entire federal government of the U.S. or it could mean any state or commonwealth government, including the agencies of that state or commonwealth government. The GPL does not require that a licensee, who has not made a distribution of the software to another, provide copies of that software to any party who so demands it. The restrictions of the GPL apply only in the case of where GPL'd software is being provided to another party, and the GPL pertains only to the preservation of its original purpose-nothing more.

Based on the foregoing, we can divide the types of Open Source usage into categories, and analyze the legal implications of the GPL for each category. The interesting categories are:

1. Users who use only GPL binaries as they would any other similar program;
2. Users who modify GPL sources to handle local configuration issues or to address internal requirements and not for distribution to others; and
3. Users who modify GPL sources and redistribute them for fun and/or profit.

In case (1), the GPL affects these users not at all; use of the Open Source GNU Emacs TM text editor does not imply that the act of saving a file changes the ownership of the file to the FSF, nor does compilation of a file by Open Source GNU C Compiler cause the resulting object code to belong to the FSF, nor does setting a breakpoint in an executable cause the executable to suddenly become the property of the FSF. Thus, the normal use of GPL software (i.e., use like one would use any other commercial software) in a commercial environment poses no extraordinary legal problems. The wide distribution of Linux operating system software in the last several years for use on commercial web and enterprise servers is ample evidence that there is no legal reason to not use Open Source software if you happen to think it is better than the proprietary alternatives.

In case (2), the locally modified software by definition confers to its users access to the locally modified sources. There is no requirement within the GPL that such local modifications be disclosed to any other party.

In case (3), we get to the group of users for whom the GPL was really written. Users redistributing modified or unmodified versions of Open Source software must obey the GPL's "Golden Rule" of licensing the distributed software under the GPL and not adding any downstream restrictions. To the extent that somebody wants to profit from GPL'd software by using traditional proprietary license restrictions, those restrictions will prove difficult if not impossible to apply. Note, however, earning profit because of the GPL is both legal and encouraged.

From this analysis we are left needing a definition of what constitutes a derivative work in software.

What is a Derivative Work?

The U.S. Copyright Act defines a derivative work as:

"a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications, which, as a whole, represent an original work of authorship, is a "derivative work"."[6]

Thus, a work that is based on one or more preexisting works constitutes a derivative work to the extent that the new material added constitutes an original work of authorship. Such new material may include editorial revisions, annotations, elaborations or other modifications. Derivative works may transform the original work, such as in a translation, including translating software from one computer language to another, or they may combine the original work with other works, such as in a compilation like Red Hat® Linux®. Copyright protection in a derivative work or compilation extends only to the material contributed by the author of such work, and does not grant rights in preexisting material included in the new work.[7]

Where does the law stand on derivative works in software?[8]

The law on derivative works in software is not well established. The U.S. Copyright Act does not specifically address derivative works in software, and there are no U.S. Supreme Court cases immediately on point. Most of the case law has developed among the various U.S. Courts of Appeals, but even there the law varies from one circuit to the next.

The Copyright Act provides an important definition in addition to that of "derivative works", that of "computer programs", which are defined as:

"a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[9]

In addition, the Copyright Act limits the scope of what is covered by copyright by excluding certain subject matter. §102(b) of the Act provides:

"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 such work."

Perhaps the most established of the tests for derivative works in software is the "abstraction, filtration, and comparison" ("AFC") test established by the Second Circuit.[10] Under the threepart AFC test, a court first determines (abstracts) the constituent structural parts of the original program. From these structural parts, the court then filters all unprotectable portions, including those unprotectable matters defined in §102(b) of the Copyright Act and elements that are in the public domain. In the final step, the Court compares any remaining code containing creative expression to the structure of the second program to determine whether the software program in question is sufficiently similar to the pre-existing work to justify a finding that the second program is a derivative work of the first. This AFC approach has been adopted by three other circuits: the Fifth,[11] Tenth[12] and Eleventh.[13]

Of the remaining nine U.S. Courts of Appeal, only one has adopted a clear test for derivative works in software. The Ninth Circuit's test is based on analytical dissection, which first considers whether there are substantial similarities in both the ideas and expressions of the

two works and then utilizes analytic dissection to determine whether any similar features are protected by copyright.[14] The similar elements are categorized by the degree of protection they are to be afforded. "Thin" protection is afforded to non-copyrightable facts or ideas that derive copyright protection only from the manner in which those facts or ideas are aligned and presented. "Broad" protection is afforded to copyrightable expression. The court uses these standards to make a subjective comparison of the works to determine whether, as a whole, they are sufficiently similar to justify a finding that one is a derivative work of the other.

How do these tests apply to derivative works in Open Source software?

In addressing derivative works, Open Source software requires special consideration. This is due principally to the fact that Open Source software, by definition, permits the making of derivative works. Under a Non-Protective license, the new portions of such a derivative work may be licensed under the license of choice of the author, and there is little likelihood of an infringement dispute.

The case is much different with a Protective license because it requires derivative works to be licensed under the same license as the original work. Here the question largely becomes one degree of copying versus adequate avoidance of derivation. Where Open Source software licensed under a Protective license appears to have been copied, in whole or in part, into a larger work, which is then licensed under a different license than the original work, the question of derivative work and infringement would be determined by the courts using the tests outlined above. However, this is not the case where the subsequent author maintains the original Protective license with respect to the original work but licenses the new work under a different license, for here the subsequent author has not infringed the rights of the original author except to the extent that the new work can be determined to be a derivative work of the original. This latter instance requires an entirely different approach to determining derivation.

Where the original work continues to be licensed under a Protective license and the new work is licensed under an alternative license, the following factors are to be considered when determining whether the new work is a derivative of the original:

1. The substantiality of the new work;
2. Whether any part of the original work has been modified; and
3. How such modification has been accomplished.

This analysis is consistent with the distinction drawn by the GPL itself. Clause 2 of the GPL states:

"Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather the intent is to exercise the right to control the distribution of derivative or collective works based on the Program. In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of storage or distribution medium does not bring the other work under the scope of this license."

For example, if the work in question is a database written entirely by you, and the Program in question is a GPL'd operating system (one of many to which the database may have been ported), the distribution of the database with the operating system on a volume of storage (such as the system hard disk) would not confer the GPL of the operating system to the database software. On the other hand, if modifications are made to the Program (the operating system) in order to accommodate the work (the database), then those modifications, which are a derivative work of the Program, would need to be made available under the GPL. No modifications to the work (the database) need be redistributed in this case.

In summary, the legal requirements of the GPL are quite straightforward for commercial software providers: if you want to use a proprietary revenue capture model, keep your works (i.e., the code) separate from GPL'd works, keep the modifications made to each fully independent, and there will be no problems protecting your primary works. At the same time, any modifications you make to software that are already covered by the GPL will be subject to the GPL.

Myths About Open Source

Before leaving this discussion of Open Source licensing it is worthwhile to address some of the myths or misconceptions that have arisen around Open Source.

Myth 1 - Open Source software is "viral" and undermines intellectual property rights.

This myth is particularly rich. First, as already noted, Open Source software is fundamentally grounded in copyright law. As with the holder of any copyright, the copyright holder for a piece of Open Source software gets to elect which rights he/she will grant to others. Open Source authors simply choose to grant more rights than proprietary vendors. The mere fact that an Open Source author using a Protective license insists that derivative works that are distributed to others be licensed under the same license should be contrasted with proprietary software licenses that simply deny the licensee the right to create derivative works or to redistribute them. Each is an exercise in intellectual property rights, and neither is wrong.

Myth 2 - Open Source software is more prone to claims of intellectual property infringement.

The suggestion of the proprietary vendor is that, because the Open Source development model relies on a vast network of Open Source developers who are not necessarily under the control of the distributor, the code produced is far more likely to be exposed to intellectual property infringement claims. The facts simply do not bear this out. While there undeniably have been such claims against some Open Source development projects and/or distributors, the claims have been few and far between.

Myth 3 - Unlike proprietary vendors, Open Source software vendors do not provide warranties or indemnity against intellectual property infringement.

That is true, but no more true for Open Source vendors than for proprietary vendors. For example, the Windows 98 license expressly disclaims any warranty of non-infringement.

Myth 4 - The GNU General Public License is risky because it has never been tested in court.

True again. But which is riskier, licensing practices that are constantly being challenged or those that, in their simplicity and effectiveness, have avoided challenge.

Myth 5 - Making your source code viewable to some users is the equivalent of Open Source.

Open Source provides value to its customers and users by giving them total control over their computing environments. The customer gets to choose whether to run the standard version or whether modifications are desirable. The customer can not only see the bugs, he/she can fix the bugs. Making source code merely viewable to a few users does not help them understand

the code, does not let them modify the code, and most importantly, does not let them fix the code when it breaks. This approach to source code "sharing" equates to entering a public library only to find there is no card catalogue and all of the books are in locked glass cases. Yes, you can root around and find the titles of the books, but you have no ability to gain knowledge from them. Proprietary software seeks to maximize its value solely in monetary terms by achieving a monopoly. Open Source software maximizes its value by assuring that a monopoly cannot be achieved.

Myth 6 - Open Source methods do not produce innovation.

This is a myth. The Open Source community: (a) developed the Apache webserver which is used to run the majority of webservers in the world today; (b) developed Sendmail, the most popular e-mail management software; and (c) developed BIND, the basis for using domain names instead of IP addresses to locate websites. Clearly, Open Source is capable of advancing the art of software.

Without belaboring this point, let us turn to best practices that a corporate law office should maintain with respect to software, whether Open Source or proprietary.

Corporate Law Office Best Practices for Software

As with any form of intellectual property, there are risks associated with licensing the use of software. Some of those risks may relate specifically to Open Source software, but most often they relate to all software, regardless of the form of license. Following are a series of best practices that every corporate law office should implement across their company:

1. Do not permit the uncontrolled importation of software onto company computers.

Do not permit employees to download freeware, shareware, or Open Source software onto company computers without first clearing the license terms with the legal department. At the same time, bar the use of proprietary software except to the extent that the company can account for the permitted licenses. In other words, know what you are putting on your machines--to do otherwise exposes your company to risk.

2. Deal with reputable software vendors with financial staying power.

One of the biggest risks a company takes is adopting software that has no future. Equally true is licensing software from a company without the financial wherewithal to maintain and protect that software. Know your vendors. Know their financial strength, know their policies on licensing, know their responsiveness, and know that their software is reliable.

3. Know how the software will be used.

It's one thing if Open Source is to be used as an operating system on a backoffice server, it is something altogether different if that same Open Source software is to be modified and embedded in a product. The former is not problematic; the latter may be. At the same time, make sure your IT folks are well aware of the typical proprietary restrictions which prohibit reverse engineering or modification. While some proprietary vendors may permit such activities under a special development license or a community source code license, they do not generally permit the activities under their general commercial licenses. It may be worthwhile to categorize each item of software and its permitted uses, e.g., approved for general use in executable form only, approved for use at the source code level in specialized applications or modified applications, and not approved for any use. Finally, nature of use is important in knowing whether the software will be distributed outside the company, potentially triggering

Open Source licensing restrictions.

4. Have a means for documenting what software, and what version of that software, is in use.

Knowing this information and having ready access to it will help assure licensing compliance and at the same time permit IT managers the ability to manage the IT architecture and its advancement.

5. Require documentation of all internal software development projects.

This includes modification of Open Source software. Such documentation should indicate the source of any base software that is modified, all of the authors of the developed software, prior projects (both internal and with prior employers) on which such developers worked, and the identification of any known related intellectual property, particularly patents.

These are but a few suggestions. They are meant to address those issues most commonly found in software, including Open Source software.

For those interested in learning more about Open Source, the following websites are suggested reading:

- Free Software Foundation - http://www.fsf.org
- Open Source Initiative - http://www.opensource.org
- Technical FAQs on Linux from IBM - http://www-106.ibm.com/developerworks/linux/library/l-faq/?open&l=252,t=grl,p=LinuxFAQ
- Link to whitepapers on the legality of the GPL - http://www.newtechusa.com/Viewpoints/GPLLegalityLinks.asp
- Quick Reference for Choosing a Free Software License - http://zooko.com/license_quick_ref.html
- Why Open Source Software - http://www.dwheeler.com/oss_fs_why.html
- Linux Security - http://www.linuxsecurity.com

Footnotes

1. When I talk about copyright law in this paper, I am discussing U.S. copyright law as embodied in Title 17 of the United States Code. The United States is a signatory to the Berne Convention covering copyright, and much of U.S. copyright law is very similar to that of other Berne signatory countries. However, there are provisions in copyright law in the U.S. that are unique to the U.S., such as copyright registration. Persons in countries other than the U.S. should consult local legal counsel specializing in copyright law.
2. §1-106, Title 17, U.S. Code.
3. §1-107, Title 17, U.S. Code.
4. 37 CFR 201.26 defines public domain computer software as software which has been publicly distributed with an explicit disclaimer of copyright protection by the copyright owner. As the Free Software Foundation has stated, public domain software means software that is not copyrighted.
5. Under the Judicial Improvements Act of 1990, which authorized the creation of a national shareware registry, software copyright owners may donate their software to the public domain by assigning it to the Machine-Readable Collections Reading Room of the Library of Congress. 37 Code of Federal Regulations Part 201.26 (1991).
6. 17 U.S. Code §101.
7. 17 U.S. Code §103.

8. For an in depth discussion of the state of the law, see "Software Derivative Work: A Circuit Dependent Determination", Dan Ravicher, October 31, 2002, http://www.pbwt.com/Attorney/files/ravicher 1.pdf.
9. 17 U.S. Code §101.
10. Computer Associates Intl. VC. Altai, Inc., 982 F.2d 693 (2nd Cir. 1992).
11. Engineering Dynamics, Inc. v. Structural Software, inc., 26 F.3d 1335 (5th Cir. 1994); Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527 (5th Cir. 1994).
12. Gates Rubber Co. v. Bando Chem. Indust. Ltd., 9 F.3d 823 (10th Cir. 1993); Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366 (10th Cir. 1997).
13. Bateman v. Mnemonics, Inc., 79 F.3d 1532 (11th Cir. 1996); Mitek Holdings, Inc. v. Arce Engineering Co., Inc., 89 F.3d 1548.
14. Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994).

Diagram 1

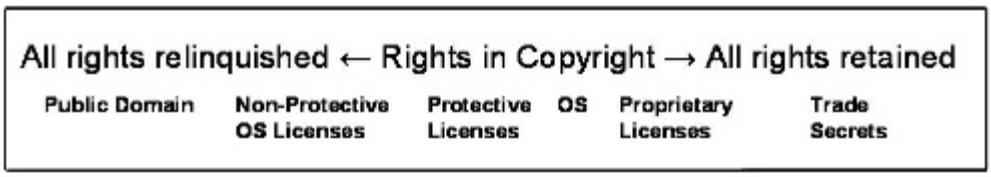

Diagram 2

Rights Granted	Public Domain	BSD	GPL	Windows 98
Copyright retained	No	Yes	Yes	Yes
Right to perform	Yes	Yes	Yes	Yes
Right to display	Yes	Yes	Yes	Yes
Right to copy	Yes	Yes	Yes	No
Right to modify	Yes	Yes	Yes	No
Right to distribute	Yes	Yes	Yes, under same license	No