Software Freedom
Law Center

1995 Broadway, 17th Floor
New York, NY 10023–5882
tel +1–212–580–0800
fax +1–212–580–0898
www.softwarefreedom.org

# Maintaining Permissive-Licensed Files in a GPL-Licensed Project: Guidelines for Developers

27 September 2007

## 1 Executive summary

The Software Freedom Law Center (SFLC) has prepared this document for developers who wish to incorporate permissive-licensed[1] code into GPL'd projects.[2] Developers are advised to take care to comply with the minimal conditions of the permissive license, which will typically require full preservation of copyright, permission, and warranty disclaimer notices. Failure to do so may result in infringement of the copyrights applicable to the incorporated permissive-licensed code. In addition to respecting the rights of the permissive licensor, preservation of notices on parts added to a GPL'd project may facilitate extraction of those parts by downstream users so that they may be used under the permissive terms instead of the GPL.

## 2 How to preserve notices

Virtually all free and open source software (FOSS) licenses have some form of notice preservation requirement. Most of these licenses, including the GPL, allow the required notices for the whole work to be collected in a single location—for example, in a file named COPYRIGHT at the top-level directory of the project's source code. Traditionally, however, FOSS developers have preferred to attach a notice to each source file in a project, thereby associating authorship and copyright permissions with the individual files contained in large programs. Our recommendations are aimed at projects that have already decided to use this common "file-by-file" method of copyright inventory rather than a "single COPYRIGHT file" method.[3]

[1] We refer to terms as "permissive terms" or "permissive licenses" if they grant relatively broad copyright permissions with few conditions and are understood to be permissive enough to allow incorporation of code into a larger work, the totality of which is governed by the more restrictive terms of the GPL. Examples of permissive licenses include the modified BSD license and the ISC license (sometimes called the 2-clause BSD license).

[2] For simplicity, we refer to versions 2 and 3 of the GNU General Public License simply as "the GPL" when our observations apply to either version. In accord with longstanding free software community usage, we use "GPL'd" to mean "licensed under the GPL".

[3] Though the file-by-file method was widely adopted in the free software community during the past two decades, it is not an ideal method, because ensuring that notices remain correct imposes a heavy burden of individual file change tracking. We recommend that project leaders begin to reconsider the file-by-file approach, as it is error-prone and can lead to inadvertent copyright infringement and improper attribution. Any method that projects choose must be properly reviewed and maintained for accuracy. If the file-by-file method continues to serve as the canonical method for copyright inventory, we believe that substantial improvements are needed in the day-to-day care of the copyright and licensing notices in most projects.

## 2.1 Including unmodified permissive-licensed files

The simplest case of notice preservation is unmodified incorporation of a permissive-licensed file from an external project into a GPL'd project without making changes to the code in the file itself. Here the developer should simply leave the file with all notices intact. If the *external* project uses the single COPYRIGHT file method, the developer should copy the names of *all* the copyright holders from that file and place them, along with any copyright, permission, and warranty disclaimer notices required by the permissive license, at the top of the incorporated source file. This may require the developer to list a large number of copyright holders, but it is always best to err on the side of inclusion of each person who might have a copyright claim.

The top of the incorpoated file should look something like this:

```
/* Copyright (c) YEARS_LIST, Permissive Project Contributor1 <contrib1@example.net>
** Copyright (c) YEARS_LIST, Permissive Project Contributor2 <contrib2@example.net>
** ...
**
** Permission to use, copy, modify, and/or distribute this software for
** any purpose with or without fee is hereby granted, provided that the
** above copyright notice and this permission notice appear in all copies.
**
** THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
** WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
** WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR
** BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES
** OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
** WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
** ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
** SOFTWARE.
*/
```

## 2.2 Adding GPL'd modifications to permissive-licensed files

A more complicated case occurs when a developer makes copyrightable changes to a permissive-licensed file that the developer is incorporating into a GPL'd program. Developers in this situation typically apply the GPL to their modifications. (However, it is possible for the developer instead to contribute new code under permissive terms, such as the permissive license that governs the unmodified file. We discuss that case in § 2.3.)

Even though the permissive license of the external project grants legal permission to incorporate code from that project into a GPL'd project, the developer of the GPL'd project must nonetheless comply with the notice preservation requirement in the permissive license. In a project that uses the file-by-file method, a developer who makes copyrightable modifications to a permissive-licensed file should place a new copyright notice and permission notice above the existing one and should make clear that the developer has modified the file. The top of the file will then appear as follows:

```
/*
 * Copyright (c) 2007  GPL Project Developer Who Made Changes <gpl@example.org>
 *
 *  This file is free software: you may copy, redistribute and/or modify it
 *  under the terms of the GNU General Public License as published by the
 *  Free Software Foundation, either version 2 of the License, or (at your
```

```
 *  option) any later version.
 *
 *  This file is distributed in the hope that it will be useful, but
 *  WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 *  General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 * This file incorporates work covered by the following copyright and
 * permission notice:
 *
 *     Copyright (c) YEARS_LIST, Permissive Contributor1 <contrib1@example.net>
 *     Copyright (c) YEARS_LIST, Permissive Contributor2 <contrib2@example.net>
 *
 *     Permission to use, copy, modify, and/or distribute this software
 *     for any purpose with or without fee is hereby granted, provided
 *     that the above copyright notice and this permission notice appear
 *     in all copies.
 *
 *     THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
 *     WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
 *     WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
 *     AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR
 *     CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
 *     OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
 *     NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
 *     CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */
```

It is very important that the developer preserve the entire copyright notice, permission notice, and warranty disclaimer as they appeared in the original code, as required by the permissive license. We sometimes see GPL notices mixed in with permissive license notices—a confusing practice that obscures both the provenance of the code and the precise permissions that were granted by the various copyright holders listed in the notices. When different copyright holders have released their contributions under different terms, the terms that each has placed on his particular contribution should be specified. We recommend making a clear separation and using indentation, as in the example above.

This manner of organizing the notices in the file makes it convenient for developers to choose whether to contribute under permissive terms or under the GPL. If they wish to make their contributions available under permissive terms, they can add their copyright notices to the lower group. If they wish to contribute under the GPL, they can add their copyright notices at the top. Note, however, that in a single source file it is typically very difficult, and often completely infeasible, to determine which parts of such a file are covered by permissive terms. If the goal is to make additional code available under permissive terms only, the method described in § 2.3 should be used.

## 2.3 Keeping modified files permissive-licensed within larger GPL'd works

Developers of GPL'd projects sometimes wish to allow certain parts of their work to remain available to other projects under permissive terms. Most commonly, the goal is to facilitate ongoing cross-project collaboration with other developers. One common example is that of a GPL'd project that includes some code that can be shared with another project under the modified BSD license.

Contributions made to a GPL'd project are typically themselves licensed under the terms of the GPL. If project maintainers have some files that should remain wholly under permissive terms, the stewards of the codebase should take great care to obtain explicit licensing assent from each contributor to (or patcher of) the file in question. The contributor should signify clear agreement that his changes are permissive-licensed.

Of the three approaches to including permissive terms in a GPL'd project, this is by far the most complex, and it requires especially careful attention to legal detail. When working in a large GPL'd codebase, it is very easy to change the work in a way that causes the GPL to cover files that, in isolation, were previously covered only by permissive terms. This is particularly true when cross-file patchsets are accepted from developers. Whenever possible, the stewards of the codebase should consult with a lawyer to be sure that the file is sufficiently independent of and distinct from the core of the GPL'd project to permit continued licensing of the file under the permissive terms despite the application of the GPL to the project as a whole.

# 3 Legal basis for GPL-incorporability of permissive-licensed code

Developers may wonder how they can be confident that a particular license is a permissive license, as we have defined it: that is, how they can know whether its terms and those of the GPL allow the permissive-licensed code to be incorporated into a larger work licensed as a whole under the GPL. While there is no single source of legal authority on the proper interpretation of these licenses, there is a well-established consensus throughout the FOSS community regarding certain relevant principles.

## 3.1 Requirements of the GPL

We consider first the GPL side of the question. The GPL is a copyleft license; it formally requires that modified versions of GPL'd programs be distributed under the terms of the GPL. Although the issue of the breadth of the GPL copyleft on modified versions is beyond the scope of this document, it is significant that most GPL licensors have refused to adopt a narrow view of what causes added code to fall under the copyleft requirement. Developers incorporating permissive-licensed code into a GPL'd project can generally assume, then, that the totality of the modified project is covered by the GPL.

The GPL clarifies its copyleft requirement by explicitly prohibiting imposition of "further restrictions" on downstream recipients' exercise of GPL-derived rights. A condition in a non-GPL license covering some incorporated code, however liberal or simple such a license is, is certain to be different from the terms of the GPL in at least a literal sense. However, the meaning of "further restrictions" under GPL version 2 (GPLv2) has not been read in a literalist fashion, but rather has been elaborated over time by the communities developing, distributing, modifying and using code under that license, as a matter of custom. The treatment of notice preservation requirements in non-GPL licenses is a case in point.

The kinds of notice preservation requirements commonly found in permissive licenses are different from counterpart requirements in the GPL, but they are, as a rule, similar in nature and purpose and no more burdensome than the GPL requirements. For example, section 1 of GPLv2 requires that anyone making or distributing a copy of the program "publish on each copy an appropriate copyright notice" and "keep intact all the notices that refer to . . . the absence of any warranty". The GPL also requires that distributors accompany all copies with the license text. The existence of such requirements in the GPL justifies regarding the comparable requirements in permissive licenses as not being "further" restrictions in relation to the GPL.

Section 7 of GPL version 3 (GPLv3) codifies this GPLv2 interpretive tradition, explicitly allowing contributors to attach "non-permissive additional terms" to the material they contribute if those terms fall within a list of acceptable categories. Such terms supplement the requirements of GPLv3 itself and are not considered "further restrictions". Among those categories are terms "[d]isclaiming warranty or limiting liability differently" from the disclaimers in the GPL and terms "[r]equiring preservation of specified reasonable legal notices or author attributions".

### 3.2 Requirements of the permissive license

The other half of the question is whether the non-GPL license presents any obstacle to incorporation of the code it covers into a larger work that is covered as a whole by the GPL's copyleft. For most who are familiar with such licenses, the answer might appear to be obvious, but it deserves some attention. The terms of permissive licenses allow unlimited modification and redistribution in source or binary form, so long as the stated minimal notice preservation requirements are met. An isolated reading of these simple licenses, in ignorance of historical community practice, can in some cases support more than one interpretation, depending on whether certain permissions are implicit or explicit and on the treatment of that difference under local law.

From the beginnings of their use, however, the permissive licenses have been understood by their licensors and licensees alike to permit the code they cover to be incorporated within larger works covered as a whole by more restrictive terms, including more restrictive FOSS licenses like the GPL as well as, indeed, by proprietary licenses. This understanding represents the uninterrupted, longstanding practice and expectation of the global information technology industry, including both its free and proprietary divisions, with vast commercial reliance on the result. As such, disruption of the established interpretation of the permissive licenses is neither likely nor desirable.

## 4 Should code be "dual licensed" under the GPL and a permissive license?

Developers sometimes attempt to explicitly "dual license" their code under the GPL and a permissive license. The term "dual licensing" is most properly used to refer to giving the recipient a choice of being a licensee under "license A" or "license B". Unfortunately, the term is also commonly used to describe different kinds of licensing arrangements. Some FOSS developers regard a dual licensing notice as a kind of meta-license giving the recipient explicit permission to choose "license A", "license B", or "the disjunction of licenses A and B" as the set of terms to impose on that recipient's distributees. The latter interpretation will (in many cases) have different consequences from the first. There are still other understandings of dual licensing; unfortunately, the concept lacks any standard meaning.[4]

Considering only the first interpretation of dual licensing, it might seem sensible for developers to use explicit dual licensing if they wish their code to be available under permissive terms as well as to be incorporable into GPL'd projects. After all, one might argue, this approach is a more careful one than simply licensing under permissive terms; it ought to do no harm to state clearly what is already implicit and understood.

This argument has merit in theory, but a consideration of the practice of this type of dual licensing suggests a different view. Common attempts by developers at stating dual license notices tend to be confusing, contradictory, and legally unclear. While there may be cases where it will make sense for a developer (with a knowledgeable lawyer's assistance) to place a dual license on his code, we generally recommend that developers *not* use dual licensing if their goal is simply to allow recipients to use the code within larger GPL'd works as well as under permissive terms. If such a developer is using a license like the modified BSD

[4] For example, "dual licensing" is sometimes used to describe the activity of a business offering a (fee-bearing) proprietary license for a GPL'd product. Arguably this usage is a misnomer; in any event, it is entirely unrelated to our discussion here.

license or the ISC license, where there is an established and widespread community understanding that the terms permit incorporation into larger programs covered by the GPL, the developer should simply use the permissive license without any further reference to the GPL.

# 5 A step-by-step guide to cross-license collaboration

One of the most challenging licensing policies to implement is one in which permissive terms are retained on files within a larger GPL'd work. This section presents procedures that should be followed by GPL'd projects that desire to maintain a subset of the codebase under permissive terms. These procedures will help such projects ensure that their work complies with all applicable licensing requirements while preserving collaborative relationships with partner projects working on permissive-licensed code.

## 5.1 Identify all contributors

Conditions cannot be imposed on recipients of a program unless all the copyright holders of the program have given their assent to the imposition of such conditions. Therefore, the licensing terms applicable to a subset of files in a project cannot be changed unless every copyright holder for that subset has consented to the change. This is one reason why some projects choose to require assignment of contributor copyrights to the project maintainer or agree in advance to give the maintainer or some other central authority the power to relicense. Such policies eliminate the need to track down individual past contributors if the maintainer should decide that relicensing is desirable.

In many projects, individual contributors retain their copyrights, and there is no central authority with power to relicense the project. For such projects, the first step in switching from the GPL to a permissive license is to secure the consent of every copyright holder. In order to do this, however, the copyright holders must be identified and located.

Identifying contributors to a project is easiest if the developers use a versioning system like Subversion to track the modification, movement, renaming, merging, and deletion of files. Once a decision has been made to change the license on a subset of files in the project, the files in that subset should be identified and their Subversion metadata used to identify contributors. Complications may arise, however, in certain cases: where patches have been committed by someone other than the author, where copyright notices have not been added to modified files, or where code has been merged from outside files without noting the source in the commit logs. The extra work created for code auditors implementing the new licensing policy in such situations is considerable. Developers should strive for strict adherence to an appropriate protocol for patch submissions: citing sources, committing their own patches, and adding appropriate copyright notices whenever changes are made to a file. Projects with improperly attributed code in their trees accumulate legal risk.

## 5.2 Identify which contributions create a copyright interest

Not every modification to a program is copyrightable by the person making the modification. Minor edits are sometimes too small or insufficiently original to give rise to a copyright interest. If a contributor has only made formatting changes or has only contributed non-expressive data, such as constant values, it is possible that the contributor holds no copyright in the work.[5]

[5] For more discussion on this topic, see *Originality Requirements under U.S. and E.U. Copyright Law*, as published by SFLC on 27 September 2007.

Most of the time, however, patches submitted to a project are sufficient to create a copyright interest. Most countries have adopted a very low standard of originality and creativity for copyrighted works, and even a few lines of code or the rearrangement of different sections of a project might be enough to cause the contributor to be a copyright holder. Because there is no way to know for certain whether a court in a given jurisdiction would find a particular contribution copyrightable, it is most prudent to assume that there is a copyright in the contribution. Such prudence is justified by a consideration of the legal consequences of error. If a notice of an invalid copyright is placed on the program, the copyright notice has no legal effect. If, on the other hand, notice of a valid copyright is omitted, there may be disastrous consequences if the contributor later objects to the use of the material contrary to the contributor's license. For these reasons, every contributor to a given work should be listed in the copyright notice, and all contributors should be consulted when projects contemplate relicensing. Even if some contributions appear to be too minor or non-creative to be copyrightable, the project should consult with a copyright lawyer before determining that the contributor need not be consulted in a relicensing decision.

## 5.3 Secure permissions from current copyright holders

Once all relevant copyright holders in the relevant files have been identified, their assent to relicensing of the work must be secured. Proper patch submission practices and documentation are again helpful here, as they help ensure that valid email addresses are recorded for each contributor.

Projects with a large number of contributors, or those that have existed for a long time, face greater barriers in securing relicensing assent from past developers. Even if logs were kept with correct names and email addresses, over time such addresses may become invalid and the identified contributors may become difficult to locate. A single missing contributor can hold up a relicensing decision indefinitely, especially if that contributor's contributions to the project are too extensive to be easily replaced with other code. This is another reason why some projects adopt governance policies specifying procedures for relicensing in the absence of affirmative consent from past contributors, and why some projects require central assignment of contributor copyrights.

## 5.4 Create a system for tracking permissions on future contributions

Having secured assent from contributors to the existing codebase for the permissive-licensed file subset, the project should ensure that future contributions to those files remain in compliance with the new licensing policy. This is best accomplished through a combination of social and technological measures. The maintainers should publicize the new licensing policy, as described below; establish regular procedures for notifying new developers of the policy at the time they make their contributions; and adopt a patch submission system that explicitly records the contributor's assent to the policy.

For a project that uses the GPL, patches submitted to project subcomponents are assumed to be governed by the GPL unless the contributor explicitly states otherwise. To ensure that no GPL'd patches are applied to the permissive-licensed file subset, the maintainers should establish a policy of asking contributors to tag their patches in the project's versioning system to indicate their intent to apply the permissive terms to their contributions. This system is a failsafe that catches contributors who are unaware of the new policy or who, for whatever reason, have decided to depart from it. It allows the maintainers of the permissive-licensed subset to double-check any untagged patch by securing explicit assent from the contributor by email or, if that is not possible, to exclude the patch from the project's repository.

## 5.5 Publicize the new licensing policy

Although the patch-submission tag failsafe will catch ambiguously-licensed patches, most license disputes concerning future contributions can be avoided if developers are aware of the licensing policy of the project. Once the new system is in place, the maintainers of the permissive-licensed file subset should publicize it widely among the current development community and take measures to inform new contributors of the policy. The web pages of the project should be updated to advertise the new policy. Announcements should be made on developer mailing lists. Every time a new contributor submits a patch, a full explanation of the licensing policy should be presented, asking the contributor to give affirmative assent to it. A contributor who has been informed of the policy implicitly agrees to it by submitting a patch, but explicit assent is much more desirable for legal and diplomatic reasons. Taking careful steps at the outset of development will help the project to avoid having to devote substantial time and resources to solving difficult legal problems in the future.