Translations of this page

Frequently Asked Questions about the GNU Licenses

This page contains answers to commonly-asked questions about the GNU licenses.

Table of Contents

- What does “rules and protocols for communication across the network” mean in GPLv3?
- Distributors that provide Installation Information under GPLv3 are not required to provide “support service” for the product. What kind of “support service” do you mean?

Using programs released under the GNU licenses when writing other programs

- Can I have a GPL-covered program and an unrelated non-free program on the same computer?
- Can I use GPL-covered editors such as GNU Emacs to develop non-free programs? Can I use GPL-covered tools such as GCC to compile them?
- Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require these these designs must be free?
- In what cases is the output of a GPL program covered by the GPL too?
- If I port my program to GNU/Linux, does that mean I have to release it as free software under the GPL or some other free software license?
- I'd like to incorporate GPL-covered software in my proprietary system. Can I do this?
- Does the libstdc++ exception permit dynamic linking?
- If I distribute a proprietary program that links against an LGPLv3-covered library that I've modified, what is the “contributor version” for purposes of determining the scope of the explicit patent license grant I'm making—is it just the library, or is it the whole combination?
- Under AGPLv3, when I modify the Program under section 13, what Corresponding Source does it have to offer?

Combining work with code released under the GNU licenses

- Is GPLv3 compatible with GPLv2?
- How are the various GNU licenses compatible with each other?
- What is the difference between an “aggregate” and other kinds of “modified versions”?
- Do I have “fair use” rights in using the source code of a GPL-covered program?
- Can the US Government release improvements to a GPL-covered program?
- If a library is released under the GPL (not the LGPL), does that mean that any program which uses it has to be under the GPL or a GPL-compatible license?
- You have a GPL'ed program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?
- If so, is there any chance I could get a license of your program under the Lesser GPL?
- Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program.
- If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?
- If a programming language interpreter has a license that is incompatible with the GPL, can I run GPL-covered programs on it?
- If I add a module to a GPL-covered program, do I have to use the GPL as the license for my module?
- If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in?
- Can I apply the GPL when writing a plug-in for a non-free program?

- Can I release a non-free program that's designed to load a GPL-covered plug-in?
- I'd like to incorporate GPL-covered software in my proprietary system. Can I do this?
- I'd like to incorporate GPL-covered software in my proprietary system. Can I do this by putting a “wrapper” module, under a GPL-compatible lax permissive license (such as the X11 license) in between the GPL-covered part and the proprietary part?
- Can I write free software that uses non-free libraries?
- What legal issues come up if I use GPL-incompatible libraries with GPL software?
- I'm writing a Windows application with Microsoft Visual C++ and I will be releasing it under the GPL. Is dynamically linking my program with the Visual C++ run-time library permitted under the GPL?
- I'd like to modify GPL-covered programs and link them with the portability libraries from Money Guzzler Inc. I cannot distribute the source code for these libraries, so any user who wanted to change these versions would have to obtain those libraries separately. Why doesn't the GPL permit this?
- If license for a module Q has a requirement that's incompatible with the GPL, but the requirement applies only when Q is distributed by itself, not when Q is included in a larger program, does that make the license GPL-compatible? Can I combine or link Q with a GPL-covered program?
- In an object-oriented language such as Java, if I use a class that is GPL'ed without modifying, and subclass it, in what way does the GPL affect the larger program?
- How can I allow linking of proprietary modules with my GPL-covered library under a controlled interface only?
- Consider this situation: 1. X releases V1 of a project under the GPL. 2. Y contributes to the development of V2 with changes and new code based on V1. 3. X wants to convert V2 to a non-GPL license. Does X need Y's permission?
- I have written an application that links with many different components, that have different licenses. I am very confused as to what licensing requirements are placed on my program. Can you please tell me what licenses I may use?
- Can I use snippets of GPL-covered source code within documentation that is licensed under some license that is incompatible with the GPL?

Questions about violations of the GNU licenses

- What should I do if I discover a possible violation of the GPL?
- Who has the power to enforce the GPL?
- I heard that someone got a copy of a GPL'ed program under another license. Is this possible?
- Is the developer of a GPL-covered program bound by the GPL? Could the developer's actions ever be a violation of the GPL?
- I just found out that a company has a copy of a GPL'ed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?
- Can I use GPLed software on a device that will stop operating if customers do not continue paying a subscription fee?
- What does it mean to “cure” a violation of GPLv3?
- If someone installs GPLed software on a laptop, and then lends that laptop to a friend without providing source code for the software, have they violated the GPL?
- Suppose that two companies try to circumvent the requirement to provide Installation Information by having one company release signed software, and the other release a User Product that only runs signed software from

the first company. Is this a violation of GPLv3?

What does "GPL" stand for?

"GPL" stands for "General Public License". The most widespread such license is the GNU General Public License, or GNU GPL for short. This can be further shortened to "GPL", when it is understood that the GNU GPL is the one intended.

Does free software mean using the GPL?

Not at all—there are many other free software licenses. We have an incomplete list. Any license that provides the user certain specific freedoms is a free software license.

Why should I use the GNU GPL rather than other free software licenses?

Using the GNU GPL will require that all the released improved versions be free software. This means you can avoid the risk of having to compete with a proprietary modified version of your own work. However, in some special situations it can be better to use a more permissive license.

Does all GNU software use the GNU GPL as its license?

Most GNU software packages use the GNU GPL, but there are a few GNU programs (and parts of programs) that use looser licenses, such as the Lesser GPL. When we do this, it is a matter of strategy.

Does using the GPL for a program make it GNU software?

Anyone can release a program under the GNU GPL but that does not make it a GNU package.

Making the program a GNU software package means explicitly contributing to the GNU Project. This happens when the program's developers and the GNU Project agree to do it. If you are interested in contributing a program to the GNU Project, please write to <maintainers@gnu.org>.

What should I do if I discover a possible violation of the GPL?

You should report it. First, check the facts as best you can. Then tell the publisher or copyright holder of the specific GPL-covered program. If that is the Free Software Foundation, write to <license-violation@gnu.org>. Otherwise, the program's maintainer may be the copyright holder, or else could tell you how to contact the copyright holder, so report it to the maintainer.

Why does the GPL permit users to publish their modified versions?

A crucial aspect of free software is that users are free to cooperate. It is absolutely essential to permit users who wish to help each other to share their bug fixes and improvements with other users.

Some have proposed alternatives to the GPL that require modified versions to go through the original author. As long as the original author keeps up with the need for maintenance, this may work well in practice, but if the author stops (more or less) to do something else or does not attend to all the users' needs, this scheme falls down. Aside from the practical problems, this scheme does not allow users to help each other.

Sometimes control over modified versions is proposed as a means of preventing confusion between various versions made by users. In our experience, this confusion is not a major problem. Many versions of Emacs have been made outside the GNU Project, but users can tell them apart. The GPL requires the maker of a version to place his or her name on it, to distinguish it from other versions and to protect the reputations of other maintainers.

Does the GPL require that source code of modified versions be posted to the public?

The GPL does not require you to release your modified version, or any part of it. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it internally without ever releasing it outside the organization.

But *if* you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the program's users, under the GPL.

Thus, the GPL gives permission to release the modified program in certain ways, and not in other ways; but the decision of whether to release it is up to you.

Can I have a GPL-covered program and an unrelated non-free program on the same computer?

Yes.

If I know someone has a copy of a GPL-covered program, can I demand he give me a copy?

No. The GPL gives him permission to make and redistribute copies of the program *if he chooses to do so*. He also has the right not to redistribute the program, if that is what he chooses.

What does “written offer valid for any third party” mean in GPLv2? Does that mean everyone in the world can get the source to any GPL'ed program no matter what?

If you choose to provide source through a written offer, then anybody who requests the source from you is entitled to receive it.

If you commercially distribute binaries not accompanied with source code, the GPL says you must provide a written offer to distribute the source code later. When users non-commercially redistribute the binaries they received from you, they must pass along a copy of this written offer. This means that people who did not get the binaries directly from you can still receive copies of the source code, along with the written offer.

The reason we require the offer to be valid for any third party is so that people who receive the binaries indirectly in that way can order the source code from you.

GPLv2 says that modified versions, if released, must be “licensed …to all third parties.” Who are these third parties?

Section 2 says that modified versions you distribute must be licensed to all third parties under the GPL. “All third parties” means absolutely everyone—but this does not require you to *do* anything physically for them. It only means they have a license from you, under the GPL, for your version.

Am I required to claim a copyright on my modifications to a GPL-covered program?

You are not required to claim a copyright on your changes. In most countries, however, that happens automatically by default, so you need to place your changes explicitly in the public domain if you do not want them to be copyrighted.
Whether you claim a copyright on your changes or not, either way you must release the modified version, as a whole, under the GPL. (if you release your modified version at all)

If a program combines public-domain code with GPL-covered code, can I take the public-domain part and use it as public domain code?

You can do that, if you can figure out which part is the public domain part and separate it from the rest. If code was put in the public domain by its developer, it is in the public domain no matter where it has been.

Does the GPL allow me to sell copies of the program for money?

Yes, the GPL allows everyone to do this. The right to sell copies is part of the definition of free software. Except in one special situation, there is no limit on what price you can charge. (The one exception is the required written offer to provide source code that must accompany binary-only release.)

Does the GPL allow me to charge a fee for downloading the program from my site?

Yes. You can charge any fee you wish for distributing a copy of the program. If you distribute binaries by download, you must provide “equivalent access” to download the source—therefore, the fee to download source may not be greater than the fee to download the binary.

Does the GPL allow me to require that anyone who receives the software must pay me a fee and/or notify me?

No. In fact, a requirement like that would make the program non-free. If people have to pay when they get a copy of a program, or if they have to notify anyone in particular, then the program is not free. See the definition of free software.

The GPL is a free software license, and therefore it permits people to use and even redistribute the software without being required to pay anyone a fee for doing so.

If I distribute GPL'd software for a fee, am I required to also make it available to the public without a charge?

No. However, if someone pays your fee and gets a copy, the GPL gives them the freedom to release it to the public, with or without a fee. For example, someone could pay your fee, and then put her copy on a web site for the general public.

Does the GPL allow me to distribute copies under a nondisclosure agreement?

No. The GPL says that anyone who receives a copy from you has the right to redistribute copies, modified or not. You are not allowed to distribute the work on any more restrictive basis.

If someone asks you to sign an NDA for receiving GPL-covered software copyrighted by the FSF, please inform us immediately by writing to license-violation@fsf.org.

If the violation involves GPL-covered code that has some other copyright holder, please inform that copyright holder, just as you would for any other kind of violation of the GPL.

Does the GPL allow me to distribute a modified or beta version under a nondisclosure agreement?

No. The GPL says that your modified versions must carry all the freedoms stated in the GPL. Thus, anyone who receives a copy of your version from you has the right to redistribute copies (modified or not) of that version. You may not distribute any version of the work on a more restrictive basis.

Does the GPL allow me to develop a modified version under a nondisclosure agreement?

Yes. For instance, you can accept a contract to develop changes and agree not to release *your changes* until the client says ok. This is permitted because in this case no GPL-covered code is being distributed under an NDA.

You can also release your changes to the client under the GPL, but agree not to release them to anyone else unless the client says ok. In this case, too, no GPL-covered code is being distributed under an NDA, or under any additional restrictions.

The GPL would give the client the right to redistribute your version. In this scenario, the client will probably choose not to exercise that right, but does *have* the right.

I want to get credit for my work. I want people to know what I wrote. Can I still get credit if I use the GPL?

You can certainly get credit for the work. Part of releasing a program under the GPL is writing a copyright notice in your own name (assuming you are the copyright holder). The GPL requires all copies to carry an appropriate copyright notice.

Why does the GPL require including a copy of the GPL with every copy of the program?

Including a copy of the license with the work is vital so that everyone who gets a copy of the program can know what his rights are.

It might be tempting to include a URL that refers to the license, instead of the license itself. But you cannot be sure that the URL will still be valid, five years or ten years from now. Twenty years from now, URLs as we know them today may no longer exist.

The only way to make sure that people who have copies of the program will continue to be able to see the license, despite all the changes that will happen in the network, is to include a copy of the license in the program.

What if the work is not much longer than the license itself?

If a single program is that short, you may as well use a simple all-permissive license for it, rather than the GNU GPL.

Can I omit the preamble of the GPL, or the instructions for how to use it on your own programs, to save space?

The preamble and instructions are integral parts of the GNU GPL and may not be omitted. In fact, the GPL is copyrighted, and its license permits only verbatim copying of the entire GPL. (You can use the legal terms to make another license but it won't be the GNU GPL.)

The preamble and instructions add up to some 1000 words, less than 1/5 of the GPL's total size. They will not make a substantial fractional change in the size of a software package unless the package itself is quite small. In that case, you may as well use a simple all-permissive license rather than the GNU GPL.

What does it mean to say that two licenses are “compatible”?

In order to combine two programs (or substantial parts of them) into a larger work, you need to have permission to use both programs in this way. If the two programs' licenses permit this, they are compatible. If there is no way to satisfy both licenses at once, they are incompatible.

For some licenses, the way in which the combination is made may affect whether they are compatible—for instance, they may allow linking two modules together, but not allow merging their code into one module.

If you just want to install two separate programs in the same system, it is not necessary that their licenses be compatible, because this does not combine them into a larger work.

What does it mean to say a license is “compatible with the GPL?”

It means that the other license and the GNU GPL are compatible; you can combine code released under the other license with code released under the GNU GPL in one larger program.

All GNU GPL versions permit such combinations privately; they also permit distribution of such combinations provided the combination is released under the same GNU GPL version. The other license is compatible with the GPL if it permits this too.

GPLv3 is compatible with more licenses than GPLv2: it allows you to make combinations with code that has specific kinds of additional requirements that are not in GPLv3 itself. Section 7 has more information about this, including the list of additional requirements that are permitted.

Can I write free software that uses non-free libraries?

If you do this, your program won't be fully usable in a free environment. If your program depends on a non-free library to do a certain job, it cannot do that job in the Free World. If it depends on a non-free library to run at all, it cannot be part of a free operating system such as GNU; it is entirely off limits to the Free World.
So please consider: can you find a way to get the job done without using this library? Can you write a free replacement for that library?

If the program is already written using the non-free library, perhaps it is too late to change the decision. You may as well release the program as it stands, rather than not release it. But please mention in the README that the need for the non-free library is a drawback, and suggest the task of changing the program so that it does the same job without the non-free library. Please suggest that anyone who thinks of doing substantial further work on the program first free it from dependence on the non-free library.

Note that there may also be legal issues with combining certain non-free libraries with GPL-covered Free Software. Please see the question on GPL software with GPL-incompatible libraries for more information.

What legal issues come up if I use GPL-incompatible libraries with GPL software?

Both versions of the GPL have an exception to their copyleft, commonly called the system library exception. If the GPL-incompatible libraries you want to use meet the criteria for a system library, then you don't have to do anything special to use them; the requirement to distribute source code for the whole program does not include those libraries, even if you distribute a linked executable containing them.

The criteria for what counts as a "system library" vary between different versions of the GPL. GPLv3 explicitly defines "System Libraries" in section 1, to exclude it from the definition of "Corresponding Source." GPLv2 says the following, near the end of section 3:

> *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.*

If you want your program to link against a library not covered by the system library exception, you need to provide permission to do that. Below are two example license notices that you can use to do that; one for GPLv3, and the other for GPLv2. In either case, you should put this text in each file to which you are granting this permission.

Only the copyright holders for the program can legally release their software under these terms. If you wrote the whole program yourself, then assuming your employer or school does not claim the copyright, you are the copyright holder—so you can authorize the exception. But if you want to use parts of other GPL-covered programs by other authors in your code, you cannot authorize the exception for them. You have to get the approval of the copyright holders of those programs.

When other people modify the program, they do not have to make the same exception for their code—it is their choice whether to do so.

If the libraries you intend to link with are non-free, please also see the section on writing Free Software which uses non-free libraries.

If you're using GPLv3, you can accomplish this goal by granting an additional permission under section 7. The following license notice will do that. You must replace all the text in brackets with text that is appropriate for your program. If not everybody can distribute source for the libraries you intend to link with, you should remove the text in braces; otherwise, just remove the braces themselves.

> *Copyright (C) [years] [name of copyright holder]*
>
> *This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.*
>
> *This program 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>.*
>
> *Additional permission under GNU GPL version 3 section 7*
>
> *If you modify this Program, or any covered work, by linking or combining it with [name of library] (or a modified version of that library), containing parts covered by the terms of [name of library's license], the licensors of this Program grant you additional permission to convey the resulting work. {Corresponding Source for a non-source form of such a combination shall include the source code for the parts of [name of library] used as well as that of the covered work.}*

If you're using GPLv2, you can provide your own exception to the license's terms. The following license notice will do that. Again, you must replace all the text in brackets with text that is appropriate for your program. If not everybody can distribute source for the libraries you intend to link with, you should remove the text in braces; otherwise, just remove the braces themselves.

> *Copyright (C) [years] [name of copyright holder]*
>
> *This program is free software; you can redistribute it 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 program 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>.*
>
> *Linking [name of your program] statically or dynamically with other modules is making a combined work based on [name of your program]. Thus, the terms and conditions of the GNU General Public License cover the whole combination.*
>
> *In addition, as a special exception, the copyright holders of [name of your program] give you permission to combine [name of your program] with free software programs or libraries that are released under the GNU LGPL and with code included in the standard release of [name of library] under the [name of library's license] (or modified versions of such code, with unchanged license). You may copy and distribute such a system following the terms of the GNU GPL for [name of your program] and the licenses of the other code concerned{, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code}.*
>
> *Note that people who make modified versions of [name of your program] are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception.*

How do I get a copyright on my program in order to release it under the GPL?

Under the Berne Convention, everything written is automatically copyrighted from whenever it is put in fixed form. So you don't have to do anything to “get” the copyright on what you write—as long as nobody else can claim to own your work.

However, registering the copyright in the US is a very good idea. It will give you more clout in dealing with an infringer in the US.

The case when someone else might possibly claim the copyright is if you are an employee or student; then the employer or the school might claim you did the job for them and that the copyright belongs to them. Whether they would have a valid claim would depend on circumstances such as the laws of the place where you live, and on your employment contract and what sort of work you do. It is best to consult a lawyer if there is any possible doubt.

If you think that the employer or school might have a claim, you can resolve the problem clearly by getting a copyright disclaimer signed by a suitably authorized officer of the company or school. (Your immediate boss or a professor is usually NOT authorized to sign such a disclaimer.)

What if my school might want to make my program into its own proprietary software product?

Many universities nowadays try to raise funds by restricting the use of the knowledge and information they develop, in effect behaving little different from commercial businesses. (See “The Kept University”, Atlantic Monthly, March 2000, for a general discussion of this problem and its effects.)

If you see any chance that your school might refuse to allow your program to be released as free software, it is best to raise the issue at the earliest possible stage. The closer the program is to working usefully, the more temptation the administration might feel to take it from you and finish it without you. At an earlier stage, you have more leverage.

So we recommend that you approach them when the program is only half-done, saying, “If you will agree to releasing this as free software, I will finish it.” Don't think of this as a bluff. To prevail, you must have the courage to say, “My program will have liberty, or never be born.”

Could you give me step by step instructions on how to apply the GPL to my program?

See the page of GPL instructions.

I heard that someone got a copy of a GPL'ed program under another license. Is this possible?

The GNU GPL does not give users permission to attach other licenses to the program. But the copyright holder for a program can release it under several different licenses in parallel. One of them may be the GNU GPL.

The license that comes in your copy, assuming it was put in by the copyright holder and that you got the copy legitimately, is the license that applies to your copy.

I would like to release a program I wrote under the GNU GPL, but I would like to use the same code in non-free programs.

To release a non-free program is always ethically tainted, but legally there is no obstacle to your doing this. If you are the copyright holder for the code, you can release it under various different non-exclusive licenses at various times.

Is the developer of a GPL-covered program bound by the GPL? Could the developer's actions ever be a violation of the GPL?

Strictly speaking, the GPL is a license from the developer for others to use, distribute and change the program. The developer itself is not bound by it, so no matter what the developer does, this is not a “violation” of the GPL.

However, if the developer does something that would violate the GPL if done by someone else, the developer will surely lose moral standing in the community.

Can the developer of a program who distributed it under the GPL later license it to another party for exclusive use?

No, because the public already has the right to use the program under the GPL, and this right cannot be withdrawn.

Can I use GPL-covered editors such as GNU Emacs to develop non-free programs? Can I use GPL-covered

tools such as GCC to compile them?

Yes, because the copyright on the editors and tools does not cover the code you write. Using them does not place any restrictions, legally, on the license you use for your code.

Some programs copy parts of themselves into the output for technical reasons—for example, Bison copies a standard parser program into its output file. In such cases, the copied text in the output is covered by the same license that covers it in the source code. Meanwhile, the part of the output which is derived from the program's input inherits the copyright status of the input.

As it happens, Bison can also be used to develop non-free programs. This is because we decided to explicitly permit the use of the Bison standard parser program in Bison output files without restriction. We made the decision because there were other tools comparable to Bison which already permitted use for non-free programs.

Do I have “fair use” rights in using the source code of a GPL-covered program?

Yes, you do. “Fair use” is use that is allowed without any special permission. Since you don't need the developers' permission for such use, you can do it regardless of what the developers said about it—in the license or elsewhere, whether that license be the GNU GPL or any other free software license.

Note, however, that there is no world-wide principle of fair use; what kinds of use are considered “fair” varies from country to country.

Can the US Government release a program under the GNU GPL?

If the program is written by US federal government employees in the course of their employment, it is in the public domain, which means it is not copyrighted. Since the GNU GPL is based on copyright, such a program cannot be released under the GNU GPL. (It can still be free software, however; a public domain program is free.)

However, when a US federal government agency uses contractors to develop software, that is a different situation. The contract can require the contractor to release it under the GNU GPL. (GNU Ada was developed in this way.) Or the contract can assign the copyright to the government agency, which can then release the software under the GNU GPL.

Can the US Government release improvements to a GPL-covered program?

Yes. If the improvements are written by US government employees in the course of their employment, then the improvements are in the public domain. However, the improved version, as a whole, is still covered by the GNU GPL. There is no problem in this situation.

If the US government uses contractors to do the job, then the improvements themselves can be GPL-covered.

Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require that these designs must be free?

In general this is legally impossible; copyright law does not give you any say in the use of the output people make from their data using your program. If the user uses your program to enter or convert his own data, the copyright on the output belongs to him, not you. More generally, when a program translates its input into some other form, the copyright status of the output inherits that of the input it was generated from.

So the only way you have a say in the use of the output is if substantial parts of the output are copied (more or less) from text in your program. For instance, part of the output of Bison (see above) would be covered by the GNU GPL, if we had not made an exception in this specific case.

You could artificially make a program copy certain text into its output even if there is no technical reason to do so. But if that copied text serves no practical purpose, the user could simply delete that text from the output and use only the rest. Then he would not have to obey the conditions on redistribution of the copied text.

In what cases is the output of a GPL program covered by the GPL too?

Only when the program copies part of itself into the output.

If I add a module to a GPL-covered program, do I have to use the GPL as the license for my module?

The GPL says that the whole combined program has to be released under the GPL. So your module has to be available for use under the GPL.

But you can give additional permission for the use of your code. You can, if you wish, release your program under a license which is more lax than the GPL but compatible with the GPL. The license list page gives a partial list of GPL-compatible licenses.

If a library is released under the GPL (not the LGPL), does that mean that any program which uses it has to be under the GPL or a GPL-compatible license?

Yes, because the program as it is actually run includes the library.

If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?

When the interpreter just interprets a language, the answer is no. The interpreted program, to the interpreter, is just data; a free software license like the GPL, based on copyright law, cannot limit what data you use the interpreter on. You can run it on any data (interpreted program), any way you like, and there are no requirements about licensing that data to anyone.

However, when the interpreter is extended to provide “bindings” to other facilities (often, but not necessarily, libraries), the interpreted program is effectively linked to the facilities it uses through these bindings. So if these facilities are released under the GPL, the interpreted program that uses them must be released in a GPL-compatible way. The JNI or Java Native Interface is an example of such a binding mechanism; libraries that are accessed in this way are linked dynamically with the Java programs that call them. These libraries are also linked with the interpreter. If the interpreter is linked statically with these libraries, or if it is designed to link dynamically with these specific libraries, then it too needs to be released in a GPL-compatible way.

Another similar and very common case is to provide libraries with the interpreter which are themselves interpreted. For instance, Perl comes with many Perl modules, and a Java implementation comes with many Java classes. These libraries and the programs that call them are always dynamically linked together.

A consequence is that if you choose to use GPL'd Perl modules or Java classes in your program, you must release the program in a GPL-compatible way, regardless of the license used in the Perl or Java interpreter that the combined Perl or Java program will run on.

I'm writing a Windows application with Microsoft Visual C++ (or Visual Basic) and I will be releasing it

under the GPL. Is dynamically linking my program with the Visual C++ (or Visual Basic) run-time library permitted under the GPL?

The GPL permits this because that run-time library normally accompanies the compiler or interpreter you are using. The run-time libraries here are “System Libraries” as GPLv3 defines them, and as such they are not considered part of the Corresponding Source. GPLv2 has a similar exception in section 3.

That doesn't mean it is a good idea to write the program so that it only runs on Windows. Doing so results in a program that is free software but “trapped” by Windows.

Why is the original BSD license incompatible with the GPL?

Because it imposes a specific requirement that is not in the GPL; namely, the requirement on advertisements of the program. Section 6 of GPLv2 states:

> *You may not impose any further restrictions on the recipients' exercise of the rights granted herein.*

GPLv3 says something similar in section 10. The advertising clause provides just such a further restriction, and thus is GPL-incompatible.

The revised BSD license does not have the advertising clause, which eliminates the problem.

If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in?

It depends on how the program invokes its plug-ins. If the program uses fork and exec to invoke plug-ins, then the plug-ins are separate programs, so the license for the main program makes no requirements for them.

If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. This means the plug-ins must be released under the GPL or a GPL-compatible free software license, and that the terms of the GPL must be followed when those plug-ins are distributed.

If the program dynamically links plug-ins, but the communication between them is limited to invoking the ‘main’ function of the plug-in with some options and waiting for it to return, that is a borderline case.

Can I apply the GPL when writing a plug-in for a non-free program?

If the program uses fork and exec to invoke plug-ins, then the plug-ins are separate programs, so the license for the main program makes no requirements for them. So you can use the GPL for a plug-in, and there are no special requirements.

If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. This means that combination of the GPL-covered plug-in with the non-free main program would violate the GPL. However, you can resolve that legal problem by adding an exception to your plug-in's license, giving permission to link it with the non-free main program.

See also the question I am writing free software that uses a non-free library.

Can I release a non-free program that's designed to load a GPL-covered plug-in?

It depends on how the program invokes its plug-ins. For instance, if the program uses *only* simple fork and exec to invoke and communicate with plug-ins, then the plug-ins are separate programs, so the license of the plug-in makes no requirements about the main program.

If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. In order to use the GPL-covered plug-ins, the main program must be released under the GPL or a GPL-compatible free software license, and that the terms of the GPL must be followed when the main program is distributed for use with these plug-ins.

If the program dynamically links plug-ins, but the communication between them is limited to invoking the ‘main’ function of the plug-in with some options and waiting for it to return, that is a borderline case.

Using shared memory to communicate with complex data structures is pretty much equivalent to dynamic linking.

See also the question I am writing free software that uses a non-free library.

You have a GPL'ed program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?

Not exactly. It means you must release your program under a license compatible with the GPL (more precisely, compatible with one or more GPL versions accepted by all the rest of the code in the combination that you link). The combination itself is then available under those GPL versions.

If so, is there any chance I could get a license of your program under the Lesser GPL?

You can ask, but most authors will stand firm and say no. The idea of the GPL is that if you want to include our code in your program, your program must also be free software. It is supposed to put pressure on you to release your program in a way that makes it part of our community.

You always have the legal alternative of not using our code.

How can I allow linking of proprietary modules with my GPL-covered library under a controlled interface only?

Add this text to the license notice of each file in the package, at the end of the text that says the file is distributed under the GNU GPL:

> *Linking ABC statically or dynamically with other modules is making a combined work based on ABC. Thus, the terms and conditions of the GNU General Public License cover the whole combination.*
>
> *In addition, as a special exception, the copyright holders of ABC give you permission to combine ABC program with free software programs or libraries that are released under the GNU LGPL and with independent modules that communicate with ABC solely through the ABCDEF interface. You may copy and distribute such a system following the terms of the GNU GPL for ABC and the licenses of the other code concerned, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code.*
>
> *Note that people who make modified versions of ABC are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception.*

Only the copyright holders for the program can legally authorize this exception. If you wrote the whole program yourself, then assuming your employer or school does not claim the copyright, you are the copyright holder—so you can authorize the exception. But if you want to use parts of other GPL-covered programs by other authors in your code, you cannot authorize the exception for them. You have to get the approval of the copyright holders of those programs.

I have written an application that links with many different components, that have different licenses. I am very confused as to what licensing requirements are placed on my program. Can you please tell me what licenses I may use?

To answer this question, we would need to see a list of each component that your program uses, the license of that component, and a brief (a few sentences for each should suffice) describing how your library uses that component. Two examples would be:

- To make my software work, it must be linked to the FOO library, which is available under the Lesser GPL.
- My software makes a system call (with a command line that I built) to run the BAR program, which is licensed under “the GPL, with a special exception allowing for linking with QUUX”.

What is the difference between an “aggregate” and other kinds of “modified versions”?

An “aggregate” consists of a number of separate programs, distributed together on the same CD-ROM or other media. The GPL permits you to create and distribute an aggregate, even when the licenses of the other software are non-free or GPL-incompatible. The only condition is that you cannot release the aggregate under a license that prohibits users from exercising rights that each program's individual license would grant them.

Where's the line between two separate programs, and one program with two parts? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).

If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program.

By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as combined into a larger program.

Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF? If I hold copyright on a GPL'ed program, should I do this, too? If so, how?

Our lawyers have told us that to be in the best position to enforce the GPL in court against violators, we should keep the copyright status of the program as simple as possible. We do this by asking each contributor to either assign the copyright on his contribution to the FSF, or disclaim copyright on it and thus put it in the public domain.
We also ask individual contributors to get copyright disclaimers from their employers (if any) so that we can be sure those employers won't claim to own the contributions.

Of course, if all the contributors put their code in the public domain, there is no copyright with which to enforce the GPL. So we encourage people to assign copyright on large code contributions, and only put small changes in the public domain.

If you want to make an effort to enforce the GPL on your program, it is probably a good idea for you to follow a similar policy. Please contact <licensing@gnu.org> if you want more information.

Can I modify the GPL and make a modified license?

You can use the GPL terms (possibly modified) in another license provided that you call your license by another name and do not include the GPL preamble, and provided you modify the instructions-for-use at the end enough to make it clearly different in wording and not mention GNU (though the actual procedure you describe may be similar).
If you want to use our preamble in a modified license, please write to <licensing@gnu.org> for permission. For this purpose we would want to check the actual license requirements to see if we approve of them.

Although we will not raise legal objections to your making a modified license in this way, we hope you will think twice and not do it. Such a modified license is almost certainly incompatible with the GNU GPL, and that incompatibility blocks useful combinations of modules. The mere proliferation of different free software licenses is a burden in and of itself.

If I use a piece of software that has been obtained under the GNU GPL, am I allowed to modify the original code into a new program, then distribute and sell that new program commercially?

You are allowed to sell copies of the modified program commercially, but only under the terms of the GNU GPL. Thus, for instance, you must make the source code available to the users of the program as described in the GPL, and they must be allowed to redistribute and modify it as described in the GPL.
These requirements are the condition for including the GPL-covered code you received in a program of your own.

Can I use the GPL for something other than software?

You can apply the GPL to any kind of work, as long as it is clear what constitutes the “source code” for the work. The GPL defines this as the preferred form of the work for making changes in it.

However, for manuals and textbooks, or more generally any sort of work that is meant to teach a subject, we recommend using the GFDL rather than the GPL.

How does the LGPL work with Java?

See this article for details. It works as designed, intended, and expected.

Consider this situation: 1. X releases V1 of a project under the GPL. 2. Y contributes to the development of V2 with changes and new code based on V1. 3. X wants to convert V2 to a non-GPL license. Does X need Y's permission?

Yes. Y was required to release its version under the GNU GPL, as a consequence of basing it on X's version V1. Nothing required Y to agree to any other license for its code. Therefore, X must get Y's permission before releasing that code under another license.

I'd like to incorporate GPL-covered software in my proprietary system. Can I do this?

You cannot incorporate GPL-covered software in a proprietary system. The goal of the GPL is to grant everyone the freedom to copy, redistribute, understand, and modify a program. If you could incorporate GPL-covered software into a non-free system, it would have the effect of making the GPL-covered software non-free too.
A system incorporating a GPL-covered program is an extended version of that program. The GPL says that any extended version of the program must be released under the GPL if it is released at all. This is for two reasons: to make sure that users who get the software get the freedom they should have, and to encourage people to give back improvements that they make.

However, in many cases you can distribute the GPL-covered software alongside your proprietary system. To do this validly, you must make sure that the free and non-free programs communicate at arms length, that they are not combined in a way that would make them effectively a single program.

The difference between this and “incorporating” the GPL-covered software is partly a matter of substance and partly form. The substantive part is this: if the two programs are combined so that they become effectively two parts of one program, then you can't treat them as two separate programs. So the GPL has to cover the whole thing.

If the two programs remain well separated, like the compiler and the kernel, or like an editor and a shell, then you can treat them as two separate programs—but you have to do it properly. The issue is simply one of form: how you describe what you are doing. Why do we care about this? Because we want to make sure the users clearly understand the free status of the GPL-covered software in the collection.

If people were to distribute GPL-covered software calling it “part of” a system that users know is partly proprietary, users might be uncertain of their rights regarding the GPL-covered software. But if they know that what they have received is a free program plus another program, side by side, their rights will be clear.

I'd like to incorporate GPL-covered software in my proprietary system. Can I do this by putting a “wrapper” module, under a GPL-compatible lax permissive license (such as the X11 license) in between the GPL-covered part and the proprietary part?

No. The X11 license is compatible with the GPL, so you can add a module to the GPL-covered program and put it under the X11 license. But if you were to incorporate them both in a larger program, that whole would include the GPL-covered part, so it would have to be licensed *as a whole* under the GNU GPL.

The fact that proprietary module A communicates with GPL-covered module C only through X11-licensed module B is legally irrelevant; what matters is the fact that module C is included in the whole.

Does the libstdc++ exception permit dynamic linking?

Yes. The intent of the exception is to allow people to compile proprietary software using gcc.

I'd like to modify GPL-covered programs and link them with the portability libraries from Money Guzzler Inc. I cannot distribute the source code for these libraries, so any user who wanted to change these versions would have to obtained those libraries separately. Why doesn't the GPL permit this?

There are two reasons for this.
First, a general one. If we permitted company A to make a proprietary file, and company B to distribute GPL-covered software linked with that file, the effect would be to make a hole in the GPL big enough to drive a truck through. This would be carte blanche for withholding the source code for all sorts of modifications and extensions to GPL-covered software.

Giving all users access to the source code is one of our main goals, so this consequence is definitely something we want to avoid.

More concretely, the versions of the programs linked with the Money Guzzler libraries would not really be free software as we understand the term—they would not come with full source code that enables users to change and recompile the program.

If license for a module Q has a requirement that's incompatible with the GPL, but the requirement applies only when Q is distributed by itself, not when Q is included in a larger program, does that make the license GPL-compatible? Can I combine or link Q with a GPL-covered program?

If a program P is released under the GPL that means *any and every part of it* can be used under the GPL. If you integrate module Q, and release the combined program P+Q under the GPL, that means any part of P+Q can be used under the GPL. One part of P+Q is Q. So releasing P+Q under the GPL says that Q any part of it can be used under the GPL. Putting it in other words, a user who obtains P+Q under the GPL can delete P, so that just Q remains, still under the GPL.
If the license of module Q permits you to give permission for that, then it is GPL-compatible. Otherwise, it is not GPL-compatible.

If the license for Q says in no uncertain terms that you must do certain things (not compatible with the GPL) when you redistribute Q on its own, then it does not permit you to distribute Q under the GPL. It follows that you can't release P+Q under the GPL either. So you cannot link or combine P with Q.

Can I release a modified version of a GPL-covered program in binary form only?

No. The whole point of the GPL is that all modified versions must be free software—which means, in particular, that the source code of the modified version is available to the users.

I downloaded just the binary from the net. If I distribute copies, do I have to get the source and distribute that too?

Yes. The general rule is, if you distribute binaries, you must distribute the complete corresponding source code too. The exception for the case where you received a written offer for source code is quite limited.

I want to distribute binaries via physical media without accompanying sources. Can I provide source code by FTP?

Version 3 of the GPL allows this; see option 6(b) for the full details. Under version 2, you're certainly free to offer source via FTP, and most users will get it from there. However, if any of them would rather get the source on physical media by mail, you are required to provide that.

If you distribute binaries via FTP, you should distribute source via FTP.

My friend got a GPL-covered binary with an offer to supply source, and made a copy for me. Can I use the offer myself to obtain the source?

Yes, you can. The offer must be open to everyone who has a copy of the binary that it accompanies. This is why the GPL says your friend must give you a copy of the offer along with a copy of the binary—so you can take advantage of it.

Can I put the binaries on my Internet server and put the source on a different Internet site?

Yes. Section 6(d) allows this. However, you must provide clear instructions people can follow to obtain the source, and you must take care to make sure that the source remains available for as long as you distribute the object code.

I want to distribute an extended version of a GPL-covered program in binary form. Is it enough to distribute the source for the original version?

No, you must supply the source code that corresponds to the binary. Corresponding source means the source from which users can rebuild the same binary.

Part of the idea of free software is that users should have access to the source code for *the programs they use*. Those using your version should have access to the source code for your version.

A major goal of the GPL is to build up the Free World by making sure that improvement to a free program are themselves free. If you release an improved version of a GPL-covered program, you must release the improved source code under the GPL.

I want to distribute binaries, but distributing complete source is inconvenient. Is it ok if I give users the diffs from the “standard” version along with the binaries?

This is a well-meaning request, but this method of providing the source doesn't really do the job.

A user that wants the source a year from now may be unable to get the proper version from another site at that time. The standard distribution site may have a newer version, but the same diffs probably won't work with that version.

So you need to provide complete sources, not just diffs, with the binaries.

Can I make binaries available on a network server, but send sources only to people who order them?

If you make object code available on a network server, you have to provide the Corresponding Source on a network server as well. The easiest way to do this would be to publish them on the same server, but if you'd like, you can alternatively provide instructions for getting the source from another server, or even a version control system. No matter what you do, the source should be just as easy to access as the object code, though. This is all specified in section 6(d) of GPLv3.

The sources you provide must correspond exactly to the binaries. In particular, you must make sure they are for the same version of the program—not an older version and not a newer version.

How can I make sure each user who downloads the binaries also gets the source?

You don't have to make sure of this. As long as you make the source and binaries available so that the users can see what's available and take what they want, you have done what is required of you. It is up to the user whether to download the source.

Our requirements for redistributors are intended to make sure the users can get the source code, not to force users to download the source code even if they don't want it.

A company is running a modified version of a GPL'ed program on a web site. Does the GPL say they must release their modified sources?

The GPL permits anyone to make a modified version and use it without ever distributing it to others. What this company is doing is a special case of that. Therefore, the company does not have to release the modified sources.

It is essential for people to have the freedom to make modifications and use them privately, without ever publishing those modifications. However, putting the program on a server machine for the public to talk to is hardly “private” use, so it would be legitimate to require release of the source code in that special case. Developers who wish to address this might want to use the GNU Affero GPL for programs designed for network server use.

Is making and using multiple copies within one organization or company “distribution”?

No, in that case the organization is just making the copies for itself. As a consequence, a company or other organization can develop a modified version and install that version through its own facilities, without giving the staff permission to release that modified version to outsiders.

However, when the organization transfers copies to other organizations or individuals, that is distribution. In particular, providing copies to contractors for use off-site is distribution.

If someone steals a CD containing a version of a GPL-covered program, does the GPL give him the right to redistribute that version?

If the version has been released elsewhere, then the thief probably does have the right to make copies and redistribute them under the GPL, but if he is imprisoned for stealing the CD he may have to wait until his release before doing so.

If the version in question is unpublished and considered by a company to be its trade secret, then publishing it may be a violation of trade secret law, depending on other circumstances. The GPL does not change that. If the company tried to release its version and still treat it as a trade secret, that would violate the GPL, but if the company hasn't released this version, no such violation has occurred.

What if a company distributes a copy as a trade secret?

If a company distributes a copy to you and claims it is a trade secret, the company has violated the GPL and will have to cease distribution. Note how this differs from the theft case above; the company does not intentionally distribute a copy when a copy is stolen, so in that case the company has not violated the GPL.

Why are some GNU libraries released under the ordinary GPL rather than the Lesser GPL?

Using the Lesser GPL for any particular library constitutes a retreat for free software. It means we partially abandon the attempt to defend the users' freedom, and some of the requirements to share what is built on top of GPL-covered software. In themselves, those are changes for the worse.
Sometimes a localized retreat is a good strategy. Sometimes, using the LGPL for a library might lead to wider use of that library, and thus to more improvement for it, wider support for free software, and so on. This could be good for free software if it happens to a large extent. But how much will this happen? We can only speculate.

It would be nice to try out the LGPL on each library for a while, see whether it helps, and change back to the GPL if the LGPL didn't help. But this is not feasible. Once we use the LGPL for a particular library, changing back would be difficult.

So we decide which license to use for each library on a case-by-case basis. There is a long explanation of how we judge the question.

Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program.

Sorry, we don't make such exceptions. It would not be right.
Maximizing the number of users is not our aim. Rather, we are trying to give the crucial freedoms to as many users as possible. In general, proprietary software projects hinder rather than help the cause of freedom.

We do occasionally make license exceptions to assist a project which is producing free software under a license other than the GPL. However, we have to see a good reason why this will advance the cause of free software.

We also do sometimes change the distribution terms of a package, when that seems clearly the right way to serve the cause of free software; but we are very cautious about this, so you will have to show us very convincing reasons.

Why should programs say “Version 3 of the GPL or any later version”?

From time to time, at intervals of years, we change the GPL—sometimes to clarify it, sometimes to permit certain kinds of use not previously permitted, and sometimes to tighten up a requirement. (The last two changes were in 2007 and 1991.) Using this “indirect pointer” in each program makes it possible for us to change the distribution terms on the entire collection of GNU software, when we update the GPL.

If each program lacked the indirect pointer, we would be forced to discuss the change at length with numerous copyright holders, which would be a virtual impossibility. In practice, the chance of having uniform distribution terms for GNU software would be nil.

Suppose a program says “Version 3 of the GPL or any later version” and a new version of the GPL is released. If the new GPL version gives additional permission, that permission will be available immediately to all the users of the program. But if the new GPL version has a tighter requirement, it will not restrict use of the current version of the program, because it can still be used under GPL version 3. When a program says “Version 3 of the GPL or any later version”, users will always be permitted to use it, and even change it, according to the terms of GPL version 3—even after later versions of the GPL are available.

If a tighter requirement in a new version of the GPL need not be obeyed for existing software, how is it useful? Once GPL version 4 is available, the developers of most GPL-covered programs will release subsequent versions of their programs specifying “Version 4 of the GPL or any later version”. Then users will have to follow the tighter requirements in GPL version 4, for subsequent versions of the program.

However, developers are not obligated to do this; developers can continue allowing use of the previous version of the GPL, if that is their preference.

Why don't you use the GPL for manuals?

It is possible to use the GPL for a manual, but the GNU Free Documentation License (GFDL) is much better for manuals.

The GPL was designed for programs; it contains lots of complex clauses that are crucial for programs, but that would be cumbersome and unnecessary for a book or manual. For instance, anyone publishing the book on paper would have to either include machine-readable “source code” of the book along with each printed copy, or provide a written offer to send the “source code” later.

Meanwhile, the GFDL has clauses that help publishers of free manuals make a profit from selling copies—cover texts, for instance. The special rules for Endorsements sections make it possible to use the GFDL for an official standard. This would permit modified versions, but they could not be labeled as “the standard”.

Using the GFDL, we permit changes in the text of a manual that covers its technical topic. It is important to be able to change the technical parts, because people who change a program ought to change the documentation to correspond. The freedom to do this is an ethical imperative.

Our manuals also include sections that state our political position about free software. We mark these as “invariant”, so that they cannot be changed or removed. The GFDL makes provisions for these “invariant sections”.

How does the GPL apply to fonts?

Font licensing is a complex issue which needs serious consideration. The following license exception is experimental but approved for general use. We welcome suggestions on this subject—please see this this explanatory essay and write to licensing@gnu.org.

To use this exception, add this text to the license notice of each file in the package (to the extent possible), at the end of the text that says the file is distributed under the GNU GPL:

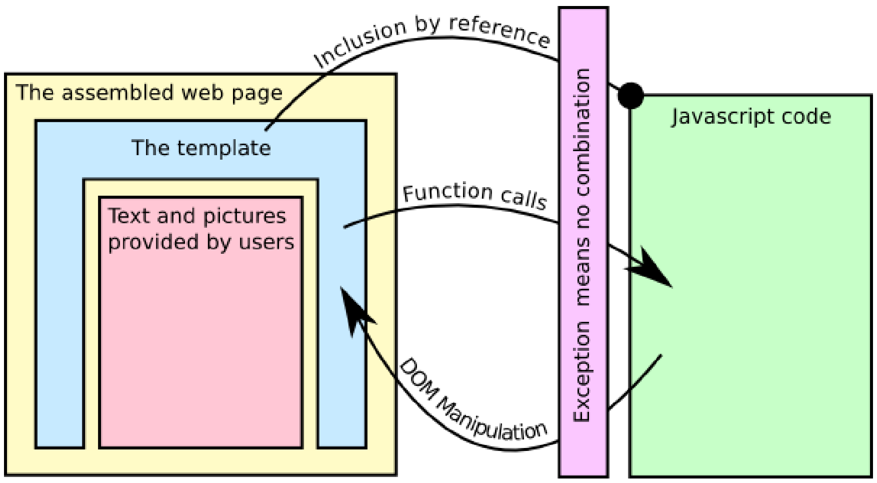
> *As a special exception, if you create a document which uses this font, and embed this font or unaltered portions of this font into the document, this font does not by itself cause the resulting document to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the document might be covered by the GNU General Public License. If you modify this font, you may extend this exception to your version of the font, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.*

I am writing a website maintenance system (called a “content management system” by some), or some other application which generates web pages from templates. What license should I use for those templates?

Templates are minor enough that it is not worth using copyleft to protect them. It is normally harmless to use copyleft on minor works, but templates are a special case, because they are combined with data provided by users of the application and the combination is distributed. So, we recommend that you license your templates under simple permissive terms.

Some templates make calls into Javascript functions. Since Javascript is often non-trivial, it is worth copylefting.

Because the templates will be combined with user data, it's possible that template+user data+Javascript would be considered one work under copyright law. A line needs to be drawn between the Javascript (copylefted), and the user code (usually under incompatible terms).



Here's an exception for Javascript code that does this:

> *As a special exception to the GPL, any HTML file which merely makes function calls to this code, and for that purpose includes it by reference shall be deemed a separate work for copyright law purposes. In addition, the copyright holders of this code give you permission to combine this code with free software libraries that are released under the GNU LGPL. You may copy and distribute such a system following the terms of the GNU GPL for this code and the LGPL for the libraries. If you modify this code, you may extend this exception to your version of the code, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.*

Can I release a program under the GPL which I developed using non-free tools?

Which programs you used to edit the source code, or to compile it, or study it, or record it, usually makes no difference for issues concerning the licensing of that source code.

However, if you link non-free libraries with the source code, that would be an issue you need to deal with. It does not preclude releasing the source code under the GPL, but if the libraries don't fit under the “system library” exception, you should affix an explicit notice giving permission to link your program with them. The FSF can give you advice on doing this.

Are there translations of the GPL into other languages?

It would be useful to have translations of the GPL into languages other than English. People have even written translations and sent them to us. But we have not dared to approve them as officially valid. That carries a risk so great we do not dare accept it.

A legal document is in some ways like a program. Translating it is like translating a program from one language and operating system to another. Only a lawyer skilled in both languages can do it—and even then, there is a risk of introducing a bug.

If we were to approve, officially, a translation of the GPL, we would be giving everyone permission to do whatever the translation says they can do. If it is a completely accurate translation, that is fine. But if there is an error in the translation, the results could be a disaster which we could not fix.

If a program has a bug, we can release a new version, and eventually the old version will more or less disappear. But once we have given everyone permission to act according to a particular translation, we have no way of taking back that permission if we find, later on, that it had a bug.

Helpful people sometimes offer to do the work of translation for us. If the problem were a matter of finding someone to do the work, this would solve it. But the actual problem is the risk of error, and offering to do the work does not avoid the risk. We could not possibly authorize a translation written by a non-lawyer.

Therefore, for the time being, we are not approving translations of the GPL as globally valid and binding. Instead, we are doing two things:

- Referring people to unofficial translations. This means that we permit people to write translations of the GPL, but we don't approve them as legally valid and binding.

  An unapproved translation has no legal force, and it should say so explicitly. It should be marked as follows:

  > *This translation of the GPL is informal, and not officially approved by the Free Software Foundation as valid. To be completely sure of what is permitted, refer to the original GPL (in English).*

  But the unapproved translation can serve as a hint for how to understand the English GPL. For many users, that is sufficient.

  However, businesses using GNU software in commercial activity, and people doing public ftp distribution, should need to check the real English GPL to make sure of what it permits.

- Publishing translations valid for a single country only.

  We are considering the idea of publishing translations which are officially valid only for one country. This way, if there is a mistake, it will be limited to that country, and the damage will not be too great.

  It will still take considerable expertise and effort from a sympathetic and capable lawyer to make a translation, so we cannot promise any such translations soon.

If a programming language interpreter has a license that is incompatible with the GPL, can I run GPL-covered programs on it?

When the interpreter just interprets a language, the answer is yes. The interpreted program, to the interpreter, is just data; the GPL doesn't restrict what tools you process the program with.

However, when the interpreter is extended to provide “bindings” to other facilities (often, but not necessarily, libraries), the interpreted program is effectively linked to the facilities it uses through these bindings. The JNI or Java Native Interface is an example of such a facility; libraries that are accessed in this way are linked dynamically with the Java programs that call them.

So if these facilities are released under a GPL-incompatible license, the situation is like linking in any other way with a GPL-incompatible library. Which implies that:

1. If you are writing code and releasing it under the GPL, you can state an explicit exception giving permission to link it with those GPL-incompatible facilities.
2. If you wrote and released the program under the GPL, and you designed it specifically to work with those facilities, people can take that as an implicit exception permitting them to link it with those facilities. But if that is what you intend, it is better to say so explicitly.
3. You can't take someone else's GPL-covered code and use it that way, or add such exceptions to it. Only the

copyright holders of that code can add the exception.

Who has the power to enforce the GPL?

Since the GPL is a copyright license, the copyright holders of the software are the ones who have the power to enforce the GPL. If you see a violation of the GPL, you should inform the developers of the GPL-covered software involved. They either are the copyright holders, or are connected with the copyright holders. Learn more about reporting GPL violations.

In an object-oriented language such as Java, if I use a class that is GPL'ed without modifying, and subclass it, in what way does the GPL affect the larger program?

Subclassing is creating a derivative work. Therefore, the terms of the GPL affect the whole program where you create a subclass of a GPL'ed class.

If I port my program to GNU/Linux, does that mean I have to release it as Free Software under the GPL or some other Free Software license?

In general, the answer is no—this is not a legal requirement. In specific, the answer depends on which libraries you want to use and what their licenses are. Most system libraries either use the GNU Lesser GPL, or use the GNU GPL plus an exception permitting linking the library with anything. These libraries can be used in non-free programs; but in the case of the Lesser GPL, it does have some requirements you must follow.

Some libraries are released under the GNU GPL alone; you must use a GPL-compatible license to use those libraries. But these are normally the more specialized libraries, and you would not have had anything much like them on another platform, so you probably won't find yourself wanting to use these libraries for simple porting.

Of course, your software is not a contribution to our community if it is not free, and people who value their freedom will refuse to use it. Only people willing to give up their freedom will use your software, which means that it will effectively function as an inducement for people to lose their freedom.

If you hope some day to look back on your career and feel that it has contributed to the growth of a good and free society, you need to make your software free.

I just found out that a company has a copy of a GPL'ed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?

No. The GPL does not require anyone to use the Internet for distribution. It also does not require anyone in particular to redistribute the program. And (outside of one special case), even if someone does decide to redistribute the program sometimes, the GPL doesn't say he has to distribute a copy to you in particular, or any other person in particular.

What the GPL requires is that he must have the freedom to distribute a copy to you *if he wishes to*. Once the copyright holder does distribute a copy program to someone, that someone can then redistribute the program to you, or to anyone else, as he sees fit.

Can I release a program with a license which says that you can distribute modified versions of it under the GPL but you can't distribute the original itself under the GPL?

No. Such a license would be self-contradictory. Let's look at its implications for me as a user.

Suppose I start with the original version (call it version A), add some code (let's imagine it is 1000 lines), and release that modified version (call it B) under the GPL. The GPL says anyone can change version B again and release the result under the GPL. So I (or someone else) can delete those 1000 lines, producing version C which has the same code as version A but is under the GPL.

If you try to block that path, by saying explicitly in the license that I'm not allowed to reproduce something identical to version A under the GPL by deleting those lines from version B, in effect the license now says that I can't fully use version B in all the ways that the GPL permits. In other words, the license does not in fact allow a user to release a modified version such as B under the GPL.

Does moving a copy to a majority-owned, and controlled, subsidiary constitute distribution?

Whether moving a copy to or from this subsidiary constitutes “distribution” is a matter to be decided in each case under the copyright law of the appropriate jurisdiction. The GPL does not and cannot override local laws. US copyright law is not entirely clear on the point, but appears not to consider this distribution.

If, in some country, this is considered distribution, and the subsidiary must receive the right to redistribute the program, that will not make a practical difference. The subsidiary is controlled by the parent company; rights or no rights, it won't redistribute the program unless the parent company decides to do so.

Can software installers ask people to click to agree to the GPL? If I get some software under the GPL, do I have to agree to anything?

Some software packaging systems have a place which requires you to click through or otherwise indicate assent to the terms of the GPL. This is neither required nor forbidden. With or without a click through, the GPL's rules remain the same.

Merely agreeing to the GPL doesn't place any obligations on you. You are not required to agree to anything to merely use software which is licensed under the GPL. You only have obligations if you modify or distribute the software. If it really bothers you to click through the GPL, nothing stops you from hacking the GPLed software to bypass this.

I would like to bundle GPLed software with some sort of installation software. Does that installer need to have a GPL-compatible license?

No. The installer and the files it installs are separate works. As a result, the terms of the GPL do not apply to the installation software.

Can I use GPLed software on a device that will stop operating if customers do not continue paying a subscription fee?

No. In this scenario, the requirement to keep paying a fee limits the user's ability to run the program. This is an additional requirement on top of the GPL, and the license prohibits it.

How do I upgrade from (L)GPLv2 to (L)GPLv3?

First, include the new version of the license in your package. If you're using LGPLv3 in your project, be sure to include copies of both GPLv3 and LGPLv3, since LGPLv3 is now written as a set of additional permissions on top of

GPLv3.

Second, replace all your existing v2 license notices (usually at the top of each file) with the new recommended text available on the GNU licenses howto. It's more future-proof because it no longer includes the FSF's postal mailing address.

Of course, any descriptive text (such as in a README) which talks about the package's license should also be updated appropriately.

How does GPLv3 make BitTorrent distribution easier?

Because GPLv2 was written before peer-to-peer distribution of software was common, it is difficult to meet its requirements when you share code this way. The best way to make sure you are in compliance when distributing GPLv2 object code on BitTorrent would be to include all the corresponding source in the same torrent, which is prohibitively expensive.

GPLv3 addresses this problem in two ways. First, people who download this torrent and send the data to others as part of that process are not required to do anything. That's because section 9 says “Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance [of the license].”

Second, section 6(e) of GPLv3 is designed to give distributors—people who initially seed torrents—a clear and straightforward way to provide the source, by telling recipients where it is available on a public network server. This ensures that everyone who wants to get the source can do so, and it's almost no hassle for the distributor.

What is tivoization? How does GPLv3 prevent it?

Some devices utilize free software that can be upgraded, but are designed so that users are not allowed to modify that software. There are lots of different ways to do this; for example, sometimes the hardware checksums the software that is installed, and shuts down if it doesn't match an expected signature. The manufacturers comply with GPLv2 by giving you the source code, but you still don't have the freedom to modify the software you're using. We call this practice tivoization.

When people distribute User Products that include software under GPLv3, section 6 requires that they provide you with information necessary to modify that software. User Products is a term specially defined in the license; examples of User Products include portable music players, digital video recorders, and home security systems.

Does GPLv3 prohibit DRM?

It does not; you can use code released under GPLv3 to develop any kind of DRM technology you like. However, if you do this, section 3 says that the system will not count as an effective technological “protection” measure, which means that if someone breaks the DRM, he will be free to distribute his software too, unhindered by the DMCA and similar laws.

As usual, the GNU GPL does not restrict what people do in software, it just stops them from restricting others.

Can I use the GPL to license hardware?

Any material that can be copyrighted can be licensed under the GPL. GPLv3 can also be used to license materials covered by other copyright-like laws, such as semiconductor masks. So, as an example, you can release a drawing of a hardware design under the GPL. However, if someone used that information to create physical hardware, they would have no license obligations when distributing or selling that device: it falls outside the scope of copyright and thus the GPL itself.

I use public key cryptography to sign my code to assure its authenticity. Is it true that GPLv3 forces me to release my private signing keys?

No. The only time you would be required to release signing keys is if you conveyed GPLed software inside a User Product, and its hardware checked the software for a valid cryptographic signature before it would function. In that specific case, you would be required to provide anyone who owned the device, on demand, with the key to sign and install modified software on his device so that it will run. If each instance of the device uses a different key, then you need only give each purchaser the key for his instance.

Does GPLv3 require that voters be able to modify the software running in a voting machine?

No. Companies distributing devices that include software under GPLv3 are at most required to provide the source and Installation Information for the software to people who possess a copy of the object code. The voter who uses a voting machine (like any other kiosk) doesn't get possession of it, not even temporarily, so the voter also does not get possession of the binary software in it.

Note, however, that voting is a very special case. Just because the software in a computer is free does not mean you can trust the computer for voting. We believe that computers cannot be trusted for voting. Voting should be done on paper.

Does GPLv3 have a “patent retaliation clause”?

In effect, yes. Section 10 prohibits people who convey the software from filing patent suits against other licensees. If someone did so anyway, section 8 explains how they would lose their license and any patent licenses that accompanied it.

Can I use snippets of GPL-covered source code within documentation that is licensed under some license that is incompatible with the GPL?

If the snippets are small enough that you can incorporate them under fair use or similar laws, then yes. Otherwise, no.

The beginning of GPLv3 section 6 says that I can convey a covered work in object code form “under the terms of sections 4 and 5” provided I also meet the conditions of section 6. What does that mean?

This means that all the permissions and conditions you have to convey source code also apply when you convey object code: you may charge a fee, you must keep copyright notices intact, and so on.

My company owns a lot of patents. Over the years we've contributed code to projects under “GPL version 2 or any later version”, and the project itself has been distributed under the same terms. If a user decides to take the project's code (incorporating my contributions) under GPLv3, does that mean I've automatically granted GPLv3's explicit patent license to that user?

No. When you convey GPLed software, you must follow the terms and conditions of one particular version of the license. When you do so, that version defines the obligations you have. If users may also elect to use later versions of the GPL, that's merely an additional permission they have—it does not require you to fulfill the terms of the later version of the GPL as well.

Do not take this to mean that you can threaten the community with your patents. In many countries, distributing software under GPLv2 provides recipients with an implicit patent license to exercise their rights under the GPL. Even if it didn't, anyone considering enforcing their patents aggressively is an enemy of the community, and we will defend ourselves against such an attack.

If I distribute a proprietary program that links against an LGPLv3-covered library that I've modified, what is the “contributor version” for purposes of determining the scope of the explicit patent license grant I'm making—is it just the library, or is it the whole combination?

The “contributor version” is only your version of the library.

Is GPLv3 compatible with GPLv2?

No. Some of the requirements in GPLv3, such as the requirement to provide Installation Information, do not exist in GPLv2. As a result, the licenses are not compatible: if you tried to combine code released under both these licenses, you would violate section 6 of GPLv2.

However, if code is released under GPL “version 2 or later,” that is compatible with GPLv3 because GPLv3 is one of the options it permits.

What does it mean to “cure” a violation of GPLv3?

To cure a violation means to adjust your practices to comply with the requirements of the license.

The warranty and liability disclaimers in GPLv3 seem specific to U.S. law. Can I add my own disclaimers to my own code?

Yes. Section 7 gives you permission to add your own disclaimers, specifically 7(a).

My program has interactive user interfaces that are non-visual in nature. How can I comply with the Appropriate Legal Notices requirement in GPLv3?

All you need to do is ensure that the Appropriate Legal Notices are readily available to the user in your interface. For example, if you have written an audio interface, you could include a command that reads the notices aloud.

If I give a copy of a GPLv3-covered program to a coworker at my company, have I “conveyed” the copy to him?

As long as you're both using the software in your work at the company, rather than personally, then the answer is no. The copies belong to the company, not to you or the coworker. This copying is propagation, not conveying, because the company is not making copies available to others.

If I distribute a GPLv3-covered program, can I provide a warranty that is voided if the user modifies the program?

Yes. Just as devices do not need to be warranted if users modify the software inside them, you are not required to provide a warranty that covers all possible activities someone could undertake with GPLv3-covered software.

Why did you decide to write the GNU Affero GPLv3 as a separate license?

Early drafts of GPLv3 allowed licensors to add an Affero-like requirement to publish source in section 7. However, some companies that develop and rely upon free software consider this requirement to be too burdensome. They want to avoid code with this requirement, and expressed concern about the administrative costs of checking code for this additional requirement. By publishing the GNU Affero GPLv3 as a separate license, with provisions in it and GPLv3 to allow code under these licenses to link to each other, we accomplish all of our original goals while making it easier to determine which code has the source publication requirement.

Why did you invent the new terms “propagate” and “convey” in GPLv3?

The term “distribute” used in GPLv2 was borrowed from United States copyright law. Over the years, we learned that some jurisdictions used this same word in their own copyright laws, but gave it different meanings. We invented these new terms to make our intent as clear as possible no matter where the license is interpreted. They are not used in any copyright law in the world, and we provide their definitions directly in the license.

I'd like to license my code under the GPL, but I'd also like to make it clear that it can't be used for military and/or commercial uses. Can I do this?

No, because those two goals contradict each other. The GNU GPL is designed specifically to prevent the addition of further restrictions. GPLv3 allows a very limited set of them, in section 7, but any other added restriction can be removed by the user.

Is “convey” in GPLv3 the same thing as what GPLv2 means by “distribute”?

Yes, more or less. During the course of enforcing GPLv2, we learned that some jurisdictions used the word “distribute” in their own copyright laws, but gave it different meanings. We invented a new term to make our intent clear and avoid any problems that could be caused by these differences.

GPLv3 gives “making available to the public” as an example of propagation. What does this mean? Is making available a form of conveying?

One example of “making available to the public” is putting the software on a public web or FTP server. After you do this, some time may pass before anybody actually obtains the software from you—but because it could happen right away, you need to fulfill the GPL's obligations right away as well. Hence, we defined conveying to include this activity.

Since distribution and making available to the public are forms of propagation that are also conveying in GPLv3, what are some examples of propagation that do not constitute conveying?

Making copies of the software for yourself is the main form of propagation that is not conveying. You might do this to install the software on multiple computers, or to make backups.

Does prelinking a GPLed binary to various libraries on the system, to optimize its performance, count as modification?

No. Prelinking is part of a compilation process; it doesn't introduce any license requirements above and beyond what other aspects of compilation would. If you're allowed to link the program to the libraries at all, then it's fine to prelink with them as well. If you distribute prelinked object code, you need to follow the terms of section 6.

If someone installs GPLed software on a laptop, and then lends that laptop to a friend without providing source code for the software, have they violated the GPL?

No. In the jurisdictions where we have investigated this issue, this sort of loan would not count as conveying. The laptop's owner would not have any obligations under the GPL.

Suppose that two companies try to circumvent the requirement to provide Installation Information by having one company release signed software, and the other release a User Product that only runs signed software from the first company. Is this a violation of GPLv3?

Yes. If two parties try to work together to get around the requirements of the GPL, they can both be pursued for copyright infringement. This is especially true since the definition of convey explicitly includes activities that would make someone responsible for secondary infringement.

Am I complying with GPLv3 if I offer binaries on an FTP server and sources by way of a link to a source code repository in a version control system, like CVS or Subversion?

This is acceptable as long as the source checkout process does not become burdensome or otherwise restrictive. Anybody who can download your object code should also be able to check out source from your version control system, using a publicly available free software client. Users should be provided with clear and convenient instructions for how to get the source for the exact object code they downloaded—they may not necessarily want the latest development code, after all.

Can someone who conveys GPLv3-covered software in a User Product use remote attestation to prevent a user from modifying that software?

No. The definition of Installation Information, which must be provided with source when the software is conveyed inside a User Product, explicitly says: “The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.” If the device uses remote attestation in some way, the Installation Information must provide you some means for your modified software to report itself as legitimate.

What does “rules and protocols for communication across the network” mean in GPLv3?

This refers to rules about traffic you can send over the network. For example, if there is a limit on the number of requests you can send to a server per day, or the size of a file you can upload somewhere, your access to those resources may be denied if you do not respect those limits.

These rules do not include anything that does not pertain directly to data traveling across the network. For instance, if a server on the network sent messages for users to your device, your access to the network could not be denied merely because you modified the software so that it did not display the messages.

Distributors that provide Installation Information under GPLv3 are not required to provide "support service" for the product. What kind of "support service"do you mean?

This includes the kind of service many device manufacturers provide to help you install, use, or troubleshoot the product. If a device relies on access to web services or similar technology to function properly, those should normally still be available to modified versions, subject to the terms in section 6 regarding access to a network.

In GPLv3 and AGPLv3, what does it mean when it says "notwithstanding any other provision of this License"?

This simply means that the following terms prevail over anything else in the license that may conflict with them. For example, without this text, some people might have claimed that you could not combine code under GPLv3 with code under AGPLv3, because the AGPL's additional requirements would be classified as "further restrictions" under section 7 of GPLv3. This text makes clear that our intended interpretation is the correct one, and you can make the combination.

This text only resolves conflicts between different terms of the license. When there is no conflict between two conditions, then you must meet them both. These paragraphs don't grant you carte blanche to ignore the rest of the license—instead they're carving out very limited exceptions.

Under AGPLv3, when I modify the Program under section 13, what Corresponding Source does it have to offer?

"Corresponding Source" is defined in section 1 of the license, and you should provide what it lists. So, if your modified version depends on libraries under other licenses, such as the Expat license or GPLv3, the Corresponding Source should include those libraries (unless they are System Libraries).

The last sentence of the first paragraph of section 13 is only meant to reinforce what most people would have naturally assumed: even though combinations with code under GPLv3 are handled through a special exception in section 13, the Corresponding Source should still include the code that is combined with the Program this way. This sentence does not mean that you *only* have to provide the source that's covered under GPLv3; instead it means that such code is *not* excluded from the definition of Corresponding Source.

In AGPLv3, what counts as "interacting with [the software] remotely through a computer network?"

If the program is expressly designed to accept user requests and send responses over a network, then it meets these criteria. Common examples of programs that would fall into this category include web and mail servers, interactive web-based applications, and servers for games that are played online.

If a program is not expressly designed to interact with a user through a network, but is being run in an environment where it happens to do so, then it does not fall into this category. For example, an application is not required to provide source merely because the user is running it over SSH, or a remote X session.

How does GPLv3's concept of "you" compare to the definition of "Legal Entity" in the Apache License 2.0?

They're effectively identical. The definition of “Legal Entity” in the Apache License 2.0 is very standard in various kinds of legal agreements—so much so that it would be very surprising if a court did not interpret the term in the same way in the absence of an explicit definition. We fully expect them to do the same when they look at GPLv3 and consider who qualifies as a licensee.

In GPLv3, what does “the Program” refer to? Is it every program ever released under GPLv3?

The term “the Program” means one particular work that is licensed under GPLv3 and is received by a particular licensee from an upstream licensor or distributor. The Program is the particular work of software that you received in a given instance of GPLv3 licensing, as you received it.

“The Program” cannot mean “all the works ever licensed under GPLv3”; that interpretation makes no sense for a number of reasons. We've published an analysis of the term “the Program” for those who would like to learn more about this.

If I only make copies of a GPL-covered program and run them, without distributing or conveying them to others, what does the license require of me?

Nothing. The GPL does not place any conditions on this activity.

If some network client software is released under AGPLv3, does it have to be able to provide source to the servers it interacts with?

This should not be required in any typical server-client relationship. AGPLv3 requires a program to offer source code to “all users interacting with it remotely through a computer network.” In most server-client architectures, it simply wouldn't be reasonable to argue that the server operator is a “user” interacting with the client in any meaningful sense.

Consider HTTP as an example. All HTTP clients expect servers to provide certain functionality: they should send specified responses to well-formed requests. The reverse is not true: servers cannot assume that the client will do anything in particular with the data they send. The client may be a web browser, an RSS reader, a spider, a network monitoring tool, or some special-purpose program. The server can make absolutely no assumptions about what the client will do—so there's no meaningful way for the server operator to be considered a user of that software.

How are the various GNU licenses compatible with each other?

The various GNU licenses enjoy broad compatibility between each other. The only time you may not be able to combine code under two of these licenses is when you want to use code that's *only* under an older version of a license with code that's under a newer version.

Below is a detailed compatibility matrix for various combinations of the GNU licenses, to provide an easy-to-use reference for specific cases. It assumes that someone else has written some software under one of these licenses, and you want to somehow incorporate code from that into a project that you're releasing (either your own original work, or a modified version of someone else's GPLed software). Find the license for your own work in a column at the top of the table, and the license for the other code in a row on the left. The cell where they meet will tell you whether or not this combination is permitted.

When we say “copy code,” we mean just that: you're taking a section of code from one source, with or without modification, and inserting it into your own program, thus forming a work based on the first section of code. “Use a library” means that you're not copying any source directly, but instead interacting with it through linking, importing, or other typical mechanisms that bind the sources together when you compile or run the code.

Skip compatibility matrix

		I want to release a project under:					
		GPLv2 only	**GPLv2 or later**	**GPLv3 or later**	**LGPLv2.1 only**	**LGPLv2.1 or later**	**LGPLv3 or later**
I want to copy code under:	**GPLv2 only**	OK	OK [2]	NO	OK if you convert to GPLv2 [7]	OK if you convert to GPLv2 [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK if you convert to GPL [7]	OK if you convert to GPL [7]	OK if you convert to GPLv3 [8]
	GPLv3	NO	OK if you upgrade to GPLv3 [3]	OK	OK if you convert to GPLv3 [7]	OK if you convert to GPLv3 [7][3]	OK if you convert to GPLv3 [8]
	LGPLv2.1 only	OK if you convert to GPLv2 [7]	OK if you convert to GPL [7][2]	OK if you convert to GPLv3 [7]	OK	OK [6]	OK if you convert to GPLv3 [7][8]
	LGPLv2.1 or later	OK if you convert to GPLv2 [7][1]	OK if you convert to GPL [7]	OK if you convert to GPLv3 [7]	OK [5]	OK	OK
	LGPLv3	NO	OK if you upgrade and convert to GPLv3 [8][3]	OK if you convert to GPLv3 [8]	OK if you convert to GPLv3 [8]	OK if you upgrade to LGPLv3 [4]	OK
I want to use a library under:	**GPLv2 only**	OK	OK [2]	NO	OK if you convert to GPLv2 [7]	OK if you convert to GPLv2 [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK if you convert to GPL [7][1]	OK if you convert to GPL [7]	OK if you convert to GPLv3 [8]
	GPLv3	NO	OK if you upgrade to GPLv3 [3]	OK	OK if you convert to GPLv3 [7]	OK if you convert to GPLv3 [7][3]	OK if you convert to GPLv3 [8]
	LGPLv2.1 only	OK	OK	OK	OK	OK	OK
	LGPLv2.1 or later	OK	OK	OK	OK	OK	OK
	LGPLv3	NO	OK	OK	OK	OK	OK

Skip footnotes

1: You must follow the terms of GPLv2 when incorporating the code in this case. You cannot take advantage of terms in later versions of the GPL.

2: If you do this, as long as the project contains the code released under GPLv2 only, you will not be able to upgrade the project's license to GPLv3 or later.

3: If you have the ability to release the project under GPLv2 or any later version, you can choose to release it under GPLv3 or any later version—and once you do that, you'll be able to incorporate the code released under GPLv3.

4: If you have the ability to release the project under LGPLv2.1 or any later version, you can choose to release it under LGPLv3 or any later version—and once you do that, you'll be able to incorporate the code released under LGPLv3.

5: You must follow the terms of LGPLv2.1 when incorporating the code in this case. You cannot take advantage of terms in later versions of the LGPL.

6: If you do this, as long as the project contains the code released under LGPLv2.1 only, you will not be able to upgrade the project's license to LGPLv3 or later.

7: LGPLv2.1 gives you permission to relicense the code under any version of the GPL since GPLv2. If you can switch the LGPLed code in this case to using an appropriate version of the GPL instead (as noted in the table), you can make this combination.

8: LGPLv3 gives you permission to relicense the code under GPLv3. In these cases, you can combine the code if you convert the LGPLed code to GPLv3.

back to top

Translations of this page

Česky [cs] English [en] Español [es] Français [fr] Italiano [it] 日本語 [ja] 한국어 [ko] Polski [pl] português do Brasil [pt-br]