

# Copyleft: Pragmatic Idealism

---

GNU philosophy

This document is part of GNU philosophy, the GNU Project's exhaustive collection of articles and essays about free software and related matters.

Copyright © 1998, 2003 Free Software Foundation, Inc.

Verbatim copying and distribution of this entire document are permitted worldwide, without royalty, in any medium, provided this notice is preserved.

---

## Copyleft: Pragmatic Idealism

Every decision a person makes stems from the person's values and goals. People can have many different goals and values; fame, profit, love, survival, fun, and freedom, are just some of the goals that a good person might have. When the goal is a matter of principle, we call that idealism.

My work on free software is motivated by an idealistic goal: spreading freedom and cooperation. I want to encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better.

That's the basic reason why the GNU General Public License is written the way it is—as a copyleft. All code added to a GPL-covered program must be free software, even if it is put in a separate file. I make my code available for use in free software, and not for use in proprietary software, in order to encourage other people who write software to make it free as well. I figure that since proprietary software developers use copyright to stop us from sharing, we cooperators can use copyright to give other cooperators an advantage of their own: they can use our code.

Not everyone who uses the GNU GPL has this goal. Many years ago, a friend of mine was asked to rerelease a copylefted program under noncopyleft terms, and he responded more or less like this: “Sometimes I work on free software, and sometimes I work on proprietary software—but when I work on proprietary software, I expect to get *paid*.”

He was willing to share his work with a community that shares software, but saw no reason to give a handout to a business making products that would be off-limits to our community. His goal was different from mine, but he decided that the GNU GPL was useful for his goal too.

If you want to accomplish something in the world, idealism is not enough—you need to choose a method that works to achieve the goal. In other words, you need to be “pragmatic.” Is the GPL pragmatic? Let's look at its results.

Consider GNU C++. Why do we have a free C++ compiler? Only because the GNU GPL said it had to be free. GNU C++ was developed by an industry consortium, MCC, starting from the GNU C compiler. MCC normally makes its work as proprietary as can be. But they made the C++ front end free software, because the GNU GPL said that was the only way they could release it. The C++ front end included many new files, but since they were meant to be linked with GCC, the GPL did apply to them. The benefit to our community is evident.

Consider GNU Objective C. NeXT initially wanted to make this front end proprietary; they proposed to release it as `.o` files, and let users link them with the rest of GCC, thinking this might be a way around the GPL's requirements. But our lawyer said that this would not evade the requirements, that it was not allowed. And so they made the Objective C front end free software.

Those examples happened years ago, but the GNU GPL continues to bring us more free software.

Many GNU libraries are covered by the GNU Lesser General Public License, but not all. One GNU library which is covered by the ordinary GNU GPL is Readline, which implements command-line editing. I once found out about a nonfree program which was

designed to use Readline, and told the developer this was not allowed. He could have taken command-line editing out of the program, but what he actually did was rerelease it under the GPL. Now it is free software.

The programmers who write improvements to GCC (or Emacs, or Bash, or Linux, or any GPL-covered program) are often employed by companies or universities. When the programmer wants to return his improvements to the community, and see his code in the next release, the boss may say, “Hold on there—your code belongs to us! We don’t want to share it; we have decided to turn your improved version into a proprietary software product.”

Here the GNU GPL comes to the rescue. The programmer shows the boss that this proprietary software product would be copyright infringement, and the boss realizes that he has only two choices: release the new code as free software, or not at all. Almost always he lets the programmer do as he intended all along, and the code goes into the next release.

The GNU GPL is not Mr. Nice Guy. It says no to some of the things that people sometimes want to do. There are users who say that this is a bad thing—that the GPL “excludes” some proprietary software developers who “need to be brought into the free software community.”

But we are not excluding them from our community; they are choosing not to enter. Their decision to make software proprietary is a decision to stay out of our community. Being in our community means joining in cooperation with us; we cannot “bring them into our community” if they don’t want to join.

What we *can* do is offer them an inducement to join. The GNU GPL is designed to make an inducement from our existing software: “If you will make your software free, you can use this code.” Of course, it won’t win ’em all, but it wins some of the time.

Proprietary software development does not contribute to our community, but its developers often want handouts from us. Free software users can offer free software developers strokes for the ego—recognition and gratitude—but it can be very tempting when a business tells you, “Just let us put your package in our proprietary program, and your program will be used by many thousands of people!” The temptation can be powerful, but in the long run we are all better off if we resist it.

The temptation and pressure are harder to recognize when they come indirectly, through free software organizations that have adopted a policy of catering to proprietary software. The X Consortium (and its successor, the Open Group) offers an example: funded by companies that made proprietary software, they strived for a decade to persuade programmers not to use coryleft. When the Open Group tried to make X11R6.4 nonfree software, those of us who had resisted that pressure were glad that we did.

In September 1998, several months after X11R6.4 was released with nonfree distribution terms, the Open Group reversed its decision and rereleased it under the same noncoryleft free software license that was used for X11R6.3. Thank you, Open Group—but this subsequent reversal does not invalidate the conclusions we draw from the fact that adding the restrictions was *possible*.

Pragmatically speaking, thinking about greater long-term goals will strengthen your will to resist this pressure. If you focus your mind on the freedom and community that you can

build by staying firm, you will find the strength to do it. “Stand for something, or you will fall for anything.”

And if cynics ridicule freedom, ridicule community. . .if “hard-nosed realists” say that profit is the only ideal. . .just ignore them, and use copyleft all the same.