Free Software Needs Free Tools

Benjamin Mako Hill
mako@atdot.cc

June 6, 2010

Over the last decade, free software developers have been repeatedly tempted by development tools that offer the ability to build free software more efficiently or powerfully.

The only cost, we are told, is that the tools themselves are nonfree or run as network services with code we cannot see, copy, or run ourselves. In their decisions to use these tools and services – services such as BitKeeper, SourceForge, Google Code and GitHub – free software developers have made "ends-justify-the-means" decisions that trade away the freedom of both their developer communities and their users. These decisions to embrace nonfree and private development tools undermine our credibility in advocating for software freedom and compromise our freedom, and that of our users, in ways that we should reject.

In 2002, Linus Torvalds announced that the kernel Linux would move to the "BitKeeper" distributed version control system (DVCS). While the decision generated much alarm and debate, BitKeeper allowed kernel developers to work in a distributed fashion in a way that, at the time, was unsupported by free software tools – some Linux developers decided that benefits were worth the trade-off in developers' freedom. Three years later the skeptics were vindicated when BitKeeper's owner, Larry McVoy, revoked several core kernel developers' gratis licenses to BitKeeper after Andrew Tridgell attempted to write a free replacement for BitKeeper. Kernel developers were forced to write their own free software replacement: the project now known as Git.

Of course, free software's relationships to nonfree development tools is much larger than BitKeeper. The source to the free software development support service SourceForge was once available to its users but its authors have returned to a completely closed model.

While SourceForge is built using free software, SourceForge users interact with the software over the web. Because users never have any copy of the SourceForge software, they can never demand source. Similar projects like CollabNet's Tigris.org, Google Code's "Open Source Project Hosting" services, and GitHub, each served similar purposes and have kept their code similarly out of reach. Their services are often provided without charge and promoted for free software development, but this commitment does not extend to their own software that runs the development platforms. The source code to each of these systems remains private and unmodifiable by the developers using the services.

These nonfree development tools present a dilemma for many free software developers. The goal of many of these tools is, through more efficient free software development, more free software and more freedom. CollabNet, Google and GitHub each claim to want free software to succeed and claim they want to help it. For a series of reasons though these companies choose to support software freedom through means that are less in line with free software ethics than the the ones they seek to create. The result is developers who are disempowered. The software freedom of the code these hackers produce is contingent on unacceptable exclusivity.

First, the use of nonfree tools sends an unacceptable message to users of the free software produced. "Software freedom is important for you as users," developers seem to say, "but not for us." Such behavior undermines the basic effectiveness of the strong ethical commitment at the heart of the free software movement. As those that are already committed to free software, we should demonstrate that we can succeed – and thrive – using free software. We should support free alternatives to proprietary systems such as Savane which can replace SourceForge or Google Code and runs GNU Savannah, or Gitorious which can replace GitHub – by using them and by improving them in the areas where they fall short.

Secondly, we should realize that, going forward, the software we produce is only as free as the software it depends on for its continued use, distribution, and evolution.

The GNU GPL license and source code mean little to a user attempting to modify a program without free access to the software required to make that modification. It is not only developers' freedom at stake but, eventually, their users and all future "downstream" developers as well. Those choosing to use nonfree tools put everyone at the whim of the groups and individuals who produce the tools they depend on.

While proprietary development tools may help free software developers create more free software in the short term, it is at an unacceptable cost. In the controversial area of private software and network services, free software developers should err on the side of "too much" freedom. To compromise our principles in attempts to achieve more freedom is self-defeating, unstable, and ultimately unfair, to our users and to the larger free software development community.

Just as the early GNU maintainers first focused on creating free tools for creating free software, we should ensure that we can produce software freely and using unambiguously free tools. Our failure to do so will result in software that is, indirectly, less free. We should resist using tools that do not allow us the freedoms we are trying to provide our users in the development of *their* software and we should apply pressure on the producers of our development tools. Free software has not achieved success by compromising our principles. We will not be well served, technically, pragmatically, or ethically, by compromising on freedom of the tools we use to build a free world.