Promoting free software on non-free platforms

Why developing free software for proprietary platforms benefits the free software community.

Chris J. Karr

In a recent discussion on the Slashdot web site (http://slashdot.org/), free software users and advocates raised the question of whether the KDE project should be ported to the Microsoft Windows platform. Advocates for porting the KDE desktop environment made the argument that porting KDE to Windows would enable a new population of users to experience the software and that this exposure would entice these new users to seek out and adopt free software for use in their daily computing lives. Opponents of porting KDE believe that since Microsoft controls the underlying platform, software like KDE will never be able to compete on a fair playing field and that the time and effort spent in porting the software would be wasted when the vendor eventually decided to exclude free software competitors. While some free software advocates are right to be concerned, current realities support the position that free software on non-free platforms should be promoted.

Motives for developing free software can be divided into ideological and pragmatic groups

Before becoming heavily involved in the dispute, one must pay attention to the motivations of those producing and using free software. Some developers approach free software development from a point of view of promoting fundamental rights. Richard Stallman, of the GNU Project, is the most well known member of this group, with his philosophies on software development and use. This group believes that there are fundamental rights that grant software users the freedom to modify the software they use to better serve their needs and that users should possess the freedoms to share and distribute these changes. Unrestricted access to source code is central to this philosophy and closed-source software with restrictions on copying and distribution is an anathema to those holding these views. Members of this group produce free software as a means to establishing alternatives to the non-free platforms and applications.

In contrast to the ideologically-driven developers, another group produces free software for pragmatic reasons. These pragmatic developers are less interested in software politics and are more interested in using the free software development model and free software licenses to achieve a practical goal of their own - such as selling services, using free software as a competitive tool in the marketplace, or simply making the software available to the widest user base possible. Many adopt the free software development model in order to build a community and promote collaboration and they solicit contributions from the community to drive further development and maintain users' interest in the software.

While there are other motives for developing free software, most can be divided into ideological and pragmatic groups. The people who primarily view free software as a philosophical movement are ideological participants, while those who release free software as a means to another end are

pragmatic participants. Despite these fundamental differences, both camps benefit when free software is available on both free and non-free platforms.

"Don't let the perfect be the enemy of the good."

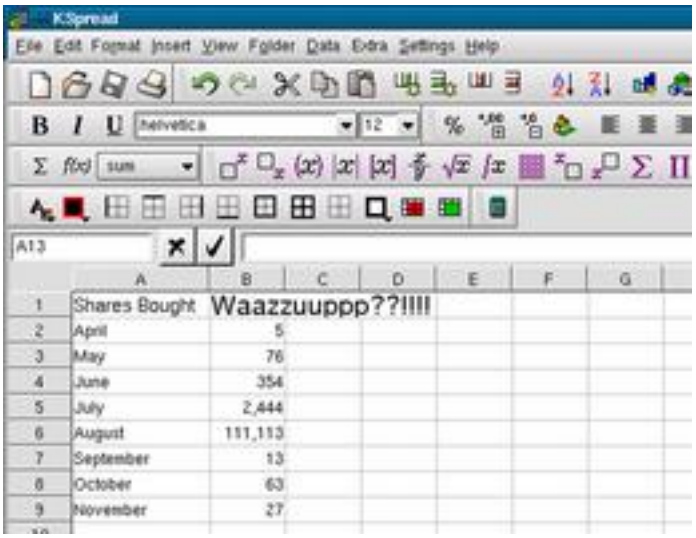
While pragmatic developers do not object to developing free software for non-free platforms, ideological developers often do. Ideological participants have a strong interest in the creation of an environment with complete freedom, so that users can use free software in all facets of their computing lives. To accomplish this goal, ideological participants believe that it is advantageous if the free platform grew and innovated independently of non-free platforms. In their view, it makes sense for the free software platform to evolve into a distinct environment that has its own advantages, not found in non-free platforms. Porting free software to non-free platforms minimizes the distinction between free and non-free platforms and the presence of free software functionality on non-free platforms provides less of an incentive for users to switch.

By any reasonable measure, the ideological developers have already achieved a remarkable goal. Using a free Unix operating system with software such as OpenOffice.org, the Mozilla internet browsers, and free software media players, an ideological user can go through their everyday tasks without using any non-free software.

However, when interacting with non-ideological users, a pure free software platform is not always sufficient. Some web sites use non-free technologies such as Flash. Document exchange is troublesome as non-free office formats complicate interoperability with open source software. New Digital Rights Management (DRM) technologies exclude free media software.

While ideological developers have succeeded in creating a pure free software platform, they now face a much more insurmountable problem – motivating non-ideological users to exclusively adopt free software. While free software use has grown greatly in recent years, the market of non-free software and platforms still dwarfs that of free software. Ideological developers prefer a computing landscape where all layers of the computing platform are free, but they face an uphill battle in convincing non-ideological users to abandon their non-free software. While ideological users were

Fig. 1: Will the KOffice suite be a competitor to OpenOffice.org and Microsoft Office?



patient and contributed time and effort to help free software reach its current level of functionality, the non-ideological user values current availability and ease-of-use over philosophical purity. Furthermore, there are no free software applications approaching the level of functionality and polish found in some popular non-free software (such as Quicken and Photoshop), and non-ideological users will not abandon their non-free platforms while suitable free software replacements are absent.

While ideological developers have succeeded in creating a pure free software platform, they now face a much more insurmountable problem – motivating non-ideological users to exclusively adopt free software

Thus ideological users are unable to convert the multitudes of non-free software users, but they benefit every time their non-ideological counterparts use free software. Each Windows or Mac OS X user that installs FireFox and browses the web using a free browser is one less user that web sites can target with non-free technologies such as ActiveX.

A bank website can afford to ignore the one percent of their clients who are pure free software users, but they cannot afford to ignore twenty percent of their clients who

Fig. 2: Firefox is winning the hearts and minds of Mac and Windows users everywhere.

are on non-free software platforms using free software web browsers. Ideological users benefit similarly when more users adopt free office suites instead of Microsoft Office. Proponents of pure free software stacks benefit from network effects each time a user of a non-free platform adopts a free replacement for a non-free application.

In addition to their own personal use, ideological users further benefit from free software adoption when promoting and educating users about freedom and software. Trying to make a case that a Windows XP user should switch to a free browser is difficult if there is no way to easily demonstrate such a browser on a non-free platform. It is difficult to sell the idea that KDE is a wonderful desktop environment when the user is required to install Linux to try it out. In this situation, the availability of free software on non-free platforms serves as an advertisement that free software is of high quality and that it can replace non-free applications.

Furthermore, as users adopt more free alternatives in place of non-free applications, the task of converting the user from a hybrid free and non-free platform to a pure free software platform becomes much simpler. Since the user is already familiar with applications, the switching costs and education required for the new free platform is greatly reduced. A user on Windows who primarily uses free software is easier to convert over to Linux, as it's simply a matter of teaching the user how to use Linux instead of having to teach them to use an entirely new suite of software and Linux. While applications exclusive to non-free platforms will prevent some users from being able to switch from a mixed system to a pure one, the adoption of high-quality free applications like Firefox and OpenOffice.org is shrinking that pool of users. As more free software replacements are made available to non-free users, the task of the free software advocate becomes much easier when convincing users to switch.

Finally, the idea that people willing and interested in writing free software should be discouraged from writing code that runs on non-free platforms is hypocritical and troubling when it comes from ideological users. One of the core tenets of the free software movement is the idea that users are free to expand and share their applications. Whenever ideological users discourage developers from free development of any kind, it violates the core principle that developers possess the freedom to develop as they feel is appropriate. Simply put, an ideological user has no right to complain when another developer targets a non-free platform with a particular application, when the ideological user is perfectly capable of donating their time or other resources to achieve the same end on a free platform. Ideological users are not owed any consideration when such a generous gift is given freely.

Making the most of non-free platforms

For the pragmatic free software developer, developing for non-free platforms provides many advantages. In terms of software engineering, a diversity of supported platforms supports encourages developers to implement their applications in more modular and portable ways. When working on non-free platforms, the pragmatic developer also has the opportunity to interoperate and collaborate with best-of-breed commercial and free software applications. Developers can craft their applications to take advantage of features unique to non-free platforms, such as using media codecs and security features. Non-free platforms also provide a stable set of libraries and base applications that free software developers can depend upon – including XML parsers, secure internet libraries, and HTML rendering engines. Finally, developers targeting non-free platforms can take advantage of these platforms' users' willingness to purchase software in order to fund the creation and further development of the free software.

As a direct result of competitive markets, non-free platforms

Fig. 3: The free software VLC media player is a popular alternative to Quicktime and Windows Media players.

typically have larger libraries of high-quality end-user applications than their free counterparts. This is not to disparage free software, but when software is purchased, there is a reasonable expectation that the purchase includes not only the basic functionality, but also things like documentation, support, and polish. Furthermore, the market is much more favorable to applications that focus on usability and consistency, whereas free software typically focuses on the quantity of features. When developing for non-free platforms, free software developers can interact with non-free applications in a variety of ways. In Mac OS X, this can be accomplished via Services and AppleScript, while Windows employs a COM-based architecture. For example, an aspiring developer can create a media player that uses native functionality (such as QuickTime or Windows Media) that allows the playback of more formats than similar applications found on free software platforms. This is becoming increasingly important as content owners adopt DRM technology.

In addition to interoperating with non-free applications on non-free platforms, the platforms themselves offer unique features that developers may wish to use. For a developer wishing to create a free software Mac OS X application, the availability of the Cocoa framework provides the developer a set of tools and programming interfaces that are more fully featured and easier to work with than comparable free software GUI toolkits. While similar frameworks such as GNUStep struggle to keep up, Cocoa developers can take advantage of KHTML-based web rendering and new search technologies in their applications with a few lines of Objective-C or Java code.

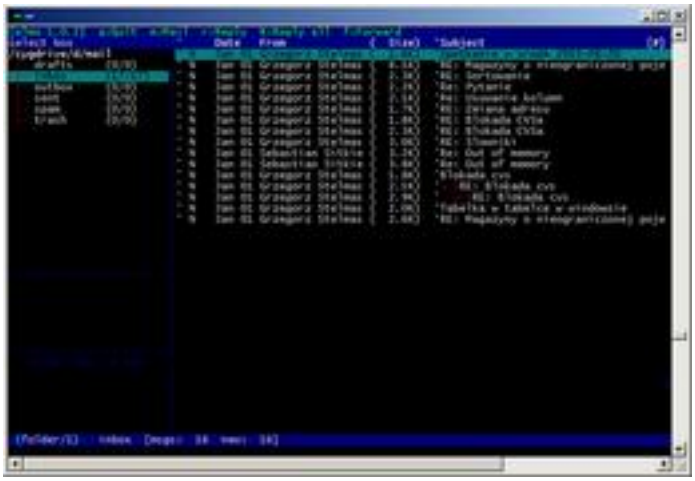
Non-free platforms offer unique features that free software developers may wish to use

Developing applications on non-free platforms also provides developers a limited amount of freedom from "dependency hell". Free platforms are often designed to allow users maximum flexibility to customize the system for uses ranging from slimmed-down routers and servers to full-featured desktop systems for daily use. In contrast, non-free platforms often include more functionality in the base installs and while users complain about bloat, this "bloat" allows developers to provide more advanced applications that are easier to install and maintain. Developers can distribute packages and installation instructions that are simpler than the typical free software application with the standard list of packages and library dependencies. This is mainly true with basic functionality, although developers may find it necessary to specify or bundle dependent packages when dealing with new or advanced features not present on the non-free system. For example, while a free software package may require that a particular XML parsing library is present on free platforms, the same package may also be able to make use of the non-free systems' own XML routines.

In terms of software engineering, developing applications that function on both free and non-free software can improve the underlying structure of application by requiring a certain amount of abstraction from the underlying platform. When the platform-specific code is separated from the platform-nonspecific code, the program can be more easily ported to new platforms by simply implementing the platform-specific code. Furthermore, separating the code into modular components can improve debugging and optimizations as the platform-specific code can be adapted to particular platforms' strengths and quirks. This tendency toward modularity assists both pragmatic and ideological developers, as the software can be adapted to new platforms as they emerge.

Finally, from the point of view of funding the creation of free software, there are cultural advantages to developing

Fig. 4: The Cygwin environment provides a Unix-like environment on Microsoft Windows.



free software that includes non-free platforms. Since users on non-free platforms are accustomed to purchasing software, free software developers can take advantage of this social trait by selling easily installed versions of the software and materials such as documentation and manuals. Some free software developers have supplemented their income by making boxed versions of their software available to new users while allowing more advanced users to download the source online and compile the software locally.

Conclusion

Ideological and pragmatic developers approach the question of whether free software should be ported to non-free platforms differently. Ideological developers are generally against the idea for philosophical reasons. Pragmatic developers are generally agnostic with respect to the question and are more open to developing for non-free platforms. Ideological users need to realize that the vision of a pure free software environment for all users will never become a reality while non-ideological users exist, because non-ideological users will not spend the time or energy to write or wait for free alternatives to all of their non-free applications,.

However, despite the continued existence of hybrid free and non-free systems, ideological users have much to gain by supporting the porting of free software on non-free platforms by limiting the amount of influence of non-free applications and file formats. Furthermore, by providing free software to users on non-free platforms, ideological users can better educate new users about the philosophy of free software and the freedom to distribute and modify their software. Pragmatic developers also gain when developing for non-free platforms, because developing for different types of platforms allow the pragmatic developers to make the most of features and functionality provided by commercial developers. Furthermore, by embracing a diverse set of platforms, pragmatic developers can leverage that experience to create more modular and portable software.

The idea that people devoted to free software should discourage the development and use of free software on non-free platforms makes little sense, considering what is to gain by developing free software for non-free platforms.

Copyright information

About the author

Chris Karr is a software developer for Northwestern University. He maintains a personal weblog (http://www.aetherial.net/).