Open Source, Free Software and Other Beasts (version 2)

Shlomi Fish <shlomif@shlomifish.org>

Copyright © 2004 Shlomi Fish

Table of Contents

Introduction

Many people will hear about Linux in the news, being the cool new operating system that everyone can use free of charge. Those who become interested in it enough or actually start working with it, will learn that it is made out of many independent "open source" components. Now, after enough time (perhaps very soon), they will learn that the term "free software" (where free is free as in "free speech" and not free as in "free beer") can be used as an alternative to the adjective "open source". But what is open source and free software? What distinguishes them from other software that is available to the public at no cost or is distributed as shareware?

Note that the terms "free software" and "open source" would be used throughout this article to refer to the same phenomenon. I do not religiously stick to either term.

Software Licences and "Proprietary" Software

This section deals with the legal details of distributing software, and the so-called licences that dictate what can be done with them.

Software out of being a sequence of bits, that can be transcribed to a paper, spoken or otherwise transported is considered speech and so is protected by the Freedom of Speech principle of Liberalism [http:// en.wikipedia.org/wiki/Freedom_of_speech]. Thus, writing software and distributing it are a constitutional right in most liberal countries.

Nevertheless, a piece of software, as any other text, can be copyrighted. Copyright involves making sure that the software as given to someone else other than its originator or copyright holder will be restricted in use or modification. An originator can outline what he believes to be a proper use of the software in a code licence (which applies to the code) or an "End-User License Agreement" (or EULA which applies to given binaries).

Proprietary software, i.e: such whose use, modification or distribution is encumbered, was a relatively new phenomenon if you take a look at the old history of computing. It actually started even before the time when Microsoft, then a very small company wrote Altair Basic, and Bill Gates published the famous (or possibly infamous) "Open Letter to Altair Hobbyists" [http://www.blinkenlights.com/classiccmp/ gateswhine.html]. In fact, IBM and other companies distributed proprietary software for mainframe systems, a long time before the Personal Computer revolution.

The PC revolution, however, made the situation more critical. Soon, computers became faster, more powerful, with larger memory, and more common as time went by. At the moment, there are hundreds of millions of Pentiums and other computers out there, and millions of newer computers are sold each year.

Yet, the majority of these computers mostly run software that cannot be modified or distributed, at least not effectively or legally. The free software (or open-source) movement started as an anti-thesis to the tendency of vendors to hide the details of their software from the public. The Linux Operating System with its various components (most of which are available to other systems as well, and are not affiliated with the Linux kernel in particular) is the most visible showcase to this phenomena. By installing Linux it is possible to turn an everyday personal computer into a full fledged UNIX-based workstation or server, which is a 100% powerful GNU system. This can cost little if any money, and the various components of the operating system are all freely modifiable and can be re-distributed in their modified form.

It is not the only place where free software can be used. It is in fact possible to turn a Windows installation into a Linux-like GNU system as well (see Cygwin [http://www.cygwin.com/] for instance).

Meaning of the terms

According to the Free Software Definition [http://www.gnu.org/philosophy/free-sw.html] free software must fulfil 4 freedoms:

1. The freedom to run the program, for any purpose

2. The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.

3. The freedom to redistribute copies so you can help your neighbour

4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits . Access to the source code is a precondition for this.

The Open Source definition [http://www.opensource.org/docs/definition_plain.php] is similar, but some licences can qualify as open-source and not as free software. This is usually not an issue, because the majority of open source software out there is free as well.

Despite common belief selling free/open-source software is perfectly legitimate. In fact one can charge as much as he pleases for it. Nevertheless, most free software is distributed for free or for very cheaply on the Internet and other mediums. This is due to the fact that its freely distributable nature does not give way much to sale value, so there usually is no point in attempting to mandate a charge for selling it.

Another common misconception is that it sometimes cannot be modified or customised for internal use. In fact, all free software (but not *all* open source software), can. Only when you wish to distribute it (free of charge or commercially), you may have to distribute your changes. (depending on the licence) The use of open source software to process proprietary content or be processed by non-free programs is also, always available. Thus, an open-source C compiler can be used to compile the code of proprietary programs like the Oracle Database Server.

History

This section is not a definitive overview of the history of the free software movement. It focuses on the issues regarding the usage of the common terms.

Early Days, AT&T UNIX, BSD

The free software movement (before it was called this way) started organically from individuals who distributed code they wrote under the Public Domain or what would now be considered open source or semi-open source licences.

AT&T UNIX that started at 1969 was the first showcase for this movement. Several Bell Labs Engineers led by Ken Thompson developed UNIX for their own use, and out of legal restrictions AT&T faced, decided to distribute it to academic organizations and other organizations free-of-charge with the source included. (that licence did not qualify as open-source but it was pretty close). UNIX eventually sported the C programming languages, which enabled writing code that would run on many platforms easier, and the UNIX sources included a C compiler that was itself written in C. Around the early 70's the only computers capable of running UNIX were main-frames and the so-called "mini-computers" so there initially weren't as many installations as only large organizations could support buying computers to deploy UNIX on.

That changed as integrated circuits, and computers became cheaper and more powerful. Very soon, cheap UNIX-based servers and workstations became commonplace and the number of UNIX installations exploded. [1]

Nadav Har'El has prepared a coverage of the BSDs and early AT&T UNIX history [http://groups.yahoo.com/group/hackers-il/message/1731].

The Berkeley University forked its own version of AT&T UNIX and started re-writing parts of the code, and incorporating many changes of its own. The original parts were licensed under the BSD license which is a copyright licence that is almost entirely Public Domain. The BSD system became very popular (perhaps even more than the AT&T one).

When Arpanet, the predecessor to the Internet was disbanded due to inadequacy, the Internet converted to running on top of 32-bit UNIX boxes such as the VAX architecture by Digital Equipment Corporation [http://en.wikipedia.org/wiki/VAX] (now part of Hewlett-Packard). This caused a merging of the UNIX culture with the Arpanet enthusiasts who exchanged code on the Arpanet, and UNIX programmers started sharing code for various components and add-ons of UNIX on the Internet.

Richard Stallman, the GNU Project, and the "Free Software" term

After a while, the legal restrictions posed on AT&T subsided, and it started to "smell money" and believe it can do better selling UNIX commercially. It created the AT&T System V system, touted it was better than AT&T UNIX and the BSDs, and sold it to vendors. System V was sold under a very restrictive licence, that forced them to hold the source code for themselves. Even cooperation between two different vendors was not allowed.

Gradually, vendors licensed the System V source code and ported it to their own architectures. This caused an explosion of proprietary UNIX systems. What Sun Microsystems initially did was actually take the BSD source code, diverge from it and distribute it without full access to the code. (something that was allowed by its licence). A similar thing happened with other software distributed under similar licences.

To answer this threat, a new phenomenon sprang into existence: the "free software" movement, the GNU project and the copyleft licences, all led by one dynamic personality: Richard M. Stallman.

Richard Stallman (aka RMS) published the GNU Manifesto [http://www.gnu.org/gnu/manifesto.html] in 1984, which coined the term "free software", and explained the rationale behind it. The Manifesto was also a creed for the the GNU project which aimed to be a complete UNIX-compatible replacement for UNIX systems, while being completely original work. The software of the GNU project was released as free software, under the terms of the GNU General Public License [http://www.gnu.org/copyleft/gpl.html] (or GPL for short).

Gradually, the GNU project created more and more C code to replace the UNIX and BSD utilities. It was already installable and usable on various flavours of UNIX, and became a fully independent system once the Linux kernel was written.

The GPL licence is a free software licence that has many fine points. The most important concepts in it are:

1. Copyleft [http://www.gnu.org/copyleft/] - making sure that derived work that are distributed to the outside includes the source and is distributed under the same licence. Note that this does not apply to modifications done for internal or private use.

---

[1] At present day, UNIX clones such as Linux or the BSDs can run on regular Pentium-based computers that can be bought from PC shops. Most PC computers nowadays can out-compete the UNIX workstations of a few generations back. This allow assembling a UNIX server which is much more powerful and much less costly than the past ones, and that suffices for most needs.

2. Restrictive Integration by Other Code bases - GPL code can only be linked against code with free software licences that match some criteria. This property has been recently referred to as "viral".

The incentive to restrict a software this way rather than following the traditional virtually public domain BSD licence, was to make sure that the core GNU system would always remain free as well.

The Linux Kernel, GNU/Linux and the Debian Free Software Guidelines

In 1992 Linus Torvalds, then a student at Helsinki University, began writing the "Linux" kernel - a 32-bit kernel for UNIX-like operating systems. The kernel development advanced rapidly and was released under the GPL licence starting from an early stage. To complete the system and make it into a usable UNIX system, the Linux developers used various existing user-land utilities and libraries from the GNU project and other sources (such as the X-Windows system), and wrote a few user-land utilities from scratch.

From an early stage, this entire system was dubbed "Linux" as well. Richard Stallman instead has advocated the name "GNU/Linux" [http://www.gnu.org/gnu/linux-and-gnu.html] (pronounced "ggnoo-Linux") which acknowledges the fact that the GNU project contributed the lion's share of the system. (as well as some pre-requisites of the Linux kernel itself). Most people haven't consistently followed this piece of advice.

The importance of the Linux kernel was that it was the last brick in materialising a fully GNU system. Since GNU tools tend to be more complete, feature-rich and generally superior to tools of other systems, this has made Linux one of the most powerful UNIX systems available. Nowadays, most UNIX servers out there and almost all workstations run the GNU/Linux system. Linux was, thus, the spearhead that guided the acceptance of free software into the mainstream.

Debian GNU/Linux [http://www.debian.org/] was a Linux distribution that was eventually endorsed by the GNU project. One of the things that made it unique was the fact it distinguished between "free" and "non-free" packages as far as the user is concerned. The guidelines for determining which software is "free" in the Debian sense [http://www.debian.org/social_contract.html] were phrased by Bruce Perens.

Note that they deviate from the free software definition (which was phrased later on) and include some licences that are not free. I.e: "Debian Free" is a superset of free software according to the Stallman definition.

This fact is important because later on, the Debian Free Software Guidelines formed the basis for the open-source definition.

The "Cathedral and the Bazaar" and the coining of the term "Open-Source"

Eric Steven Raymond (now also known as ESR) wrote an essay titled "The Cathedral and the Bazaar" [http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/] and presented it to the Linux Kongress at 21 May 1997. This contrasted the Bazaar way of managing a software project and the old "Cathedral" way, that was used by almost all non-free projects and (until that point at least) by most free ones.

"Bazaar" projects are characterised by frequent and incremental release schedules, treating the users as co-developers, and generally getting a lot of peer review, ideas, input and cooperation. Despite a common misconception, the core group of the project contributors still usually remains relatively small except for very large projects.

The article is considered one of the seminal works on free software and was followed by other works in what is collectively known as the "Cathedral and the Bazaar" (or CatB for short) series [http://www.catb.org/~esr/writings/cathedral-bazaar/]. It has made Eric Raymond a famous person at least among the community of free software hackers.

In February 3, 1998, in Palo Alto California, a brainstorming session which Raymond attended, coined the term "open source" as an alternative for "free software". Their incentive was that when talking to a businessman, either free software will be understood as software that costs nothing, or it will be associated with the relatively anti-Capitalistic views held by Richard Stallman (who claims non-free software is immoral). They decided that the term "open source" would be a better candidate for acceptance in the corporate world.

Consult the opensource.org history document [http://opensource.org/docs/history.php] for further coverage of the history of the term.

During the following week, Eric Raymond, and Bruce Perens launched the opensource.org [http://www.opensource.org] web-site, and formed the Open source definition. This was based on the Debian Free Software Guidelines.

The term "open source" caught on. Very soon, Richard Stallman decided to reject it on the premise that the freedom of software is more important than the "openness" of its code. While he does not oppose the openness of the code, and acknowledges the fact that free software is open source as well, its freedom remained more important. For more information read the document "Free Software for Freedom" [http://www.gnu.org/philosophy/free-software-for-freedom.html] on the GNU web-site.

While some people have continuously stuck to the term "free software" and a few others converted to using "open source" entirely, most knowledgeable people don't completely reject either term, and use each one whenever they see fit. Nevertheless, the term "open source" is more commonly used by both open source developers and even more so by non open source developers. See Eric Raymond's "Terminology Wars" [http://catb.org/~esr/writings/terminology/] for more details.

Linux Becomes More Popular

Since 1997, Linux and other open-source systems have become more and more popular. Linux saw a lot of success in the server market, where cheap PCs that can be bought in stores can serve as an almost full replacement for more costly UNIX servers by installing Linux. Even if the latter are used, they very often run open source servers and other open source programs, utilities and frameworks.

Linux has become the number one choice for constructing clusters, a large set of computers that are networked together to form a fast computation system, with powers that rival or exceed super-computers. There are various kinds of clusters around. Some of them are performed at a relatively high level. Others, try to make the system believe it has as many processors as there are nodes.

Linux also had a lot of success in the embedded market, serving as the framework for creating software that is embedded in hardware.

The Internet boom not only made free software more essential for its operation , but also enabled more and more users and developers to share their code, get help and work together for advancing it.

At the moment, Linux had a much more limited success as a choice for a desktop system. While being the only operating system that is gaining market share, it still has a very low one, in comparison to Microsoft solutions. Many projects started to supply users with desktop and GUI environments and applications. Some of them are very mature, usable and successful. Only time can tell if and when Linux becomes the default solution for the desktop.

Apple's Mac OS X was released and is based on Darwin, which is an open-source BSD-derived system. Mac OS X can run UNIX applications natively, and supports the X-Windows system, which is the de-facto GUI framework for UNIXes (including Linux). It is therefore a popular UNIX choice for PowerMac computers (and more recently for Macintosh computer based on Intel-based chips), albeit not the only one since Linux, and various open-source BSD clones and other UNIXes can run there as well.

The recent recession in the information technology market, did not seem to slow down the development of open source software. Freshmeat [http://freshmeat.net/] is still busy as ever with releases of new software, and since the recession started, many important new releases were done for a lot of major applications and even many more less important ones.

Open Source and Open Content Become Mainstream

While open source software has existed for DOS and Microsoft Windows practically since the beginning, and some of it was relatively popular among people, most of the software available for these platforms has been non-open-source binary-only software, a lot of it from Microsoft.

This has started to change recently. The Firefox browser [http://www.mozilla.com/firefox/] from the home of the Mozilla Foundation (and now also the Mozilla Corporation), is an open-source, modern and sophisticated browser, that has been virally publicised by various means such as the various "Spread Firefox" campaigns [http://www.spreadfirefox.com/]. It has become popular and as of July 2006 has passed the 10% usage in web site hits according to some firms, and in some countries much more so. It is still gaining some market share, even if it's growth has declined somewhat.

Other cross-platform open source software includes OpenOffice.org [http://www.openoffice.org/], a powerful and usable office productivity software for Windows, Linux and other platforms, the GIMP (GNU Image Manipulation Program) [http://www.gimp.org/], a sophisticated raster image editing program, and Inkscape [http://www.inkscape.org/], a vector editing program, and also most Peer-to-Peer networking clients. These have probably seen less popularity than Firefox, but are still providing cheap, open, modifiable alternatives to traditional binary-only software.

In 2003, a study was published that estimated that by 2004, more developers will work on Linux than on Windows [http://discuss.fogcreek.com/joelonsoftware3/default.asp?cmd=show&ixPost=98232&ixReplies=39]. While it definitely does not mean that more people will use Linux at home, it is still a good indication for its general mainstream acceptance and usefulness.

Another important recent trend was **the rise of open content**. The first edition of this article [http://fc-solve.shlomifish.org/oss-fs/docbook/], included a small section about "open content" [http://fc-solve.shlomifish.org/oss-fs/docbook/oss-fs/x181.html], where I concluded by saying that "Only time can tell whether other elements of open source besides its freely distributable nature will have an impact in other areas of creative arts besides software.". Now, about 3 years later, I can say that by all means open content has already proven to be a great success.

Among the landmarks of open or semi-open content are:

1. The Creative Commons project [http://creativecommons.org/] that specifies licences for open content, semi-open content or just freely redistributable artworks for individuals and organisations to use in their artwork, as well as supplying several resources for facilitating their publishing and use.

   Creative Commons licences have proven to be very popular among many web publishers for use in their works.

2. The Wikimedia Foundation [http://wikimedia.org/] publishes several online multi-lingual wikis - web sites that are editable by the surfers - all under an open content licence. The most famous and important one are the Wikipedias [http://www.wikipedia.org/], which are free, online encyclopaedias. The English

Wikipedia (which is still the largest) is larger than Encyclopaedia Britannica and Microsoft Encarta combined and is growing rapidly.

3. There are many sites for independent musicians, such as GarageBand.com [http://www.garageband.com/], ccMixter [http://ccmixter.org/], and Magnatune [http://magnatune.com/] (a record label that publishes artists whose songs are under a freely redistributable licence).

4. From weblogs and weblog comments, to wikis, to audiocasts or video-blogs - open or semi-open content is everywhere.

Difference between "Open Source" and "Free Software".

The term "free software" was coined by Richard Stallman, and is associated with the Free Software Foundation [http://www.fsf.org/]. The term "open source" was coined by Eric Raymond and is advocated by him and other people at the Open Source Initiative [http://www.opensource.org/]. Nevertheless, those who consider themselves in either camp, much less those who use either or both terms, do not necessarily hold the opinions of these figures. Therefore I will not globally associate them with the "free software movement" or the "open source movement" because both include many users and developers with heterogeneous opinions on the subject. Moreover, they are pretty much one and the same.

Nevertheless, it is important to summarise their opinions, because they are recurring in many places.

Stallmanism

Proprietary software is legal but illegitimate and immoral. Manufacturing and using proprietary software causes a lot of unhappy social and psychological side-effects. The knowledge that a software cannot be shared causes people to become reluctant to sharing, which is a natural and good part of living in a human society. The inability of people to modify software for their own needs makes them feel helpless, and at the mercy of external software.

Free software, on the other hand, is the natural conclusion derived from the basic facts of information, computing and software, and is highly moral. People, companies and other organizations can modify it, customise it and distribute it for their own use should the need arise, and so it actually benefits them.

Raymondism

Proprietary software is not illegitimate, just problematic from the economic sense. Open Source software gives many advantages to the end-users and is a generally a good thing. Copyleft licences are important in making sure certain software is not abused. It is not immoral to use proprietary software, it's just risky. Using or producing software that is not 100% open-source but pretty close, can be a good idea, depending on its licence and the general attitude of its developers.

In for Free Beer

This approach basically says this:

"I like free software because I can get a lot of useful software without charge. I may like contributing to free software because it helps other people, makes me happy and may indirectly benefit me technically or financially. But proprietary software is perfectly valid as well, if it's done right, and I may choose to use it or contribute to it.

In short: write code, use whatever tool you wish, and be happy."

The most prominent figure who holds this view is Linus Torvalds, but there are many others, some of them quite prominent. Such figures, however, tend to be less loud than the "religious" advocates of the other two views, and thus it may seem that they are at a minority. Part of the reason is that many of them inherently tend to value productive code and decisions over advocacy.

Note: I have prepared a longer (and unofficial) manifesto for this view [http://fc-solve.shlomifish.org/oss-fs/in_for_free_beer_manifesto.html] which you may wish to consult for further information.

Conclusion

While some figures out there prominently stick to either ideology, most people hold a mixture of the three (or more?) approaches, or are just happy using free software or contributing to it, without thinking too much about its philosophy.

The terms themselves are used interchangeably by many people. "Open source" has become more common, partly because free software can mean software that is given free of charge. (the standard "free as in free speech" or "free as in free beer" distinction). Moreover, both the Free Software Foundation, and the people associated with the Open Source Institute are on friendly terms with each other and answer questions, give feedback, accept contributions, etc. from each other or from people that do not belong to either camp.

Like I said earlier, the fact that some licences would qualify as open-source and not as free software is usually a negligible fact. While some esoteric software has been released under custom licences that are open-source while not free software, most of the important software out there (and most software generally started by individuals) is free as well. [2]

Other Criteria of Open Source Software

GPL Compatibility

Making a software free is not necessarily enough to make it compatible with the GNU GPL licence. The GPL makes some restrictions regarding which licences it can link against, and some perfectly good free software does not qualify to it. (examples are the Mozilla Public License, the Qt Public License, and even the original BSD licence). It is advisable that whenever possible a developer or vendor should choose a licence that is compatible with the GPL [http://www.dwheeler.com/essays/gpl-compatible.html], because otherwise there may be problems integrating his code with GPLed one or using both a GPL and a non-GPL compatible library. (I am not a lawyer, so I cannot conclusively say when it is legal or not).

Mozilla [http://www.mozilla.org/] is an example for a large project that started out with its custom (albeit now relatively common), non-GPL compatible licence, and recently adopted a triple licence of the Mozilla Public License, the GNU General Public License, and the GNU Lesser General Public License in order to make it compatible with the GPL and to standardise its integrability. The Qt library whose commercial vendor and originator is Troll Tech Inc. [http://www.trolltech.com/], also adopted the GPL as well as its own QPL, to relieve the various legal problems that KDE [http://www.kde.org/] (a desktop system for UNIXes which is based on it) faced when using GPL code.

Copyleft

The copyleft definition [http://www.gnu.org/copyleft/copyleft.html]

---

[2] It is advisable not to use a custom licence anyhow, as this tends to confuse users and fellow developers. There are many common licences to choose from. Check the GNU licences list [http://www.gnu.org/philosophy/license-list.html] and the list of open source licences [http://www.opensource.org/licenses/] for such lists.

Copyleft means that a derived work of a copyleft software, that are not used for internal or personal use, must include the source code and released under the same terms of the original work. Copyleft is common in many licences including the GPL, the Lesser General Public License, the QPL, etc.

Many licences are not Copyleft - most notably the BSD and MIT/X11 licences. Software released under such licences can be derived into a proprietary software product by a third party, and often have been.

Open Source vs. Sourceware

Open source does not mean any software that is accompanied by its source, albeit many people who are new to the term would be tempted to think that. It is possible to write non-OS software while accompanying it with the source.

Examples for such cases are:

1. The Microsoft Visual C++ Run-Time Library and the Microsoft Foundation Classes, that are accompanied with their source.

2. xv [http://www.trilon.com/xv/] - a very popular shareware image viewer and manipulator for X-Windows that has been distributed with its source code.

3. qmail [http://www.qmail.org/] - a popular mail server whose source code is available and can be deployed free of charge, but its licence specifies that it is illegal to distribute modified binaries (at least outside the organization) This is enough to make it non-open-source, but it is still a very popular program.

None of these packages qualify as free software, but they are all accompanied with the source. There are many others around. A quick search on Freshmeat [http://freshmeat.net/] will find many such packages.

In order for a program to be open-source it needs to be free of various restrictions as specified in the open-source definition [http://www.opensource.org/docs/definition.php]. To be free software as well, it must be also free of some other restrictions. [3]

I believe the term open-source is a bit dangerous in this regard. Then again, free software may not automatically be associated with freedom and liberty, so it isn't perfect either. But I guess finding a description that accurately describes it in a short space is not very possible, so these terms will have to do.

Myths about Open Source and Other Issues

Sharing software huh? Isn't it a bit like Communism?

This analogy is not new but very deceptive. First of all, there's nothing anti-Capitalistic about sharing something voluntarily. While in a Capitalistic country, goods are generally sold and have to be paid for, people can *voluntarily* dedicate their time and money for any cause they wish, possibly altruistic. Communism in fact *forces* the sharing of all good, including physical ones that take time and money to manufacture each unit of.

Secondly, because manufacturing and distributing a unit of software costs practically nothing, it is not necessary that it will be sold. While the development cost can be very large, a developer of the software will not be encumbered by it being used by a million people instead of a thousand.

Furthermore, by making a software open-source and keeping it so, it is possible to gain other economical and psychological advantages: you'll make sure it is maintained, gain feedback and admiration of others,

---

[3] In a recent query given to an Israeli Defence Force Official (IDF) by an Israeli member of the parliament, the former interpreted open-source as software that the IDF has access to its source. This is an even more radical deviation from the correct meaning.

and may be able to eventually receive input and contributions from the outside. Distributing software as commercial proprietary packages does not automatically yield good advantages and it takes a lot of time and money to make it usable as well. Such a vendor is actually risking that his software will work at all, and not be out-competed by something better.

This analogy was rejected and treated in a semi-jokily manner by most people who did not oppose Capitalism as a whole. A true understanding of why open-source does not contradict the liberal ideals of Capitalism and Individualism originated from Eric Raymond's "Homesteading the Noosphere" [http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/] and "The Magic Cauldron" [http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/], which are a very good read anyway you look at it.

The facts themselves do not give way to it either. The free software movement is almost entirely limited to liberal Capitalistic countries, which are free enough to allow unrestricted programming and distribution of software to prosper. Furthermore, I cannot name a single prominent figure who is Socialist or Communist, or otherwise entirely anti-Capitalistic (albeit some may hold certain views of this kind). Lastly, open source or some open source software was recently endorsed by many IT and non-IT businesses who greatly benefit from it, including some vendors of proprietary software.

The "Programmers Will be out of Work" because of Free Software Myth

It is unlikely that assuming Open Source software becomes the dominant software, it will imply that programmers will "starve to death". As Eric Raymond notes in the Magic Cauldron [http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/ar01s03.html], the vast majority of software out there is software without a sale value: be it software that large organization like banks, insurance companies or armies use to power their critical systems, customizations, scripts or code used within smaller organizations (Access customizations, Perl or shell scripts, Macros), embedded software whose source code is not released to the public, software that powers web-sites and was not released to the public, etc.

The majority of programmers out there are employed for developing such software, whose codebase dwarfs that of the marketplace software that includes all commercial and open-source software put together.

In due times, packages are developed and become available that makes some tasks that were once hard to do internally almost straightforward to set up and run. Nevertheless, these packages still require a clueful person to operate, diagnose problems, communicate with the vendor or developer and manage the configuration. For example, a complete computer beginner will probably not know what to do with a spreadsheet program (such as Microsoft Excel) without thoroughly studying it. Afterwards, it becomes a very useful tool.

Even if programmers do become out of job as a result of free software, then it will not necessarily a bad thing. It means that it solved problems that otherwise required extra hands, and so those programmers can be allocated for something else entirely that is more productive.

Other Myths about Linux and Open Source Software

One can very often hear many myths or generally accepted "truths" about open source software and Linux, some of which negative and other positive. Examples include:

1. Open source software is less secure than software whose source code is not revealed, because people can find bugs at it by looking at the code.

2. Open source software is more secure than closed-source software because more people can review the code and discover bugs in it.

3. Linux is harder to use than Windows.

4. Linux is not compatible with Windows.

5. Open source gives way to forking more easily.

And many others. The Linux Myth Dispeller [http://www.linuxmafia.com/~rick/myths.html] attempted to answer some of these, with a focus on negative myths. Myth #1 is completely false as bugs can still be found by analysing the disassembly of the machine code. Also often such bugs are found by accident due to a certain valid use of the software) There were many closed-source packages out there in which many bugs have already been discovered. (like Microsoft Outlook, Microsoft IIS or Microsoft Internet Explorer [http://www.shlomifish.org/no-ie/]). Some of these already became widely exploited a long time before a vendor patch was made available.

Myth #2 has a grain of truth in it. However, some open source packages nevertheless had very poor security records out of poor programming practices. Some closed-source offerings, on the other hand, have a very good security record. In most packages, security bugs occurred due to sloppy programming practice or lack of auditing of the code. They can be mostly avoided whether or not the package's source code is available to the public.

Myth #5 is not entirely true. While it is possible to fork a piece of open-source software, most packages were not effectively forked. Eric Raymond covers the customs that relate to forking a package in "Homesteading the Noosphere" [http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/], and Rick Moen explains why when major packages forked, it was not necessarily a bad thing in his "Fear of Forking essay" [http://www.linuxmafia.com/~rick/essays/forking.html].

Moreover, many times proprietary software was forked as well. There are many flavours of System V UNIX out there, and there used to be many more. Microsoft released three different lines of Windows flavours with two or more simultaneously, and has many localised versions. (Which are many times incompatible with one another.)

Where I Stand

It is customary in documents of this kind to convey the personal opinion of the author in this case. This document will not be an exception.

I am a user, developer and advocate of free software. However, I do not think that proprietary software is inherently immoral or destructive. I know some vendors of such software abuse their customers. However, I generally see them as suppliers of goods, which took a lot of time to develop, and which they perfectly naturally wish to sell for money.

The fact that open-source developers develop similar goods and distribute them for no cost or little cost under a less restrictive open-source licence, does not invalidate this fact. I agree with most of what Eric Raymond said in the "Cathedral and the Bazaar" series, part of which is that proprietary software is problematic. However, I think that a world dominated by free software (which I hope to see soon) can inhibit some proprietary software without it having a generally harmful effect on the computer world at large.

I do not hate Microsoft, just think that their systems are much inferior to GNU/Linux, which I like better. I still use Windows when I find it appropriate, or when I need to. (I'm not an "I only use free software" kind of guy). I realise the superiority of Linux may have stemmed from the fact it is free software, but otherwise don't use it only because it is free software. I just like to work with it better.

I don't see Microsoft or other suppliers of proprietary software as enemies of the free software movement. I expect that people will continue to buy some proprietary software even after Linux and free software take over, assuming they do. I think Microsoft will eventually port their software to Linux if it gains enough

market share. While they may lose the revenue generated from selling Windows and providing various services for it, I don't think they will disappear entirely. And they may be able to find different revenue streams.

Open-source, however, can change the rules of the game, and I believe it will. In a world dominated by open-source, proprietary vendors must realise that they need to supply their customers with quality software, listen to what they say and act upon it, and constantly try to keep it above the open-source competition. There is no point in hiding the details of the Specs or protocols, and completely hiding the source code is not as important as many of them now think. If Microsoft survives in a Linux environment, we will see a much less abusive Microsoft.

My general ideology used to be a variation of "in for free beer". Use, code, and be guiltless and happy. A recent encounter with a free for some uses proprietary software whose licence changed and I became unable to use it any longer, slightly modified it. In the future, I'll be more careful in relying upon proprietary software, because it may become inaccessible to me, but otherwise still don't hold the vendors of it as immoral. I still use some not-entirely-free software because I like it and am used to it or it gets the job done.

My stance regarding the war between the term "open source" and "free software" is that I use either one when I find it appropriate, and am not fanatical to either term. It depends on the context of using it, who I speak to, what I wish to imply, what sounds right, or the first thing that pops out of my head. I usually prefer saying "Linux" over "GNU/Linux" because it is shorter, and more snappy and people will understand what I talk about. I do sometimes resort to "GNU/Linux", but not very often, and use the term a "GNU system" even more.

The fact I don't stick to either open-source or free software, stems from the fact that I respect both the Free Software Foundation, and the Open Source Institute, and believe that the free software movement and the open source movement is pretty much one and the same. I also like both terms.

I don't use GNU/Linux despite the fact that a large and integral part of it stems from the GNU project, for marketing reasons. GNU/Linux is longer than Linux and does not add more information, just a lot of pseudo-ideology. Add to the fact that many people will pronounce it "djee-enn-you-slash-Linux" when they first see it, and you'll get something that makes a very bad marketing name. Here's a nice quote from Linus Torvalds on why "Linux" is superior to "386BSD" (a 90's BSD clone that was free software as well):

```
> > Other than the fact Linux has a cool name, could someone explain why I
> > should use Linux over BSD?
>
> No.  That's it.  The cool name, that is.  We worked very hard on
> creating a name that would appeal to the majority of people, and it
> certainly paid off: thousands of people are using linux just to be able
> to say "OS/2? Hah.  I've got Linux.  What a cool name".  386BSD made the
> mistake of putting a lot of numbers and weird abbreviations into the
> name, and is scaring away a lot of people just because it sounds too
> technical.
```

Well, GNU/Linux is a step in the wrong direction in this regard.

As a developer I try to use Public Domain licences (like a pure PD, or variations of the BSD licence or the MIT/X11 licence) for software I produce. I don't mind people making a derived code proprietary much less integrating it inside proprietary products. If the original code, which I modify or derive from, is distributed under a different licence, I respect the original licence, whatever it may be.

I agree that some systems are critical enough to justify GPLing or LGPLing them. Even so, there are many BSD, MIT X11, or otherwise public domain projects out there, and they don't seem to suffer from it. Assuming that the GPL is the perfect licence for everything is a relatively immature notion, in my opinion.

Links and References

Resources for Further Reading

The GNU Project Philosophy [http://www.fsf.org/philosophy/philosophy.html]

This is a comprehensive sub-section of the site of the Free Software Foundation and the GNU project; it covers its philosophy (as put forth by Richard Stallman and others) in detail. Several articles from there were references in the site, and there are many others of interest.

Eric S. Raymond's Writings [http://www.catb.org/~esr/writings/]

Contains many interesting articles and writing about Open Source, from one of its most prominent figures, including the immortal "The Cathedral and the Bazaar Series".

- The Cathedral and the Bazaar [http://www.catb.org/~esr/writings/cathedral-bazaar/] - a document explaining the "Bazaar" way of managing a project.

- Homesteading the Noosphere [http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/] - an analysis of the "ownership" customs of the open-source developers community and their culture in general.

- The Magic Cauldron [http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/] - an analysis of the Economics of open-source.

- How To Become a Hacker [http://www.catb.org/~esr/faqs/hacker-howto.html] - a how-to document explaining how to become a "hacker", an expert and enthusiastic computer developer.

- Software Release Practice HOWTO [http://www.tldp.org/HOWTO/Software-Release-Practice-HOWTO/] - explains good conventions in the management and release of an open source software package.

Joel on Software [http://www.joelonsoftware.com/]

Joel Spolsky is a veteran software engineer that conveys his sometimes unusual views on software management and design in his site. Please don't take them too seriously, because he obviously doesn't either. Not all of his articles are relevant to the topic hand and here is a selected few which are.

- Strategy Letter V [http://www.joelonsoftware.com/articles/StrategyLetterV.html] - explains why big companies support open source and what's in it for them.

- Five Worlds [http://www.joelonsoftware.com/articles/FiveWorlds.html] - explains the differences between the different "worlds" of software development and the rules that apply to each.

David A. Wheeler's Site [http://www.dwheeler.com/]

A free software consultant and analyst which has many interesting pieces of research on his site.

Related Books

- The Mythical Man-Month : Essays on Software Engineering [http://www.aw.com/catalog/academic/product/1,4096,0201835959,00.html] by *Fredrick P. Brooks, Jr.* - one of the first books on software

management by the team leader of the OS/360 IBM Mainframe operating system. Despite the fact that it was originally written in 1975 , it still contains many useful insights on software engineering.

The Cathedral and the Bazaar aims to explain how the problems raised by this book can be easily resolved.

• Open Sources: Voices from the Open Source Revolution [http://www.oreilly.com/catalog/opensources/]

This book is available online and gives the opinions of many of the most prominent leaders of the Free Software and Open Source world.

Document Information

Author

Shlomi Fish, http://www.shlomifish.org/

Thanks

Thanks to Chen Shapira for taking the time to read several early drafts of this document and provide useful comments. Thanks to Nadav Har'El for summarising the early history of UNIX and the BSDs for me.

And naturally thanks to Richard Stallman for starting the GNU project, Eric Raymond for writing the "Cathedral and the Bazaar" series and so all the way back to thanking God for creating the universe.

To Do List

• Mention some other projects aside from Linux and other operating systems. (?)

Copyright

This document is copyrighted by Shlomi Fish under the Creative Commons Attribution licence version 2.5 [http://creativecommons.org/licenses/by/2.5/] or at your option a greater version.