

Open Source: History

Andrew Katz provides a companion piece to his article in the June/July issue of the magazine, Open Source Software: An Opening Resource, in this article outlining the history of open source.

Until recently, software was provided to the business world used software on a 'proprietary' basis. In other words, business paid for software provided by systems houses who licensed it to them. These licences typically restricted the number of users who could use the software, limited the CPUs it could be operated on, and specified the location where the software could be used.

The software house would, frequently, provide maintenance of the software, for a recurring fee. Any maintenance to the software was provided by the software house, for two reasons: one, the contract made sure this was the case, and two, for practical reasons, only the software house could, in practice, maintain the software as it alone had access to the source code.

Academic institutions, however, tended to rely on a different model. Software developed in university computing science laboratories was freely shared between those laboratories, and this involved sharing the source as well as the object code. Accordingly, a lab would frequently take a piece of code developed elsewhere (say to run the university's e-mail systems), improve or adapt it, and then let other labs have access to that software. Because the source code was always passed around with the object code, the labs always had the wherewithal to make those amendments.

Human nature meant that if you improved a piece of software you received in this way, you would pass the improvements back to the source (either through an obligation of reciprocity or because you wanted to prove you were the better programmer!) and the upshot of this was that software developed in this way tended to be robust, efficient and, crucially, had a widely dispersed cadre of experts across the different labs which had contributed to it who understood the functioning of the program in depth. This tended to be a pretty informal process, and indeed the use of the programs themselves was very rarely governed by licence agreements. When they were developed by the universities, these licences tended to be very permissive and have led to the software released under them (two of the most prominent licences being BSD from the University of California at Berkeley and MIT from the Massachusetts Institute of Technology) being used by commercial organisations, including Microsoft, as well as academic institutions.

And so, there seemed to co-exist two separate models: publicly funded academic software development which typically led to open-source software which was released under very permissive zero-cost licences, and privately funded commercial software development which was released under paid-for commercial licences under which the source code was not made available.¹

Occasionally, the commercial software companies, entirely legitimately, made use of software developed under the academic licences, and nothing prevented them from incorporating that software into their own, proprietary products provided that they complied with some (usually fairly) straightforward obligations on releasing the product, such as placing an acknowledgment in the product's manual that it contained software from, for example, MIT.

And this state of affairs would probably have continued to this day, were it not for two radical insights from one man: Richard Stallman.

¹ Except, generally, under escrow agreements, but that's a topic for another day.

During the 1980s it began to occur to Stallman that it was unfair that commercial enterprises should be able to benefit from all the work done by these academic programmers, by making a commercial profit from what they had obtained for free. He also realised that the range of software available under the free academic licences was pretty patchy. There were plenty of utilities for dealing with issues which universities often came across in their IT departments (such as how to network the computers to each other, and how to handle e-mail) and also tools for manipulating text, but no coherent suite of software from operating system, through to applications and programming tools.

Having worked in academic computer laboratories, Stallman saw how powerful their collaborative model was in creating excellent, high quality software, but he wanted to foster an environment in which this software could be created without being unfairly exploited by the corporate world.

He foresaw a world in which all types of software were made in this open environment, and that such software would be free to use, and free to amend and distribute. Unlike most idealists, he established a mechanism by which this aim could be achieved. Stallman realised that he needed a licensing mechanism which would ensure that, once software was released on this basis, it could not be (as he saw it) unfairly exploited by the corporates.

Therefore, in 1989, after a brief period of development, Stallman (aided by attorney Eben Moglen) started to release software under a licence called the GPL – the GNU Public General Licence. The most commonly used version of the GPL was issued in 1992 as version 2, and this is the version still in use today.

The GPL differed crucially from the more liberal licences issued by the academic institutions in that it required any person who received code licensed under the GPL, and went on to distribute it, either in combination with other software, or in a modified form, to license that combination or modification, at no charge, to all third parties.

Thus once you had incorporated some GPL code into your code, and went onto distribute it, the whole of that code became freely available to everyone under the terms of the GPL.²

In contrast to the academic licences, which allowed you take code issued under them and reissue it in a commercial context, once you use code issued under the GPL, anything you derive from it will have to be issued under the GPL. This concept, called copyleft by Stallman, is really the embodiment of his philosophy that once free, software should not be allowed to become proprietary again. Proprietary software houses had become used to copying academically licensed software repackaging and selling it on. Software licensed under the GPL looked superficially similar, but the copyleft licence terms prevented them from exploiting it in this way. It is this characteristic of the GPL which causes more caustic observers to describe it as 'cancerous' and 'viral'.

Stallman and the FSF had developed some excellent software tools, including the multi-purpose text editor EMACS and a suite of compilers such as GCC, as well as other significant parts of an operating system (called 'GNU'). The main thing which was missing was the core of the operating system itself. The operating system is the all-important framework software on which all other software on the computer has to communicate to provide access to the disks, screen, networking capabilities, peripherals and memory of the computer. It arbitrates between the different pieces of software

² This is a massive simplification, and the terms of the licence remain subject to debate.

running on the computer to ensure that they all run and communicate effectively and shares out the computer's available memory between them. Microsoft Windows, in its various guises, is an operating system.

The FSF's operating system made slow progress, and in 1991 the core (technically, kernel) of a new independently-developed operating system called Linux was released by Linus Torvalds under the GPL. It progressed rapidly using the collaborative model made possible by the Internet³ and hundreds of widely dispersed programmers worked on the various different facets of the system to produce the first versions of Linux capable of running quickly and reliably. Since then it has been developed to produce the stable and mature versions of Linux which we see today and which underpin everything from Google's search engine to mission-critical banking systems operated by banks like UBS Warburg and Morgan Stanley.

When combined with the suites of other software required to make a computer do productive work (programming tools, word processors, web development tools, games etc), the whole combination is called a distribution. Distributions of Linux are available from companies like Red Hat, Mandriva and SuSE. In practice, these distributions all rely heavily on GNU components, so are frequently called GNU/Linux. In practice, these companies products are available either for free (as in free beer), or are available at a price, the price including various levels of warrant, maintenance and support (which is of a comparable standard, and price, to that provided by traditional proprietary software vendors).

Since then, thousands of pieces of software have been developed to run on Linux. Many are open source. Many are proprietary. Many are excellent, and many are dreadful.

The increased commercial profile of Open Source software has led, perhaps inevitably, to a schism in the philosophy of open source. The fundamentalists are represented by the Free Software Foundation, with Richard Stallman at its helm (which prefers the term 'free software' to open source. The more liberal and corporate wing of the movement is represented by the Open Software Initiative, which concentrates less on copyleft as a mechanism for the promotion of open source than the FSF. Both have criteria which licences must comply with to be regarded as Free Software (for the FSF) or Open Source (for the OSI). In practice, there is little distinction between the two definitions.

The most widely used applications are based on 'stacks'. A stack is a hierarchy of software modules, based on top of one another. The most common is called the 'LAMP' stack, which stands for 'Linux, Apache, MySQL, Perl/Python'. Increasing numbers of applications within business are delivered to the users on a web browser, even for internal use. These include accounting systems, call centre systems, CRM (customer relationship management systems), management information systems and banking systems.

When run on a LAMP stack these applications will run on a server or servers running the Linux operating system, with the Apache web server providing the web pages, MySQL providing access to the database containing the accounts or customer data, and Perl or Python (two programming languages) providing the core software which drives the functionality of the application.

An attractive feature of the LAMP stack is that because each element of the stack is designed to be standards based, in theory, any component can be swapped out, whether for another Open Source application or a proprietary one. So for example. SCO is pushing

³ For more details on how this collaborative model works, see Eric S Raymond: The Cathedral and The Bazaar

the 'SCAMP' stack, which uses the same Apache MySQL and Perl/Python components as the LAMP stack, but substitutes the SCO's Unix as the underlying operating system.

Open source software has penetrated many other areas of computing: for example, versions of Linux are available to run on many different platforms from Dreamcast and Playstation games consoles, to mobile phones, TiVo personal video recorders and TomTom satnav devices.

Open source is still to penetrate the desktop where Microsoft still is the undisputed leader. However, the most recent releases of Linux (for example, Fedora Core 5 and Ubuntu 6.06) are designed to be easy to install, easy to use, and have a user-interface as attractive as Microsoft Windows XP's. There are also open source office applications available (such as Sun's Open Office – which I am using to write this article) which are comparable with Microsoft Office in terms of ease of use, functionality and stability. It is also important to note that they will read and write Microsoft's file formats.

However, this does miss the point to a certain extent. An increasing number of desktops are purely running applications running within browsers (built on the infamous LAMP stack), and in this case a lightweight version of the Linux operating system with a browser on top (such as Firefox) provides an easily installable, reliable, stable, zero cost (at least in terms of software licensing) system which can easily run on low-end hardware. The reduction in hardware, software, energy, support and training costs is very attractive to organisations whose software is browser based.

In practice, it's less easy to justify the transition to Open Source where users are used to the traditional Microsoft desktop and office suite: especially where they are using some pieces of software (such as a solicitors' accounts system, for example) which has been written only to run in a Windows environment.

There are now open source products which interface with Microsoft Exchange Server to provide equivalent functionality to Outlook, but these still have some way to go to match the richness of Outlook itself. Ironically, the biggest threat to Outlook sitting on the Microsoft Desktop may turn out to be Microsoft's own server-based 'Outlook Web Access' system. This lets you access an Outlook-like interface to your mailboxes, calendar and contacts through a browser directly from your Exchange server, without needing Outlook installed on your desktop.

For a company which tries to concentrate applications on the desktop, Microsoft has created a cracking piece of ASP software! OWA works just fine on Firefox and Opera, and does not need Internet Explorer to run.

It's also worth pointing out that a lot of open source desktop software will run on a Windows desktop just as happily as on Linux.

Looking to the future: Open Source software has an extremely strong position in the server room and there is no reason to believe that that position is under attack from proprietary products. It is also becoming increasingly popular in embedded devices, such as MP3 players, SatNav systems, industrial automation systems and network infrastructure. The future on the desktop is not so clear, but my money is on open source finding its way there over the next few years through a combination of applications such as call-centre and banking systems, which are mainly browser based, and its use in desktop extender appliances, such as tablet PCs and PDAs, which people can use while away from their desks to access their e-mails, documents, accounts systems and contacts.

Andrew Katz: Partner at Moorcrofts LLP, where he specialises in technology law with a bias towards open source software. In a former life he was a developer and has released software under the GPL.

20/07/2006