THE FIGHT FOR FREEDOM

Free Software isn't just about getting shiny new programs for no cash – it's part of a much larger social movement. Mike Saunders and Graham Morrison explore the history and future of FOSS.

There's a problem with the word 'free'. Specifically, it can refer to something that costs no money, or something that isn't held down by restrictions – in other words, something that has liberty. This difference is crucial when we talk about software, because free (as in cost) software doesn't necessarily give you freedom. There are plenty of no-cost applications out there that spy on you, steal your data, and try to lock you in to specific file formats. And you certainly can't get the source code to them.

To make the distinction clearer, many people refer to free (as in liberty) software as a proper noun: Free Software. But it's important to note that Free Software didn't just pop up as an idea one day, as a "wouldn't it be cool" notion from some hackers in a pub. The principles behind Free Software go back to the early days of computing, and many people have fought long and hard to protect freedom in computing, even when all hope looked lost.

So this issue we want to delve deep into the world of Free Software: where exactly did it come from, why is it important, and what challenges are ahead. We also look at the differences in licences, one of the thorniest issues in FOSS, especially when people have different definitions of "free". But let's start by going back to the early days of computing, when the world was a simpler, happier place...

FOSS before there was FOSS

Free Software goes back to the 1950s – it just didn't have a name back then.

The idea of releasing software as binary-only executables, without access to the source code that generated them, is relatively new. Yes, commercial software has existed for several decades, but back in the 50s and 60s, as mainframe computers started finding their way into businesses and universities, it was completely normal to get source code with a machine or software package.

Take the UNIVAC 1, the second commercial computer produced in the US: its A-2 compiler was supplied with source code, and customers were encouraged to send their modifications back to UNIVAC. This is FOSS just as we know it, but back in 1953! And it made absolute sense, because improved code was better for users, for the computer makers, and for everyone else who needed data generated by those enormous machines.

So this was the norm at the time, and there are plenty of other examples, such as IBM distributing operating system source code with its mainframes. When Richard Stallman joined the AI Lab of MIT (the Massachusetts Institute of Technology) in 1971, source code was everywhere: "Sharing of software was not limited to our particular community; it is as old as computers, just as sharing of recipes is as old as cooking. But we did it more than most."

But the times were changing. Companies started to see software as commercially viable products, and not just handy things to bundle with hardware. Stallman saw this happening at MIT, where more and more computers were being supplied with proprietary (closed) operating systems. He saw his beloved community of hackers, engineers and sharers being destroyed.

The straw that broke the camel's back was a printer driver: Stallman needed the source code to add some vital features. But in order to access the source code, he had to sign a non-disclosure agreement, which essentially prohibited him from sharing his improvements with his co-workers. What kind of a world was this becoming, where companies deliberately try to stop you from helping your fellow man? Why set hackers against each other, when they could work together to make a better world?

Richard Stallman started the Free Software movement not just to make low-cost programs, but to encourage sharing and benefit the world. (Image: Richard Stallman CC-BY-ND, **https://stallman.org/photos**)

> **"Why set hackers against each other, when they could work together to make a better world?"**

So a deeply despondent Stallman had a choice. He could either choose to leave the computing world altogether, or create a new project comprised entirely of software that's free from non-disclosure agreements and other attempts by companies to eliminate collaboration and sharing. In 1983, Stallman created GNU (GNU's Not Unix), a new operating system with a Unix-like design, for everyone to share. The announcement is one of the most famous Usenet posts in internet history: **www.gnu.org/gnu/initial-announcement.html**.

GNU alone wouldn't save the software community, though. Stallman also founded the Free Software Foundation, and created the GNU General Publish Licence, which described software freedom in legal terms and prevented anyone from taking his work and locking it up in proprietary software.

By 1991, much of the GNU system was complete, although the kernel (HURD) hadn't seen much work. However, a non-GNU kernel project called Linux was becoming usable, and paired with the GNU software, a complete operating system could be made. Stallman, and many others from the GNU project, prefer to call the operating system GNU/Linux for this reason, and to emphasise that GNU is a project for computing freedom, and not just some useful bits and bobs that run on "Linux". For brevity we use "Linux" to describe the OS in this magazine, but we appreciate the argument that it should be called "GNU/Linux".

The BSD alternative

Even while companies were trying to monetise software, code sharing remained common in academic circles. BSD, the Berkeley Software Distribution, was a Unix flavour that started life in 1977. Its source code ended up in legal tangles in the early 1990s, as GNU/Linux was beginning to take off, but the situation was resolved and today we have three major spin-offs: FreeBSD, OpenBSD and NetBSD. They share a lot of similarities with GNU/Linux, but the licensing is different (more over the page) and the developers tend to focus on the practical aspects of source code availability, rather than societal implications of freedom and sharing. Some BSD fans regard BSD as the original Free Software, and GNU just happened to pick up on it later.

So many licences...

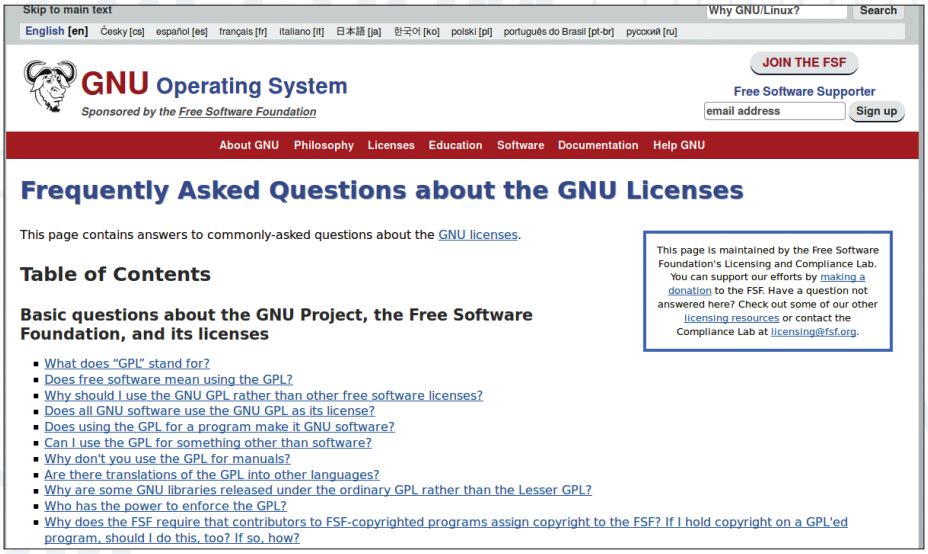
GPL, LGPL, Affero GPL, BSD... there are many ways to make code free (as in liberty).

Free Software, according to Richard Stallman, should grant users four essential freedoms:

- Freedom to run the program for any purpose.
- Freedom to study how the program works (ie look at the source code).
- Freedom to distribute copies to help your neighbour.
- Freedom to distribute your changes in source format.

Now, you could easily knock together a quick 100-word licence based on these preconditions, but to make it last over the years and have a significant legal foundation, you need something longer. This is why Stallman created the GPL, the General Public Licence, which is quite long but makes it very hard for malicious types to subvert it.

Consider, for instance, source code. A dodgy company using GPLed code could release its modifications as assembly language listings, generated by a disassembler. This is, strictly speaking, "source code", but it's of little use to developers who want to incorporate modifications back into the main tree. So the GPL describes source code as the "preferred



The GNU project takes its licences seriously – the FAQ for the GPL is over 22,000 words long!

A quick chat with: Richard Stallman

The creator of GNU, the Free Software Foundation and the GPL

What do you see as the biggest challenges facing Free Software right now?

Richard Stallman: Computers designed to make it impossible to run free software. These include Apple and Microsoft phones and tablets, the modem processors of all new portable phones, computers in cars, and so on. Many of them check for manufacturers' signatures to make it impossible for users to change the software in their own computers.

Also, services that refuse to function except through nonfree apps or nonfree code sent to the browser in a web page. Many of these are nasty in other ways too – for instance, they track people and collect dossiers, thus endangering democracy. See **www.gnu.org/philosophy/surveillance-vs-democracy.html**.

Are there any problems approaching that could make a GPL v4 necessary?

RMS: Not that I know of.

From a wider perspective: tens of millions (if not more) of people now benefit from Free Software, and a free platform in the form of GNU/Linux. It's perfectly possible to do almost every mainstream computing task without being restricted by proprietary software. Obviously there are still some battles to fight, but are you satisfied on the whole? Is there anything else outside of software that you'd like to tackle?

RMS: The idea of the free software movement is that users should have control over their computing, so also over software they use. (See **www.gnu.org/philosophy/free-software-even-more-important.html**) Given that nonfree software is nowadays typically also malware (see **www.gnu.org/philosophy/proprietary** for examples), a free society calls for replacing all nonfree software with free software.

We have advanced a long way starting from near zero in 1983, but we have a long way left to go. As of yet, we have freed only a small fraction of the inhabitants of cyberspace, and that is mostly limited to the field of PCs.

Finally, are you still using the Lemote netbook you had for a while, or have you moved on to the Free Software Foundation-approved refurbished Thinkpad?

RMS: It's called the Gluglug, and yes I have switched. In practical terms it is a lot better. (**www.fsf.org/news/gluglug-x60-laptop-now-certified-to-respect-your-freedom**).

form for making modifications" – in other words, code in the original language.

The GPL uses copyright law to make sure that the rights to distribute and modify Free Software remain in the code, and nobody can suddenly lock it down under a different license. This strategy is known as "copyleft" in the FOSS world. But there are various versions of the GPL:

- **GPL v2** Provides the rights given above, and is used in Linux (the kernel).
- **GPL v3** As above, but with extra clauses relating to software patents, DRM (you can freely break DRM that's implemented in Free Software) and the right to replace GPLed software on locked-down hardware such as TV set-top boxes.
- **LGPL** The "lesser" GPL, which allows linking with proprietary applications. The GNU C Library (**glibc**) uses this. But why does it exist, when the FSF is against proprietary software? Basically, it's better to have non-free programs using free libraries rather than proprietary equivalents – as it gives users slightly more freedom.
- **Affero GPL** Like the GPL, but if you run GPLed software on a server and users run communicate with it (like a web app), users should also have the right to access the source code.

There are other GNU licences, such as for documentation, with the full list at **www.gnu.org/licenses**. Interestingly, Linux (the kernel) hasn't upgraded to the GPL v3 due to objections from Linus Torvalds. He doesn't think it's wrong if hardware manufacturers want to restrict users from modifying software, noting that he installs Linux on his children's computers, and has the right to stop them from upgrading it. The GPL v3 is "overreaching" accordingly to Torvalds, and isn't "morally" where he wants to be. (See his full explanation at **http://tinyurl.com/npmfwvz**).

"Why set hackers against each other, when they could work together to make a better world?"

Free, or even freer?

While the GPL v3 is over 5,600 words long, there are alternative and much simpler licences. The most notable is the BSD licence, used by FreeBSD among other projects, at just 233 words. This basically says: do what you want with the code, but credit the original source, and don't sue us for anything that goes wrong.

Now, this leads to an involved philosophical debate about which licence is more free. From one side, FreeBSD fans would argue that their licence is the freest, as it enforces fewer restrictions on its users. You really can do what you want with the code, including folding it into proprietary software, just like Sony did with its PS4 operating system (which was based on the FreeBSD kernel).

GPL fans counter with: yes, the GPL has more restrictions, but these are put in place to maintain the user's freedom down the road. The GPL is the freer licence as it actively fights for freedom.

Who's right? The arguments will go on for years, no doubt. But the general consensus tends to be that the BSD licence provides more freedom for developers, while the GPL is better for end users.

A quick chat with: Microsoft!

Gianugo Rabellino, senior director of open source communities at MS Open Tech.

LV Microsoft today is heavily involved in various open source projects and releases a lot of code under OSS licences. What brought about the change in attitude since the early 2000s? Is it a grass-roots campaign in Microsoft, or a bigger corporate strategy?

Gianugo Rabellino: Microsoft sees openness as a way to satisfy our customers and grow our business. This involves enabling open source applications to run better on and with our Microsoft platforms, but also to deliver great Microsoft experiences to other device platforms.

Our open source strategy has evolved based on conversation with our customers, many of whom operate heterogeneous IT environments with traditional commercial software, commercial open source software and community-based open source software working side-by-side.

So just how far we have come? Our CEO Satya Nadella recently said "Microsoft loves Linux," and described how there are 1,000 Linux virtual machines to choose from for Microsoft Azure, and that Linux and various packages of Linux comprise 20% of Azure's workloads. But those who follow our Linux work more closely will know that we've had Microsoft engineers actively contributing to the Linux kernel for over five years.

Openness is increasingly becoming part of the company's DNA – multiple teams across Microsoft are involved in open source, standards and interoperability efforts. It's both a top-down and bottom-up approach – with customers and developers at the centre.

LV How is open source being used in Microsoft now? What would you describe as the company's biggest open source projects?

GR: Microsoft currently participates in over 800 open source projects on GitHub, and that number is growing. We work with many open source communities to identify valuable opportunities, projects and initiatives in which we want to participate, often focusing on improving open source application interoperability with our products, and using an open source development approach when it makes sense for specific products and solutions.

One of our most significant open source projects was our recent announcement that Microsoft is open sourcing the full server-side .NET stack and expanding .NET to run on the Linux and Mac OS platforms. A large chunk of .NET was already open in the ASP.NET family of technologies, and this change builds from that successful initiative.

Real GNU/Linux distributions

Our recommendations for maximum freedom.

We know there are hundreds of distributions to choose between. But there are far fewer choices and some compromises to make if you want to use a GNU/Linux distribution that's endorsed by The Free Software Foundation for adhering to Richard Stallman's guiding principles. Choice is reduced because the Linux kernel can't contain any of the proprietary blobs of firmware that are tolerated by most other distributions. These kernels are given the name 'libre', and they can have an impact on hardware compatibility and performance.

If there's no open source driver, you'll also need to invest in different hardware. This used to be a much bigger problem 10 years ago, with many modems, wireless dongles, printers, touchpads and graphics cards rendered useless without their manufacturer's proprietary blobs. Fortunately, the Linux kernel is in much better shape, and most modern hardware we use will 'just work', which means there's a good chance a free distribution won't require new hardware unless you're using something esoteric. The main decision is which distribution to try, and while there are quite a few, not many receive the same number of updates you'd expect from an active distribution, which is why we're only going to look at four.

The biggest difference between these distributions and the ones we usually cover is their emphasis on freedom – that sometimes means sacrificing features, but it's in a noble cause.

Trisquel 7.0 LTS

It used to be the case that 'free' GNU/Linux distributions weren't as usable as their non-free counterparts. That could make switching difficult for non-technical users. That things have changed is partly thanks to what we'd consider the most popular GNU-centric distribution, Trisquel. Trisquel has been downloaded 344,786 times since its 2.0 release, and now uses Ubuntu as its foundation, making it an easy migration for millions of Ubuntu users. The latest release is a re-working of the 14.04 Long Term Support version of Ubuntu, which means you'll get updates until 2019. There's a wide variety of download choices, from a 3 GB ISO that includes source code to a 25MB ISO that needs a network installation. We opted for the 1.5GB DVD image, which can also operate as a live desktop. Its Gnome-based installer looks amazing, and while it's great that the *Orca* screen reader was enabled by default and speaking many of the options you make and see on-screen, we couldn't an easy way to disable it (it's Alt+Super+S). And that's all there is to installation. Within minutes, we had our new system up and running.

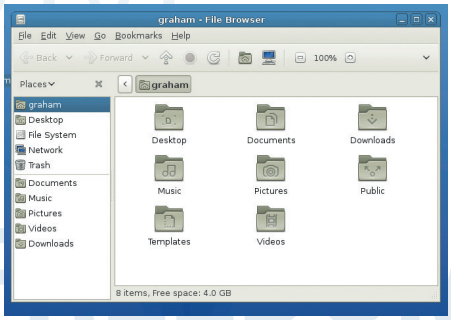
Trisquel defaults to running Gnome in its classic mode, and like the installer, we really like the appearance of the default theme. Most of the default applications are identical to a standard Ubuntu release. There's *LibreOffice*, *Rhythmbox*, *Gimp* and *Evolution*. The web browser is based on version 33 of *Firefox* and it's called *Abrowser*. It worked well for us, defaulting to DuckDuckGo for searches as well as offering clear options for disabling JavaScript or installing the more privacy focussed *GNU/IceCat*.

http://trisquel.info

gNewSense 3.1

Second to Trisquel in popularity, gNewSense is a little more austere in the appearance category. This is primarily because it's using a little-themed version of Gnome 2.x and older packages than most distributions, and this is because of its choice of base distribution. After a few years of using Ubuntu as its base, version 3 switched to Debian 6, first released in 2011, and many of the version numbers of packages stretch back from that point – hence Gnome 2.x. Like Trisquel's Ubuntu repositories, it Debian that many packages can be installed very easily, and because the non-free repositories are disabled automatically, you don't have to worry about. Despite its age, installation is straightforward although slightly more intimidating than Trisquel. You're asked for your network's DNS address and manual confirmation of how your partition table is going to be generated, for example.

Using this old version of Gnome is a reminder of how much has changed. It's quick and functional, but doesn't have any of the bells and whistles of newer versions



Based on an older version of Debian, gNewSense looks a little dated and may not work on the latest hardware.

– we still find *OpenOffice* here rather than *LibreOffice*, for instance, and the desktop doesn't look anything like as good as Trisquel. The older kernel, 2.6.32, is a little more worrying. Trisquel sports version 3.13, complete with low latency patches and bfq scheduling, but more importantly, many more hardware drivers and updates, making gNewSense less likely to work with modern hardware, at least until it catches up with the latest Debian release.

www.gnewsense.org

Parabola

Parabola is a relatively recent addition to the Linux Foundation's list of free distributions, being ordained in 2011, and it's a little different to both gNewSense and Trisquel. This is something you find out within 30 seconds of booting the 500MB Live ISO because you're dropped as root into the command line and curtly told that if you want to install Parabola, you'd better have a working networking connection, and that to work out how this is done, open **network.html** into *Lynx*. Things could be worse. They could have insisted on loading the html file into *Emacs*.

If this sounds familiar, it's because Parabola is built atop Arch, a distribution that's spawned a couple of 'libre' kernel distributions. You can even migrate from a regular Arch installation if you'd rather rid yourself of those pesky proprietary bits. We like Arch a lot here at LV Towers, but it's not for the uninitiated. Fortunately, the barking words thrown onto the screen at login are worse than their bite, and we found networking already up and running, putting off our Herculean struggle with Systemd for another day. And while you'll need to configure and install everything else you want to use, including a graphical environment, and carefully follow the installation instructions, this is still a wonderful distribution. In some ways, building your own installation with an Arch distribution is a great way of appreciating the amount of work that goes into creating a working system, especially when you know that every package is untarnished.

https://www.parabola.nu

```
Parabola GNU/Linux-libre 3.10.10-1-LIBRE  (parabolaiso) (tty1)

parabolaiso login: root (automatic login)

================================================================================

        Parabola live media 2013.09.01

    To install Parabola, the system must be connected to the internet.
    For instructions, enter this command:
      lynx network.html

    Press the number keys while holding Alt to switch virtual terminals.
    This allows entering commands without closing lynx.

================================================================================

root@parabolaiso ~ # _
```

It's quite a shock how little a 500MB actually gets you these days. Judging from the text, maybe not even a network connection.

> **"There are some compromises to make if you want to use a distro that's endorsed by the FSF."**

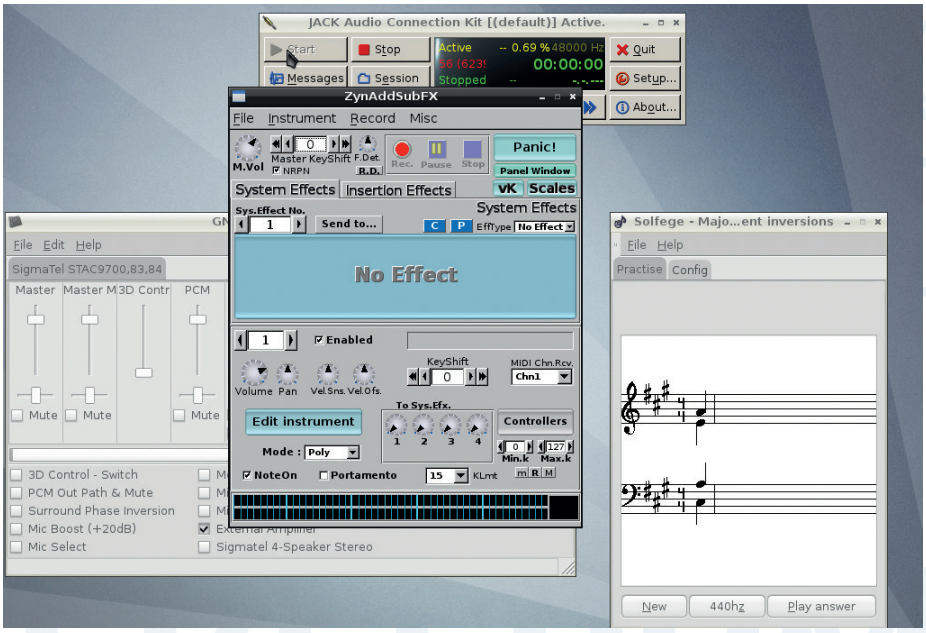
Musix v3.0.1

The three distributions we've looked at so far have been functional and modifiable just like any other distribution. Musix is a reminder that not all 'libre' distributions need to focus on sober functionality, and it does this by being a Debian-based distribution designed for music and media creation, which we think is a great idea. There's not too much choice on the download medium, and the 2GB download can take a while from the limited server capacity, and unlike the other three distributions we've looked at here, there's no torrent we could fine. Installation is easy though. The Live DVD defaults to Spanish, but there's English, French and Portuguese too, and they're all selectable from the boot menu, which is an excellent idea. This is also the only distribution we've looked at that boots to an augmented KDE desktop (username: **live** password: **user**).

There's pretty much every audio application and effect you've ever heard of installed, along with some lots of other multimedia tools like *Kdenlive* for video editing and *Blender* for 3D generation. The most important feature is that the Jack audio system is already running, and you can control it's parameters and connections with the *QJackCTL* application that's also included.

One tool we'd not seen before is *GNU Solfage*. This is music educational and training tool. It can play intervals and rhythms, for example, and ask you to identify them, you can create and train yourself about scaled, and chords and keep on top of your progress. The user interface is simple, but it's crammed full of essential content that can really help.

https://musixdistro.wordpress.com



Forget about proprietary plugins and formats with a music-making distro focussed on freedom.

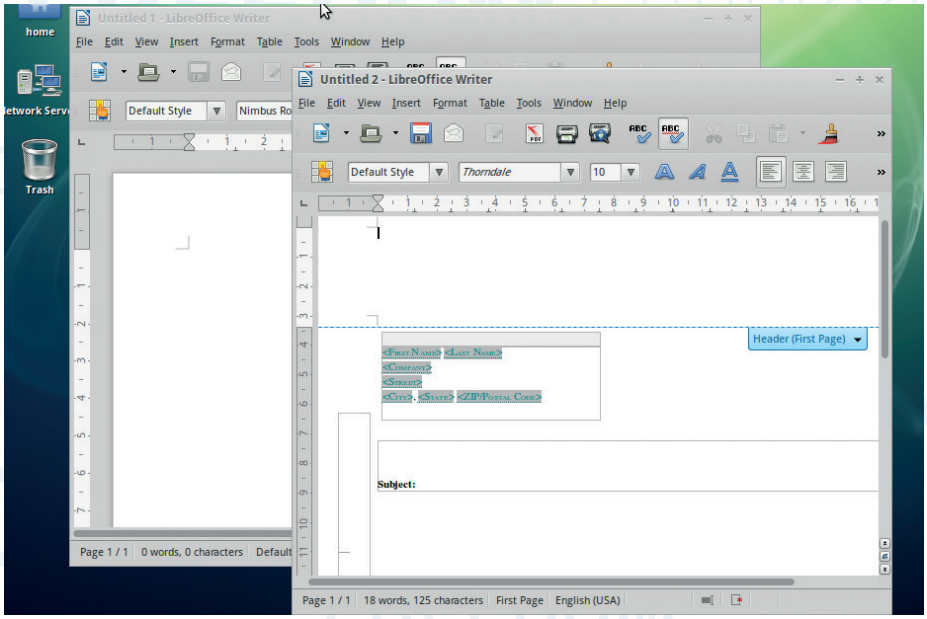
LibreOffice vs OpenOffice.org

Our essential office suite proves that Free Software licences are important.

The recent history of both the *OpenOffice.org* and *LibreOffice* projects encapsulates a lot of what is good in open source philosophy, and what wider good can be achieved. We think it's also a great example that exposes many of the issues brought about when these kinds of projects are large and successful, and how they interact with both their community and their corporate sponsors.

Having an 'office' suite of applications for Linux has always been absolutely vital. At work, we all know that text documents and spreadsheets spend their lives in perpetual motion between colleagues' email accounts, and nearly of these documents will have been created by *Microsoft Word* or *Microsoft Excel*. These two applications are a cornerstone of Microsoft's still unrivalled business strategy and unapproachable influence, and they have been dominant for over 20 years. As such, they've been fiercely guarded jewels in Microsoft's crown. Microsoft famously blocked its rival, IBM, from selling Windows 95 in an attempt to undermine IBM's own office suite, Lotus SmartSuite. Late in the same decade, another rival, Sun Microsystems, acquired a commercial office suite called *StarOffice* and open sourced the code in an attempt to subvert the influence of Microsoft's *Office*, eventually releasing version 1.0 of *OpenOffice.org* in May 2002.

OpenOffice.org has always had a strong focus on embedding excellent import and



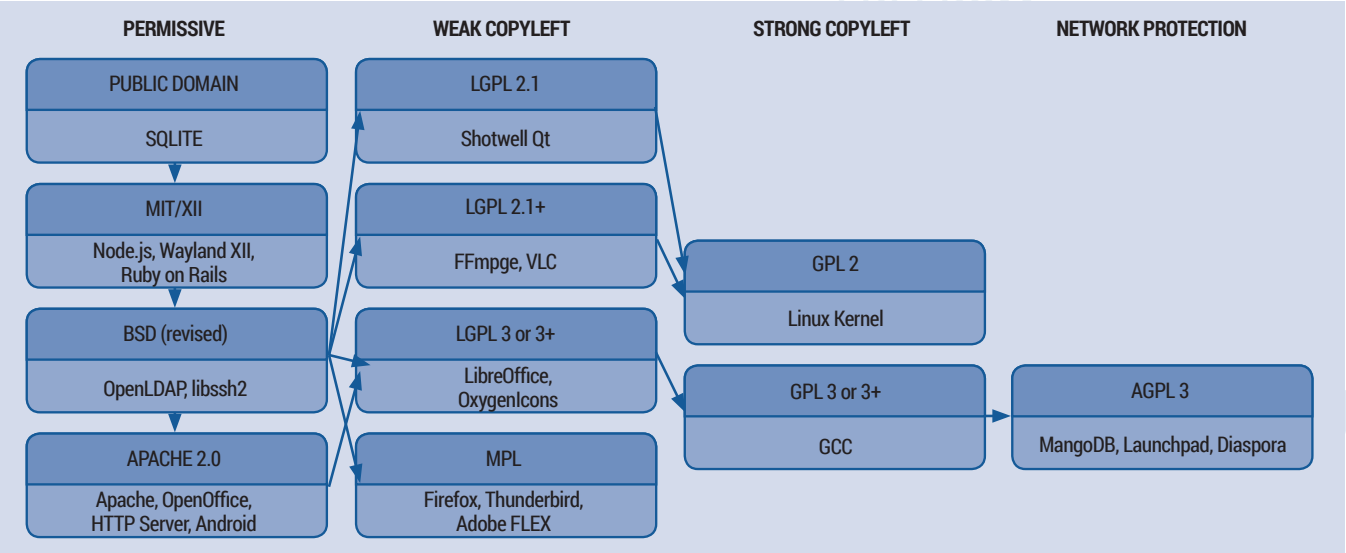
LibreOffice **has supplanted** ***OpenOffice*** **in nearly all Linux distributions, but it still faces a battle for recognition outside of the open source community.**

The open source licence spectrum

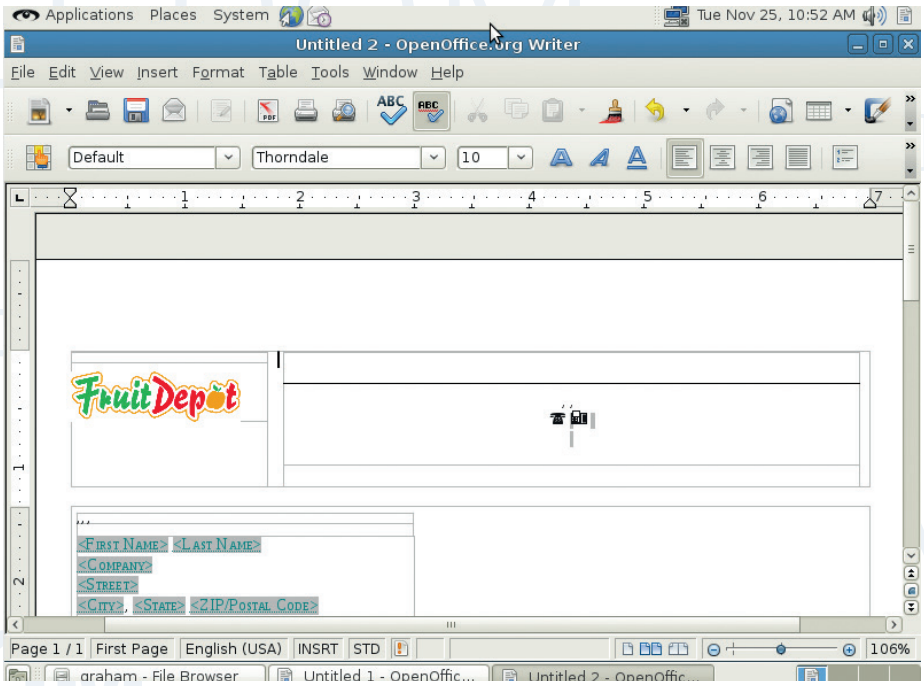
Not all licences are the same. Some enable you to do more things than others. With a BSD-style licence, for example, you can often create proprietary code without having to release your own changes. This is what allowed Apple to take parts of FreeBSD and NetBSD for its own Mach kernel without having to provide its own changes.

This is great in certain circumstances, especially when a developer just wants to get their idea out there, but the Free Software movement is built upon the users of that software having the same access to the new developments, and that means having access to the code and being able to make their own modifications. These licences are less permissive and are stronger 'copyleft'. Finally, we have the Affero General Public Licence, which is recommended by the Free Software Foundation when code is running across a network.

Based on a 2007 illustration by David A Wheeler (CC BY-SA 3)



export compatibility with Microsoft's own formats, which meant that not only did Linux inherit a fully fledged office suite, it also gained vital compatibility with the file formats everyone was emailing between themselves. This has subsequently helped Linux become a real viable alternative to Windows, as shown in many places such as Munich city council's LiMux project. But more importantly, it has also been instrumental in making its OpenDocument Format (ODF) an ISO standard, and it could be argued that *OpenOffice.org*'s viability for document editing and interoperability has paved the way for a change in attitude for many institutions who previously saw Microsoft's applications and formats as the only possible options.



***OpenOffice* is in the very capable hands of The Apache Software Foundation, but a merge with *LibreOffice* now seems almost impossible.**

Oracle

We'd argue that none of this would have been possible without *OpenOffice.org* being an open source project (LGPL v3 has been used since version 2), and the project is significant because the licence has both ensured its openness and its survival. Oracle acquired Sun Microsystems in 2010 and muddied the future of many of Sun's open source projects like *OpenOffice.org*, *MySQL*, *VirtualBox* and of course, Java.

Oracle also reduced the number of developers working on what was now called Oracle Open Office, and this apparent lack of progress led to what's best described as a 'fork', in a similar way that Xfree86 became X.org, and both forks were only possible because of the licences used to host and share the source code.

Names, such as *OpenOffice.org*, are trademarked and not typically part of the open source side of a project. The Document Foundation (TDF) was created to take control over the project, after initially hoping that Oracle would rather hand over *OpenOffice.org* to TDF than run it itself. When this didn't happen, The Foundation created its own fork of the source code and voted to rename the liberated version *LibreOffice*, which has since gone on to replace *OpenOffice.org* in all of the major Linux distributions.

However, the story doesn't end there. *OpenOffice.org* still has some powerful brand recognition and is still downloaded and used in many places, despite *LibreOffice*'s superiority, and Oracle did eventually decide to give the project away, *OpenOffice.org* was given to The Apache Software Foundation, a *bona fide* repository for open source projects, and this donation must have left it with something of a quandary. On the one hand, the Apache Software Foundation now had an important piece in the free software puzzle under its control, a gateway suite for people migrating from proprietary systems. On the other hand, *OpenOffice.org* had been morally supplanted by *LibreOffice* in the hearts of many open source users. The Document Foundation also made it clear that it had no intention of shifting direction when it made its own statement shortly after Oracle's relicensing, "The Document Foundation and *LibreOffice* represent already a future path of development for the *OpenOffice.org* community and the *OpenOffice.org* code base, as was originally announced on September 28, 2010."

"As users and advocates, we still have LibreOffice, which is by far the most important thing."

OpenOffice rides again

The Apache Software Foundation has since developed the suite on its own, which has revealed another final twist to the saga. Without the same resources, *OpenOffice* development has been slower; both projects have worked on their own aspects to the code, but there's also some sharing. Unfortunately, this is always where subtle differences in their licensing begin to have an effect. *Apache OpenOffice* uses the generally more liberal Apache Licence, as you might expect. *LibreOffice*, by contrast, has inherited LGPLv3 and the MPL 2.0 (Mozilla Public Licence). This makes moving code from one project to other the much easier in one direction – from the more liberally licensed code to the less liberally licensed code, and that means from *Apache OpenOffice* to *LibreOffice*.

It's a shame that we have two such similar projects, but it's difficult to see how it could have happened any other way without Oracle donating the code to The Document Foundation at an early stage. That this didn't happen could have simply been a lack of understanding at Oracle, or the subtle machinations of a large corporation with its wide and convoluted approach towards the open source projects it curates. Either way, as users and advocates, we still have the project and the software, which is by far the most important thing, and something that would never have happened with a similar proprietary piece of software.

Despite all this manoeuvring and strategy, and despite the long history of the software, it's open source that has ensured its survival and continued growth. And we don't think there's any other system that could have produced the same result.