HOW TO WIN FRIENDS AND INFLUENCE PEOPLE

ILLUSTRATION: STUART HARRISON

or: EFFECTIVE EVANGELISM FOR GEEKS

Professional evangelists Zak Greant and Jennifer Zickerman show you how to turn friends, colleagues and clients on to free and open source software.

You use Linux; you almost certainly use *Firefox*; you probably use *Apache*, PHP or Perl or Python, maybe *Gimp* and *OpenOffice.org*. Stand up and take a bow: you've just been promoted to the position of free and open source software evangelist.

Have you ever been collared at a party by someone who has recently adopted a new diet or exercise regime – or, even worse, quit smoking – and can't shut up about it? As they rattle on about the joys of their new lifestyle, telling you why you should do the same, you take guilty slugs of your beer and notice that, invisible under your shirt, the top button of your jeans is undone.

That's ineffective evangelism. After an encounter like that, the last thing you want to do is spring out of bed early the next morning, ingest a high-fibre, caffeine-free breakfast and go for a jog. You probably feel like you should, but some statistic somewhere undoubtedly proves what we all know – the things we should do are often the things least likely to get done.

Effective evangelism is not a holy crusade, and it's not accomplished by self-righteous tirades about things that other people should do. Instead, it's the process of using your technical expertise and your experience with FOSS tools to solve real problems. Pick the right problems, apply the right tools, and *voilà* – your friends, your clients and your co-workers become FOSS users. Not because they should use FOSS, but because FOSS tools solved their problem and extended the usefulness of their own systems and of the services they use.

A history of sharing

In the early days of software development, most programs were distributed as source code that users configured and compiled for themselves. They were free to alter the source code to fix bugs or extend the program's functionality. The fixes and enhancements were freely distributed as a community service and as a way of gaining status within the group.

Few programs were commercial. Instead, software apps were the result of hackers working at various academic institutions, scratching their various itches and sharing their code with each other. This model of sharing and building on one another's work forms the philosophical basis of free software and open source.

In contrast, the early PC boom was characterised by commercial software distributed in compiled, binary format on physical media. Collaborative technologies like UUCP were not easily available to PC users; instead of a community of users who wrote their own programs and shared the source code, PC users were mostly restricted to compiled software distributed via standard commercial channels, such as retail outlets and catalogues.

While the commercial distribution model had financial benefits for the companies who made software, it hampered two natural human tendencies: the desire to organise into communities, and the desire to understand how things work. PC users responded to the first desire by copying and sharing compiled programs, and to the second desire by, wherever possible, tinkering with the compiled programs so that they could do more with them.

Software companies responded by adding security measures to prevent such copying and tinkering, and the race was on. One early event in this race was Bill Gates's famous Open Letter to Hobbyists (archived at **http://en.wikipedia.org/wiki/Open_Letter_to_Hobbyists**), in which he equated copying without permission to piracy.

When the world wide web emerged, the internet became easier for PC users to access. Advances in hardware – and the sheer number of PC users – meant a growing number of people considered their machines both as tools for running software and as platforms for writing software. It's easy to understand why so many people consider source-code access to be a right: if you own your toaster, you're allowed to take it apart and investigate how it works. Why wouldn't you be allowed to take your software apart in the same way?

They were allowed, of course – they just needed to use a different kind of software. FOSS enabled a generation of early PC users to become software creators. The early generations of Unix users had had no choice. There hadn't been any precompiled binary software options;

what they needed they created. PC users did have a choice, and many thousands chose to become involved in FOSS projects, where their desires for community and the freedom to explore and learn could be met.

What FOSS isn't

FOSS is not the revolution. It's not even *a* revolution. Nor is it a religion, in spite of the cargo cults that have sprung up around it. Even those folks focused on the ethical aspects of unhindered access to source code, such as the Free Software Foundation, primarily limit their focus to the issues of software licensing and free speech. They don't seek a New World Order.

Presenting FOSS as a revolution tends to put you firmly in the loony camp in the eyes of proprietary software supporters, and with good reason: revolutions usually either fail horribly or, if they succeed, have many unintended negative consequences. Rather than being a revolution, FOSS is merely a sane set of responses to a fundamental change in the information ecosystem, evolving from a combination of inexpensive computers, simple and open standards for data exchange, widespread computer literacy and a ubiquitous network.

This emerging infosystem highlights a key property of information, which is this: distribution and use don't reduce the quantity available for others. It's not a 'rival' good like most material commodities (food, clothing, shelter, and so on). After food is consumed, it can't be consumed again. Information, on the other hand, is never used up; it can be used by as many people as can access it. As Thomas Jefferson wrote: "He who receives an idea from me receives instruction himself without lessening mine; as he who lights his taper at mine, receives light without darkening me."

Information (especially of the software source code variety) has another interesting property: its value can increase through sharing. This concept is fundamental to the purpose of publishing academic papers – if someone publishes a paper on the ecology of mangrove swamps in the Zambezi delta, and somebody else uses the paper to further their own studies of juvenile bull sharks in the area, the original research is more valuable than if its knowledge had just stayed in the heads of the original authors, or had only limited distribution among their students. With FOSS, this process of increasing the value of information happens at an extreme rate. Ideas, debates, patches and distributions can occur within days, and involve hundreds of people. And the process is not purely serial, proceeding from one person to the next. Instead, many people can contribute value to the same idea at the same time.

RESOURCES FOR EVANGELISTS

The Cathedral and the Bazaar
by Eric Raymond
www.catb.org/~esr/writings/cathedral-bazaar
The Creative Commons
Free, open and permissive licences.
http://creativecommons.org
The Electronic Frontier Foundation
US digital civil liberties group.
http://eff.org
First Monday
Peer-reviewed journal of the internet.
www.firstmonday.dk/issues/index.html
***Free as in Freedom* by Richard Stallman**
www.oreilly.com/openbook/freedom
The Free Software Foundation:
USA www.fsf.org
Europe www.fsfeurope.org
India http://fsf.org.in
The Open Source Initiative
http://opensource.org
The Semasiology of Open Source
by Robert M. Lefkowitz
www.itconversations.com/shows/detail169.html
The Software Freedom Law Center
Law office that deals exclusively with FOSS-related legal issues.
www.softwarefreedom.org
Wikipedia entries on free software (http://en.wikipedia.org/wiki/Free_software) and open-source software (http://en.wikipedia.org/wiki/Open_source)
The Tipping Point
by Malcolm Gladwell (Abacus)
Describes the mechanics of epidemics and other mass phenomena.

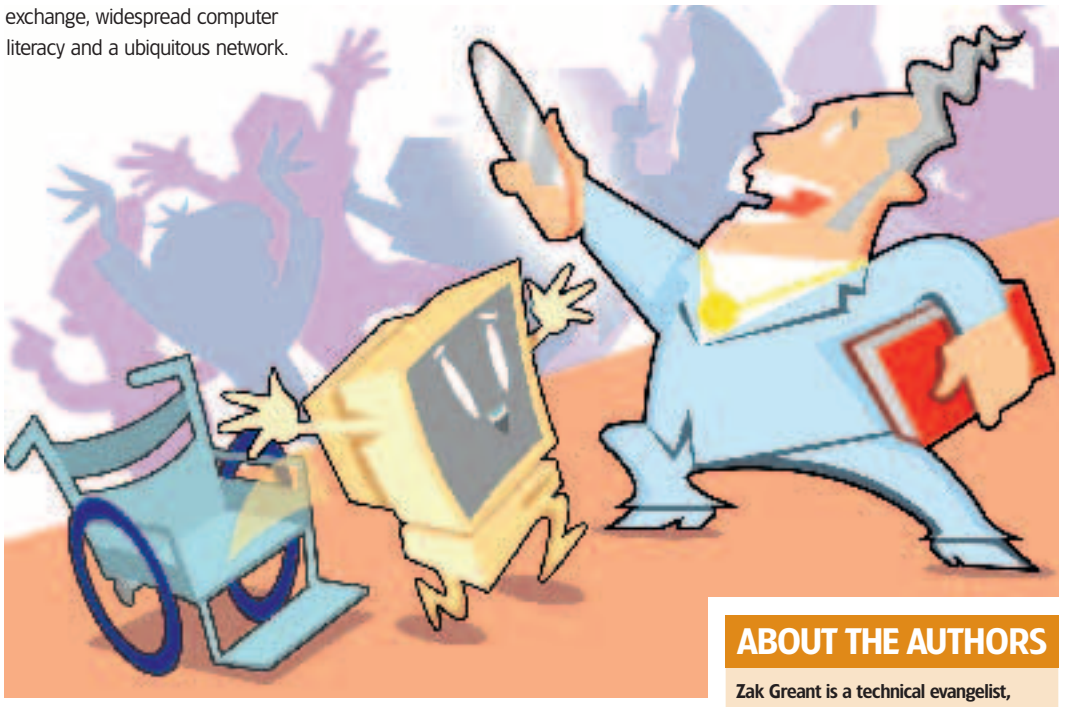
How, then, can you take these ideas and begin to convert those around you to Linux? At its simplest, FOSS evangelism is about one user and one application. Think about *Firefox*. Its usability benefits are so overwhelming compared with *Internet Explorer* that people only continue to use *IE* either because they must (due to company policy, say) or because they don't know about *Firefox*.

So tell them. Focus on the issues that matter to them. If you're discussing *Firefox* with a web designer, for instance, show them the Web Developer extension (**www.chrispederick.com/work/firefox/webdeveloper**), which provides embedded tools for developing and debugging CSS and HTML. Remember that people are primarily interested in features and cost; development methodology, philosophy and licensing are secondary issues.

Mini-evangelism

Simple evangelism – one user, one application – might not change the world, but remember that FOSS adoption is a process, and an exponential one, at that. Success with one user and one application leads to more users and more applications, as users come to realise that FOSS applications are viable replacements. Even a small migration from *Microsoft Office* to *OpenOffice.org* can save a company tens of thousands of dollars in licensing fees.

Successful evangelism requires foresight to minimise the inevitable loss of productivity during the switch. Even when migrating one user to one new



"THE BEST METHOD FOR EVANGELISING WITHIN AN ORGANISATION IS TO SOLVE A REAL PROBLEM."

ABOUT THE AUTHORS

Zak Greant is a technical evangelist, author and programmer whose deep love of free software and open source is turning him into a penguin. He works at eZ publish, where he gets to work on evangelism, licensing and other cool things. Jennifer Zickerman launched her career in IT when she became, by default, sysadmin of a brand-new Novell network. She is senior technical writer at Sxip Identity Corp.

application, it's the evangelist's responsibility to ease the transition as much as possible. A bumpy transition will reflect badly on FOSS (regardless of the fact that switching applications always incurs a productivity drop, whether the switch is to a commercial or a FOSS application). Similarly, a smooth transition will reflect well on FOSS, and make users more open to expanding their FOSS use.

Mega-evangelism

Evangelising and implementing free and open source software within a business is inevitably more complicated than migrating a single user to a single FOSS application. Many users will

usually be affected, interactions between components in the organisation's infrastructure often need to be altered, processes (rather than mere tasks) may need to be changed to support the new application.

Similarly, the risk of implementing FOSS within an organisation is greater. More people need to support (or at least accept) the project in order for it to succeed, the cost of temporary productivity losses is magnified, and the investment in making the switch is greater.

Happily, to balance these risks, the pay-off of evangelism within an organisation is greater – not only in terms of the number of users who are exposed to FOSS technologies, but in the benefits that accrue to the organisation by adopting FOSS components and a FOSS strategy.

Successful FOSS evangelism and adoption within a business largely depends on three strategies: fixing the right problem, gaining the interest and support of people who will be affected by the change, and adhering to established FOSS procedures and protocols.

Advocating FOSS for its own sake wins few converts. The same principle applies whether converting a single user or a thousand users: people – especially people within organisations – are interested in features and cost. Therefore, your best chance for introducing your employer to FOSS is to suggest they use it to solve a real problem that has some of the following characteristics:

- **Familiar** Choose a problem that lies within the scope of your skills and knowledge. Stray too far from what you know and you risk failure.
- **Small** Small problems are easier and faster to solve than big problems. They also help you build the skills and experience required to solve larger problems. A successful FOSS implementation, even for a small project, builds confidence in FOSS and is acknowledged by the people who make choices about FOSS adoption.
- **Low budget** Projects that are being crippled by high software-licensing costs are excellent candidates for rescue. Often an organisation has only a few licences for a commercial product that could, in fact, be used by many users. If a project's bottleneck is access to a commercial application, replacing that application with an equivalent FOSS product is an excellent advocacy project.
- **Low risk** Avoid risky situations, such as modifying core systems, touching late projects or working on politically sensitive issues. The pathology of the project will be inherited by the FOSS implementation solutions among and beyond the users directly involved in the project.
- **Tangible** Look for problems that are well known. There's little benefit to solving a problem that isn't causing anyone significant pain, or, frankly, isn't causing pain that for the people who make choices about FOSS adoption.
- **Failing** Contrary to the advice above, if a project is doomed but could be saved with FOSS, it might be a candidate. Ensure, however, that the failure is acknowledged by the key stakeholders – get it in writing. Otherwise, the failure of the project could be blamed on FOSS, rather than the project's own dysfunction.

Once you have a solid candidate for a FOSS implementation, get other people involved. Depending on the nature of the project and the organisation, the process of buy-in might be formal or informal, done in stages or done with all 'stakeholders' at the same time.

The rule is: no surprises. Everyone affected by the project should know about it. Communication is a fundamental feature of FOSS advocacy. People are much more likely to support a FOSS initiative if they understand its purpose, its implementation time-frame and the way it will affect them.

This doesn't meant that you should blab to everyone about the great FOSS solution you're going to build for Problem X before you've cleared it with the decision-maker responsible for Problem X. It's more likely that you'll put together a proposal (and maybe even a prototype or demo), get the decision-maker's approval to continue, then start discussing and demonstrating the FOSS solution to the people affected by the problem.

If your proposal is not approved, be gracious – and then go looking for a new problem. Graciousness in defeat will increase the likelihood that a future proposal to solve a different problem with FOSS technology will be accepted. Your tenacity will keep FOSS on the organisation's agenda.

Code for coders

Most FOSS development projects share certain principles. These principles have evolved over the course of many years and many development projects, in response to the challenges of building software with a team of distributed volunteers. Some of these principles are explicit, such as a licensing agreement. Others are implicit, and have been adopted to ensure that the community of developers and users functions efficiently, ethically and harmoniously.

As a FOSS advocate, you're an ambassador for these principles, as they're the premise on which FOSS development is predicated. These principles are not limited to free and open-source software development culture, though – they're a set of values and ethics that benefit the members of many kinds of communities. Here are those rules to follow:

- **Be open** Once you've won support for Linux, make sure your project is visible as it evolves, especially when it reaches the point of affecting other systems. Openness is one of the core values of FOSS, for many reasons: it's inclusive rather than exclusive; it encourages others to look at your stuff and offer suggestions and comments; it helps people to prepare for implementation; and it encourages people to get involved.
- **Scratch what itches** Work on real problems. FOSS communities tend to attribute the greatest value to people who work on bugs and enhancements that affect lots of users, rather than building arcane features that may only be of use to the programmer himself. And, most important, if it ain't broke, don't fix it. If it's not a problem, it's not a candidate for a FOSS solution.

> **"NO ONE LIKES FANATICS – THEY TEND TO RUIN OTHERWISE FUN PARTIES."**

LINUX SUPPORT *DOES* EXIST!

In the early days, people avoided FOSS because it lacked the support they were used to from proprietary systems. Now, however, there are thousands of FOSS vendors who provide quality-assured distros and commercial support. Be a good evangelist by letting them know what's out there.

Databases

IBPhoenix (*Firebird*) www.ibphoenix.com
MySQL http://mysql.com
PostgreSQL http://pgsql.com
Sleepycat http://sleepycat.com
SRA http://osb.sra.co.jp

Linux

Mandriva http://mandriva.com
Red Hat http://redhat.com
SUSE www.suse.com
Ubuntu http://ubuntu.com

Programming languages

ActiveState (Perl, Python, Tcl) http://activestate.com
Stonehenge (Perl) www.stonehenge.com
Zend (PHP) http://zend.com

Miscellaneous

Covalent (*Apache*) www.covalent.net
Open Source Development Methods Collabnet http://collab.net
Sendmail www.sendmail.org

■ **Postel's Law** "Be conservative in what you do, be liberal in what you accept from others." Generally, Postel's Law (**http://en.wikipedia.org/wiki/Jon_Postel**) is used to describe applications or components that are fault-tolerant regarding input, but exact and well-formed regarding output. Robust implementation of specifications, most notably on the internet, often follow this law. In the context of FOSS evangelism, this epigram can also express the way that you and your FOSS project should interact with other people and systems.

So, don't blindly advocate FOSS – provide people with a broad perspective on the ramifications of the solutions you propose. And be a responsible representative of FOSS. No belittling other software.

■ **Release early, release often** Don't lock yourself in a broom closet for ten months while you architect the über-solution that will fix all the problems and also make toast.

It's better to break big problems into a series of small problems. Then solve a small problem, show the solution to the people who care, get their input, solve another small problem. Rinse, lather, repeat. This builds your knowledge of the problem space, and tends to engage more people as they witness the gradual creation of the über-solution.

■ **Don't break the build** Never break working systems. It will damage your personal credibility and, by extension, the credibility of FOSS. If you need to overhaul a working system, always try to model and test on a private system first. Then, before altering the live system, make sure the people who use it are aware of the change and the inherent risks. Make sure you can revert to the original system if things go badly wrong, and make sure that any data stored while trying the new system is migrated back to the old system.

We'd like to add one more 'principle': don't be a jerk. FOSS evangelism is both a technical project and an interpersonal project. You'll have to win hearts as well as minds if you are to succeed. Don't discount others' questions and concerns but, conversely, don't let others denigrate your project merely because it is based on FOSS technology. Challenge spurious assumptions, regardless of their source. For example, assertions that all Microsoft software is low-quality are just as incorrect as assertions that all FOSS users are anarchists.

Extreme prejudice

You'll need plenty of energy to bat away the prejudices and myths many people hold about FOSS. Here are the most common misconceptions that you should be ready to fight.

MYTH 1 FOSS is communism

The 'People's Republic of PHP' has yet to issue passports or make a bid to host the Olympics. A diversity of political opinion occurs in FOSS communities, as in every other community. And, as in every other community, the fringes get the most attention. FOSS itself is not political.

MYTH 2 If I use FOSS, I will have to give my software away for free

Untrue. There are many open source licences, covering many uses. None of them requires you to give your software away. One class of licence – strong 'copyleft' licences like the GNU Public License – have strong requirements about sharing your work, but these requirements only come into effect if you distribute your software. If it's used privately or within the bounds of a single legal entity (such as a company), you are not required to give it away.

INSIDE THE TENT

Where to find out how the big vendors are facing up to Linux

HP www.hp.com/linux **and** http://opensource.hp.com
IBM www.ibm.com/linux **and** www.ibm.com/developerworks/opensource
Novell www.novell.com/linux **and** http://forge.novell.com/
Oracle www.oracle.com/linux **and** www.oracle.com/technology/tech/opensource
Sun www.sunsource.net, www.sun.com/software/linux **and** www.opensolaris.org

MYTH 3 FOSS isn't popular

FOSS (such as the *Apache* web server and the PHP and Perl programming languages) and open standards (like TCP/IP and HTML) form the backbone of the internet. These are not marginal technologies. For example, as of April 2005, *Apache* was powering about 69% of all web servers. Internet users rely on FOSS every day in their web travels; most don't even realise it.

MYTH 4 FOSS is insecure

The comparative security of FOSS versus proprietary applications and operating systems is the most hotly contested aspect of FOSS, especially regarding Linux. There's no room here to untangle the myriad studies, arguments, claims, misinformation campaigns and flame wars. Regardless, access to the source allows security problems to be found more easily by any skilled person who invests the effort. Also, the modular design of most FOSS projects makes individual components less risky to patch, as they are more loosely coupled both with other components and with the operating system itself.

MYTH 5 FOSS is just for hobbyists

Software giants like HP, IBM, Oracle and Novell are major supporters of the Linux operating system and have developed broad open-source strategies. Even Sun, with its competing, proprietary Solaris operating system, has adopted a Linux strategy and has started distributing Solaris under a FOSS licence. Apple based the Mac OS X on the FreeBSD operating system and has licensed the core of the OS under a FOSS licence (**http://developer.apple.com/darwin**). See *Inside The Tent* box, above.

FOSS evangelism goes beyond implementing FOSS solutions for computing problems experienced by users and organisations. Remember earlier when we said that FOSS is not a religion? Well, this is the part where it starts to get religious – or at least philosophical.

Possibly the most interesting aspect of FOSS is that it exists at all. Circumstances unique to our current technological era, coupled with the wealth and education enjoyed by people in the rich world, have created new scope for collaborative creation.

This is worthy of discussion and research beyond the practical bounds of FOSS implementation. As an evangelist, try to talk over the pros and cons of FOSS in many different settings. Have conversations with academics, non-technical users, developers who rely on proprietary tool stacks, and your dad. Find out what they think; this will benefit not only your FOSS advocacy, but also your broader understanding of the FOSS phenomenon.

Most of all, relax and enjoy the ride. No one likes fanatics – they tend to ruin otherwise fun parties. FOSS thrives because it is flexible, open and inclusive. Don't become inflexible, closed and exclusive as you work to promote it. LXF