



Jon Hall consented to the publication of this image under the GNU-FDL

Interview with Jon “maddog” Hall, Linux international

1. Can you give us some information about the experiences of transforming the proprietary Software from Microsoft OS and third party vendors to FOSS in some countries which you know?

When we want to move to FOSS, where should we start? From client or server side?

You can actually start moving to FOSS while using Microsoft as your operating system. A lot of FOSS works on top of multiple operating systems. Firefox as a browser, Thunderbird as a mail reader, OpenOffice as an office package, MySQL as a database and there are many others. At OpenCD.org you can find a whole CD of applications that are all FOSS and can be introduced into your current environment.

After that, I usually recommend starting on the server side. Your systems administrators are easier to train and limited in number to your end-users. A lot of the implementations of FOSS can be made invisible to your end users. Using FOSS for firewalls, DNS servers, file-and-print servers (using SAMBA) and database servers (utilizing high-availability components of FOSS) can all be done without your end-users knowing anything except how much more stable the servers seem to be once they were switched to FOSS. Web-based applications are often fueled by an Apache web server on top of a Linux operating system.

You do not necessarily have to switch database engines to use an FOSS operating system. Oracle runs fine on top of the Linux(R) Operating System. In the meantime your end-users are getting used to using FOSS on top of their already existing desktop systems, while your systems administrators are being trained and getting used to solving issues and answering questions on FOSS.

In addition to all of this, there is no real reason why you should not be looking at using FOSS for your high-performance computing needs (Beowulf systems) or your embedded system needs. Your high-performance programmers and users are usually sophisticated, and may already be using FOSS. Embedded system users hardly ever see the operating system, only the application. Yet there are many reasons for using FOSS for embedded systems other than the fact that it is royalty free.

2. Which difficulties would we face when transforming to open-source?

Finally, I hardly ever recommend a “movement” of users from one system to another unless there is a true economic or value reason for having them move. Do not move them just to say that you have gone to FOSS unless you can demonstrate a real benefit for doing so. I call this the “all pain, no gain” port. Even if everything goes right, the end users will resent the interruption and changes to their work. On the other hand, the upcoming movement to Vista (formerly known as Longhorn) will give the perfect opportunity to migrate to FOSS rather than migrate to this new operating system.

I recommend looking hard at FOSS during the planning stages for new projects. Do not list the requirements using product names such as “Microsoft SQL” or “Microsoft Office”. Instead list the requirements as “relational database engine” and “office package that has word processor, spreadsheet and presentation capabilities”. You may then find answers that are less expensive or higher quality for your specific needs.

Think about using projects such as the Linux Terminal Server Project (LTSP) for setting up a server and some thin clients. This makes an easy to install, easy to administer, easy to backup and restore system that can utilize older systems for the clients. Classroom situations, banks, point-of-sale terminals, all could make use of this basic configuration.

Some of the other difficulties that you will experience are things like document formatting incompatibilities. However, you might have these problems even if you stick with a proprietary solution such as Microsoft Office as you go from version to version. You need to specify an Open Data Format for your documents, just in case some company goes out of business and one hundred years from now you no longer can read the documents created with their word processor. You need them to FULLY define their format, even if it means that some competitor of theirs can easily create a competing product to read and write that format. This will protect you from losing the ability to read your

own documents, but it will also allow document preparation systems to compete on ease of use and features rather than market share.

3. Which are the leading countries that migrated or are migrating to FOSS?

I like to use Brazil as my “Shining Star of FOSS”. They started several years ago just using FOSS as a way to save money. But then they also started to realize that it created local jobs instead of sending a lot of money to other countries for software. They also realized that they could tailor the software to meet their cultures' needs, and that the software belonged to them, rather than some company. This was important, since it allowed them to use the software for any purpose that they wanted to use it. No one told them that they could not have multiple people logging into one “personal computer” or that they could not transfer a piece of software from one system to another.

For a long time Brazil had most of its IT work done by private enterprise at their specification. With FOSS, they extended that to work with the FOSS community and create projects faster, with less cost, and with functionality tailored to their needs. I use a lot of examples from Brazil in my talks.

4. Which countries are famous for recommending FOSS in the area of e-Government? Can you give some examples?

A lot of other countries are very visibly “migrating to FOSS”. Malaysia has a national program to migrate. Germany is famous for its central government moving its servers, then its clients to FOSS. The city of Munich is very publicly moving to FOSS, as well as a lot of other cities in Germany.

Most countries using FOSS have not made a “movement” to abandon closed source software, but they have made a policy either to start using FOSS for new projects or to consider FOSS along with proprietary solutions for new projects. Thus, as new needs and new requirements for old systems into retirement or re-write, FOSS software often replaces a proprietary solution.

The best place I know to see examples of this are at:

<http://wiki.go-opensource.org/taskforce/FrontPage>

a Wiki formulated by the “Go Open Source” campaign.

Often it is particular places inside of each government that FOSS starts to be used, not over-all. A good example of an exception to this rule is Malaysia, which has a national plan to move to FOSS.

5. In which field can we apply open-source most effectively nowadays in the world?

E-government, certainly. Education and library sciences seem to have also embraced FOSS, and there are many projects for these groups.

From a commercial side, banking and finance, health care, and medical research (particularly GENOME research), communications (VoIP and embedded systems in mobile phones), retail sales (especially Point-of-Sale systems), Geographical Information Systems work (particularly in locating and mining gas, oil and other resources), motion picture industry (rendering movies such as “Titanic”, “Shrek”, “Lord of the Rings”)...the list goes on.

In reality, many people are using FOSS today and do not realize it. Sendmail has been used to transport email from the beginning, and is a FOSS project. Apache as a web server is FOSS, DNS servers are based on BIND, a FOSS program. A lot of software development engineers use emacs as a text editor, one of the first FOSS projects.

I have been using FOSS in my daily life since 1969.

6. Should we use FOSS in the standard applications which do not have business/security critical relevance and **should not** for the critical applications? Why? Could you please give us some examples of using FOSS for critical applications in banking, aviation, petroleum, space, or defense?

There is nothing inherent about proprietary software that makes it any more secure than FOSS. If this was true, then Microsoft would not be having so many problems with security. On the other hand, the only secure system is one that has constant diligence in applying patches and fixing holes.

I do believe that FOSS has two advantages over proprietary software in both the areas of security and stability of mission critical applications.

First, when a proprietary system goes down, you have a limited number of people who can find the problem and fix it. Only the company that produced the program typically has the source code, and then only a limited number of engineers (perhaps just one) inside the company might have an understanding of that code. If that engineer is on vacation when your application fails, then you may have to wait for that fix. Comparing that to FOSS, where every person has access to the source code, and can hire an independent engineer to fix their problem if the original coder is busy, this allows for less down time, with a shorter mean time to repair.

The second advantage is in testing. Most proprietary code companies put out a “field test” (also called a “Beta”) of their code to a limited set of customers when they are almost ready to release it. Those limited set of customers may or

may not put the software on their systems and really test it. In FOSS, the source code is developed in the open, and lots of interested parties can download the code and try it out, giving constant feedback to the developers as it moves towards release. This model tends to build more stable code.

The Open Development model also allows end users to give direct feedback and to develop new features as “patches” to the existing code, so these new features can be tried out in real-world situations before they are implemented in main-stream code.

7. How to work well in the heterogeneous environment of using both Windows and Linux based applications (or mixed environment of open-source and commercial one)?

As I said earlier in the interview, using FOSS applications on Windows systems is one way of moving into the FOSS environment, and it works very well.

Linux can mount, read and write to FAT, FAT-32, VFAT and NTFS file systems. With Samba, Linux can be either a client or a server for Microsoft systems. Linux also works well with Apple systems, talking both Appletalk and NFS.

Through third-party X Window System servers that run on Windows and through browsers, a person can set up a single “desktop” that allows all applications to be displayed on it, with the applications sharing data through a network-based file system below.

8. It is said about a trend of developing eGovernance applications based on open-source . What's about your opinion on this trend? Is it true or not?, Why should we need to do that?

There is definitely a trend in doing this, and for several reasons.

First of all, computers are no longer a luxury in running a country. No modern country can go back to using paper and pencil for administering their government. Therefore computers, and software, become a “mission critical” aspect of running a country. When software comes from a foreign government, and you have no real way of telling what code is inside of the software that is running the ships, planes, tanks, and missiles of your country, you are at risk. When you may not be able to get the patches or fixes to the programs you need because your country is under embargo from the country that created the software you are using, you are at risk. When some foreign country has a policy of not allowing you to use the same fast computers and advanced software that they are using, you are at risk. How can we have peace in the

world when one country feels constantly at the mercy of others? When every country is strong and secure, we will be much further along in the peace process.

Even with the issue of country security taken out of the picture, there is the issue of longevity. Vietnam has a history going back 500,000 years. The oldest software company on the face of the earth is 40 years old. What happens when the company that Vietnam bought its software from goes out of business, or is merged with another company, and that software is now discontinued? Does Vietnam have to migrate its governmental software just because a company in a foreign nation decides to change it? Recently the world's largest hydroelectric dam (located in Brazil) decided to switch to using FOSS because they realized that their facility had to survive for at least a hundred years, and they needed software that THEY could maintain, if the company that supplied it disappeared.

Balance of trade is another issue. A lot of countries have problems with software piracy, and they have to stop this in order to join the World Trade Organization (WTO). Unfortunately this means that they would have to pay millions of United States Dollars to Microsoft for the continued use of the software they have been using. In some cases this is more than the gross national product (GNP) of their country for a year. FOSS allows these countries to use the software off the net, tailor it to their needs locally (creating local jobs and local expertise) and not have to send money outside their country.

National pride is my last issue. With pirated software there is no real development of a national software economy and infrastructure. People are dependent on software developed by another person, and used against that person's desires. If this piracy was of a solid thing, like a cyclo, it would be called "stealing". How can a society condone this? Even without the moral argument about who owns software, with piracy there is no incentive to learn to program well. FOSS, as a service-based economy, gives the opportunity of a highly trained and skilled job to people who wish to charge for the services of modifying software to people's needs. Your children will belong to a world society of programmers, not as serfs, but standing shoulder to shoulder with them.

9. It is said that all open-source software follow open standards, is this true or not? Does commercial software follow open standards? What about this issue? Could you please introduce some fundamental open standards? Which URLs that can be used as a reference?

Many standards have been created around FOSS projects. POSIX was fashioned around Unix(R), both System V(R) and the Berkeley Software Definition (BSD). The Internet Standards that drive the Internet were created as open Request For Comment (RFCs) by many people interested in the formation of the Internet. Language standards of the IEEE and others allow FOSS projects to create compatible compilers. We strive to follow freely available, openly published standards.

Where standards do not exist, we produce FOSS software that not only helps us to implement the standard, but in a lot of ways helps us document the standard. By having FOSS code available, even proprietary vendors can better understand what was said in a formal, written standard. FOSS is good for standard development and implementation.

Some vendors claim their code is “standard” just because a lot of people use it. The industry calls this “de-facto standards”, but they do not guarantee you that the standard will not change dramatically in the next release of the product. Nor does it give open access to everyone to help determine what the standard will be. Nor does it create an even playing field for other people trying to implement that “standard” and give competition to the owner of the “pseudo-standard”.

Some vendors even take publicly accepted standards and change them in their own products so they are no longer compatible with the accepted standard. Microsoft, as an example, changed both Kerberos and JAVA to make incompatible versions.

For examples of the FOSS movement toward standards, please see the Free Standards Group web pages (www.freestandards.org) which is a non-profit set up to create a definition of a binary interface for all FOSS systems.

10. It is said that one should not use FOSS if they don't want to be at the court of being claimed about copyright. Is this true or not? Software patents exist in the US. Europe voted against just recently. Do you have any comments on that and what's the trend in Asia?

Software patents are evil and obsolete. There may have been a reason for having them when computers cost trillions of dong and programmers were very few. But today you can buy a very powerful computer for a couple of million dong, and most high school and college students know now to program and can solve problems with new ideas.

Patents are not an act of nature or god. They are an act of law which is set to help move society and technology forward. Is it better for government or the

people to share ideas freely when billions of people can contribute ideas, or to create a protective monopoly for the good of a few? I am not saying that people should not have a right to their ideas, but there are other ways of protecting that right than having the law create a horribly broken method called “software patents”. I will point out that the golden age of software idea creation actually happened BEFORE the introduction of software patents.

Software is now everywhere. It is in everything. Thus it is like art, music and mathematics. Imagine trying to paint a painting if someone patented a particular brush stroke, or writing music if someone had patented the idea of the “triplet” or “riffle”. Now try to imagine writing a program with over 10,000 patents applying to things like “use of a mouse” or “window refresh”.

Asia should refuse to honor software patents.

Copyrights are another issue. When someone rights a piece of code, they should have the right to say what happens with it. All FOSS software is copyrighted. Yet, the owner has the right to license it any way they want, and often it is licensed multiple ways, depending on the user's needs. Copyright allows people to re-implement ideas in different code, allowing off-shoots of those ideas.

FOSS code is neither better or worse, neither more secure or less in the area of copyright infringement than proprietary code is. I can say, however, that all of the FOSS programmers that I know would rather die then infringe on another programmer's copyrights. And since all of their code will be open for inspection, this makes the odds of a real FOSS programmer using other's code without permission very unlikely.

11. If we want to create a new project and license it GPL, what are your recommended steps ?

Study other projects first. If there has been other work was done in the area see why it succeeded or failed. Try not to re-invent what others have done.

One thing you should always consider doing is to take the changes you make to other people's software and try to give it back to the original developers. This will keep your code changes in the mainstream, so when a new version comes out, you will not have to make the changes all over again.

12. How to avoid being trapped by using projects that go commercial? e.g “closing an FOSS project” (companies used FOSS for developing products, then close them).

When the developers decide to close a project, the GPL guarantees that the people using the last GPLed version of the code can use that code forever, and

also use it as the basis of their own new project. We call this “forking” the project. If enough people are interested in continuing the project as a completely open, GPL'ed project, they can.

Forking happens for other reasons. Sometimes the developers can not agree on a future direction, and the project divides into two, both of which might be GPLed. Then the community, both through the developers and the users, decide which of the two forks survives and moves forward. Again, this offers competition and choice, and is not necessarily a “bad thing”.

13. Could you predict the future of applying and developing FOSS? How will be the situation of applying FOSS and commercial software?

While I believe that closed source software will be around for a long time, I do think that the concept of FOSS will be the way that software is created in the future.

Proprietary software came about in the early 1980s, and it was mostly developed for a smaller, more homogeneous environment than the world computer economy of today. The issue of proprietary software is that it can not meet the needs of everyone in the world, and the companies that make it can not afford to meet those needs and still be profitable. This removes the ability of the end consumer (even if they are a government) to get the features and bug fixes that affect them in a timely manner.

FOSS, with its availability of the source code, and the freedom to change it, allows the end customer, big or small, to make a business decision as to whether they want to use the software as it is, and wait until the main developer implements what they need, or to move ahead and hire someone to implement those changes when they are needed.

Note:

FOSS: free and open-source software

Jon Hall consented to the publication of this interview under the terms of the Creative Commons License (Attribution-NonCommercial-NoDerivs 2.5)

http://creativecommons.org/licenses/by-nc-sa/2.5/deed.en_GB