
What is Free Software?¹

V Vinay

Author photo
Author intro

¹ This article is based on a talk delivered on the '*Free Software: New Avenues for IT in India*' on 4th December 1998 at NIAS.

² On a suggestion by CS Yogananda

Any article on *free software* starts with a definition of *free*: Free as in free speech not as in free mangoes. Most readers comprehend the meaning of this line quite well but within a few minutes, they revert to the free-mangoes mode. Part of the confusion is because many of the current products of free speech are also free mangoes.

Some of this ambiguity is due to the English language in which '*free*' has more than one meaning. There have been some attempts to rectify this: a newly coined word which is attracting some attention is *open source* software. While this is gaining some currency, the name unfortunately does not truly reflect the spirit of free speech. To fix the concept once and for all, here is an Indian equivalent many of us can easily relate to: *muktha*².

The free software movement as we know it, was founded in 1985 by R M Stallman. The basic tenets of free software are:

- Freedom to study
- Freedom to change
- Freedom to share or distribute
- Source *has* to accompany binaries
- Right to sell free software

We would want to use *free software* for a variety of reasons. First of all, it is generally more reliable than its counterparts. *Linus Law* explains why: Given enough eyeballs all bugs are shallow. As many more people look into the code, it is less likely that the bugs will escape the attention of all of them. In fact, free software, especially GNU software has been found to perform more robustly than their fettered counterparts.

Reliability is not the only issue. Sometimes, we want to customize the software to our liking and not be restricted to what the designer thinks should be our choices. Or perhaps we want to



experiment with the software, because that is the *only* way to learn!

It would also be nice to be able to make changes if required. We can make the changes if we are competent to do so; or find somebody to make the changes for us. Let me illustrate from a personal experience. Recently, for some numerical computations, I was using a free software called *rlab*. Upon running a certain input, I noticed an error in its functioning. Fortunately, since the source was available, I was able to isolate the source of error in a couple of hours, fix it and inform the author of it. The author acknowledged the bug and released a new version of the software sometime later. In the meanwhile, I was able to continue my work unhindered as I had patched my version of the program. This freedom to alter, to change, is what makes free software so different from *fettered software*. I would not want to give up this freedom at any cost!³

³ By the way, not all commercial softwares are fettered softwares. I hope the distinction is clear.

If freedom is dear to our hearts, must it not percolate every aspect of our lives? Must it not be zealously guarded in all spheres? What then would explain our support and surrender to proprietary software? Allow me to sketch a rough analogy: say you are driving from Bangalore to Mysore. Suppose you have a flat tyre along the way, what would you do? You have many options. You may fix it yourself. Or you may get a mechanic from a nearby garage. Or ring up your manufacturer 60 kms away for help! Note that the knowledge of fixing a flat is in the 'public domain' and not with any particular manufacturer. And surely, you will not buy a car if the manufacturer insists that only he can change the flat! Why then do you buy a fettered software under those terms?

There are a variety of reasons for buying such software. (1) Insurance: knowing whom to blame when something goes wrong. (2) My neighbour bought a copy; I should buy one too! (3) Intellectual property rights (IPR): we should all respect it, and how else can we reward the people for their effort. While the first two are the reasons why we buy proprietary software, IPR is used



as a rationale for doing so.

IPR is tricky business and a paragraph about it here is not going to change the way the world will go about it. We might as well save our breath and instead argue *for* free software instead of trying to counter proprietary software. At the very least, this would be a positive approach.

The mechanics of free software derives from science itself! What drives a scientist? Peer recognition. All our papers to journals are reviewed by our peers. And a scientist having many such quality papers becomes a peer. There is no formal structure to this process itself (hopefully, I am a professor because I am a peer and not the other way around) but we know it works and works well. A scientist builds a reputation over the years based on one's work, as opposed to being a celebrity. Reputation allows a scientist to define what is *reputable*; information that a new generation of scientists would need to be aware of, to make its mark! Indeed a consequence of this internal mechanics of science is that we would like to work on questions that gains us peer recognition.

Science was not always like this. We all know how science was practiced some centuries ago and its need for anagrams to make progress! Over the centuries, it has evolved to its present traditions; and who knows where evolution will take it next!

A programmer writes free software and *publishes* the software. Other programmers (peers) recognize its value by acknowledging it as a useful addition and by probably adding a few new ideas to the software. Pretty soon, one has a long list of contributors to the software; this enhances the breadth and depth of the software. Scientists will immediately recognize this script.

Peer recognition is quick to come by when one works on deep problems. A word (a long pause now!) processor is a large program but does not have much depth. So what gets written is not a simple word processor but a complex text composer like T_EX. Again think of scientists who publish papers *without giving*



⁴The practice of such a 'science' can be quite disconcerting!

away much in details to protect 'intellectual property.'⁴ What can a peer judge from such a paper? Indeed, how can a programmer evaluate proprietary software? On its algorithmic merits or on its user interface!

⁵University dropouts, of course, have other ideas and values ...

It should come as no surprise that free software derives much of its strength from academic and research institutes; these are the places that encourage openness and sharing. This is not to say that all free software comes from educational institutions but that these values are nurtured in such environments.⁵

With this as a backdrop, I will now explain the phenomena of free software through the language of *memes* first expounded by Richard Dawkins in his book *The Selfish Gene*. A meme is in Dawkins' words a self replicating information pattern '... leaping from brain to brain' much like a gene that jumps from body to body. Memes are like genes: they propagate, mutate and like to be around for as long as they possibly can.

Until the early 1970's all software was free in all sense of the word free! And then a new meme of proprietary software was born. What compulsions generated this new meme and its reasons for its success will make an interesting study which we will skip here. Instead, we will concentrate on the original meme.

In 1984, Richard Stallman, then a programmer at MIT, started a movement for creating free software. To Stallman, creating free software was about a co-operative society, where neighbours talk to each other, exchange information and the freedom to help a community to build by adding new capabilities. To quote Stallman from an interview, 'The most fundamental way of helping other people is to teach people how to do things better, to tell people things that you know that will enable them to better their lives. For people who use computers, this means sharing the recipes you use on your computer, in other words the programs you run.'

Stallman created or helped create several hundred free software



programs, among them the compiler *gcc* and the powerful editor *emacs*. He also came up with a remarkable license called GPL, a so called *copyleft license* in a world otherwise copyrighted! GPL allows one to share a program with anyone you wish, to mutate it as you desire, to sell if you so feel (and find someone to buy it!) but **forbids you to forbid!** For example, you cannot take a GPL'ed software and convert it into a proprietary software or prevent your neighbour from sharing it with someone else. The value of this clause for the survival of the meme is uncontested: it has virtually guaranteed the free software meme of its immortality!

Stallman's desire was to build a operating system. In the meantime Tannenbaum wrote Minix, a small operating system for teaching and exploratory purposes. Unfortunately, minix was not free software and hence could not mutate! Pretty quickly minix gave way to *Linux* in the early 1990's, a creation of Linus Trovalds, a Finnish student.

Linus Trovalds released his software under the GPL license. This meme spread like wild fire with hundreds of programmers joining the linux movement. By now, linux has an installed base of about ten million machines; which probably means about fifty million users around the world are linux aware.

Elsewhere, the programming language *perl* was created by Larry Wall. By the mid 1990's the Internet was more or less run completely on free software: perl for CGI scripts, Apache as a web server, sendmail for email and bind for domain name service.

In 1997, Eric Raymond wrote an influential paper called the 'The Cathedral and the Bazaar.' The paper primarily distinguished the manner of software development of the 1980's (Stallman) to the 1990's (Trovalds). He also suggested the use of 'open source' as an alternative terminology to free software to win over the industry. The industry now sees the open source model as an evolutionary model for cutting costs while creating quality software. Many important companies such as IBM and Netscape



have now open sourced some of their software.

What does all this mean to India? Foremost is its impact on education. Free software allows teachers and students to look into the software and not just treat it as a mystical black box. Children like to play with things, tear them apart and (if we are lucky!) put things back together. Free software encourages such exploration, allows interaction with other children (without inducing any guilt of being a 'pirate!'), and learn to understand large complex programs. Dexterity in creation and not in usage is crucial if a developing country like India has to create its own niche. Or else, we will merely be followers.

The other major impact of free software is paradoxically on the security of the country. Free software is software that can be trusted as we have the source code.

Eventually, if computing for all has to be realized, even a conservative estimate of about ten lakh servers at Rs one lakh per server, will result in an expenditure of about ten thousand crores rupees. Using free software will spare the country of a huge financial burden, as in this case, free software is luckily also free mangoes!

Suggested Reading

- [1] The Free Software Foundation's official page: <http://www.fsf.org/>
- [2] In 1986, Richard Stallman received an Ph.D from the Royal Institute of Technology, Stockholm. This is an URL to his acceptance speech. An absolute must read. <http://www.gnu.org/philosophy/stallman-kth.html>.
- [3] Richard Dawkins, '*The Selfish Gene*', Oxford University Press 1976.
- [4] E S Raymond's influential article '*The Cathedral and the Bazaar*': <http://www.earthspace.net/~esr/writings/cathedral-bazaar/>

Address for Correspondence
V Vinay
Department of Computer
Science
Indian Institute of Science
Bangalore 560 012

