Gnowgi

is Nagarjuna's press

Copyleft Society

This essay contains issues that are usually part of my speech introducing the free software philosophy. This is transcribed and edited from a few of such talks.

The main purpose of this communication is to orient you to free software and its philosophy, without going into the technical and economic reasons. To begin with I will introduce the special nature of digital resources, and why in this new digital society and what we need to know about it. Later, I will begin by explaining the term 'free software' and its relation to the culture of copyleft, followed by the history of free software movement, and how it is different from the open source software. I will then clear the doubts people often have regarding the possibility of a business, and the motivation of the people who come forward to contribute to free software. Then I will turn my attention to the core issue of why software and knowledge should be free, and how dangerous proprietary software is to human culture. We will also see how proprietary software companies are working with Government and educational institutions to take hold of the power, and how we can fight this evil nexus. Free software is an eminent choice for education, and is an ethical and political choice to exercise.

Introduction

Digitization of cultural resources can facilitate the conduct of education, or for that matter any domains where ICT (information and communication technology) is being used or recommended, if and only if it is conducted in the lines of free software movement, else it is going to be against social and democratic values. I will elaborate my argument below.

The argument can be best understood if we focus on what happens when we digitize any document, whether text or other kinds of media. Digitization uses a computing model to write (encode) the data, and when we try to retrieve the data, the computer reads (decodes) it for us in a human readable form. Normally, we expect that the computing model used for encoding and decoding is part of computer science, and so we rely on it.

However, since code is by nature arbitrary, each company can invent (mind you an arbitrary invention is not necessarily an innovation) its own model of digitization and provides a computing service to its customers. The arbitrary computing model they use is protected under the various forms of IPR, and the current Governments not only respect this but also protect and promote the interests of the companies.

Free Software Movement (FSM) identifies this as the root cause of betrayal that happens in the digital society in various forms. The computing model used must be published, just as any scientific or technological models are published for use by the society. More important than providing access, by publication, is the freedom to use the computing model by other agencies. In addition to this, FSM also seeks the freedom to modify the model, as well as the freedom to republish the model either without any restrictions or with the restriction that other users cannot transform them into private property.

Considering that computer science is a strange mix of deep theory as well as sophisticated technology, it is very vital for any society to use this transparently. Else, we will let some agencies become monopolies. This is a serious danger to digital society because, the data that is digitized belongs to you and me, and not to the company. But, in reality today, our data has been handed over to the proprietary companies, since they alone have the license to decode our own documents. This will create a possibility for computer crime, which is happening all around our eyes. Why should we let this happen?

The only way of stopping this is to follow the guidelines provided by FSM to correct this serious defect in the current digital society.

All cultural resources are at stake

All of us know well that the process of education is an important agency of cultural transmission, whether formal or informal mode of education. In traditional (non-digital) society, this happened by using natural languages, where encoding and decoding knowledge (syntax and semantics) gets stored only in the "way of life". The language games we play naturally stores this knowledge, and therefore we can use these languages in the process of education. For formal languages (used heavily in scientific and mathematical models), where in the syntax and semantics is artificially declared, the coding and decoding rules are publicly archived, and accessible as cultural resources to all the people.

What is so special about computer languages that we cannot continue to practice as we do in the case of natural languages and formal languages? This is the serious question FSM raises, and asks all the policy makers and social engineers to take immediate notice of.

We can learn a lesson from the cave writings and art. Our ancestors inscribed them on a hard stone with the hope that their inscriptions are passed onto us. However, they forgot to add in their inscriptions how to decode them. Since that code is not used, we lost them completely, leaving all of us to decipher what they meant. Though the art of deciphering is an interesting engagement, we should not let that happen to our digital documents that were inscribed only a while ago. Most of us think that, by keeping the CDs, hard disks, and taking backups of files in a number of locations, we are safe. This is a myth. Because, this digital code is nothing but the caveman's stone. For most of us are taking backups of only the stone (code) and not the rules and the means of interpreting that code. The rules and means of interpreting the code that contained in our own writings are already appropriated by several companies. We hold onto our precious 'stones' thinking that they belong to us. What is the use of holding onto stone engravings if we have no means of reading them?

If we understand how digital reading and writing takes place, and if we understand how users are prevented from reading, and also understand how the companies encourage us to write and create more and more cultural resources using the privately appropriated means, what we are doing is nothing but pouring our cultural resources into their kitty, which is a black hole for us, and a property for them. This is how the current digital business is working. We are all made slaves of this new so called knowledge society. I recommend, by the way, not to use the term, 'knowledge society', for human society has always been a knowledge society. We should on the other hand use the more appropriate term 'digital society', for that is what it is. What we therefore need is a digital commission and not a knowledge commission.

The danger is that every cultural resource (text, audio, video, music, dance, drama, educational tools, governing policies, nothing is spared) is getting digitized, in the name of efficiency, speed, modernity, and what not. If the society cannot claim them back, if we do not take precautions, and follow the model of FSM, the human society is in great danger of loosing our precious memories.

All the hype that is created in the name of emerging, so called, knowledge society is a way of promoting this process, where the commons loose all the freedom and thus also the power.

What are the implications of this picture for the use of ICT in education?

It helps us to define what is a free (mukta) digital learning resource (digitized culture) and how to protect it from private appropriation. Having made the point clear in the above section, that the use of a digital resource lies in the ability to decode the 'content' (code), and not in having access or possession to it, it should be clear why we must not separate the two. Interpretation (decoding) is performed by the software or the embedded software/logic in the hardware. If the model of decoding is not public knowledge, and even if there exists 'open' access to content, our lives are continued to be appropriated by the private agencies. That is why open access and open content is not enough, we need mukta knowledge.

A public cultural resource, which is otherwise eminently imitatable, can be commodified (for want of a better word) by technically separating the code from the decoding mechanism. Best example is a digitized song, or a CBT (computer based tutorial) material available to us in the form of e.g. a CD. If you do not also have a particular operating system, these resources cannot be decoded. The company that makes the CBTs, or the songs have private agreements to twist the arms of the customers, and often they have several patents and other agencies must pay license fee to decode the songs and CBTs for us (At the end, we, as customers, pay for all this.). Together, the OS and software makers and the content creators are let loose, with support from Governments, to loot the people. The current draft that is being circulated makes a special room for this in the name of PPP (public private partnership). PPP is not bad per se, but what service PPP can provide and what not, and the rules for this must be clearly defined in the policy. We must warn the Govt, and ask them to take immediate corrective steps.

In the case of mukta software, the scenario is different. The songs and the CBTs are encoded in a public model (for they use free and open standards), and the software that decodes and encodes is also made available always with source code. Disengaging mukta software with the source code is legally not permitted, for mukta software is protected by copyleft (GNU GPL or other such mukta licenses). Thus, by using copyleft model of legislation (policy) we can ensure that reading and writing are never disengaged from one another. This is how we can grant the freedom to read and write. I consider this right more important than the right to information, for what is the use of access to information without an ability to read.

If we do not take care of this, digitization will have a devastating effect on society. We will be dumping all our cultural resources that carry our memories and will be permanently lost.

Considering that education is a part of the process of facilitating the cultural transmission of past memories (mimetics), we must not allow this digital devastation to happen. Unless, of course, we pledge to use exclusively mukta software.

What is this thing called free software?

The first thing we need to know about the term “free software” is that its meaning does not arise from the combination of the terms “free” and “software”. The meaning of this term arises from the definition, and not from the terms it contains. The term “free software” is defined by the Free Software Foundation (gnu.org) as that software which gives the user the freedom

1. to use it for any purpose,
2. to know how it works,
3. to improve it by modifying, and
4. to share or propagate or distribute the modified code to others.

Any software that meets these four criteria can be called free software (henceforth called FS). We must notice that there is no mention of price of software in its definition. This means that there exists a possibility to pay or charge for FS. Since FS is intended to give the users the freedoms mentioned above, it is better called as “freedom software”. In Indian languages there are more options: we may call it “swatantra software” (a preferred term in southern and western India), or call it “ajaadi” software (a preferred term in north-eastern India), or else call it “mukta software” (a preferred term in northern India). The last option is nice, since we can have pun with the word to say: we are talking about mukta, and not mufta (gratis) software. Let us therefore bear in mind that free software is not about price, but it is about freedom. Richard Stallman, who founded FSF and the GNU project, says aptly: “free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free [cup of tea].

Copyleft is an exemplary hack

To understand how FS works, it is necessary to see how copyright can be used for creative works we write, including software. All of us are aware of the copyright law, and how most people use it. For example, a book published by MIT press, titled “Philosophy of Computing Information” has this on the copyrightpage: ©2004 by Blackwell Publishing Ltd All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs, and Patents Act 1988, without prior permission of the publisher.

Many of us have seen such statements in most books. The copyright law says, the author or the copyright holder can specify the conditions of using the resource. But, as the history of its usage suggests, almost everyone in the world used it to restrict the way the resource can be or cannot be used by others. Only one person thought differently about this copyright act. He is Richard M. Stallman (popularly known as RMS). He began to invent different ways of writing a copyright statement. For example the book, a must read, that contains selected essays of RMS: ©2002 Free Software Foundation Permission is granted to make and distribute verbatim copies of this book provided the copyright notice and this permission notice are preserved on all copies.

This is a creative use of the copyright. By imposing the condition, “provided the copyright notice and this permission notice are preserved on all copies", the author can ensure that only unrestricted copying of the work can take place. Else someone can reimpose restrictions. This line of thinking is the beginning of a new era of publishing not only written materials but also software. Though the above copyright statement shows 2002 as the year, RMS invented this art around 1983. Such a creative act of exploring a new possibility when other people don't see any, is called a hack. Around that time it is popular among the computer scientists to use the term 'hacking' for this kind of use. From that time, whenever someone finds an intelligent way of getting around a problem, it is called hacking. Popular usage by the media of the same term mostly means unauthorized intrusion by somebody into your computer, often causing damage or stealing information etc. For this kind of evil act, free software community has a different term, it is called 'cracking'. When we say, FS like GNU or BSD or Linux is made by hackers, we meant the former creative people, not the latter criminals. Cracking is a criminal act, but not hacking. Please therefore don't popularize the term hacking for the negative connotation, use cracking for this purpose instead.

Thus the greatness of RMS is not that he is one of the greatest programmers, though he is indeed one. History will remember him for the most important invention, nowadays aptly called, copyleft. A copyleft is an intelligent use of copyright for giving and protecting freedom of using a cultural resource, than imposing restrictions. RMS invented GPL (General Public Innocence) for this purpose. It is mostly used for software. GPL is a long carefully crafted document, a must read to understand the spirit of free software movement. You can find it in any copy of such software or better read it from http://www.gnu.org/copyleft/gpl.html, among other interesting essays about free software movement.

How old is free software?

I began using FS around 1989 at Indian Institute of Technology, Kanpur. It is an interesting incident that introduced me to FS and its philosophy. I joined to do a PHD at IIT Kanpur in 1988 August. In my second semester I wanted to type my term paper for the graduate course and wanted to nicely format and submit a printed copy. The computer lab at IITK at that time got four new HP mainframe computers with hundreds of terminals spread out in spacious halls in the computer centre. After figuring out which tools to use and how to use them, I quickly became a regular user. On one of the late night sessions while I was jotting some notes, a strange thing happened. I don't know what I did to the keyboard, suddenly on the screen I saw some strange screen titled "GNU Manifesto". It was certainly not the document I ever created. I got puzzled. How did it appear on the screen? Curious, as any one will be in that kind of situation, I began to read. Strange! What is this document doing in a computer center talking about a way of life, sharing kitchen recipes, code and knowledge. How did it get into my terminal? Later I realized that I accidentally pressed a help key on the GNU Emacs editor, which I was using to type my notes, that shows the full text that tells about the GNU project. Till that time I didn't know that I was using free software and it was developed to fight some evil tendencies in the society. As a student of philosophy and science, and an activist of a sort, I took more interest. Then I figured out that the Emacs source directory on the computer actually contained lots of things to read that introduces us to free software philosophy.

What is interesting in this incident is not the accidental hit on the keys that diverted my attention to FS. But, I was puzzled by the fact that no one in the computer lab, and not many in the computer science department there knew about the philosophy of the software they were all using. Not even those who introduced me to Emacs. The Center purchased the four mighty mainframes from HP, and most students thought the stuff there was proprietary software contributed by HP. After some exploration I figured out that most of the tools that we used were made by the GNU project. For example, the bash shell, gzip, tar, gcc, and the libraries etc. Of course not to forget the Emacs editor. Several senior engineering students who oriented me to the technology taught me several tricks of using Emacs. One of the tricks was to add a line to execute Emacs at the last line of the .bashrc script, which invoked Emacs as soon as I logged into the terminal. For over five years Emacs was my desktop, it gave me everything I needed, a programming environment, an editor to type at the speed of thought, spell checker, structured document with LaTeX, file manager, email client, news reader etc. and last but not the least a philosophy of FS. It was then I realized that in GNU Emacs, editing a file is an option. For most others in that high tech institute, the software was a mere practical utility, they didn't care much about the philosophy. One of those who introduced me to several tricks within Emacs is now working for the largest proprietary software.

This episode informs us at least two points. One is, lot of people use free software without realizing that they are doing so, and even if they knew, they used it for practical purposes without realizing the significance of the ideology. Second, GNU software is already part of the mainstream Unix systems at least around 1988.

Similarly, most people who are using the revolutionary operating system, GNU/Linux, often and improperly called only as Linux, do not know the philosophy of free software either. They use it because it is becoming a trend. They use it because it provides some technical advantages. What is missing is that this software is powered by the copyleft movement founded by FSF. Open source movement did not want to propagate the ideology of freedom along with the technology. The result is a misconception that Linux is an OS, while it is just the name of the kernel. Also the impression that FS is recent, since Linux or OSS is recent. Linus Torvalds released the first version of the kernel in 1991. At that time, while GNU hackers are still struggling with the Hurd kernel (the original GNU kernel) he came out with a working Linux kernel. In order to turn this kernel into an OS, he needed a system of tools, compilers, libraries, shell, and other file utilities etc. GNU had all of them, except the kernel. Both of them are released as free software, and so each of them embraced what they lacked. The result is the GNU + Linux or GNU/Linux operating system. This, needless to say, became a revolution in the software world.

The story of GNU/Linux is the story of a revolution that is still happening in front of our eyes. Very few revolutions can be witnessed this way. I do not want to go on listing the success stories and tell you that IITs use it, TIFR uses it, Google uses it, NASA uses it, it works on the space ships and satellites, it has been deployed in weather calculations and gene pool analysis etc. I almost started listing the success stories. But this listing is not what I intend to do here. Since these stories can be heard from other sources, I will concentrate on the arguments that take home the point that we should exclusively use FS for all our needs, and the use of proprietary software is nothing short of propagating an evil.

Free software and open source software

Most people think that 'open source' is clear in meaning while 'free software' is confusing. This is a misconception propagated by the OSS advocates. The confusion is due to imposing a specific meaning from other uses of the term 'free', without realizing that 'free software' is a well defined term — i.e., any software that has the above mentioned four freedoms is a free software. This definition helps us to arrive at an unambiguous class of software that gives users the four freedoms. The philosophy guiding the free software movement (FSM) is an exemplar par excellence of clear practical thinking.

Another way to make the issue clearer is: take the terms 'energy', 'work', 'force' as used in Physics.

They all have a meaning in folklore. If we think they have the same meaning in science too as in folklore just because they are spelled similar, we are wrong. The terms, when used in Physics are well defined technical terms stated by operational criteria. Similarly, 'free software' is re-defined concept though the two terms are taken from folklore. Just as scientists have accustomed to the re-defined concepts of 'work', 'force' etc. we may accept the term 'free software'.

The term 'open source' on the other hand is defined by the set of licenses that are approved by OSF (open source foundation) from time to time as open source. A few of the licenses that are declared OSS (open source software) are not FS licenses. Therefore there is a slight difference in the reference too. Due to these reasons, OSS and FS are not same concepts. The term 'open source' is invented to make FS business friendly with the assumption that the ideology of freedom is not good for business. OSS advocates talk mostly about an efficient development methodology. Open source movement and free software movement are therefore not same movements. Time will prove that freedom is good for everyone interested in preserving human values, including ethical business. I learned in my graduate courses in Philosophy about 19 years ago the difference between a term called 'evening star', and another called 'morning star'. They both stand for the same thing, namely the planet Venus. Since this planet appears first to the naked eye, and is very bright that can be seen even before the sun sets completely or about the time sun rises completely, it is called by these very descriptive names. The terms have different meanings, but the same reference.

Similarly, the terms 'free software' and 'open source' are very different in their meaning, but they refer to more or less the same set of software with negligible exceptions. The meaning of being open is implicit in freedom, for freedom is not possible without being open, but being open does not necessarily imply freedom. Thus, semantically FS is inclusive of openness.

Some people use the term 'freeware', assuming that FS is similar to it. But, freeware does not give any of the freedoms, but it is only free of charge. Another term 'shareware' is also used, but this is another name for proprietary software that is meant to be given free of charge for a trial period.

Some others misrepresent FS as 'public domain' software, without realizing that public domain software does not state clearly the conditions of usage, usually they have none. Public domain software is FS, but not vice versa. It is therefore very important to realize that FS is different and novel from others. The only antecedent of FS philosophy in human history is good human values that nurtured our societies in the past and preserved the culture. FSM though is rooted in virtuous cultures of human kind, it is only now an overt attempt to transform the knowledge dynamics of society with a clearly stated manifesto got initiated. Do read the GNU manifesto from http://www.gnu.org/gnu/manifesto.html, to see that this is more an ethical and political movement than a technical movement. It is incidental that FS is also technically superior and economical.

Is it legal to sell free software?

There is no mention of anything about the price of FS in its definition. Therefore it is very clear that the act of selling does not make it non-free software. Still, people ask us this question, if software is free, how to make money? When they ask this question they completely forgot about the definition. In any case, let me explain one of the ways of making money.

Let us say I have a CD full of FS, containing say the gnusense GNU/Linux distribution. This CD contains a complete OS plus a number of useful applications. What I can do is to make copies of this CD, and sell each copy for say Rs. 50/- (approximately for a dollar). I may have spent about Rs.15/- on the cost of a blank CD and cover, and my time to burn the CD. I may have also spent a lot of time in downloading it, and learning about it etc. I get a profit of about Rs.30/-. I may also price the CD at 500/-, 1000/- 10,000/-. For each customer, the software I give is same. But for the one who payed me 500/- I may give him a printed manual, and tell him, if he has any problem, he can send an email for about six months. For the customer who payed 1000/-, I may visit their place and do the installation, and provide some useful tips. I may also provide the printed documentation, and further promise him that for about an year he can take help from me through email. For the customer who payed me 10,000/-, I may do everything that I did for the 1000/-, but give him an additional guaranty that I will attend at the installation site whenever there is a problem for two years. In all these cases, since all of them got the same software, the money that they paid for is not for the software at all, but only for the service. This way an engineer who knows the system very well can earn livelihood by never selling software, but only his time.

In this kind of a scenario, people will have a lot of motivation to learn FS. A knowledgeable engineer will earn more, since he can provide more sophisticated service. But poorer customers on the other hand can begin to learn as much as possible, since that will save them money, though at the cost of their time. But poor people usually have time to spend, but no money! On the whole FS and its model creates a good atmosphere for learning, and it is win win situation for everyone. Therefore only FS can create a good ecosystem.

What motivates FS hackers?

It is very often asked by the community: if money is not the motivation for developing FS, what other factors motivate them? Why do they develop such powerful software and give it away? Though it is not true that FS hackers cannot make money, it is interesting to look at other motivations.

Usually I take a small quiz in my advocacy talks on FS. Let me repeat the quiz, for it informs us a very important point about FS. The quiz consists of three sets of questions.

The first set of questions goes like this: 1. Who discovered the theory of relativity? 2. Who discovered the structure of DNA? I am yet to enter into a hall that never gave me the answer. Even when I spoke to outside academic communities, such as artists, journalists, trade unions, or women's organizations, I always found that someone in the crowd could give me correct response. Even if I was speaking to an ignorant community, they could note down my question and search for an answer in a library, or ask a friend, or as is more often among the networked people now – look up in an Internet search engine. The answers to these questions are part of human history. The culture remembers them, since they have become part of folklore and culture. Even though most people do not understand either the theory of relativity or the structure of DNA, the society made the inventors their heroes, and they were immortalized for eternity. If the questions were about the authors of "The Time Machine", or "Jurassic Park" or any such famous books, we will get similar responses.

Now let us take the second set of questions: 1. Who are the author of your favorite word processor? 2. Who is the author of the Win2K kernel? When I asked these set of questions, till date there was no reply. The only reply was 100% silence. At times when I was doing a workshop to a small group of teachers or students, who had connectivity to Internet, I requested them to search for the answers instantly, and still could not find proper answers. They do teach and learn about both these technologies as part of the curriculum, the people who contributed to these technologies did not form part of the story. Those heroes were swallowed by the proprietary companies, they sucked them into a black hole. Those heroes are neither part of the folklore nor history. Even if there were a team of contributor's, all of them deserve to be immortalized. What were often left in this part of history are only "windows" and "gates"! I say this because, when I ask them "Who is the owner of Microsoft?"

almost everyone replied. Here people who invented creative tools did not matter, people who own them did.

Then I go to ask the third set of questions: 1. Who is the author of GCC compiler? 2. Who is the author of GNU Emacs? 3. Who is the author of TeX or LaTeX? Who is the author of Perl or Python? Who is the author of Linux kernel? Even though most of the time I was not addressing to a community that is rooted in free software culture, I was never disappointed. Even if no one could find an answer instantly, if I let them use Internet or visit a library or ask friends, they will get back with correct answers. It will not take more than a moment to get answers to these questions.

Both the second and third set of questions though were taken from software scene, the situation is so different. Why? The answer is obviously because of the cultural practices of science, literature and free software, differ radically from the culture of proprietary software practices. On the one hand, science, literature and free software create history, the proprietary software culture wipes out the authors of software from the story, but does record about the owners of their software.

You may repeat this kind of quiz at any part of the world, you will confirm my results. Just as scientists publish the results of their research in Journals or books, free software authors publish their software usually on the modern publishing media, Internet. So, instead of answering the question directly, what motivates FS hackers, I gave an indirect answer: the motivation of FS hackers is similar to the authors of literature, art, science or mathematics. In other words, they are practicing what is practiced by human culture thus far. FS hackers are immortalized for their contributions, while the proprietary software 'coolies' are eaten up or used by others for other motivations. Please do think to see the connections yourself, you will understand how stark is the contrast.

However, this culture of the proprietary companies is not new. To know this, you can take another set of quiz: Who constructed Taj Mahal? Obviously the owner of this historical monument, and neither the architects nor the workers, who were also eaten up by the black hole. We do not have much time, so I have to leave you here with such indications. Storm your brain with more such questions to uncover the truth.

Why all knowledge should be free?

Let us take a blank CD and take its mass on a balance. Now, having recorded its mass, let us copy about 700MB of software or any digital documents into it. Record its mass after it is full. What is the difference of the CD before and after it was full? Zero. How can a CD that contained 700 megabytes of information did not increase in any weight? Is software a kind of levitating gas like phlogiston? (Phlogiston was thought to be the reason why metal oxides lost weight when heated. This theory was later questioned by Lavoisier, arguing against Priestly, leading to the discovery of oxygen, and eventually the chemical revolution.)

If you have an apple and I have an apple and we exchange apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas. George Bernard Shaw (1856-1950)

Yes, knowledge and code are not conserved like matter and energy. Therefore if you have one instance, you can make several copies of them. As technology advanced, the effort involved in making such a copy eventually became so negligible. The cost is negligible today because communication and digital technologies have become highly efficient in producing verbatim copies and the media is inexpensive. The efficiency of copying has become so good that we cannot tell which is a copy of which.

From now on, when we are asked what is this world made of, we must say at least three things: matter, energy and code (assuming space and time as given). Of these matter and energy are conserved, i.e. neither of them can be created nor destroyed. Code, is a different 'matter'. If a substance is conserved, it follows that we cannot make copies of them. Copyable things like code (including all symbolic forms like music, dance, gestures, languages etc.) are not conserved. Without depleting the original, we can make more of them.

Since knowledge is impossible without code, whatever was said about code applies to knowledge too. By Nature, or by God (if you are a believer), knowledge is born copyable.

Based on what we now know about the Nature around us, if we take away code, there will be no life on earth. We know that reproduction is a defining character of life, and genetic code (the language needed for inheritance) is necessary for reproduction. Thus, code is not only necessary for human existence, it is actually the very essence of life.

Why are we digressing and talking about the world around, matter, energy, life and all? What is its relation to software or to free software? Well, the relation is where ever there is code, there is also information processing. Though information processing happens in Nature too, it is not controlled by any political, or business corporations. It happens seamlessly there. However, in human life, information processing is subjected to and controlled by a very complex socio-political process. Unless we understand it clearly, we cannot participate in the game with confidence. Understanding the dynamics of software creation, management, propagation, modification and the factors that control it, are important in order to understand the relevance of free software movement. The knowledge that we create, if stored only in our mind, perishes with us. But in human beings, and in a very few of the other higher animals, mechanisms evolved that pass their knowledge to the children through what is often called behavioral or cultural inheritance. Older generation of beings engage in teaching the younger generations the art of living. This includes to a large extent, how to 'read' the world around. The better we can understand it, the better is the survival chance. One of the essential differences between humans and animals is that we developed several layers of complexity into this process of cultural inheritance, so much so that we have institutionalized the process in the form of schools. Though a lot of knowledge automatically gets through from a generation to another by merely learning a language, special skills are learned through special institutions. In all these cases, what we learn is mostly, how to interpret different forms of knowledge, and how to create artifacts that sustain this process. Thus we not only learn what was gained by the older generations, we create more by standing on the history. Recalling these facts from our study of civilization is necessary to understand what is happening now in the name of proprietary software, knowledge industry etc. We evolved into humans and reached this far due to transmission of knowledge from one generation to another. Human languages are responsible for harboring and transmitting the knowledge through social and cultural dynamics.

Creating, modifying, managing, transmitting, and copying of code or ideas take human effort. That is why, when we want to read the ideas in a book, we pay for it. This payment is essential to ensure that the human effort that created it, printed it, transmitted it are compensated for their service. Though, in principle, this appears easy, human effort is not easy to measure. This is the core contribution of Karl Marx who explained how economic inequalities get created by misjudging the measure of human effort that goes in creating and maintaining the creations.

Having understood that code is not conserved the way matter and energy are, and also that knowledge in the form of code is an essential part of human cultural life, several seers in the past invited us to participate in this very noble process and warned us to live a life of freedom and dignity. Having realized that knowledge is the essence of human life, Rabindranath Tagore – among the tallest Indians who had the vision of free society and free knowledge – wrote this poem:

```
Where the mind is without fear and
the head is held high
Where knowledge is free
Where the world
has not been broken up into fragments
By narrow domestic walls
Where words come out
from the depth of truth
Where tireless striving stretches its arms towards perfection
Where the clear stream of reason
has not lost its way
Into the dreary desert sand of dead habit
Where the mind is lead forward by thee
Into ever-widening thought and action
Into that heaven of freedom, my Father,
let my country awake

Rabindranath Tagore from Gitanjali.
```

Though he was not aware of what is happening in today’s modern world in the name of IT revolution, this was a visionary call to protect our culture, particularly our freedom. Let us see how proprietary software culture is working against the vision of Tagore.

Proprietary software is a black hole of knowledge universe.

When programmers write instructions, they do so in one of the programming languages. These instructions can be read only by trained programmers or by the compilers or interpreters. Only they know how to parse them and decode the meaning of the instructions. However, a compiler is not capable of carrying out the instruction, because it is the processor of the computer that carries out the instruction, it merely acts as a barter in between the programmer and the computer. Since, computer does not understand the programming language, the compiler decodes our instructions, and then re-encodes (re-writes) in a language the computer can decode (understand). Such a rewritten code is called compiled software, which is written exclusively for the machines, therefore it is often called the machine language. This code is humanly impossible to decode, though in principle possible. Since there are several kinds of processors made by different vendors, the compilers have to create different versions of software suitable for each processor. Thus, a program made for an x86 processor is not suitable for a PPC processor.

When software vendors distributes their software, they distribute it for a particular machine, and the code that is distributed is not the code the programmer made for the company, but a compiled version. Added to this complexity is that the program cannot be directly passed onto the processor without an operating system (OS). An operating system is another mediator that helps in passing the instructions from the program to the hardware, and vice versa. It is therefore also necessary to keep in mind for which OS are the programs compiled, apart from which processor they were compiled. Some times, it is possible to use the same byte code on all the operating systems, provided that there exists a separately compiled interpreter for each processor and OS. Language like Java and Python, for example, work this way. This makes the code inter-operable. Now that we briefly looked at how our instructions before reaching a processor get re-written into a series of ‘languages’, we can now see how to convert software into a proprietary software. If any one wants to take control of the software (code) that is made, a proprietary software company can do several things. One way is to provide the source code of a program to everybody, but restrict the interpreter (say a compiler of that language) to only those who pay license fee. It is also possible to embed the interpreter in a hardware, and who ever has the hardware can make use of the software. Other way is to provide neither the source code nor the compiler, but only the result of a compiled software, and restrict the operating system to only those who pay license fee for it.

It is possible also to restrict the use at all these multiple stages — the compiler, the operating system, and the access to the compiled code. The possibility of restriction does not end here. It is possible to create a special hardware, which may also contain another layer of interpreter, and lock the hardware to a single user who enters into a license agreement with the manufacturer. It is possible to invent more and more such stages of control to restrict the use of a software.

But, in all these stages, what is kept under control is mostly the interpreter and sometimes the code. Since, code per se is of no value, the process that makes it valuable is the interpreter, which de-codes the meaning contained in it. By making the interpreter a scarce commodity, it is therefore possible to enhance its value. Since, even interpreters are codified instructions to the computer, one copy of an interpreter can be copied to make several copies. That is why, a proprietary software vendor is looking at technical innovations that prohibit copying, or find other ways of prohibiting. One common way is to de-couple the interpretation process into two or more, and embed a part of that into hardware. This way an interpreter can be made a scarce commodity, making it available to those who can afford to pay license for it or buy the hardware and software together.

Another very important way of controlling is to write user's creations (such as a digitized text, audio, video etc.) in a specific language that can be interpreted only by the system that created it. This is the most popular way of enhancing the value of a software, again by restricting the user to use only one kind of application. MS's 'doc' format is a good example of this.

The companies that indulge in this kind of practice do provide a justification, which is to collect fee for the interpretation, claiming the ownership of the interpreter. This is the reason for calling the money that users pay as a license fee, and not the price of the software. This is a tact to say that this is a service oriented business. Mind you, they alert us in the fine print that a very few of us read: the customer is not the owner of the software, you are merely given the license to use it for a purpose.

Currently the human effort on the operations involving code are slowly getting taken care by the artificial agents (computers). Owners of the computers, instead of asking lesser compensation, are demanding more. Demanding more compensation is not justified since, the precious human effort has come down substantially. As a result, in place of substantial decrease in the compensation of human time, they sought increased fee for artificial agents. Since the new software technology got a fashionable image that it will be creating more value, people began paying the fee as per the demands of the software industry. This, to my understanding, increased the wealth of software industry by several folds.

On the other hand, hardware is also getting embedded with increasing amount of software. Increasingly, even hardware is entering into a licensing regime. Software and hardware industry are together creating more and more artificial agents. The main problem with this model is, society began to pay for the services of the artificial agents. The manufacturers of these agents are pocketing the money in the name of the service time of these gadgets. In this new economy, it is not the goods and service time of human agents that is on sale, but the service time of the gadgets. Do the manufacturers of the devices deserve to extract the compensation? Yes. But only if they do not insert unnecessary locks, that is if they do not prevent free dissemination of cultural resources. Technical innovation should go towards a way of finding out how to preserve cultural resources for a long time, rather than decrease their life time. What is the problem with this business model? I think, the problem lies in charging for the service of an artificial and copyable agent (interpreter). As long as the interpreters were human beings, we sought to buy their time when we needed an expert. Currently, most of human expertise is getting re-written as programmed instructions, and they are interpreted by the artificial agents. Scarcity of artificial agents is controlled artificially, so that their demand increases.

It is important to realize that there are two kinds of artificial agents: the hardware and the software. The hardware is a substantial thing, fabricated in a generic way to carry out programmed instructions. It is not possible to make copies of hardware without spending lots of matter and energy. But, the software set of instructions on the other hand are copyable with least effort and high fidelity. Writing programs is a creative act, just as writing a formula to calculate in mathematics. The compensation should go to the author, and not to the agency that copies the program. Often programs written by several authors is collected and compiled to produce a re-written form of the program, which the author too lost the freedom to interpret. This is the right the author of the program lost. The only way to regain is to keep the entire compilation process accessible. By becoming a custodian of the latter stages of converting the program into a machine code, and its interpreter, proprietary software industry invented a technical method of taking away the right to know. This is not required for making the technology work, but only required for the business interests. As we saw in the previous section, code is eminently and naturally copyable. Copyability is code's essence. When people indulge in such a natural process, the proprietary world called them 'pyrates'.

To understand that this model of exploitation is not new, let us look at what happened in Indian history. There was a time when the traditional wisdom in India, often called Vedas, was available only as spoken (verbal) code and was part of common knowledge (folklore). Later, this wisdom was rewritten (re-encoded) in an artificial language called Sanskrit. Sanskrit is artificial because of its generative structure of its grammar. There is a sense in which all natural languages are artificial, but Sanskrit is not natural in the sense that it is a strict rule based construction, just as our programming languages are. After re-writing the traditional wisdom in Sanskrit, it is accessible only to those who spoke or wrote this language. Obviously it was the elite section of the society – the Brahmins – who had this access. They were the 'compilers' of Sanskrit, so to speak! There was a time when the right to learn Sanskrit was prohibited, saying that only the royal caste (kshatriyas) and scholarly caste (Brahman) could only decipher what was in there. Even kshatriyas were prohibited from accessing some portions. This restriction to knowledge was accomplished by creating a private language. This is very similar to the way proprietary companies are making private languages to prevent knowledge from flowing freely. Brahmins called a shudra (person belonging to the lower caste) a papi (a sinner), just as proprietary world called those who copy software 'pyrates'. Since knowledge is a very useful thing for survival, in the game of survival it became part of the political process. The powerful people always harbored the 'compilers', buttressed them, so that the rulers could remain rulers for ever. This is the same game. Old wine in new bottle! A corporate version of multinational brahminism!

The question is, are these mechanisms to take ownership of the code or knowledge ethical? Are they even necessary for doing ethical business? Certainly not. In the last ten years or so, a large number of companies world wide began distributing FS and provided services around it, e.g., Redhat, Debian, Mandrake, Ubuntu etc. Instead of selling software per se or claiming ownership of what they created, they sell only software service. Indeed they are harboring a large number of hackers in their business houses to create more software and enhance the efficiency and value, without claiming ownership of what they produce. This proves that ownership of tacit property is not essential for business. The rise in service business is an evidence to its success.

The objection is not about making money, but the means of making money. They are not using technology to enable the society to access knowledge freely, they are actually using technology to curb the free access to knowledge. Their inventions are counter to preserving human values. The story does not end here. There are other aspects to it.

To understand how criminal are their intentions, let us do an activity. Some of you may be using computers for the last 10 to 20 years. Collect all the documents you created during this time. Begin to open those files today using your modern OS. Take my word, several of you will find that most of the older documents are unusable. Either the document could not be opened, because the software that you have on your modern OS does not know how to decode the older documents or after decoding you see that there is some significant loss of data. Now the question is, why did this happen? This obviously happened because the underlying encoding technology is modified. If our work is inscribed in digital media like this, what is the justification for using computers for storing data. Printed books, old audio tapes etc can still be used, because they are not encoded in a secret language. We have not lost the ability to decode them. But our modern operating system lost the ability to decode some of the older computer documents. Who is responsible for this loss? If the agency which lost this is a public body like Government, who is responsible for this loss of data? Often, people were using the software made by the same company all along. Still this loss took place. Why?

The answer is not technical. True, advancement in technology requires that there be some changes. But, what is the justification for changing the encoding of data, without converting them to newer form. Computing industry has norms to follow, such as encoding standards, e.g., ASCII, Unicode, HTML. All the modern computers did not loose the ability to decode the older encoding standards. The loss took place because the software that was used was proprietary and the encoding of the data was also proprietary. A few people like me were fortunate, for I never lost data, since I was using exclusively free software all along.

If using computers for our work means such loss of data, isn’t this a sufficient reason to stop using computers? Even our inscriptions in the ancient caves still exist, although deciphering them is often a challenge. This indicates that preserving code is not enough, we must also preserve how to decode. The only way out of this problem is to make sure that we save our work in a standard format, and record in the museums the process of decoding. Currently our museums store only the code, often forgetting their meaning. Most users of computers may not be able to understand these subtleties of code dynamics. In such a situation, it is the responsibility of educational institutions, media, and the companies themselves to advice the users to follow the best practices. There must be governing policies to preserve cultural records, and not to preserve knowledge in a proprietary format. Otherwise, they must at least warn, that there could be loss of data.

If we collect all the digital documents which cannot be usable in today’s operating systems from all the users from merely the last fifteen years, we will realize that this is not an ordinary loss. It is nothing short of wiping away history, since it is the documents that contain what we did in the past. If the documentation cannot persist, we cannot create history. It is like walking on quick sand, where we find it difficult to trace the steps we took. If we used FS all these years, like me, you would not have lost anything.

This digital crime must stop. Using technology for destroying culture for making profits and becoming powerful must be considered an illegal act. Currently, Governments instead of protecting the society from this mishap, they are indulging in signing special memorandums of understanding with big multi-national companies in the name of developing the society by introducing digital practices. Policy makers and educators all over the world must guide the companies and the Governments so that this crime doesn’t continue, before we lost our history to electrons. All major public interest bodies, such as UNO, should recognize this problem, and create a policy to prevent human culture from vanishing into a black hole. Proprietary software culture is no less dangerous than an atomic bomb.

How Governments and other public bodies are helping Proprietary Software

We hear number of times in the news that a big software company signed a MOU with a Government, a school, or a college or an university. If we look deep in to these MOU, we see some clauses, that make these organizations to use that software produced by that company exclusively. If it was a school or college, this meant teaching that only the companies software to the students. As a result, what we see are brand names in the syllabus as well as text books. For example, visit the website of the most popular open university in India. Their courses train students and examine their knowledge of only proprietary software. Instead of teaching generic software knowledge, they indulge in branded education. This is another serious crime committed by these public bodies. When asked, one state secretary told me that Government does not have enough funds to either buy computers or the software to run them. When companies come forward wearing philanthropic hat, Governments lay red carpet for them. Usually whenever such MOUs were signed, I was told, the companies gave away a number of licenses to schools either gratis or with a substantial reduction of the original price.

This act is nothing short of giving tobacco to small children gratis, till they get addicted, so that addicted people will begin to buy cigarettes. The Governments and public bodies must realize the evil intentions of the companies in doing these so called philanthropic gestures. The education department of the state may have saved some money, but other departments in the state will loose lots of money, for they need to import the software. Our education system is actually spending money and most precious of all time, for the benefit of a multinational company. Most importantly, we must not loose our freedom this way.

FS is an eminent choice for education

When a car breaks down on a high way, the driver opens the bonnet and checks if he can fix the problem. Often, experienced drivers do so. If he does not know, he will take the car to the garage or call a mechanic if he cannot tow it to the garage. The mechanic in the garage fixes the problem, and the car is back on the road. This is what happens with every other technology. Does this happen with software?

How many times your proprietary operating system yelled at you: “The program committed an illegal operation, please contact the program vendor” and quits from running the application. Several times, the OS hangs. Most of you will reboot the system, some times reinstall the ‘corrupted’ application, and it is not uncommon to install the OS all over again. These steps are not same as repairing the system. If you wanted to repair, the first requirement is to obtain the source code. Assuming you are one of the best programmers in the town, you will experience the misery. All the knowledge and training you had remained useless. Even though you knew how to fix the problem, you could not. This is how an educated and well trained engineer is treated with proprietary software. This does not happen with FS. When the program did not work, you could download the source code, look at the program, identify the bug, and possibly fix it. After fixing the problem, you write to the main developers, informing about your feat. They congratulate you, and then you become part of that team. This way, a knowledgeable user is invited to contribute to further the development process. Even if we are not knowledgeable, there exists a possibility to get the problem fixed from another geek in town, whom you could compensate by paying for his time.

I do not know a single concept in computer engineering that cannot be taught using FS exclusively. Since FS does not belong to any one company, the training will not lead to any monopoly. Since source code for all programs is accessible, students can learn by actually looking at the program, better still by modifying them. This is the way all other technologies are taught. But due to lack of a good policy with regard to IT training, IT education in our country, and also in several other parts of the world, got associated with learning a particular brand. All this learning is restricted to how to use, and never how to repair.

How can we make good engineers if they cannot be trained to repair? how can we repair a car, if the car comes with a closed bonnet? It is therefore very important to work towards a policy to teach IT exclusively with FS.

Education as we all know is a cultural institution that transmits knowledge from one generation to the other. No learning is possible if writing (encoding) and reading (decoding) are restricted. PS forces you to write in a language that we cannot understand directly. And you have no right to read unless you pay license fee. If you understand the anatomy of their mind set, you will begin to hate them, unless you are one of them.

Run for freedom even if it is expensive

I did not spend time explaining giving technical or economic reasons for using FS. This is not because FS is not a superior technology. FS is already a success story. From small industries to big one, everyone is using it. The list of success stories is very big. You will possibly hear about them from other sources. Most people who use FS often talk only about the technical features, and much often about the economical aspect of it. Since there is little awareness about the social, ethical and political reasons mentioned above, I chose to talk about them instead of the technical and economical arguments. Putting all the arguments together the case for FS is strong. It is only due to lack of awareness that good citizens in the world are using proprietary software. Once they realize that it is evil, they will make effort to transform themselves. This effort is not without any pains. This may also cost you money. But this effort will not go waste.

There will always be forces in our society who will try to take away our freedom. Now we know the mechanism of how software freedom can be taken away from us. So, it is possible for us to act wisely, and prevent this from happening. We must make this a policy of life, not to ever separate decoding from encoding. Once this separation takes place, there is a possibility of loosing freedom. Therefore the only way to protect it is by making a deliberate effort to ensure the freedom is not lost. This effort is expensive, for it will take your time to begin with. This is the price we must pay for protecting the cultural resources like knowledge and software from the enemies of civil society. Let us recall the reason why Mahatma Gandhi gave the call to weave our own cloth during the freedom struggle. He argued, any technology that cannot be repaired among the community of a village in India, should not be used. The reason is clear. If the village intends to stand on its feet, and not depend on others, it is important to keep the technology simple. Secondly we adopted Charkha as a symbol of the movement, because that enabled every single individual to participate in the process. Every citizen, with a little of bit of training, can be part of weaving the cloth, a basic need of civil life. In the current society, apart from the basic needs, such as food, shelter, clothing, health and education, we need to add an implicit component of the last of the basic needs, namely information, more explicitly. Consequently we need to expand the basic human rights. This need is often mentioned as the right to information. What is forgotten in the assertion of right to information is the right to decode the digital information. At a time when all of human culture is getting re-written in the digital mode, asserting this right is very important. Richard Stallman's demand for the four software freedoms, defining what is free software, is an integral part of the GNU manifesto. The four freedoms explicitly suggest, in a practical terminology, what do we have to do in order to protect our right to decode the digital information.

Today Free Software Foundation of India (http://gnu.org.in) is calling us to participate in another freedom movement. And the slogan of this movement is: Let us weave our own code! Just as we won our freedom for the nation at a huge cost, and so much sacrifice, we must defend our software freedom with the same fervor. Fortunately, this time the enemy we are fighting is not in any one country. The enemy exists in every country. That is why this movement is a global movement to eliminate proprietary software. As mentioned already the effort that we all have to put in this is not small. This may also mean to be critical of the Government, Industry and other public bodies that are working closely with proprietary software by signing memorandums of understanding. This effort is the only price we all have to pay to get rid of proprietary software and achieve freedom. Fortunately the option already exists in a mature form, due to the 30 years of free software movement. Currently, your role is to gain more awareness about this movement, spread the word around, and exercise your freedom by opting for free software for all your work, and participate in it to help ourselves to sustain our culture, our knowledge for ever.

Let us run for freedom even if it is expensive!

3 Responses to "Copyleft Society"

P2P Foundation » Blog Archive » Proprietary software is a black hole of knowledge universe: Why all knowledge should be free Says:

November 10, 2009 at 10:43 am
[...] from a longer text (introducing free software to an Indian audience) by [...]

Reply
Streit um die Neufassung des European Interoperability Framework (EIF) | OpenData Network Says:

November 11, 2009 at 1:25 pm
[...] software is a black hole of knowledge universe: Why all knowledge should be free. Excerpt from a longer text by [...]

Reply
Compensation Calculator Says:

April 13, 2010 at 8:34 pm
Thank you for your fantastic work in putting this together! This article is far batter then others i have found on this topic. Thank you very much for your help in helping me understand this subject.

Reply