FLOSS IS NOT JUST GOOD FOR YOUR TEETH!

www.sarai.net

ISBN 81-901429-8-4

9 788190 142984

FLOSS Is Not Just Good for Your Teeth

Editorial/Design Coordinator: Monica Narula
Text: Ben Olin, Aniruddha 'Karim' Shankar, G Karunakar
Illustration: Parismita Singh
Design: Mrityunjay Chatterjee

32 pages, 6.5" x 9"
ISBN: 81-901429-8-4
Delhi, 2006

Produced and Designed at the Sarai Media Lab
Centre for the Study of Developing Societies
29 Rajpur Road, Delhi 110054, India
www.sarai.net
dak@sarai.net

FLOSS Is Not Just Good for Your Teeth is a media work produced as part of the activities of 'Towards a Culture of Open Networks', a collaborative initiative of Sarai-CSDS (Delhi), Waag Society (Amsterdam) and t0 (Vienna), focused on bridging "Information Society" in Europe and India through culture and communication. 'Towards a Culture of Open Networks' is supported by the EU-India Economic and Cross-Cultural Programme under its Media, Communication and Culture dimension. http://www.opencultures.net

Disclaimer: This document has been produced with the financial assistance of the European Union. The contents of this document are the sole responsibility of the partners in the 'Towards a Culture of Open Networks' initiative, and can under no circumstances be regarded as reflecting the position of the European Union.

WHAT IS FREE SOFTWARE?

One of the components of the umbrella term Free/Libré/Open Source Software (FLOSS), free software is software that anyone can use, copy, improve, examine or distribute, either at no cost or for a price.

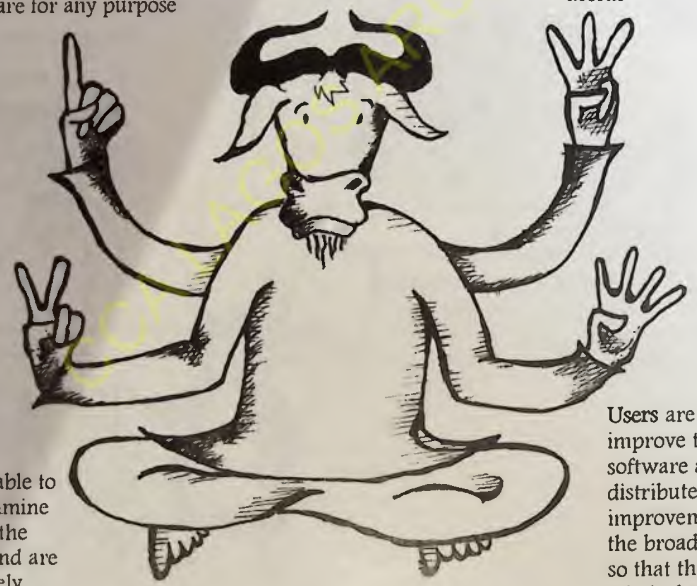
More precisely, it refers to four fundamental freedoms which users of the software have:

1
Users are allowed to run the software for any purpose

2
Users are able to closely examine and study the software and are able to freely modify and improve it to fill their needs better

3
Users are able to give copies of the software to other people to whom the software will be useful

4
Users are able to improve the software and freely distribute their improvements to the broader public so that they, as a whole, benefit



Hold on a minute!!

I thought the word "user" just meant someone who simply uses a piece of software.
While the common understanding of the term "user" suggests someone who simply "uses" software, one of the most interesting characteristics of free software is the way in which the distinction between "user" and "creator" blurs. Creators of free software are also its users, and users of free software are also often (though not always) involved in the creation of the software. For a number of reasons, as we shall see, it's also extremely easy for someone who uses free software to become a creator. That's right, **you don't need to be a programmer to be a creator of free or open source software!**

Open source software? What's that?
For someone who is not familiar with the intricacies of FLOSS, there are few differences between free software and open source software. In almost all cases, they deal with the same things – works that people can improve, study and redistribute. They differ only in their primary emphasis. While open source software proponents believe that this methodology of software creation tends to create the best software, proponents of free software believe that even as a "mere user", the freedom to be a co-creator and co-sharer of software you use should be protected and sustained. We'll expand on these two concepts as we go on.

What's stopping the user from being involved in the creation of other types of software? In what ways are they not "FREE"?

If you go to a computer store and pay money in return for a box containing software CDs, you might think that you've bought something. However, most "closed", "non-free" or conventional software in the world today is not sold; it is licenced. From complex operating systems to tiny games and screen savers, the final users of the software **do not buy software but pay money in return** for a licence to use it under certain conditions, which are laid out in an End User Licence Agreement. In software such as these, restrictions are often imposed on the uses to which the software can be put. In almost all cases, users are explicitly prohibited from "taking the software apart" to study how it works. They **cannot modify or improve it, and are commonly only allowed to make a single copy of the software** (for backup purposes). They're certainly not allowed to pass around copies. In most countries, if you get a copy of conventional, proprietary software from a friend or even install software that you've paid money for on more than one computer, you could be guilty of what the law calls "software piracy" and would be liable to pay punitive fines and, in some cases, even spend time in jail.

You mentioned people being creative – working on software. Why should I care about free software if I'm interested in something totally different – like composing music or making movies?

Whether it's composing, editing, remixing and DJ-ing music, importing, editing, encoding and distributing movies or just simply creating professional-looking publications, free and open source software have a host of high-quality tools that speak to almost every single creative niche. Moreover, as creators are almost always extremely interested in the tools they use to create, free and open source software tools give you unprecedented control over your favourite application – even if all the control you want to exercise is to tell the authors about the problems, say, with the latest version, or what YOU think is a good idea for future development. Oh, and it's important to mention that contrary to closed and non-free creative tools, which are often prohibitively expensive – especially for beginners, hobbyists and people in the third world – **free software tools for media creation are freely available** for download off the internet.

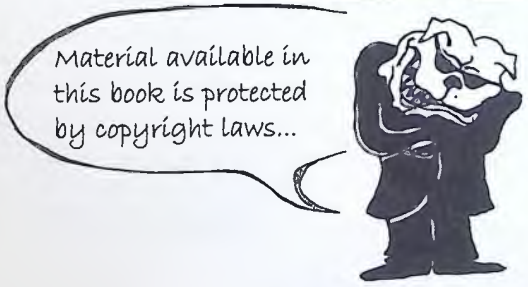
Surely if I duplicate free software and pass it around, it means that someone is losing out somewhere along the way, right?

The moment a person or an organisation decides to licence a piece of software as free software, they have automatically agreed to allow all people who get a copy of that software the right to redistribute the software. Rather than see this as a deficiency of free software, it should be considered an advantage – the more your software is spread, the more users it garners – and as we've said before, the more users free software gets, the more co-creators it gathers. A piece of free software is never a closed, finished product. Free software is always already a collaborative enterprise. No one loses anything when free software is copied and copied again. Rather, **people who copy and distribute free software are taking the first step in collaboration and co-creation,** in a spirit of civic mutual aid from which everyone gains. This is very different from duplicating commercial software, which could also get you into a lot of legal trouble!

So you're saying that users of non-free software have a totally different relationship with the software that they use?

In almost all cases, the use of commercial software is governed by an End User Licence Agreement. These agreements tend to put limitations on your use of the software and go to great lengths to absolve the makers of the software from any responsibilities whatsoever. In contrast, Free software licences spend most of their time specifying the rights allocated to users, and the responsibilities for the use of the software. Broadly, all users of free software are, at the bare minimum, considered to be potential creators of free software. The licences reflect this by making sure that it is extremely easy for someone to go from being a mere consumer of the software to a co-creator or contributor. Of course, no user of free software is compelled to tinker with it to get it to a functional state – the vast majority of users of free software "use" software in the conventional sense.

In general, all users of commercial software are considered to be mere passive consumers of the software. Great pains are taken to ensure that the passive user remains passive, regardless of any possible desire to be more involved with the software. What few rights commercial software licences grant users are given carefully and grudgingly.

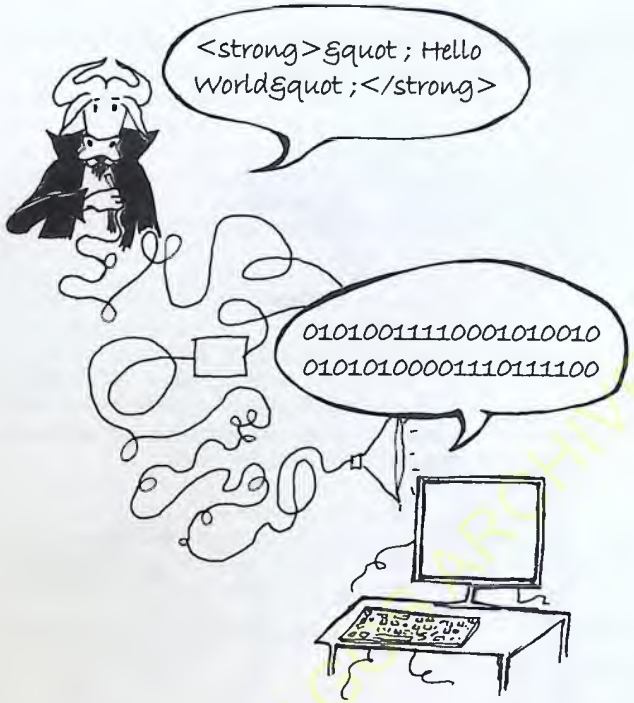


Okay. But "free" still means that I don't have to spend any money, right?
The word "free" is slightly misleading in this case. The free in free software refers to the freedoms that we've talked about that people have. The freedom to not just be a passive "user". While it might sound confusing, there's nothing in the definition of free software that says that you cannot sell it to someone for a price. Indeed, there are companies whose entire business model is centred around collecting, creating, compiling and selling free software. These companies make money by providing customised manuals and supporting their customers if they have any problems with the software. However, since someone to whom free software is licenced is free to sell, or give it away in turn, you can almost always easily find free software openly (and legally) downloadable on the internet. **Companies that sell free software make most of their money by customising the software for clients and by offering support for their products in return for money.**

So commercial organisations make free software too?
Yes, and for a number of good reasons. In traditional methods of creating software, programmers are paid by a company to work on software that the company owns. In the free/open software world, more programmers, testers and bugfixers work on software than the company could ever expect to hire. In effect, if a company decides to "free" its software, it suddenly finds itself with a vast amount of potential contributors to the software. While they can't now stop anyone from redistributing, changing or using the software, they can make handsome amounts of money by selling services and support for the software: if a user of the software wants a particular improvement or feature, or if they have a problem with the software that they need resolved, they can pay the company to extend the software or help them with their problem. Another reason why commercial organisations might make free software is because all hardware, to be useful, needs software. Traditionally, hardware companies also provide, for free, the software required to make their hardware usable. If one of these companies decides to "free" their software, they suddenly have access to many more programmers, testers and bugfixers than they could possibly afford, were they to keep their software closed.

How is free software actually made?
To properly understand how free software is made, we have to have a good idea how **all** software is made.

Okay then, how is software made?
A software programme is simply a set of instructions to a computer to do something. Since the computer is a machine without any capacity to think for itself, it can only understand instructions written in a particular language – this format is called **"object code"**. Unfortunately, to human beings object code looks like gibberish. While they are creating software, humans use a particular format called **"source code"**, that they can easily understand. Source code uses letters, numbers and punctuation and, like any human language, can be understood by humans who learn to do so. So now we have source code, a format that humans create and write programmmes in, but which looks like gibberish to a computer, and object code, a set of instructions that a computer can understand but which look like gibberish to a human. **A special programme, called a compiler, transforms the source code into the object code.**
Let's go over that again. Human beings lay out what they want the computer to do in programming languages, which are written out using alphabets, numbers and punctuation marks. A compiler takes the instructions that are expressed in programming languages and transforms it into a language that computers can understand.



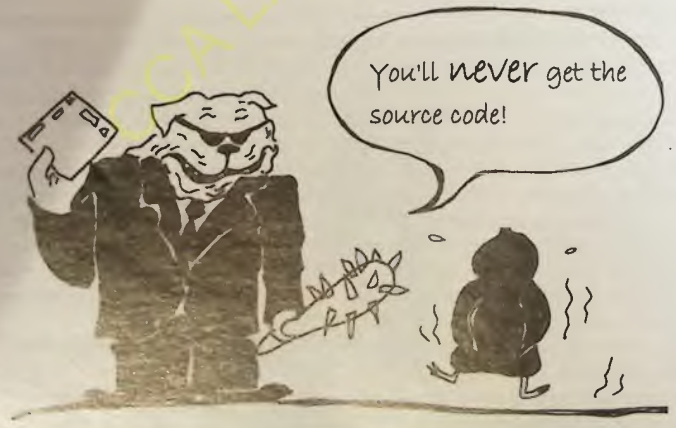
So, now that I understand how software is made, how is free software made?

Remember, humans cannot understand object code. Therefore, if a human wants to closely study, modify or improve a piece of software, s/he has to have access to the source code of the software. Since being free to examine, modify or improve software is central to the concept of free software, it follows that humans have to have access to the source code of a piece of software for it to be considered free. Unlike closed software where only the original software creators (or those they explicitly provide access to) have access to the source code, anyone who is interested can get access to the source code of free software. Therefore, if a user of free software wants to modify or improve it, s/he is free to do so. In many cases, the people who make the improvements make the improved software available to the broader public via the internet. By definition and in practice, people who in most cases are connected to each other just through the internet create free software collaboratively.

One critical aspect of the creation of free software that is often overlooked is the feedback in the form of complaints and suggestions from normal users. This feedback is actively sought, and many tools exist that make it easy for lay users to integrate these complaints, bug reports and suggestions into the production methodology.

So how is this different from the production of other kinds of software?
Typically, entities that produce non-free software have very tight restrictions on who has access to the source code of their programmes, and distribute their software only in object code format. The reason for this is that while it is very easy to compile source code into object code, it is very difficult to derive the original source code from an existent object code.
A good analogy would be curd. While it's easy to make curd from milk, it's pretty much impossible to get milk from curd.

But what's stopping someone or some company from taking free software and making it into a closed product for their own financial gain?
This has happened. People have taken software that had been placed in the public domain and used what was once a commonly-owned and commonly-produced resource to make software that is closed. To stop this from happening, an ingenious legal device was created that made sure that no one could take the four freedoms – the freedoms to run, examine, improve and distribute – away from free software. In a tongue-in-cheek play on the word copyright, this protection was called **"copyleft"**.

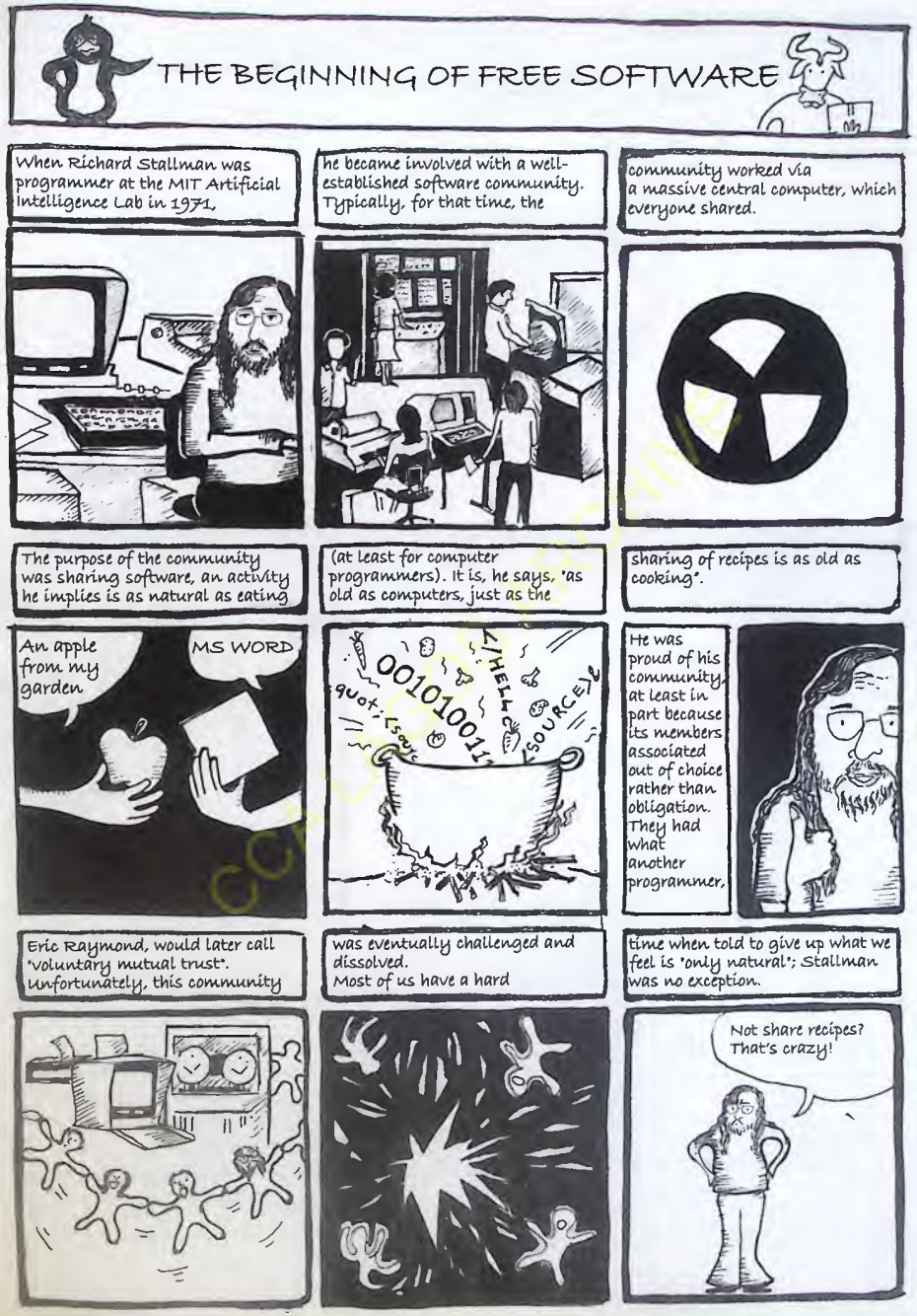


What do you mean by "copyleft"?
Copyleft is a general method for making sure that all terms under which something is licenced for distribution apply to all subsequent redistributions. It's a tool that many licences employ to ensure that the terms under which the original authors distribute the software or piece of work are not modified by people who redistribute the software. The simplest way to make a programme free is to put it in the public domain, uncopyrighted. This allows people to share the programme and their improvements, if they are so minded. But it also allows uncooperative people to convert the programme into proprietary software. They can make changes and distribute the result as closed software. This means that people who receive the programme in that modified form do not have the freedom that the original author gave the software when s/he placed it in the public domain; the middleman has stripped it away. **Copylefting free software ensures that anyone who redistributes the software, with or without changes, automatically passes along the freedom to further copy, examine, improve and run it.**

So how would I go about copylefting a programme?
It sounds strange, but **before you can copyleft something, you first have to copyright it**. Broadly, when you copyright something you've created, you are stating that you have certain rights of distribution, use and exclusion over it. Once you've copyrighted it, you then add distribution terms in the form of a licence document – these terms comprise a legal instrument. To copyleft it, your licence must state that all users of the programme can only redistribute it under the terms of the original licence. So, when you copyleft free software, the code and the freedoms become legally inseparable. **According to the Free Software Foundation, "proprietary" software developers use copyright to take away the users' freedom; we use copyright to guarantee the users' freedom.**
That's why we reverse the name, changing "copyright" into "copyleft".

So you're saying that free software is not automatically copylefted?
That's right. There's nothing in the definition of free software that states that the subsequent redistribution of free software must be on the condition that the software remains free. However, several free software licence agreements exist that automatically incorporate copyleft. The most famous and widely used one is the **General Public License (GPL)**.

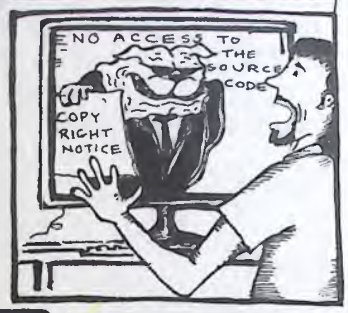
Is this copyleft against the law?
No. Legally, a licence is a contract between the licencor and the licencee, and legally, parties have the freedom of contract – meaning that they can enter into contracts with each other after mutual agreement on the terms. Unless the terms incorporate elements that are illegal, parties can agree on any terms that they wish. When we say that something is copylefted, we refer to a particular feature in a legally binding licence agreement between the parties. While copyleft may be a novel concept, it certainly is not illegal.



The community's demise in 1981 was precipitated, as he tells it, through a combination of events related to people leaving for new jobs and changes in the computer operating systems being used at the lab. Members found that their community had become illegal. They were shut out by 'non-disclosure agreements',

or copyright notices that accompanied the operating system software and forbade the sharing and modification of its

source code, the programme that made it work. As sharing and modifying (and improving) 'code' was what the



community was all about, this licence change presented a serious problem. Stallman was faced with the choice of

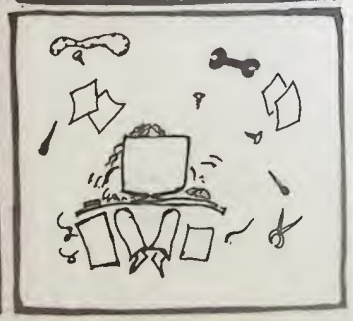
joining a company as a programmmer of the same type of closed-licence programmes

that he thought were causing the problem; quitting programming; or breaking the law.

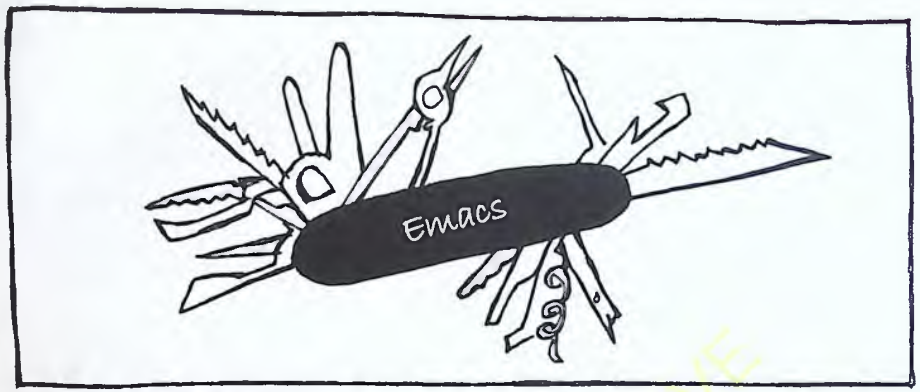
Rather than accept these options, however, he set out to build a way around the impasse: he decided

that if he wanted to be able to write and share software with fellow programmers, then he

would have to secure his independence by creating his own computer system.



He began planning an operating system and all the tools that he needed and wanted to use, which would be completely free of legal restrictions and open to modification, piece by piece. Only in this way could he create a system that he need fear that it would suffer the fate of its predecessor...



In 1984, Stallman took the further practical step of resigning from his job so that his employer could not claim ownership of his work. The task he had set himself was gargantuan by anyone's definition, but he had a plan. He would build on the foundation of existing public domain software for UNIX, a computer operating system that was widely in use at that time, and a UNIX-like system would be his goal.

The first piece of the puzzle, which he began putting together, was a Swiss Army knife of text editors called **Emacs**. By early 1985, he had early versions of it ready to share with others. It is important to note that Stallman's personal story prefigured the interest in free and open software shown by the commercial world in the past 15 years. While he wanted the right to collaborate and to create a legal and free community of collaborators, he was in no way against the programmer's right (including his own) to be paid for work. "Freedom" for him was about intellectual rights, not price. After quitting his job, he obviously needed to support himself, and he wanted to find a way to make money for his work on free software. He uploaded Emacs to a networked machine (so others could download it if they wanted to), but, as he tells it, many people still had no internet access and, therefore, could not access his programme. His solution was to spread the word that he would mail the programme to anyone who sent him $150 – this price was to cover his time and mailing costs, not the software itself, which was "free". Thus he began to solve his money problem while being able to stick to his ideals.

But how is something like a software programme created by people who never meet face to face? Doesn't it get terribly confusing?

If you've ever tried to help in the organisation of a family function, such as a wedding, you know that there are a hundred things that can go wrong even with people remaining in constant contact with each other. To think of software development happening between people who might never meet face to face seems quite impossible, at first glance. The problems of coordination, of assigning responsibilities and of submitting and reviewing changes to software are very real indeed, and a number of very smart people have literally spent decades trying to create tools and procedures to resolve them – not just in theory, but

in practice. Just to take an example, a tool called CVS exists which allows people to submit changes to the source code of a programme. At a later date, it is possible to roll back any number of changes to the source code made through CVS as easily as you can pull the blankets off a bed. Because these tools use the internet, **anyone with internet access and even someone with just an email account can participate in the development of free software without the whole affair dissolving into chaos.**

So are you saying that in some ways the history of free software and the internet run parallel to one another?

In some ways, yes. The internet is built on UNIX networking technology. From the early 1970s onwards, academic institutions were able to access UNIX and its source code. Because the source code of UNIX was kept open, allowing a whole range of experts to work on it, internet technologies developed at a very fast pace. This idea of open collaboration on which the internet was built is the guiding force behind free software, which similarly benefits from the constant modifications made possible through an involved community of users. **A really good example of free software benefiting from the broad reach of the internet is the collaboratively developed computer operating system Linux.**

Okay, but now what is Linux? When did that come about?

Linux is a computer operating system. It's risen into prominence these days because many commercial companies have started to build their businesses around providing support for software, hardware and services that they build around Linux. It's garnered lots of media attention in the past year or so, and we've even started to see broadcast advertisements on mainstream television that promote it. It's probably the best known example of free/open source software. While Linux, in a strict sense, refers to the kernel, or the core of the

software that runs on a computer, it is often used to refer to the entire operating system, plus the free software tools and system software. These tools and system software were created by the GNU foundation, so some people prefer to call the entire collection GNU/Linux. **Linux can run on a staggering variety of computers – from mobile phones and handheld devices to personal computers to massive supercomputers.** While it was initially developed by a group of hobbyists connected to each other through the internet, it has since gained the support of large companies such as IBM, Hewlett-Packard and Novell. In some areas, it challenges the dominance of Microsoft's (proprietary and closed) Windows line of operating systems and is seen by that company as a major competitor. The Linux kernel was initially written in 1991 as a hobby by Linus Torvalds, a student at Helsinki University.

How do you see the future development of free software?

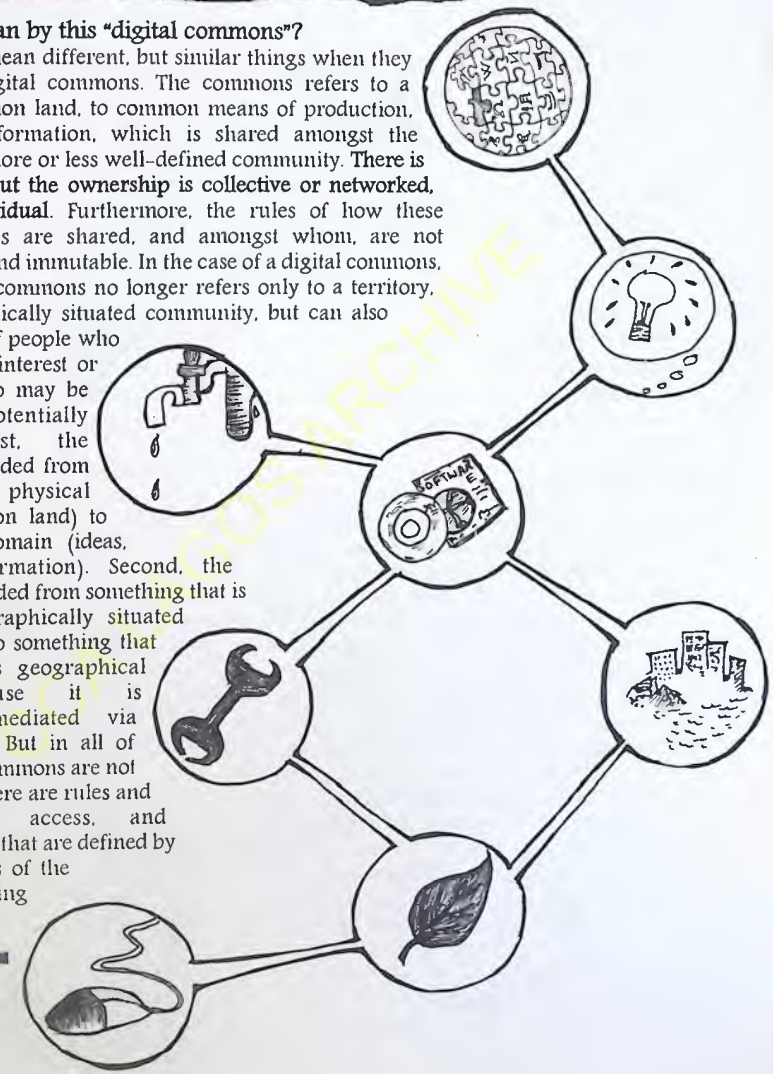
If we take a look at the history so far of free software, and merely extrapolate from there, we should see free software gaining greater and greater prominence in areas where it has been, traditionally, weak. While free software is widely used on servers (computers that provide resources to other computers), it has had limited success in the home and desktop PC markets. In a few years' time, we should see an increasing number of people opting for free software on their personal machines.

What kinds of people make free software?
Most people who contribute towards free software are volunteers who typically have an unrelated day job. These people spend their free time developing free software. In addition to this, academic institutions, non-profit organisations and even commercial entities contribute towards free software.

But I still don't understand why anyone would want to give away their work for free? What's in it for them!?
We have to consider that financial gain isn't the only possible "reward" that motivates people. The culture of collaborative interaction that has flourished on the internet stems largely from the long established practice of information-sharing common to the scientific community. Historically, **if scientists insisted on keeping their findings "closed" or secret from other scientists, almost every major scientific advance that you could think of would have been impossible – as all of them build on the work done by other scientists earlier.** Individual scientists gain personal recognition for their discoveries, which, along with the satisfaction they receive for advancing scientific knowledge, we might see as their "payment". We can better understand the motivations of programmers working with open source technologies in the light of this example. Programmers build their reputations through their contributions to the development of free software within a **digital commons** from which they are also constantly receiving value – both as software and as information.

Wait a minute...

What do you mean by this "digital commons"?
Different people mean different, but similar things when they talk about the digital commons. The commons refers to a resource, to common land, to common means of production, knowledge or information, which is shared amongst the constituents of a more or less well-defined community. **There is ownership here, but the ownership is collective or networked, rather than individual.** Furthermore, the rules of how these common resources are shared, and amongst whom, are not necessarily fixed and immutable. In the case of a digital commons, the notion of the commons no longer refers only to a territory, i.e., to a geographically situated community, but can also refer to a group of people who share a common interest or set ideas, yet who may be distributed potentially worldwide. First, the commons is extended from a set of shared physical resources (common land) to an immaterial domain (ideas, knowledge, information). Second, the commons is extended from something that is necessarily geographically situated (walking paths) to something that is shared across geographical divides, because it is electronically mediated via digital networks. But in all of these cases, the commons are not entirely "free". There are rules and mechanism of access, and limitations on use that are defined by the shared values of the community sharing these resources.



In what ways could I benefit from involving myself in this digital commons?
Everyone has everything to gain though their involvement in the digital commons. Some people have compared the structure of the digital commons to that of a cooking pot in ancient societies. Here people would contribute their different ingredients to the pot, with the end product being available to all contributors. All the tastier, thanks to the varied offerings of the contributing parties! In the same sense, **we can all benefit by sharing information, programmes, attitudes and ideas through the digital commons.** In fact, software actually develops at a faster pace through being 'open' and available to all through the digital commons, where users and programmers can share their experiences using and modifying a piece of software. You don't have to be a programmer to be a part of the digital commons – **the moment you correct a mistake in a help file, or test out some software on your machine, or even constructively criticise the software, you are a part of the digital commons.**

It seems to me that this entire thing (free software, the digital commons) is focused on people working together. Right?
You're right, cooperation and collaboration are central themes in the free software world.

So that means that people should be able to use the same ideas – collaboration and openness – in other areas!
You're right again. There are lots of projects where the core concepts of free software – freedom, collaboration, sharing – have been used. One very famous example amongst many is **Wikipedia – the largest encyclopaedia in the world, with over 1.5 million articles in the English edition.** It's an open-access encyclopaedia to which anyone can contribute. In less than 6 years, 99 million changes have been made by millions of authors, including over 3 million registered users; and after a mere $3\frac{1}{2}$ years of operation, it's many times the size of any other encyclopaedia, whether online or in print. What's also quite fascinating is

that since its design is open and well-documented, it's possible for people to start writing encyclopaedias in their native languages – it even includes articles in Cherokee, Amharic and Urdu. Since it's open and there's no central editorial board which has to answer to publishers for whom profit motive might be a large consideration, it's possible for people to write on controversial subjects, or place on record pieces of information that, for whatever reason, have not found inclusion in other encyclopaedias. It's also possible for people to write or contribute to articles on subjects that other encyclopaedias would consider trivial. All in all, what "freedom" means in this context is that people have a repository of self-written knowledge and culture that is not under any one person's control.

In keeping with the principles of free software, **people are allowed to make copies of the entire Wikipedia encyclopedia and modify, tinker, improve and redistribute it – and some people have.** A prominent journalist lost his baby son to Sudden Infant Death Syndrome, or "crib death" – where, for no apparent reason, a newborn baby dies. He proposed the idea of a simplified health monitor to warn caregivers of patient symptoms. Volunteers initiated an online community site to coordinate the project, which has already received contributions and ideas from thousands of people. Another example of a knowledge-based distributed initiative is OpenLaw, an experiment in the collaborative development of legal arguments. Open to both lawyers and the general public, it provides relevant documents regarding legal cases and discussion tools to allow users to interact and propose potential arguments, and find weaknesses in each other's strategies before cases are brought to court. From designing low-cost, low-maintenance incubators for premature babies born without access to healthcare in third world countries to HIV/AIDS research, people have been taking the ideas of collaboration and openness and implementing them in sometimes totally unrelated fields.

In fact, this very booklet has been designed using free software principles – a "bare framework" was posted onto the internet as a wiki and on mailing lists for additional questions and answers. This bare framework was fleshed out with lightning speed by a cohort of mostly anonymous contributors. Once we got to a stage where we had a coherent framework, we started building on a local copy of the wiki – **at all times incorporating suggestions and feedback from a wide range of people.**

Arguments for Using Free Software

I'm still not convinced. Surely a big computer company knows best when it comes to designing software? Why would I want to use software designed by an unskilled amateur?

When we engage with free software, we are using a product that has been crafted by an entire community of developers rather than any one single programmer. Crucially, the developers of free software are also its users, so they have already encountered the same kind of problems, bugs and glitches that any user might expect. Since free software remains open, problems become ironed out through an ongoing process of collaborative interaction among interested people. The disparate interests and requirements of the developer-user free software community mean that a wide range of problems are encountered, and these find solutions from other developer-users along the way. In this way, free software benefits from daily improvements. This kind of collaborative problem solving is impossible with closed software where the producers are not the consumers, and where collective refinement is effectively prohibited.

Another very important point is that in almost all cases, **free software uses open specifications for its operation – the structure and format of the files that it saves its data in, the way in which the programmes are configured – are all openly available.** What this means is that unlike closed source software, which, in the majority of cases, uses closed protocols and formats for its data, you are guaranteed access to your data and are not forced to rely on the original vendor in order to view, edit or work with your own files.

But what about bugs? Surely free software is more likely to be virus prone?

The question really involves two different types of computer-related problems – bugs and viruses. **A bug is an unintended and unwanted property in a piece of software that causes it to malfunction.** Unfortunately, bugs go hand-in-hand with software, and free software is no exception; but free software does get rid of bugs at a faster pace. When you find a bug in a free software programme, you can exercise the freedom to help yourself and correct the programme. If you are not a programmer, you are free to hire any programmer to correct the programme. You are not under the mercy of any single organisation. Also, by submitting the bug fixes to the maintainer of the package, the software package steadily improves. Viruses are malicious programmes that infect other programmes by embedding copies of themselves within them. When these programmes are executed, the embedded virus is executed too, thus propagating the infection. Viruses are prevalent in systems that lack proper security. Because free software can be examined by everyone and not just the programmers who originally created it, security flaws are found faster and fixed faster than closed software. Many huge organisations around the world choose free software to power their websites and email. Amazon.com, Rediff.com and even Google.com, a search engine so popular that it has become a word in the English language, run Linux, a free software operating system, and Apache, a free software web server.

So you're saying that free software actually evolves at a faster pace than closed software?

It's impossible to make a sweeping statement that all free software, by virtue of its free character, evolves faster than all closed software. Having said that, it's certainly true that many examples of free software exist – from word processors to operating systems, to the infrastructure on which substantial portions of the internet run – which prove that a free software development model can result in software evolving at a blinding pace that can match and even outstrip, in quality and in pace of evolution, the best efforts of the closed software development model.

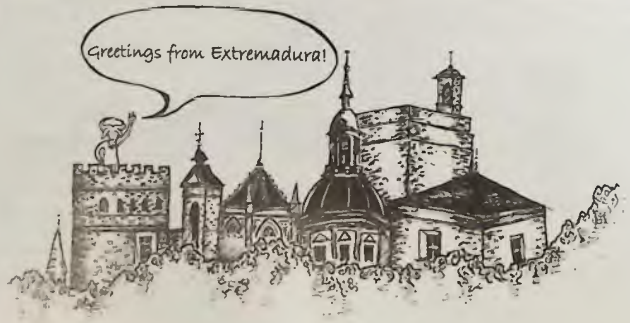
Really? You mean to say that free software can be more advanced than software developed by a conventional company?

It might sound crazy at first, but it's true! In some areas, free software is at the cutting edge of technology. One such area that is getting lots of attention these days is wireless technology. Using free software tools to create something called a **"mesh network"**, it's now possible in some cities to actually provide virtually free wireless internet and phone services to all people within a certain area. Because free software is developed in the large part by enthusiasts and by professionals, it's possible for them to work in areas and create tools, which, if created by a company, might decrease their revenue stream in the future. Also, if you're a free software developer/user/creator, there are no artificial limits to your creativity – feel free to tinker, examine and tweak to your heart's content. Here's another example of free software being at the cutting edge – the computers used to control and direct the Mars Rovers run free software.

But does software that can be freely modified run better than closed software?
Again, we can't state emphatically that all free software, by virtue of its free character, runs better than all closed software. Having said that, many reputed authorities believe that **free software tends to crash less than closed source software.** One of the virtual axioms of the free software development process is that the more people who can look at and suggest changes to a piece of software, the easier it is to find, study and fix problems in the software. Given that no software company could hire the millions of free software programmers, bug fixers, testers and users that are in existence today, it's safe to say that the potential of free software to improve is far greater than that of commercial software.

Free software is only something used by computer enthusiasts, right?
Wrong. From stockbrokers to ecologists, from gamers to musicians, from storefront cash registers to the Human Genome Project, free software is used by a massive spectrum of users, definitely broader than those for whom computers are merely a hobby. **Free software is also used by artists and writers, who like using the software because it is, according to them, the best they can use.**

Have any established organisations actually used free software to their advantage?
The biggest and best-known internet search engine in the world – Google – runs on free software. Each time you send an email on Yahoo Mail, you're relying on free software to send your messages. At the time of writing, many state governments around the world have switched to free software – the government of the Spanish state of Extremadura, the city of Munich and others. In many institutions and contexts where security and reliability are prime concerns, such as at financial brokerages, stock exchanges, government research installations and electronic infrastructure, free software is the preferred, and in some cases the mandatory, tool of choice.
We are slightly smaller than Extremadura! But Sarai is nonetheless proud to use free software in every area it engages with – print, audio, video, animation, email, word processing, archival, software development and network infrastructure.



Personal Relationship to Free Software

Okay, but why would I want to modify my software anyway?
You might not. There are plenty of reasons, however, why you might. Say, for example, a software does not support your local language. You would like this software to be available in your local language so that you can use it. If the software is proprietary, you will have to go and beg the "owner" of the software. If he finds that making the change wouldn't be profitable, he will not make the change. With free software, you can make the change yourself, or you can go to a programming company and ask them to make the changes for you. With free software you are not helpless. Not all free software users might have the necessary technical expertise to change the code. But with free software, somebody who knows how to change it can change it. What's more you can help that person by providing valuable feedback.
Another very compelling area where free software is of great importance is in **"internationalisation"** and **"localisation"** – these two impressive terms simply refer to the process of making software usable to people who speak different languages. Because it's open and well documented, it's easy to incorporate translations for different languages into free software. An example would be having the menus, messages and the entire interface of a word processor readable in, say, Arabic or Hindi or Chinese. It can, with the same ease, be translated into "niche" languages such as Tulu, Inuit or Amharic. In the commercial/closed software world, a big corporation might decide to create versions of its software in Hindi and in Bengali, but not in Assamese or Oriya. If you're from Assam or Orissa, you might have to create your tools yourself... and the natural choice would be to use free software as a foundation. In this fashion, free software places computing resources and networks within the grasp of people who do not speak English.

Look, I'm no computer whiz! Isn't it easier for me to just use packaged software? Who do I turn to when something goes wrong?
Free software also comes in packaged form like most proprietary software. If you buy it in this form, the seller provides support when things go wrong. Additionally, the free software community has excellent documentation, support channels, internet forums and user groups who will, in most cases, help you out for free.

But how do I know I can trust someone not linked to a big company that has a reputation to uphold?

A programmer working for a big company assigns the copyright in all things that s/he creates to the company. Even if they were the sole person who conceptualised, designed and created a piece of software, it is owned by the company they work for, and it is the company's complete discretion whether or not to give the individual programmers any public credit for the work they have done. Just as the praise for software that is well-crafted goes to the company, not the individual programmers who worked on it, so do the brickbats. Additionally, since the software is closed source, it is very difficult to examine it for hidden bugs or wrong design decisions. When a free software programmer decides to publish something on the net, s/he knows that anyone can examine the work s/he has done, and that the strongly critical, irreverent and egalitarian user community will express their vocal scorn at shoddy work. Or, worse in some people's eyes, will ignore it. Because of this, free software traditionally goes through many cycles of development – "unstable", "alpha" and "beta" stages before developers gather the courage to put their reputations on the line and state that the software is "in release form" or ready for public use. This paradigm tends to create software that is of a high quality and that becomes better over time.

Additionally, let's recognise that the primary motivation of a big company is to enhance shareholder value. Everything else that a big company does is epiphenomena, i.e., things that are incidental to their prime motivation. Therefore, if a big company feels that a certain course of action will increase their profits, and that the benefits that will accrue to them are greater than the costs in reputation or other factors that the course of action will impose upon them, they will take that course of action. An example might be a corporation selling software knowing that it is defective or unstable, hoping to force the users to upgrade to the company's "new and improved" version.

Why should someone who is perfectly happy with closed/non-free software care about open or free software?

An excellent reason for engaging with free software is something many people can understand – it's great fun, especially when one starts to transcend national and linguistic boundaries while working with it. By engaging with free software in the ways that we've spoken about, we resist impulses that ensure that people remain mere consumers of news, culture, art and information, and we help foster a climate of collaboration, creativity, respect, friendship and sharing.

While it's heartening to see the increasing use of free software, it is also important to remain aware of legal developments that have, increasingly, posed threats to creativity and to the freedom to collaborate and share. Recent developments in the law relating to software patents could threaten the livelihood of hundreds of thousands of people involved with free software. More importantly, such legal developments threaten the expression of inherent human creativity, as they threaten the tools with which these creative people work. By working with and on free software, we reaffirm one of humanity's essential attributes – that of creativity – and we display our solidarity to the millions of people who live and work with free software.

What kinds of problems might I expect to encounter using free software?

Often when people start to use one of the many free software operating systems and user environments, they find themselves understandably bewildered because certain things like floppy drives and installing new hardware or software work differently from commercial operating systems like Windows or MacOS. Additionally, if you run software that is not adequately tested, you might find it crashing or behaving in unpredictable ways. Sometimes, companies that make hardware keep the inner details of their products secret. When that happens, free software developers who want to create software for those particular pieces of hardware have a difficult job, as they have to make educated guesses about the inner workings of the hardware. When that happens, the hardware does not work as well as it should.

Okay, so how would I begin to install free software on my machine?

We recommend you try out KNOPPIX (available at http://www.knoppix.org), a really easy-to-use distribution of Linux that can be downloaded for free off the internet. KNOPPIX is try-it-out-and-see-if-you-like-it software and it comes on a CD that you insert into your computer before it starts up. When your computer starts up, it will load a complete working environment from the CDROM without changing anything on the hard disk. Since it's not using your hard disk, the programmes will start up much slower than they would otherwise, but you can easily see for yourself what it's like to run a free operating system and free software programmes. If you like KNOPPIX, you can install it onto your machine easily.

Another distribution of Linux that is easy for people new to free software to use is Ubuntu Linux. You could try that if you want terrific support for your hardware and a setup that holds your hand the entire way.

You can still use free software if you run Windows or MacOS. Try Mozilla Firefox, a great browser that is free software! There's a list of some of the free software that you can run on Windows:

http://theopencd.sunsite.dk/

http://www.jairlie.com/oss/suggestedapplications.html

Check it out! We can help you get KNOPPIX or Ubuntu Linux -- email us at deb@sarai.net or send us a letter at 29 Raipur Road, Delhi 110054, India, and we'll get back to you as soon as we can with instructions on how to enter the world of free software - today!

Links and Resources

General

Creative Commons
http://creativecommons.org/

Free Software Foundation, India
http://www.fsf.org.in

GNU
http://www.gnu.org/philosophy/

Unix history
http://www.levenez.com/unix/

Wikipedia
http://en.wikipedia.org/wiki/Main_Page

User Groups

LUGs, GLUGs, FSUGs in India, Asia
http://en.wikipedia.org/wiki/LUGs,_FSUGs,_GLUGs_in_India_and_Asia

Indic Computing

Indic Free Fonts List
http://www.indlinux.org/wiki/index.php/IndicFontsList

IndLinux project
http://www.indlinux.org/

Padma - plugin for Firefox
http://padma.mozdev.org

Sarai FLOSS
http://floss.sarai.net

Internet

Firefox web browser
http://www.mozilla.org/projects/firefox/

Gaim - multiprotocol chat client
http://gaim.sourceforge.net

Kmail - email client
http://kmail.kde.org

Konqueror web browser
http://www.konqueror.org/

Thunderbird - Email client
http://www.mozilla.org/projects/thunderbird

Xchat - IRC chat client
http://xchat.org/

Desktops

GNU Network Object Modelling Environment (GNOME)
http://www.gnome.org

KDE - K Desktop Environment
http://www.kde.org

XFCE
http://www.xfce.org

Distributions

Debian GNU/Linux
http://www.debian.org

Fedora
http://fedora.redhat.com

Ubuntu Linux
http://www.ubuntu.com

Text Editors

Emacs
http://www.gnu.org/software/emacs/

Vim
http://www.vim.org/

Office Suites

Koffice
http://www.koffice.org/

OpenOffice.org
http://www.openoffice.org

Publishing

LaTeX
http://www.latex-project.org/

Scribus
http://www.scribus.net/

Graphics

Dia - Diagram editor
http://www.lysator.liu.se/~alla/dia/

GIMP
http://www.gimp.org/

ImageMagick - graphic tools
http://www.imagemagick.org

Inkscape - Vector graphics
http://inkscape.org/

SodiPodi - vector graphics
http://sodipodi.sourceforge.net/

Educational

Edubuntu
http://www.edubuntu.org

KDE educational packages
http://edu.kde.org/

Entertainment

GNOME games
http://www.gnome.org/

KDE games
http://games.kde.org/

Multimedia

Audacity - Audio editor
http://audacity.sourceforge.net/

Audio/video streaming
http://www.gstreamer.net/

Cinelerra - video production
http://heroinewarrior.com/cinelerra.php3

CinePaint - GIMP for films
http://filmgimp.sourceforge.net/

K3b - CD/DVD burning
http://k3b.sourceforge.net/

Mplayer - universal player
http://www.mplayerhq.hu/

TV Viewer
http://bytesex.org/xawtv/

XMMS - winamp clone
http://www.xmms.org/

Software Development

Anjuta
http://www.anjuta.org/

Eclipse
http://www.eclipse.org/

KDevelop
http://www.kdevelop.org/