WikiReader

Free Software and Free Contents

A collection of articles from Wikipedia, the free encyclopedia

Last change: June 28th, 2004

WIKIMEDIA
FOUNDATION

IMPRINT

Authors: The volunteer writers of the english Wikipedia

Editor: Thomas R. "TomK32" Koll

Notable Wikipedians for this WikiReader:

Used fonts: FreeSerif und FreeMono
Cover:

Last change in this edition: June 28th 2004 at 22:40 CEST

Webdress of Wikipedia: http://en.wikipedia.org

ISSN (Onlineedtion): 1613-7752
ISSN (Printedtion): not known yet

A complete list of used articles and the names of all registered authors who worked on these articles can be found in the appendix.

On Wikipedia

Wikipedia is a free encyclopedia which was started to give everyone a free source of knowledge which you not only can read but also extend in form of writing for it. On the website http://en.wikipedia.org you can not only find the current articles of Wikipedia but you can also start writing imediately without registration or identification. With this revolutionary method more than 700.000 articles were written since 2001 in more than 40 languages, and it's growing faster. In some languages Wikipedia is the first encyclopedia ever.

Since 2003 the Wikimedia Foundation is taking care of running the farm of webservers and also hosts and supports other projects like the multilinugual dictionary Wiktionary and the textbooks project WikiBooks.

On WikiReader

WikiReader is a randomly published series of collections of Wikipedia articles, a detailed overview over a certain topic presented in a editored form. It don't claim to be complete or even perfect but is more or less a "snapshot" of the topic. We encourage our readers to do further researches, include their results into Wikipedia and give new impluses for the next edition of this WikiReader.

The first WikiReader, on the topic "Schweden", was published at the German Wikipedia in February 2004 by Thomas Karcher and others followed.

On WikiReader this WikiReader

The "WikiReader Free Software and Free Contents" is the first one published in English, still by a German Wikipedian, but surely soon to be editored by native English speakers.

The WikiReader's target is to give a overview on the various aspects of free software and contents which started in the 1970s and became a counter-movement to propietary software and restrictive copyright in the late 1990s.

Copyright

Table of Contents

Definition of Terms

Open Content

Open content, coined by analogy with *open source,* describes any kind of creative work (for example, articles, pictures, audio, video, etc.) that is published under a copyright license, or in the public domain, in a format that explicitly allows the copying of the information. One example is the GNU Free Documentation License, which is used by Wikipedia and Nupedia. "Open content" is also sometimes used to describe content that can be modified by anyone. Of course, this is not without prior review by other participating parties - but there is no closed group like a commercial encyclopedia publisher which is responsible for all the editing.

Just as open source software is sometimes described simply as Free Software (not to be confused with Freeware), open content materials can be more briefly described as free materials. But not every open content is free in the GNU GPL sense (for instance the Open Directory). Some licenses attempt to maximize the freedom of all potential recipients in the future, while others maximize the freedom of the initial recipient.

Free software

The term **free software** is used in essentially two different ways:

1. Software that can be used, copied, studied and modified and redistributed by the user;
2. Software which may be copied and used without payment, also referred to as **freeware** (or **gratis software** by advocates of the first variety).

These definitions may conflict, and a piece of software that is free in the first sense may not be free in the second, and vice versa.

Amongst software developers and free and open source software enthusiasts, the first sense is traditionally called "free as in speech", while the second is called "free as in beer". In this context, the term "free software" more commonly refers to the first sense. Advocates of "free as in speech" software call the second type "gratis", which translates to the "free" of "free beer," because there is no charge to receive a copy.

In many languages the terms for free as in freedom and free as in "free beer" is different; in French, for example, "libre" translates to "free" in the sense of "freedom". Hence, free software of the "free speech" type is sometimes called "software libre", from the French *"logiciel libre"* and the Spanish *"software libre"*.

Free software of the first type is often made available online without charge, or available offline for the cost of distribution; however, this is not required, and free software can also be sold for profit.

Free software as in "Free Speech"

Developers in the 1970s frequently shared their software in a manner similar to the principles of free software. In the late 1970s, companies started routinely imposing restrictions on users with the use of license agreements. In 1984, Richard Stallman started working on the GNU project, founding the Free Software Foundation (FSF) one year later [1] (*http://www.gnu.org/fsf/fsf.html*).

Stallman introduced the concepts of "free software" and "copyleft", which he specifically devised to give users freedom and to restrain the possibilities for proprietisation [2] (*http://cisn.metu.edu.tr/2002-6/free.php*).

The FSF has produced a specific free software definition, by which software is "free" in this sense if it grants:

- the freedom to run the program for any purpose (freedom 0)
- the freedom to study and modify the program (freedom 1, which they state requires access to the program's source code)
- the freedom to copy the program so you can help your neighbor (freedom 2)
- the freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3)

A list of compliant licenses is available from FSF's web site (see below). The term "proprietary software" is used for software distributed under more restrictive licenses which do not grant these freedoms. Copyright law reserves most rights of modification, duplication and redistribution for the copyright owner; software released under a free software license specifically rescinds most of these reserved rights.

The FSF definition of free software does not touch on the issue of price; a commonly used slogan is "free as in speech, not as in beer", and it is common to see CDs of free software such as Linux distributions for sale. However, in this situation the buyer of the CD would have the right to copy and redistribute it. *Free beer* software can include restrictions that do not conform to the FSF definition — for example, gratis software may not include source code, may actively prohibit redistributors from charging a fee, *etc.*

To avoid confusion, some people use the words "libre" and "gratis" to avoid the ambiguity of the English word "free". However, these alternative terms are still used mostly within the free software movement and are only slowly spreading to the outside world. Others advocate the term open source software, but the relationship between "open source" and "free software" is complex.

There are several variations on free software in the FSF sense, for example:

- The freedoms defined by the FSF are protected through copyleft licenses, the most prominent of which is the GNU General Public License. The author retains copyright, and permits redistribution and modification under terms designed to ensure that all modified versions of the software remain under copyleft terms.
- Public domain software, in which the author has abandoned the copyright. Public-domain software, since it is not protected by copyright at all, may be freely incorporated into closed, proprietary works as well as free ones.
- BSD-style licenses, so called because they are applied to much of the software distributed with the BSD operating systems. The author under such licenses retains copyright protection solely to disclaim warranty and to require proper attribution of modified works, but permits redistribution and modification, even in proprietary works.

Note that the original copyright owner of copyleft-licensed software can also make a modified version under their original copyright, and sell it under any license they like, in addition to distributing the original version as free software. This technique has been used as a business model by a number of free software companies; this does *not* restrict any of the rights granted to the users of the copyleft version.

Examples and evolution

A large and increasing amount of software is made available under free software licenses; observers of this trend (and adherents) often refer to this phenomenon as the free software movement. Notable free software projects include the Linux and BSD operating system kernels, the GCC compiler, GDB debugger and C libraries, the BIND name server, the Sendmail mail transport server, the Apache web server, the MySQL and PostgreSQL relational database systems, the Perl, Python, Tcl and PHP programming languages, the X Window System, the GNOME and KDE desktop environments, the OpenOffice.org office suite, the Mozilla web browser, and the GIMP graphics editor.

Like all free software, these projects distribute their programs under licenses that grant users all the freedoms discussed above, but because of technicalities in the licenses, combining programs by mixing source code or directly linking binaries may be problematic unless both applications are under mutually compatible licenses. When programs are not directly linked together into a single program, these problems do not exist. Much free software can run on non-free platforms such as Microsoft Windows, and non-free software can be run on free platforms, although purists prefer to use all-free software running on a free platform such as Linux.

Free software packages constitute a software ecosystem where different pieces of software can provide services to one another, leading to co-evolution of features: in one simple example, the Python programming language provides support for the HTTP protocol, and the Apache web server that provides the HTTP protocol can call the Python programming language to serve dynamic content.

The Debian Project, which produces an operating system entirely composed of free software, created a set of guidelines that are used to evaluate the compatibility of a license with Debian's free-ness goal. The Debian Free Software Guidelines are used to delineate the *free* from *non-free* software. Debian had by 2003 collected over seven and a half thousand software packages compliant with the above guidelines.

Debian developers also argue that the same principles should apply not only to programs, but to software documentation as well. Many documents written by the Linux Documentation Project, and all documents licensed under the GNU Free Documentation License, do not comply with all of the above guidelines.

Comparison with Open Source software

The Open Source movement is philosophically distinct from the free software movement. It was created by a group of people, notably Eric S. Raymond and Bruce Perens, who formed the Open Source Initiative (OSI). They sought (1) to bring a higher profile to the practical benefits of sharing software source code, and (2) to interest major software houses and other high-tech industry companies in the concept. These advocates see the term *open source* as avoiding the ambiguity of the English word "free" in *free software*.

Many people recognise a qualitative benefit to the software development process when a program's source code can be used, modified and redistributed by developers. (*See also* The Cathedral and the Bazaar.) The free software movement places primary emphasis on the moral or ethical aspects of software, seeing technical excellence as a desirable by-product of its ethical standard. The Open Source movement sees technical excellence as the primary goal, regarding source code sharing as a means to an end. As such, the FSF distances itself both from the Open Source movement and from the term "Open Source".

Since the OSI only approves free software licenses as complying with the OSD, most people interpret it as a distribution scheme, and freely interchange "open source" with "free software". Even though there are important philosophical differences between the two terms, particularly in terms of the motivations for developing and using such software, they seldom make any impact in the collaboration process.

Whilst the term "Open Source" removes the ambiguity of Freedom versus Price, it introduces another: between programs that meet the Open Source Definition, giving users the freedom to improve upon them, and programs that simply have source available, possibly with heavy restrictions on the use of that source. Many people believe that any software that has source available is open source because they can tinker with it themselves. However, much of this software does not give its users the freedom to distribute their modifications, restricts commercial usage, or otherwise restricts users' rights.

Political significance

Once a free software product has started to circulate, it soon becomes available at little or no cost. At the same time, its utility does not decrease. This means that free software can be characterized as a pure public good rather than a private good.

Since free software allows free use, modification, and distribution, it often finds a home in third world countries for whom the cost of proprietary software is sometimes prohibitive. It is also easily modified locally, so translation efforts into languages which are not necessarily commercially profitable are also feasible. See also internationalization.

Most free software is produced by international teams cooperating through free association. Teams typically are composed of individuals with a wide variety of motivations. There are many stances about the relation of free software to the current, capitalist economic system:

- Some consider free software to be a competitor to capitalism.
- Some consider free software to be another form of competition within free markets, and that copyright is a governmental restriction on the market.
- Groups like Oekonux and Hipatia consider that everything could be produced in this manner and that this mode of production has the potential to supersede the capitalist mode of production.

Free software as in "no charge"

Various types of free software in this sense exists:

- Freeware, software that can be distributed and used without cost. Few strings are attached; sometimes only private, non-commercial use is allowed. The software may not be modified, and sometimes may also not be redistributed.
- Adware, software which displays advertisements during use. Legit adware is often a kind of shareware which may be used for free with ads, other adware is a kind of spyware which comes with advertising. This second kind is often installed without the consent of the installee.
- Spyware, collects market research data and/or credit card numbers from the host computer. Also often (if not always) installed without consent.
- Crippleware, software which can be used in a limited form for free; the enhanced version typically requires payment (see shareware).

The following are freely available to a greater or lesser degree, but are not properly "free software" in this sense:

- Shareware's license requires payment for use beyond a specified trial period. The payment typically has to be made by the user on an "honor system".

- Warez is current commercial software which is distributed for free by a third party in violation of its copyright license.
- Abandonware is software which is used and distributed in violation of copyright license, like warez, but is no longer sold or developed by its owner. Its copyright may or may not be enforced by the owner.

Open Source

Open source is a work methodology that fits the Open Source Definition, and generally is any computer software whose source code is either in the public domain or, more commonly, is copyrighted by one or more persons/entities and distributed under an open-source license such as the GNU General Public License (GPL). Such a license may require that the source code be distributed along with the software, and that the source code be freely modifiable, with at most minor restrictions, such as a requirement to preserve the authors' names and copyright statement in the code, a concept known as copyleft. In some cases, as with Apache or FreeBSD, there are only very minor conditions on use of modified versions. When used as an adjective, the term is hyphenated, e.g. "Apache is *open-source* software."

These are rights for users of the software. An open-source license itself does not necessarily require that the software, or its source, initially has to be freely (in both senses of the word) available on the Internet. Most popular open-source software is, however.

The term *open source* in common usage may also refer to any software with publicly available source code, regardless of its license, but this usage provokes strong disapproval from the open source community. Examples of such "disclosed source" software include some versions of Solaris and PGP. There are also shared source licenses which, on the surface, appear to be open source, but have critical differences.

Source Code

Source code (commonly just **source** or **code**) refers to any series of statements written in some human readable computer programming language. In modern programming languages, the source code which constitutes a software program is usually in several computer files, but the same source code may be printed in a book or recorded on tape (usually without a filesystem). The term is typically used in the context of a particular piece of computer software. A computer program's *source code* is the collection of files that can be converted from human-readable form to an equivalent computer-executable form. The source code is either converted into object code by an assembler or compiler for a particular computer architecture, or executed from the human readable form with the aid of an interpreter.

Purposes

Thus, source code is either used to produce object code, or to be run by an interpreter. Modifications are not carried out on object code, but on source code, and then converted again.

An other important purpose of source code is for the description of software. Also, source code has a number of other uses. It can be used as a tool of learning; beginning programmers often find it helpful to review existing source code to learn about programming techniques and methodology. It is also used as a communication tool between experienced programmers, due to its (ideally) concise and unambiguous nature. The sharing of source code between developers is frequently cited as a contributing factor to the maturation of their programming skills. Source code can be an expressive artistic medium; consider, for example, obfuscated code or PerlMonks.org (*http://www.perlmonks.org*).

Source code is a vital component in the activity of porting software to alternative computer platforms. Without the source code for a particular piece of software, portability is generally so difficult as to be impractical and even impossible. Programmers frequently borrow source code from one piece of software to use in other projects, a concept which is known as Software reusability.

Organization

The source code for a particular piece of software may be contained in a single file or many files. A program's source code is not necessarily all written in the same programming language; for example, it is common for a program to be written primarily in the C programming language, with some portions written in Assembly language for optimization purposes. It is also possible for some components of a piece of software to be written and compiled separately, in an arbitrary programming language, and later integrated into the software using a technique called library linking.

Moderately complex software customarily requires the compilation or assembly of several, sometimes dozens or even hundreds, of different source code files. This complexity is reduced considerably by the inclusion of a Makefile with the source code, which describes the relationships among the source code files, and contains information about how they are to be compiled. The Revision control system is another tool frequently used by developers for source code maintenance.

Licensing

Software, and its accompanying source code, typically falls within one of two licensing paradigms: Free software and Proprietary software. Generally speaking, software is *free* if the source code is freely available, and *proprietary* if the source code is kept secret,

or is privately owned and restricted. The provisions of the various copyright laws are often used for this purpose, though trade secrecy is also relied upon. For a further discussion of the differences between these paradigms, and the divisions within them, see software license.

Legal Issues

As of 2003, court systems are in the process of deciding whether source code should be considered a Constitutionally protected form of free speech in the United States. Proponents of the free speech argument claim that because source code conveys information to programmers, is written in a language, and can be used to share humour and other artistic pursuits, it is a protected form of communication. The opposing view is that source code is functional, more than artistic speech, and is thus not protected by First Amendment Rights of the U.S. Constitution.

One of the first court cases regarding the nature of source code as free speech involved University of California mathematics professor Dan Bernstein, who had published on the internet the source code for an encryption program that he created. At the time, encryption algorithms were classified as munitions by the United States government; exporting encryption to other countries was considered an issue of national security, and had to be approved by the State Department. The Electronic Frontier Foundation sued the U.S. government on Bernstein's behalf; the court ruled that source code was free speech, protected by the First Amendment.

In 2000, in a related court case, the issue was again brought under some scrutiny when the Motion Picture Association of America (MPAA) sued the 'hacker' magazine 2600 and a number of other websites for distributing the source code to DeCSS, an algorithm capable of decrypting scrambled DVD discs. The algorithm was developed to allow people to play legally purchased DVDs on the Linux operating system, which had no DVD software at the time. The US District court decision favored the MPAA; 2600 magazine was prohibited from posting or linking to the source code on their website. This ruling was widely considered a victory for the supporters of the Digital Millennium Copyright Act, as it established a legal precedent for the notion that source code is not Constitutionally protected free speech. It was affirmed by the Appeals Court and as of late 2003 is being appealed to the US Supreme Court.

Copyleft

Copyleft is the application of copyright law to force derivative works to also be released with a copyleft license. So long as all of those wanting to modify the work accept the terms, the net effect is to facilitate successive improvement by a wide range of contributors. Those who are unwilling or unable to accept the terms are prohibited from creating derivative works.

No restrictions apply to works in the public domain. They may be freely modified, and the creator of the derivative work may license any new portions of the derivative work, but not the public domain portion, under any terms, or none. The resulting derivative work may not be available to the creators of the original or may compete with them.

In copyleft, the copyright holder grants an irrevocable license to the recipient of a copy, generally permitting the free unlimited use, modification and redistribution (often including sale of media or auxiliary materials which may carry a different copyright license (e.g. documentation)) of copies. The distinctive condition to that license is that any modifications to the work, if redistributed, must carry the same permissions (i.e. license terms) and be made available in a form which facilitates modification. For software, this means in source code.

The concept of copyleft arose when Richard Stallman was working on a Lisp interpreter. Symbolics asked to use the Lisp interpreter, and Stallman agreed to supply them with a public domain version of his work. Symbolics extended and improved the Lisp interpreter, but when Stallman wanted access to the improvements that Symbolics had made to his interpreter, Symbolics refused. Stallman then, in 1984, proceeded to create a software license that would prevent this behavior which he named software hoarding. The term "copyleft" came from a message contained in Tiny BASIC, a free distributed version of Basic written by Dr. Wang in the late 1970s. The program listing contained the phrases "All Wrongs reserved" and "CopyLeft."

There are definitional problems with the term "copyleft" which contribute to controversy over it. The term originated as an amusing backformation from the term 'copyright', and was originally a noun, meaning the copyright license terms of the GNU General Public License originated by Richard Stallman as part of the Free Software Foundation's work. Thus, 'your program is covered by the copyleft'. When used as a verb (i.e. 'he copylefted his most recent version'), it is less precise and can refer to any of several similar licenses, or indeed to a notional imaginary license for discussion purposes.

Because of complications caused by use of software library routines, there developed the GNU Library General Public License (subsequently renamed the Lesser GPL), which changes the requirement of further distribution in ways which are compatible with actual library routine use.

Copyleft is one of the key features in free software/open source licences, and is the licenses' legal framework to ensure that derivatives of the licensed work stay free/open. If the licensee fails to distribute derivative works under the same license he will face legal consequences - the license is terminated, leaving the licensee without permission to copy, distribute, display publicly, or prepare derivative works of the software.

Other free software licenses, such as those used by the BSD operating systems, the X Window System and the Apache web server, are not copyleft licenses because they do not require the licensee to distribute derivative works under the same license. There is an ongoing debate as to which class of license provides a larger degree of freedom. This debate hinges on complex issues such as the definition of freedom and whose freedoms

are more important. It is sometimes argued that the copyleft licenses attempt to maximize the freedom of all potential recipients in the future (*freedom from* the creation of proprietary software), while non-copyleft free software licenses maximize the freedom of the initial recipient (*freedom to* create proprietary software).

An example of a free software license that uses strong copyleft is the GNU General Public License. Free software licenses that use weak copyleft include the GNU Lesser General Public License and the Mozilla Public License. Examples of non-copyleft free software licenses include the Q Public License, the X11 license, and the BSD licenses.

Copyleft licenses for materials other than software include the Creative Commons ShareAlike licenses and the GNU Free Documentation License. The latter is being used for the content of Wikipedia. The Free Art license is a license that can be applied to any work of art.

Copyleft licenses are sometimes called **viral copyright licenses** , mainy by people who stand to lose much money from them, because any works derived from a copylefted work must themselves be copylefted. The term "viral" implies propagation like that of a biological virus through an entire organ of similar cells or species of similar bodies. In context of legally binding contracts and licenses, "viral" refers to anything, especially anything memetic, that propagates itself by attaching itself to something else, regardless of whether the viral assertions themselves add value to the individual work. The viral metaphor is over-used but is reasonable to help distinguish between free software and open source in software and documentation projects. Most advocates of copyleft argue that the analogy between copyleft and computer viruses does not apply. As they point out, computer viruses generally infect computers without the awareness of the user, whereas the copyleft actually grants the user certain permissions to distribute modified programs, which is not allowed under copyright law without permission of the copyright holder. Most proprietary software licenses do not allow such distribution. Furthermore, copyright itself is "viral" in this sense, since any works derived from a copyrighted work must have permission from and obey any conditions set by the original copyright holder.

The view that copyleft licenses are viral is supported by Microsoft, who say that if a product uses GPLed code, that product automatically escapes the creator's control and becomes GPLed, leaving the creator no recourse. Obviously, working for a software company will have a like effect. Advocates, including Eben Moglen, Professor of Law at Columbia University and counsel for the Free Software Foundation, note that this is not true since the GPL is a license, not a contract.

Microsoft, and others, in describing the GPL as a "viral license", may also be referring to the idea that any release of something new under the GPL would seem to create a positive feedback network effect, in which over time there will be an ever-expanding amount of copylefted code. Code reuse is of course tempting, as a way to save effort and get on with a project, especially when a perfectly sensible design and implementati-

on has already been done and is available. In contrast, those working on non-copylefted programs will have to "reinvent the wheel" for their own programs.

Copyleft licenses are desirable and popular for shared works precisely because they *are* viral, and apply to all derivative works, which are thus "infected" by the requirement to re-integrate changes deemed desirable by any party down the line. This guarantee is important because it ensures uniform license terms and free access, and makes copyleft projects resistant to unnecessary forking because all maintainers, of the original work or other versions, may use any modifications released by anyone. Useful changes tend to be merged, and different versions are maintained only to the extent that they are useful. Without the "viral" license, variant terms can apply to the forks, derivative works can be controlled commercially by the parties that extend or translate them, and the project would degrade to an open source one. It is thought that Linux has not suffered the same fragmentation as Unix because it is copylefted.

Copyleft-like ideas are increasingly being suggested for patents, such as open patent pools that allow royalty-free use of patents contributed to the pool under certain conditions (such as surrendering the right to apply for new patents that are not contributed to the pool).

Copyleft is also starting to inspire the arts with movements like the Libre Society and open-source record labels emerging.

Open Patent

The **open patent** movement seeks to build a portfolio of patented inventions that can freely be distributed under a copyleft-like license. These works could be used as is, or improved, in which case the patent improvement would have to be re-licensed to the institution that holds the original patent, and from which the original work was licensed. This frees all users who have accepted the license from the threat of lawsuits for patent infringement, in exchange for their surrendering the right to build up new patents of their own (in the specific domain for which the original license applies).

The open patent idea is actually quite old and has traditionally been practiced by consortia of research-oriented companies, and increasingly by standards bodies. These also commonly use open trademark methods to ensure some compliance with a suite of compatibility tests, e.g. Java, X/Open both of which forbid use of the mark by the non-compliant.

Thus the model already has a strong legal framework. Patent improvement licensing is already practiced by some global institutions, notably the government of China and MIT. Each of these manage a large patent portfolio and often require improvements to be licensed back as part of the original portfolio, although this is not the default license.

Critics question whether the promoters of truly 'open' and mandatory improvement licensing, having spent most of their lives opposed to software patents, can actually attract donors of patents, or would actually participate in a process that they claim to despise. This criticism probably focuses unduly on the personality and ideology of Richard Stallman, who has nonetheless sought to solicit donors for such schemes. He found, not surprisingly, that software patent holders were not so interested in talking to him.

However, despite confrontation between Stallman/GNU and the patent system, the open patent movement got going and attracted some support. It remains to be seen if it can become a major phenomenon - patents are difficult and costly to obtain and require extensive documentation, unlike copyrights. Patent rights on software and on life forms are controversial and many activists believe that they can successfully prevent such patent rights from existing at all, and so would be less inclined to patent and contribute to any such portfolio.

See also *http://www.openpatents.org/*

Open Hardware

Open Hardware (OH) is a part of the GNU project in which hardware designers share their work by disclosing the schematics and software (GNU drivers) used in their designs.

Open Hardware designers meet, discuss what they are doing and ask each other for assistance in finding parts, or seek ideas to solve design problems. OH is also an opportunity to exhibit designs, so some may learn from what others have done.

With the rise of reconfigurable logic devices, the sharing of logic designs is also a form of Open Hardware. Instead of sharing the schematics, HDL code is shared. This is different from Open Software. HDL descriptions are commonly used to set up SoC systems either in FPGAs or directly in ASIC designs. HDL modules, when distributed, are called "cores" or "IP" (intellectual property).

External Links

- Openhardware *http://www.openhardware.net*
- Debian Open Hardware *http://opencollector.org/Whyfree/open_hardware.html*
- Embedded Linux/Microcontroller Project *http://www.uclinux.org*
- OpenCores *http://www.opencores.org/*

The Cathedral and the Bazaar

The Cathedral and the Bazaar is an essay by **Eric S. Raymond** on open-source software engineering methods, based on his observations of the Linux kernel development process and his experiences managing an open source project, fetchmail. It was first presented by the author at the Linux Kongress on 27 May 1997.

The essay contrasts two different free software development models:

- The *Cathedral* model, in which source code is available with each software release, but code developed between releases is restricted to an exclusive group of developers. GNU Emacs and GCC are presented as examples.
- The *Bazaar* model, in which the code is developed over the Internet in view of the public. Raymond credits Linus Torvalds, leader of the Linux kernel project, as the inventor of this process. He also provides anecdotal accounts of his implementation of this model for the fetchmail project.

The essay's central thesis is Raymond's proposition that **Given enough eyeballs, all bugs are shallow** (which he terms Linus's law): if the source code is available for public testing, scrutiny and experimentation, bugs will be discovered at a rapid rate. In contrast, Raymond claims that an inordinate amount of time and energy must be spent hunting for bugs in the Cathedral model, since the code is available only to a few developers.

The essay helped convince most existing open source and free software projects to adopt Bazaar-style open development models, fully or partially — including GNU Emacs and GCC, the original Cathedral examples. Most famously, it also provided the final push for Netscape to open the source of Netscape Communicator and start the Mozilla project.

The Cathedral is also the typical development model for proprietary software — with the additional restriction in that case that source code is usually not provided even with releases — and a common usage of the phrase *"the Cathedral and the Bazaar"* is to contrast proprietary with open source. However, the original essay concerns itself only with free software, and does not address proprietary development in any way at all.

The terminology has been extended to describe non-software projects. Wikipedia is a Bazaar-style project, while Nupedia and the Encyclopædia Britannica are Cathedral-style projects.

Eric S. Raymond: *The Cathedral and the Bazaar* (O'Reilly, January 2001; paperback ISBN 0596001088) — includes *"The Cathedral and the Bazaar", "Homesteading the Noosphere", "The Magic Cauldron"* and *"Revenge of the Hackers"*

Website: *http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/*

SOFTWARE LICENSES

OPEN-SOURCE LICENSES

An **open-source license** is a copyright license for computer software that follows the principles of the open source movement. More formally, a license is considered open-source when it has approved by the Open Source Initiative, with the criteria being the Open Source Definition. Software in the public domain (that is, with no copyright license at all), meets those criteria as long as all source code is made available, and is therefore recognized by the OSI and entitled to use their service mark.

In addition, OSI has approved the following licenses as of 2003:

- Academic Free License
- Apache Software License
- Apple Public Source License
- Artistic license
- Common Public License
- Eiffel Forum License
- BSD License
- GNU General Public License (GPL)
- GNU Lesser General Public License (LGPL)
- Historical Permission Notice and Disclaimer
- IBM Public License
- Intel Open Source License
- Jabber Open Source License
- MIT License
- MITRE Collaborative Virtual Workspace License (CVW License)
- Motosoto License
- Mozilla Public License 1.0 (MPL)
- Mozilla Public License 1.1 (MPL 1.1)
- NetHack General Public License
- Nokia Open Source License
- Open Software License
- Open Group Test Suite License
- Python license
- Python Software Foundation License
- Q Public License (QPL)
- Ricoh Source Code Public License
- Sleepycat License
- Sun Industry Standards Source License (SISSL)
- Sun Public License
- Vovida Software License v. 1.0
- W3C License
- X.Net License
- zlib-libpng license
- Zope Public License

FREE SOFTWARE LICENSES

It should be noted that the Free Software Foundation has different criteria for evaluating whether or not a license qualifies a program as free software. See Free software license.

Generally speaking, **free software license** is a phrase used by the free software movement to mean any software license that grants users of the software the following four freedoms:

1. The freedom to run the program for any purpose
2. The freedom to study and modify the program
3. The freedom to copy the program
4. The freedom to redistribute modified versions of the program

A license which preserves those freedoms for modified works is a copyleft license. See Free software movement for more information.

The Free Software Foundation maintains a list of free software licenses at their web site. The list distinguishes between free software licenses that are compatible or incompatible with the FSF license of choice, the GNU General Public License, which is a copyleft license. The list also contains licenses which the FSF considers non-free for various reasons. The list, which differs slightly from the open source license list, can be found at *http://www.gnu.org/philosophy/license-list.html*

MIT LICENSE

The **MIT License** is an agreement for the use of certain types of computer software.

It is most similar to the 3-clause BSD license, which is essentially different only in the fact that it contains a notice prohibiting the use of the name of the copyright holder in promotion. The 4-clause BSD license also includes a clause requiring all advertising of the software to display a notice; the MIT License has never had this clause. The MIT license, however, more explicitly states the rights given to the end-user, including the right to use, copy, modify, merge, publish, distribute, sublicense, and/or sell the software.

A 2-clause BSD-style license, found in software such as Apple Computer's WebCore is, in practicality, the same as the MIT License, as it does not contain the "promotion" clause.

According to the Free Software Foundation's license list, the MIT license is more accurately called the X11 license, because MIT has many licenses for software. However, the Open Source Initiative refers to it as the MIT License, as do many other groups.

Many groups use the MIT license for their own software; examples of these include expat, MetaKit, XFree86, and X11.

Because the MIT License is not copyrighted, other groups can elect to modify the MIT License to suit their own needs. For example, the Free Software Foundation uses a license identical to the MIT License for its ncurses library, except for the addition of this text:

```
Except as contained in this notice, the name(s) of
the above copyright holders shall not be used in ad-
vertising or otherwise to promote the sale, use or
other dealings in this Software without prior
written authorization.
```

Adding text like this makes the license almost identical to the BSD license.

Still other groups prefer to dual-license their products under the MIT license; an example of this is older versions of the cURL library, which allowed you to choose either the Mozilla Public License or the MIT License.

http://www.opensource.org/licenses/mit-license.php - The MIT License template

BSD LICENSE

The **BSD license** is the license agreement that the BSD software (largely, a version of UNIX) is distributed under. The owner of the original BSD distribution was the "Regents of the University of California". This is because BSD originally came from the University of California, Berkeley.

Versions of the current BSD template (and the older version with the "advertising clause", *see below*) are often used by other organizations.

The BSD License does not prohibit the use of the material licensed in products for resale. A notable example of such use is the use of BSD networking code in Microsoft products, or the use of numerous FreeBSD components in MacOS X.

It is possible for something to be distributed with the BSD License and some other license to apply as well. This was in fact the case with very early versions of BSD Unix itself, which included proprietary material from AT&T.

As originally distributed the license had an extra clause, the so called **advertising clause**:

```
* 3. All advertising materials mentioning features or use of this software
*    must display the following acknowledgement:
*    This product includes software developed by the University of
*    California, Berkeley and its contributors.
```

The GNU project referred to it as the "obnoxious BSD advertising clause". Along with offending people, the clause caused a practical problem. People who made changes to the source code tended to want to have their names added to the acknowledgement. With large numbers of people working on a single project (or for many separate projects in a software distribution), the advertising clause quickly created large and unwieldy acknowledgements. Another practical problem was legal incompatibility with the terms of the GNU General Public License (which does not allow the addition of restrictions beyond those it already imposes), forcing a segregation of GNU and BSD software. The GNU project went so far as to suggest people not use the phrase "BSD-style" licensing when they wanted to refer to an example of a non-copyleft license, in order to prevent inadvertent usage of the original BSD license.

On July 22, 1999, William Hoskins, the director of the office of technology licensing for Berkeley, revoked the clause. The document enacting that revocation is available at *ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change.* The original license is now sometimes called "BSD-old" or "4-clause BSD", with the revised license sometimes called "BSD-new", "revised BSD", or "3-clause BSD". More often than not however, the revised license is called the "modern BSDL," or simply, the "BSDL."

A 2-clause BSD-like license also exists; the clause deletes the third section, which prohibits use of the name of the copyright holder for endorsement purposes.

BSD AND GPL LICENSING

Two of the most common free software licenses are the **BSD and GPL licenses**. There has been continuing discussion over the relative merits of the use of either license in free software projects.

The BSD license essentially allows the user of source code released under this license to be free to do whatever that user wishes to do with the code, with as few restrictions as possible. This means that code released under this license can be used in both open source *and* closed source situations. Proponents of copyleft-style software licenses such as the GPL argue that the non-copyleft nature of the BSD license becomes detrimental to open source in general, as it does not expressly request of a user, who wishes to extend BSD-licensed software, to release openly that user's extensions. However, it is argued that the non-copyleft nature of the BSD license encourages inclusion of well-developed standard code into closed source projects, and it is further argued that licenses such as the GPL discourages this, forcing people who wish to develop closed source projects to reinvent the wheel to maintain their ability to keep the project closed source, doing it in possibly a less-efficient way than the open source version does it (as it has been open to more scrutiny and patching).

Those who advocate the use of a copyleft license such as the GPL argue that the requirement that software licensed under the GPL allows the freedom to to copy, use, study,

modify, and distribute the source code, with the advantage that other improvements to GPL-licensed software are ensured with its requirement that the extensions to the software be freely available as well (however, it is said by some that the requirement that the source code be also freely available does not give the user "freedom" to make closed source software under such a license). GPL supporters often point out that since BSD licensed code allows distribution of closed source modified versions, that the work of the contributor has been somehow "stolen", and as such is not fair and therefore no code should be licensed under BSD-style licenses -- however, it still remains that the orginal free, BSD licensed code is still available for all to use and improve, with the copyrights still remaining to the author and that nothing has been lost.

Traditionally, Linux associated software is licensed under the GPL, whilst BSD derivatives often use the BSD license. Code licensed under the BSD license can be relicensed under the GPL (the BSD license is said to be "GPL-compatible"), but code under the GPL cannot be relicensed under the BSD license as the BSD license does not necessarily require the source code to be again freely available.

Content Licenses

Public Domain

Internationally, the **public domain** is the body of creative works and other knowledge —writing, artwork, music, science, inventions, and others—in which no person or organization has any proprietary interest. (Proprietary interest is typically represented by a government-granted monopoly such as a copyright or patent.) Such works and inventions are considered part of the public's cultural heritage, and anyone can use and build upon them without restriction (not taking into account laws concerning safety, export, etc.).

While copyright was created to temporarily enclose creative work as a means to encourage more creative work, works in the public domain just exist as such, and the public have the right to use and reuse the creative work of others without financial or social burden.

Without some kind of grant of monopoly rights—so-called "intellectual property rights"—all works belong to the public domain. When copyright or other protections reach the end of their life, works are said to revert to the public domain.

Absence of legal protection

Creative works are in the public domain wherever no law exists to establish proprietary rights, or where the subject matter is specifically excluded from existing laws. For example, most mathematical formulas are not subject to copyrights or patents in most of the world (although their application in the form of computer programs can be patented). Likewise, works that were created long before such laws were passed are part of the public domain, such as the works of William Shakespeare and Ludwig van Beethoven and the inventions of Archimedes (however, *translations* of the works of Archimedes, Shakespeare, etc., may be subject to copyright). Also, works of the United States Government are excluded from copyright law.

Expiration

Most copyrights and patents have a finite term; when this expires, the work or invention falls into the public domain. In most of the world, patents expire 20 years after they are filed. Trademarks expire soon after the mark becomes a generic term. Copyrights are more complex; generally, they expire in all countries except Guatemala, Mexico, Samoa and Colombia when all of the following conditions are satisfied:

- The work was created and first published before 1 January 1923, or at least 95 years before January 1 of the current year, whichever is later.
- The last surviving author died at least 70 years before January 1 of the current year.
- No Berne Convention signatory has passed a perpetual copyright on the work.
- Neither the United States nor the European Union has passed a copyright term extension since these conditions were last updated. (This must be a condition because the exact numbers in the other conditions depend on the state of the law *at any given moment*.)

These conditions are based on the intersection of United States and European Union copyright law, which most other Berne Convention signatories recognize. Note that copyright term extension under U.S. tradition does not *restore* copyright to public domain works (hence the 1923 date), but European tradition does because the EU harmonization was based on the copyright term in Germany, which had already been extended to life plus 70. Note further that works created by a United States government agency fall into public domain at the moment of creation.

The situation with respect to British government works is a little more complex, but still relatively easy to understand. British government works are restricted by either Crown Copyright or Parliamentary Copyright. Published Crown Copyright works become public domain at the end of the year 50 years after they were published, unless the author of the work held copyright and assigned it to the Crown. In that case, the copyright term is the usual life of author plus 70 years. Unpublished Crown Copyright documents become public domain at the end of the year 125 years after they were first created. However, under the legislation that created this rule, and abolished the traditional common law perpetual copyright protection of unpublished works, no unpublished works will become public domain until 50 years after the legislation came into effect. Since the legislation became law on 1 August 1989, no unpublished works will become public domain under this provision until 2039. Parliamentary Copyright documents become public domain at the end of the year 50 years after they were published. Crown Copyright is waived on some government works provided that certain conditions are met.

These numbers reflect the most recent extensions of copyright in the United States and Europe. Canada and Australia have not as of 2004 passed similar twenty-year extensions. Consequently, their copyright expiry times are still life of the author plus 50 years. As a result, characters such as Mickey Mouse, and works ranging from Peter Pan to the stories of H. P. Lovecraft are public domain in both places. (The copyright status of Lovecraft's work is debatable, as no copyright renewals, which were necessary under the laws of that time, have been found. Also, two competing parties have independently claimed copyright ownership on his work.)

As with most other British Commonwealth countries, Canada and Australia follow the general lead of the United Kingdom on copyright of government works. Both have a version of Crown Copyright which lasts for 50 years from publication. New Zealand also has Crown Copyright protection, but has a much greater time length of protection

at 100 years from the date of publication. Ireland also has a fifty year term on government works, although since it is no longer a monarchy, such protection is, of course, not called Crown Copyright. India has a government copyright of sixty years from publication, to coincide with its somewhat unusual life of the author plus sixty years term of copyright.

Examples of inventions whose patents have expired include the inventions of Thomas Edison. Examples of works whose copyrights have expired include the works of Carlo Collodi and most of the works of Mark Twain. Examples of works under a statutory perpetual copyright include many of the *Peter Pan* works by J. M. Barrie; this was granted by the British government and applies only within the United Kingdom. Other works, such as the works of The Walt Disney Company are not under a *de jure* statutory perpetual copyright because the United States Constitution requires copyrights to last "for limited Times" (Article I, section 8, clause 8). However, the limits have been retroactively extended several times, leading to longer and longer protections. Critics have observed that the extensions have taken place right before noteworthy works from Disney and others were about to expire, concluding that such copyright term extensions add up to *de facto* perpetual copyright. Disney and other large publishers routinely provide millions of U.S. dollars in campaign money to legislators, allegedly in exchange for these continued extensions.

Disclaimer of interest

In the past, in some jurisdictions such as the USA, a work would enter the public domain with respect to copyright if it was released without a copyright notice. This is no longer the case. Any work receives copyright by default and copyright law generally doesn't provide any special means to "abandon" copyright so that a work can enter the public domain (in the USA, the Computer Software Rental Amendments Act of 1990 provides a registration mechanism for public domain computer programs at the Library of Congress, but it is still not explained how the work should be placed in the public domain in the first place).

A copyright holder can explicitly disclaim any proprietary interest in the work, effectively granting it to the public domain, by providing a licence to this effect. A suitable licence will grant permission for all of the acts which are restricted by copyright law.

With regards to patents on the other hand, publishing the details of an invention before applying for a patent will generally place an invention in the public domain and prevent its subsequent patenting by others. For example, once a journal publishes a mathematical formula, it may no longer be used as the core of a claim in a software patent. There is an exception to this, however: in US (not European) law, an inventor may file a patent claim up to one year after publishing it (but not, of course, if someone else published it first).

Ineligibility

Laws may make some types of works and inventions ineligible for monopoly; such works immediately enter the public domain upon publication. For example, US copyright law, 17 U.S.C. § 105, releases all works created by the US government into the public domain, patent applications as part of the terms of granting the patent to the invention are public domain, patent law excludes inventions that obviously follow from prior art, and agreements that Germany signed at the end of World War I released such trademarks as "aspirin" and "heroin" into the public domain in many areas.

Licensing

Note that there are many works that are not part of the public domain, but for which the owner of some proprietary rights has chosen not to enforce those rights, or to grant some subset of those rights to the public. See, for example, the Free Software Foundation which creates copyrighted software and licenses it without charge to the public for most uses under a class of license called "copyleft", forbidding only proprietary redistribution. See also Wikipedia, which does much the same thing with its content under the GNU Free Documentation License. Sometimes such work is mistakenly referred to as "public domain" in colloquial speech.

Note also that while some works (especially musical works) may be in the public domain, U.S. law considers transcriptions or performances of those works to be derivative works, potentially subject to their own copyrights.

The role in society

"Public access to literature, art, music, and film is esssential to preserving and building on our cultural heritage. Many of the most important works of American culture have drawn upon the creative potential of the public domain. Frank Capra's *It's a Wonderful Life* is a classic example of a film that did not enjoy popular success until it entered the public domain. Other icons such as Snow White, Pinocchio, Santa Claus and Uncle Sam grew out of public domain figures." ([1] (*http://www.creativecommons.org*)

Public domain and the Internet

Historically, the vast majority of copyright and other licensing issues arising from misunderstandings about the legal definition of "public domain" fell into two camps:

1. Businesses and organizations who could devote staff to resolving legal conflicts through negotiation and the court system.

2. Individual and organizational use of materials covered by the fair use doctrine, reducing the need for substantial governmental or corporate resources to track down individual offenders.

With the advent of the Internet, however, it became possible for anybody with access to this worldwide network to "post" copyrighted or otherwise-licensed materials freely and easily. This aggravated an already established but false belief that, if something is available through a free source, it must be public domain. Worse yet, once such material was available on the net, it could be perfectly copied among thousands or even millions of computers very quickly and essentially without cost.

These factors have reinforced the false notion that "freely obtained" means "public domain". One could argue that the Internet is a publicly-available domain, not licensed or controlled by any individual, company, or government; therefore, everything on the Internet is public domain. This specious argument ignores the fact that licensing rights are *not* dependent on the means of distribution or consumer acquisition. (If someone gives you stolen merchandise, it is still stolen, even if you weren't aware of it.) Chasing down copyright violations based on the erroneous idea that "information is free" (see Footnotes below) has become a primary focus of industries whose financial structure is based on their control of the distribution of such media. Though this is legally correct, public support for these companies' efforts is significantly undermined by the belief that they are receiving their "just desserts" for decades of price-gouging for licensed media. Ironically, this puts many creators of such work, like musicians and authors, on both sides of the issue, since they have frequently fought media distributors over inadequate compensation for their work, but depend on distributors' revenues for that compensation.

Another complication is that publishing exclusively on the Internet has becoming extremely popular. According to US law, at least, an author's original works are covered by copyright, even without a formal notice incorporated into the work. But such laws were passed at a time when the focus was on materials that could not be as easily and cheaply reproduced as digital media, nor did they comprehend the ultimate impossibility of determining which set of electronic bits is original. Technically, any Internet posting (such as blogs or emails) could be considered protected material unless explicitly stated otherwise. (Many Internet content providers attempt to assert copyrights by claiming all ownership and reproduction rights to any material posted to their servers, but the potential for conflicting claims has not been adequately tested.) Traditional methods of proving original work, such as physically mailing a sealed copy of one's work to oneself, thereby gaining a dated stamp from a governmental agency (i.e., the local Postal Service), are irrelevant for this new source of creative work.

FOOTNOTES

In response to the frequently-championed concept that "information is free", technology columnist Nicholas Petreley once wrote, "Those who want information to be free as a

matter of principle should create some information and make it free." This statement concisely illustrates the conflict between the cultural desire to make original material readily and cheaply (or freely) available and the right of original-work creators to receive compensation for their work.

External links

- Chris Sprigman's article *THE MOUSE THAT ATE THE PUBLIC DOMAIN: Disney, The Copyright Term Extension Act, And Eldred v. Ashcroft* (*http://writ.news.findlaw.com/commentary/20020305_sprigman.html*)
- Copyright Research and Information center (*http://www.cric.or.jp/cric_e/index.html*) - about the copyright law in Japan
- MPEG video recordings of panel discussions from the Conference on the Public Domain (2001) (*http://www.law.duke.edu/pd/mpegcast.html*) panelists include Eben Moglen, Robin Gross and Lawrence Lessig
- Short list of uncopyrightable things in the US (*http://www.copyright.gov/circs/circ1.html#wnp*)
- Summary list of copyright terms in other countries (*http://onlinebooks.library.upenn.edu/okbooks.html#whatpd*)
- Union for the Public Domain (*http://www.public-domain.org*)
- When U.S. works pass into the public domain (*http://www.unc.edu/~unclng/public-d.htm*).

GNU Free Documentation License

The **GNU Free Documentation License** (GFDL) is a copyleft license for free content, designed by the Free Software Foundation (FSF) for the GNU project. The official text of version 1.2 of the license text can be found at http://www.gnu.org/copyleft/fdl.html.

The license is designed for software documentation and other reference and instructional materials. It stipulates that any copy of the material, even if modified, carry the same license. Those copies may be sold but, if produced in quantity, have to be made available in a format which facilitates further editing. Wikipedia is the largest documentation project to use this license.

Many people and groups, notably the Debian project (based on their Debian Free Software Guidelines), consider the GFDL a non-free license, due to both the usage of "invariant" text that cannot be modified or removed, and the well-meaning but overly-broad prohibition against DRM systems which affects valid usages as well. See the documents in the "External links" section for more information.

Secondary Sections

The license explicitly separates any kind of "Document" from "Secondary Sections", which may not be integrated with the Document, but exist as front-matter materials or appendices. Secondary sections can contain information regarding the author's or publisher's relationship to the subject matter, but not any subject matter itself. While the Document itself is wholly editable, and is essentially covered by a license equivalent to (but bothways incompatible with) the GNU General Public License, some of the secondary sections have various restrictions designed primarily to deal with proper attribution to previous authors.

Specifically, the authors of prior versions have to be acknowledged and certain "invariant sections" specified by the original author and dealing with his or her relationship to the subject matter may not be changed. If the material is modified, its title has to be changed (unless the prior authors give permission to retain the title). The license also has provisions for the handling of front-cover and back-cover texts of books, as well as for "History", "Acknowledgements", "Dedications" and "Endorsements" sections.

Wikipedia and GFDL

All Wikipedia articles are licensed to the public under the GNU Free Documentation License. See Wikipedia:Copyrights for the details. The local copy of the license, as required by the terms of the GFDL, is at Wikipedia:Text of the GNU Free Documentation License.

Materials for which commercial redistribution is prohibited

Materials for which commercial redistribution is prohibited generally cannot be used in a GFDL-licensed document, e.g. a Wikipedia article, because the license does not exclude commercial re-use. However in some specific cases, commercial re-uses may be fair use and in that case such materials do not need to be licensed to fall within the GFDL if such fair use is covered by all potential subsequent uses. One good example of such liberal and commercial fair use is parody.

Creative Commons License

The **Creative Commons License** refers to the name of several copyright licenses released on December 16, 2002 by *Creative Commons*, a US nonprofit corporation founded in 2001.

There are four key license terms, in brief:

- **Attribution** (by): Permit others to copy, distribute, display, and perform the work and derivative works based upon it only if they give you credit.
- **Noncommercial** (nc): Permit others to copy, distribute, display, and perform the work and derivative works based upon it only for noncommercial purposes.
- **No Derivative Works** (nd): Permit others to copy, distribute, display and perform only verbatim copies of the work, not derivative works based upon it.
- **Share Alike** (sa): Permit others to distribute derivative works only under a license identical to the license that governs your work. (See also copyleft).

Some combinations of the above restrictions is possible, e.g. "CC by-sa" (Creative Commons Attribution-ShareAlike)

(Portions of this article are taken from the Creative Commons website, published under the Creative Commons Attribution License v1.0)

Non-Free Licences

Copyright

A **copyright** provides its holder the right to restrict unauthorized copying and reproduction of an original expression (i.e. literary work, movie, music, painting, software, mask work, etc.) Copyright stands in contrast to other forms of intellectual property, such as patents, which grant a monopoly right to the use of an invention, because it is not a monopoly right to do something, merely a right to prevent others doing it.

Background

Rights of copyright holder

A copyright holder typically has exclusive rights:

- to make and sell copies of the work (including, typically, electronic copies)
- to import or export the work
- to make derivative works
- to publicly perform the work
- to sell or assign these rights to others

What is meant by the phrase "exclusive right" is that the copyright holder and *only* the copyright holder is allowed to do these things; everyone else is prohibited from doing them without the copyright holder's consent. Copyright is often called a "negative right", to stress that it has less to do with permitting people (e.g. authors) to do anything, and more to do with prohibiting people (e.g. readers, viewers, or listeners) from doing something: reproducing the copyrighted work. In this way it is similar to the Unregistered Design Right in English Law and European Law.

Transfer of rights

Copyrights may be granted, sold, or relinquished. Very often, a copyright holder will, by contract, transfer his copyrights to a corporation. For example, a writer who writes a novel will sign a publication agreement with a company such as Random House in which the writer agrees to transfer all copyrights to Random House in exchange for royalties and other terms. One might ask why a copyright holder would ever give up his rights. The answer is that large companies such as Random House generally have production and marketing capabilities far beyond that of the author. In the digital age of music, music may be copied and distributed for a minimal cost through the Internet, but

record labels attempt to provide the service of promoting and marketing the artist so that his work can reach a much larger audience. A copyright holder does not have to transfer all rights completely. Some of the rights may be transferred, or else the copyright holder may grant another party a non-exclusive license to copy and/or distribute the work in a particular region.

Idea-expression dichotomy and the Merger Doctrine

The **idea-expression divide** is a concept in copyright law which states that copyright does not protect ideas, information or function, but only the form of expression of ideas. Only the way in which something has been expressed is protectable by copyright.

For example, if a book is written describing a new way to organize books in library, a reader can freely use that method without being sued, but what is written in the book, the original expression of the idea, may not be copied. One might be able to obtain a patent for the method, but that is a different body of law. If there is art on the cover of the book, that art may also be copyrighted. If the book lists only facts, one might say that it is not original, but if the facts are selected and arranged in an original manner, the book may be copyrighted.

In the English decision of *Donoghue v. Allied Newspapers Ltd* [1938] Ch 106, the court argued that "the person who has clothed the idea in form, whether by means of a picture, a play or a book" owns the copyright. Even more eloquently, Latham CJ in the Australian decision of *Victoria Park Racing and Recreation Grounds Co. Ltd v. Taylor* (1937) 58 CLR 479 and 498 argued that if you are the first person to announce that a man has fallen off a bus, you cannot use the law of copyright to stop other people from announcing that fact.

Some courts have recognized that there are particular ideas that can only be expressed intelligibly in only one or a limited number of ways. Therefore even the expression in these circumstances is unprotected, or extremely limited to verbatim copying only. In the United States this is known as the **merger doctrine**, because the expression is considered to be inextricably merged with the idea. U.S. courts are divided on whether **merger** constitutes a defense to infringement or prevents copyrightability in the first place, but it is often pleaded as an affirmative defense to infringement.

Doctrine of first sale

Note that copyright law does *not* restrict resale of copies of works, provided those copies were made by or with the permission of the copyright holder. Thus it is legal, for example, to resell a book or a CD that you have purchased, provided you do not keep a copy for yourself. In the US this is known as the First Sale Doctrine, and was established in the US court system to clarify the legality of reselling books in used book stores. Elsewhere it has other names; in the United Kingdom it is known as "Exhaustion of rights" and is a principle which applies to other Intellectual property rights.

Subject to moral rights, copyright also does not prohibit the owner of a physical copy of a work from modifying, defacing, destroying, etc. the work, so long as this does not involve duplication

Fair use

Copyright also does not prohibit all forms of copying. In the United States, the fair use clause of the Copyright Act (17 U.S.C. Section 107) allows copying and distribution when done fairly. The statute does not clearly define fair use, but examples of fair use are given, and four non-exclusive factors are offered for a fair use analysis. Copyrighted works may also be available for copying through a statutory compulsory license scheme or via a copyright collective or performing rights organisation such as ASCAP or BMI.

How copyrights are obtained and enforced

Typically, works must meet minimal standards of originality in order to qualify for a copyright, and the copyright expires after a set period of time. Different countries impose different tests, although generally the test is low; in the United Kingdom there has to be some 'skill, originality and work' which has gone into it. However, even fairly trivial amounts of these qualities are sufficient.

In the United States, the original owner of the copyright may be the employer of the actual author rather than the author himself if the work is a "work for hire". Again, this principle is widespread; in English Law the Copyright Designs and Patents 1988 provides that where a work in which copyright subsists is made by an employee in the course of his employment, the copyright is automatically assigned to the employer.

Copyrights are generally enforced by the owner in a civil law court, but there are also criminal infringement statutes. Criminal sanctions are generally aimed at serious counterfeiting activity.

How copyrighted material is identified

In general when a piece of material such as a film (including DVDs and Videos) and/or music is registered with the appropriate country's copyright office, the material at the beginning or end may contain a copyright notice which can be a c inside a circle ©, or the word "copyright", followed by the year(s) of the copyright and the copyright owner's name. However, such notice is not *required* in nations that have acceded to the Berne Convention. In one way or another, under nearly any copyright regime, a work is generally protected by copyright from the moment of its creation (in the United States the usual phrase is "fixed in a tangible medium of expression") whether it displays a notice or not.

Year(s) of copyright

The year(s) of copyright are listed after the © symbol. If the work has been modified (ie. a new edition) and recopyrighted there will be more than one year listed.

All rights reserved

The phrase, *All rights reserved*, is a formal notice that all rights granted under existing copyright law are retained by the copyright holder and that legal action may be taken against copyright infringement.

Copyrighting fonts

In the United States, font design is not copyrightable, but it is patentable if novel enough. Stone and Lucida are the only two patented typefaces, and this may not hold up in court.

Europe used to have the same "can't copyright typefaces" laws as the United States, but Germany (in 1981) and the UK (in 1989) have passed laws making typeface designs copyrightable. The UK law is even retroactive, so designs produced before 1989 are also copyrighted, if the copyrights wouldn't have already expired (the German one is not retroactive).

Rights beyond copyright

Many European countries (and other countries as a result of the GATT Trade Related Intellectual Property or "TRIPs" agreement) further provide for moral rights in addition to copyrights possessed by authors, such as the right to have their work acknowledged and not be disparaged. (Famously, the Monty Python team managed to use these rights to stop the Monty Python TV programme being shown in the US because the US TV station was putting so many adverts into the program the Monty Python team claimed that it was being ruined as a serious comedy programme.)

While copyright is normally assigned or licensed to the publisher, authors generally retain their moral rights (although in some jurisdictions these can be excluded under contract). In most of Europe it is not possible for authors to assign their moral rights (unlike the copyright itself, which is regarded as an item of property which can be sold, licensed, lent, mortgaged or given like any other property). They can agree not to enforce them (and such terms are very common in contracts in Europe). There may also be a requirement for the author to 'assert' these moral rights before they can be enforced. In many books, for example, this is done on a page near the beginning, in amongst the British Library/Library of Congress data.

Some European countries also provide for artist resale rights, which mean that artists are entitled to a portion of the appreciation of the value of their work each time it is

sold. These rights are granted on the background of a different tradition, which granted *droits d'auteur* rather than copyright also granting all creators various moral rights beyond the economic rights recognized in most copyright jurisdictions. (see also parallel importation.)

History of copyright

While governments had previously granted monopoly rights to publishers to sell printed works, the modern concept of copyright originated in 1710 with the British Statute of Anne. This statute first recognized that authors, rather than publishers, should be the primary beneficiary of such laws, and it included protections for consumers of printed work ensuring that publishers could not control their use after sale. It also limited the duration of such exclusive rights to 28 years, after which all works would pass into the public domain.

The Berne Convention of 1886 first established the recognition of copyrights between sovereign nations. (Copyrights were also provided by the Universal Copyright Convention of 1952, but that convention is today largely of historical interest.) Under the Berne convention, copyright is granted automatically to creative works; an author does not have to "register" or "apply for" a copyright. As soon as the work is "fixed", that is, written or recorded on some physical medium, its author is automatically granted all exclusive rights to the work and any derivative works unless and until the author explicitly disclaims them, or until the copyright expires.

Critique of copyright

Critiques of copyright fall broadly into two camps, those who assert that the very concept of copyright has never been of net benefit to society, and has always served simply to enrich a few at the expense of creativity, and those who feel that the current copyright system doesn't work in the new Information society. The general problem is that the current (international) copyright system undermines its own goal (Boyle 1996, 142). The concept of public domain, needed as a pool for future creators, is far too often forgotten or repressed, due to the strong position of the concept of the romantic author, and selective blindness for the possibilities concerning copyright that the Internet and computers offer. Except for unlimited copying, it offers, as said, also new ways for marketing and, more important, the possibilities of code; much depends of course on how code is used (code can be used and is in most of the cases also used in a positive way), but in various cases it threatens not only the public domain in a serious way, but is also ignored when talking about "restoring the balance" which is said to be gravely disturbed by the so called unlimited copying possibilities the Internet creates. [1] (*http://akira.arts.kuleuven.ac.be/andreas/english_paper_gaidai.html*)

Others believe that irrespective of contemporary advances in technology, copyright has been and remains the fundamental way by which authors, sculptors, artists, musicians

and others can fund the creation of new works. This view espouses that copyright is the only reason some valuable books and art would be created.

In the US in 2003, controversial changes implemented by the Sonny Bono Copyright Term Extension Act extending the length of copyright under U.S. copyright law by 20 years were constitutionally challenged unsuccessfully in the Supreme Court. The Court, in the case called Eldred v. Ashcroft, held inter alia that in placing existing and future copyrights in parity in the CTEA, Congress acted within its authority and did not transgress constitutional limitations.

Further reading

- Bruce Lehman: Intellectual Property and the National Information Infrastructure (Report of the Working Group on Intellectual Property Rights, 1995)

Fair Use

The **fair use** doctrine is a body of law and court decisions which provides for limitations and exceptions to copyright protection in the United States. **Fair use** is also a doctrine that applies to other areas of intellectual property law such as trademarks. Fair use attempts to balance the interests of copyright holders with the public interest in the wider distribution and use of creative works, by allowing certain limited uses that would otherwise be considered infringement. It is also considered to be an accommodation of the free speech protections of the First Amendment to the U.S. Constitution.

Comparable copyright limitations can be found in many nations' copyright statutes, though these differ in scope. Most other common law countries have a related doctrine known as *fair dealing*, which is defined in a constrained manner through an enumerated list of causes for exemption that allows little room for judicial interpretation. Civil law countries have codified similarly specific and narrowly drawn exceptions. Fair use, however, tends to be an open-ended legal doctrine, as statutory factors are balanced by U.S. judges on a case-by-case basis rather than strictly applied.

Fair use under United States law

Fair use in the U.S. grew out of the English common law doctrine of *fair abridgement.* It was first applied in the U.S. in *Folsom v. Marsh* (1841), where the defendant had copied 353 pages from the plaintiff's 12-volume biography of George Washington to make a two-volume work. The court rejected the defendant's fair use defense with the following explication of the doctrine:

> [A] reviewer may fairly cite largely from the original work, if his design be really and truly to use the passages for the purposes of fair and reasonable cri-

> ticism. On the other hand, it is as clear, that if he thus cites the most important parts of the work, with a view, not to criticise, but to supersede the use of the original work, and substitute the review for it, such a use will be deemed in law a piracy.

The *Folsom* court went on to formulate the basis for the factors used today in an analysis of the fair use defense. It continued to be a purely judge-made and applied law until it was finally codified as part of the 1976 Copyright Act at 17 USC § 107, excerpted here:

> Notwithstanding the provisions of sections 106 and 106A, the fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include--
>
> > (1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
> >
> > (2) the nature of the copyrighted work;
> >
> > (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
> >
> > (4) the effect of the use upon the potential market for or value of the copyrighted work.
>
> The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

Purpose and character

This is basically the intention and motivation behind the use; for example, there is a difference between a few shots taken from a film for a nonprofit review of the film and taking a few shots to include in a for-profit compilation of film reviews (though a for profit review may be considered news reporting). If it is obvious that the user is attempting to make a profit from the use, then it suggests that it is more likely that the use is infringement unless the use fits within the various "preamble purposes". However as can be seen in the parody cases discussed below such a commercial use is not dispositive as there are ways to use substantial portions of the work and still successfully claim fair use. Nonprofit or educational uses are generally seen to be given more latitude than for profit endeavors.

This first factor is divided into several subfactors: (1) the commercial or nonprofit educational nature of the use (discussed above); (2) the "preamble purposes", i.e. criticism, comment, news reporting, teaching, scholarship, and research (this list is not restrictive, and falling within one of these purposes does not create a presumption of fair use, it is just one factor to consider) (3) the degree to which the work has been transformed, has the fair use added to the original work in some way giving it a different character, or adding the original and giving it a new meaning or messsage. Not all educational use is protected by fair use, see *Macmillan Co. v. King.*

Nature of the copied work

Though according to the Supreme Court, copyright law is not supposed to discriminate based upon the quality or artistic merit of the work at issue, fair use analysis nonetheless looks at whether the copied work was creative or informative. Facts and ideas are unprotected--the particular expression of those facts or ideas is what merits copyrightability. (see idea-expression divide) In application to written works, this factor will tend to weigh for a copying defendant if the original work was a work of nonfiction rather than fiction or fantasy. Functional images--those that are merely illustrative of their subject matter or serve a purely utilitarian purpose--are also more likely to support a finding of fair use than more fanciful, expressive ones.

Also considered critical under this factor is whether or not the original work has been published and/or distributed to the public. Copyright law highly values the author's right to control how his work is first released to the public, and so a work's unpublished nature will tend to weigh against a finding of fair use.

Amount and substantiality

This relates to how much of the original copyrighted work is used in the new work; if only a very small amount is used in relation to the original (perhaps a few sentences for a book review) then chances are that the sample is a case of fair use. However, if a very substantial amount is used (perhaps an entire chapter, taken verbatim) then this will often be considered copyright infringement. See *Sony Corp. v. Universal City Studios* for an example of substantial copying that was upheld as fair use.

For several years some decisions led many to suppose that one of the few cases where this factor was irrelevant was in sampling a piece of a copyrighted sound recording. Subsequent decisions have shown that this is not the case and a normal fair use analysis must be performed. The US National Association of Music Retailers is one trade group which believes that sampling is not inevitably infringement, in the case of its members when the sample is used as part of the process of selling new or used musical works (position statement) (*http://www.narm.com/Content/NavigationMenu/Public_Affairs/Position_Papers/sampling/sampling.htm*).

In regards to the digital reproduction of images it may be argued that a lower resolution sample of the image (i.e. thumbnails) is a lesser sample of the image (the sound re-

cording sample is not analogous here) and thus the whole image is only being approximated by the lower resolution sample (limiting further reproduction outside an informational context) see the *Kelly v. Arriba Soft Corporation* case below.

Effect upon work's value

This fourth factor considers the effect that the use of the copyrighted material has upon the copyright owner's ability to exploit the original work. Thus if an image of a copyrighted audio CD cover is reproduced in a catalogue of published musical works this image will most likely not prevent the copyright owner of the CD from further exploiting the CD (indeed the listing of the CD may encourage purchasers to buy it). An individual who takes copyrighted images off an Internet web site, prints them, and then sells them in front of the art museum where the original copyrighted images are displayed would be interfering with the sales of these images in the museum gift shop (note that the harm to derivative works is also significant).

Practical effect of fair use defense

The practical effect of this law and the court decisions following it is that it is usually possible to quote from a copyrighted work in order to criticize or comment upon it, teach students about it, and possibly for other uses. Certain well-established uses cause few problems. A teacher who prints a few copies of a poem to illustrate a technique will have no problem on all four of the above factors (except possibly on amount and substantiality), but some cases are not so clear. All the factors are considered and balanced in each case: a book reviewer who quotes a paragraph as an example of the author's style will probably fall under fair use even though he may sell his review commercially. But a non-profit educational website that reproduces whole articles from technical magazines will probably be found to infringe if the publisher can demonstrate that the website affects the market for the magazine, even though the website itself is non-commercial.

Fair use as a defense

Fair use is an affirmative defense to copyright infringement. This means that if the defendant's actions do not constitute an infringement of the plaintiff's rights (for example, because the plaintiff's work was not copyrighted, or the defendant's work did not borrow from it sufficiently), fair use does not even arise as an issue. However, it also means that, once the plaintiff has proven (or the defendant concedes) that the defendant has committed an infringing act, the defendant then bears the burden of proving in court that his copying should nonetheless be excused as a fair use of the plaintiff's work.

Because of the defendant's burden of proof, some copyright owners frequently make claims of infringement even in circumstances where the fair use defense would likely

succeed in hopes that the user will refrain from the use rather than spending resources in his defense. Because paying a royalty fee may be much less expensive than having a potential copyright suit threaten the publication of a completed work in which a publisher has invested significant resources, many authors may seek a license even for uses that copyright law ostensibly permits without liability.

Fair use and parody

Producers or creators of parodies of a copyrighted work have been sued for infringement by the targets of their ridicule, even though such use may be protected as fair use. To be protected as a parody, a work must use the copyrighted material in a manner that is intrinsically a commentary, ridicule, or criticism of it. Fair use will not apply if the use was merely for attention getting.

In *Campbell v. Acuff-Rose Music* (1994), the Supreme Court recognized parody as a fair use, even when done for profit. Roy Orbison's publisher, Acuff-Rose Music Inc., had sued 2 Live Crew in 1989 for their use of Orbison's "Oh, Pretty Woman" in a mocking rap version with altered lyrics. The Supreme Court viewed 2 Live Crew's version as a ridiculing commentary on the earlier work, and ruled that when the parody was itself the product rather than used for mere advertising, commercial sale did not bar the defense. The *Campbell* court also distinguished parodies from satire, which they described as a broader social critique not intrinsically tied to ridicule of a specific work, and so not deserving of the same use exceptions as parody because the satirist's ideas are capable of expression without the use of the other particular work.

In a more recent parody case, *Suntrust v. Houghton Mifflin*, a suit was brought unsuccessfully against the publication of *The Wind Done Gone*, which reused many of the characters and situations from *Gone with the Wind*, but told the events from the point of view of the slaves rather than the slaveholders. The Eleventh Circuit, applying *Campbell*, recognized that *The Wind Done Gone* was a protected parody, and vacated the district court's injunction against its publication.

Fair use on the Internet

A recent court case, *Kelly v. Arriba Soft Corporation,* provides and develops the relationship between thumbnails, inline linking and fair use. In the lower District Court case on a motion for summary judgment Arriba Soft was found to have violated copyright without a fair use defense in the use of thumbnail pictures and inline linking from Kelly's website in Arriba's image search engine. That decision was appealed and contested by Internet rights activists such as the Electronic Frontier Foundation, who argued that it is clearly covered under fair use. On appeal, the 9th District Court of Appeals found that the thumbnails were fair use and remanded the case to the lower court for trial after issuing a revised opinion on July 7, 2003. The remaining issues were resolved with a

default judgement after Arriba Soft had experienced significant financial problems and failed to reach a negotiated settlement.

Fair use and trademark law

In the U.S., there is also a fair use defense in trademark law based on similar principles as the doctrine under copyright (such as free speech), but with different exceptions. Fair use is consistent with the more limited protection granted to trademarks, generally specific only to the particular product market and geographic area of the trademark owner.

Most trademarks are adopted from words or symbols already common to the culture (such as Apple), instead of being invented by the mark owner (such as Kodak). Courts have recognized that ownership in the mark cannot prevent others from using the word or symbol in these other senses, such as if the trademark is a descriptive word or common symbol such as a pine tree. This means that the less distinctive or original the trademark, the less able the trademark owner will be able to control how it is used.

Trademarks may also be used by a nonowner *nominatively*--to refer to the actual trademarked product or its source. In addition to protecting product criticism and analysis, U.S. law actually encourages nominative usage by competitors in the form of comparative advertising.

Both of these exceptions require that the mark not be used by the nonowner in a way that would be likely to confuse consumers about the source of their (or the trademark owner's) product. Generally this translates into the requirement, similar to that in fair use under copyright, that no more of the trademark is used than is necessary for the legitimate purpose.

Fair Dealing

Fair dealing is a doctrine of limitations and exceptions to copyright which is found in many of the common law jurisdictions of the Commonwealth of Nations (the former British Empire).

Fair dealing is an enumerated set of possible defenses against an action for infringement of an exclusive right of copyright. Unlike the related United States doctrine of fair use, fair dealing cannot apply to any act which does not fall within one of these categories. In practice, common law courts might rule that actions with a commercial character, which might be naively assumed to fall into one of these categories, were in fact infringements of copyright as fair dealing is not as flexible concept as the American concept of fair use.

Fair dealing in Australia

In Australia, the grounds for fair dealing are:

- Research and study
- Review and criticism
- "Reporting the news"
- Legal advice (although the Crown is deemed to own copyright in federal statutes, and each State in state statutes).

Australia has a deeming provision which guarantees that fair dealing applies if you photocopy either (not more than one chapter), or (less than 10%) of a book or journal (this was a result of a successful lawsuit brought against a university library for "authorisation" of patrons' copyright infringement).

Regarding fair dealing under Crown copyright the Australian Copyright Act 1968, ss.176-178. Section 182A (inserted by Act 154 of 1980, s.23) provides that the copyright, including any prerogative right or privilege of the Crown in the nature of copyright, in Acts, Ordinances, regulations etc., and judgments of Federal or State courts and certain other tribunals, is not infringed by the making, by reprographic reproduction, of one copy of the whole or part of that work for a particular purpose (this does not apply where charge for copy exceeds cost).

Fair dealing in Canada

The Canadian concept of fair dealing is similar to that in the UK and Australia. The fair dealing clauses of the Canadian *Copyright Act* allow users to make single copies of portions of works for "research and private study." Unlike the United States where fair use is seen as an allowable infringement, Canada's fair dealing is seen as not an infringement at all.

The 2004 ruling by the Supreme Court of Canada in *CCH Canadian Ltd. v. Law Society of Upper Canada* has gone far in clarifying the concept of fair dealing in Canada.

In considering fair dealing it makes the following general observation: "It is important to clarify some general considerations about exceptions to copyright infringement. Procedurally, a defendant is required to prove that his or her dealing with a work has been fair; however, the fair dealing exception is perhaps more properly understood as an integral part of the Copyright Act than simply a defence. Any act falling within the fair dealing exception will not be an infringement of copyright. The fair dealing exception, like other exceptions in the Copyright Act, **is a user's right**. In order to maintain the proper balance between the rights of a copyright owner and users' interests, it must not be interpreted restrictively. ... 'User rights are not just loopholes. Both owner rights and

user rights should therefore be given the fair and balanced reading that befits remedial legislation.'"

It then establishes six principle criteria for evaluating fair use.

1. **The Purpose of the Dealing** Is it for research, private study, criticism, review or news reporting? It expresses that "these allowable purposes should not be given a restrictive interpretation or this could result in the undue restriction of users' rights."
2. **The Character of the Dealing** How are the works were dealt with? Was there a single or multiple copies. Were these copies distributed widely or to a limited group of people? Was the copy destroyed after after its purpose was accomplished? What are the normal practices of the industry?
3. **The Amount of the Dealing** How much of the work was used? What was the importance of the infringed work? Quoting trivial amounts may alone sufficiently establish fair dealing. In some cases even quoting the entire work may be fair dealing.
4. **Alternatives to the Dealing** Was a "non-copyrighted equivalent of the work" available to the user? Could the work have been properly criticized without being copied?
5. **The Nature of the Work** Copying from a work that has never been published could be more fair than from a published work "in that its reproduction with acknowledgement could lead to a wider public dissemination of the work - one of the goals of copyright law. If, however, the work in question was confidential, this may tip the scales towards finding that the dealing was unfair."
6. **Effect of the Dealing on the Work** Is it likely to affect the market of the original work? "Although the effect of the dealing on the market of the copyright owner is an important factor, it is neither the only factor nor the most important factor that a court must consider in deciding if the dealing is fair." A statement that a dealing infringes may not be sufficient, but evidence will often be required.

"These factors may be more or less relevant to assessing the fairness of a dealing depending on the factual context of the allegedly infringing dealing. In some contexts, there may be factors other than those listed here that may help a court decide whether the dealing was fair."

The Association of Universities and Colleges of Canada (AUCC), a well established lobbying group representing the educational sector in Canada is of the opinion that making a copy of the following for the purposes of private study and research is fair dealing:

- a periodical article of a scientific, technical or scholarly nature from a book or a periodical issue containing other works;

- a newspaper article or entry from an encyclopedia, annotated bibliography or similar reference work; or
- a short story, play, poem, or essay from a book or periodical containing other works.

The AUCC believes that faculty members or students can make a copy of parts of a book or other complete works under fair dealing. No hard or fast rules are available in Canadian law but a rough indicator would be 10% of a complete work. The AUCC also maintains that fair dealing applies not just to photocopying but also to other methods of reproduction -- including the making of copies onto slides, microfiche or transparencies. For multiple copies and for copying in excess of the extent mentioned above, AUCC recommends acquiring licences from Access © , the Canadian Copyright Licensing Agency, one of the copyright licensing societies or copyright collectives in Canada.

Fair dealing in the United Kingdom

In the United Kingdom, "fair dealing" has always been the subject of dispute because the law never defines clearly the exact number of copies and the amount of the original materials allowed.

Under the *Copyright, Design and Patents Act* (1988) (CDPA), fair dealing is defined as "private study and criticism and review and news reporting" (s. 29, 30) Although not actually defined as a fair dealing, copyright in works is not infringed by incidental inclusion in an artistic work, sound recording, film, broadcast or cable program. It is also important to note that new regulations came into force at the end of October 2003 which reduced the research fair dealing exception to non-commercial research only.

CDPA permits individuals to make a single copy of a "reasonable proportion" of literary, dramatic, musical and artistic works for "research and private study" and "criticism, review and news reporting" (s. 29, 30) under the terms of "fair dealing". The extent of "reasonable proportion" is not defined in the act.

Some higher education institutions in the UK interpret "reasonable proportion" as:

- One article in a single issue of a periodical or set of conference proceedings.
- An extract from a book amounting to 5% of the whole or a complete chapter.
- A whole poem or short story from a collection, provided the item is not more than 10 pages.
- In general, copying of sheet music is not allowed.
- Making more than one copy is also not allowed.

For copying beyond the boundaries set forth by these guidelines, universities and schools in the UK obtain licences from the UK Copyright Licencing Agency (CLA) for their staff and students a local copyright collective. Under these licences, multiple copies of portions of copyrighted works can be made for educational purposes.

Intellectual Property

The concept of **intellectual property** (IP) treats certain intangible products similarly to physical things. In most countries, IP laws grant certain kinds of exclusive rights over these intangibles on the analogy of property rights, some expiring after a set period of time, and others lasting indefinitely. (See also intellectual capital.)

The purposes of these laws has varied, but most grant the "owner" a monopoly on the use of copying of the protected "property". This was done historically to both to grant a boon to a king's favourite, as well as "to promote the progress of science and useful arts". In the latter sense, patents and copyrights serve as incentive to inventors and authors to produce works which benefit the public. These creators can exact a fee from those who wish to copy their invention or publish their compositions.

Seen as an incentive to benefit the public, patent rights in particular promoted innovation by ensuring that someone who devoted, say, ten years of penury while struggling to develop vulcanized rubber or a workable steamship, could recoup his investment of time and energy. Using monopoly power, the inventor could exact a fee from those who wanted to make copies of his invention. Set it too high, and others would simply try to make a competing invention, but set it low enough and one could make a good living from the fees.

In latter years, the public benefit idea has been downplayed in favor of the idea that the primary purpose of "property rights" is to benefit the holder. This view places a priority on the benefit of the patent or copyright holder, even to the detriment of society at large, and has attracted some opponents, notably Richard Stallman.

In some fields, patent law has had an unintended, indeed, a perverse consequence: treating mental products like physical ones has stifled innovation in those fields, rather than aiding it.

The four main types of non-physical things considered by this point of view are copyrights, patents, trademarks and trade secrets. Common types of intellectual property rights include conflicting areas of law:

- Copyrights, which give the holder some exclusive rights to control some reproduction of works of authorship, such as books and music, for a certain period of time.
- Patents give the holder an exclusive right to use and license use of an invention for a certain period, typically 20 years from the filing date of a patent application.

- Trademarks are distinctive names, phrases or marks used to identify products to consumers.
- Trade secrets, where a company keeps information secret, perhaps by enforcing a contract under which those given access to information are not permitted to disclose it to others.

These rights, conferred by law, can be given, sold, rented (called "licensing") and, in some countries, even mortgaged, in much the same way as physical property. However, the rights typically have limitations, sometimes including term limits and other exceptions (such as fair use for copyrighted works.)

It is important to understand that it is the rights that are the property, and not the intellectual work they apply to. A patent can be bought and sold, but the invention that it covers is not owned at all. For this and other reasons, some people think that the term *intellectual property* is misleading. Some use the term "intellectual monopoly" instead, because such so-called "intellectual property" is actually a government-granted monopoly on certain types of action. Others object to this usage, because of potential confusion with the economic sense of the term "monopoly." Others still prefer not to use a generic term because of differences in the nature of copyright, patent and trademark law, and try to be specific about which they are talking about.

Legal status

Intellectual property rights are generally divided into two categories: those that grant exclusive rights only on copying/reproduction of the item or act protected (e.g. copyright) and those that grant not only this but also other exclusive rights. The difference between these is that a copyright would prevent someone from copying the design of something, but could not stop them from making that design if they had no knowledge of the original held by the copyright holder. A patent, on the other hand, can be used to prevent that second person from making the same design even if they had never heard of or seen the original. Patent rights can thus be more powerful, and generally harder to obtain and more expensive to enforce.

There are also more specialized varieties of so-called sui generis intellectual property rights, such as circuit design rights (called mask work rights in USA law, protected under the Integrated Circuit Topography Act in Canadian law, and in European Community Law by Directive 87/54/EEC of 16 December 1986 on the legal protection of topographies of semiconductor products), plant breeder rights, plant variety rights, industrial design rights, supplementary protection certificates for pharmaceutical products and database rights (in European law).

TYPES AND SCOPE OF INTELLECTUAL PROPERTY

Intellectual property may be analysed in terms of its subject matter, the actions it regulates in respect of the subject matter, the duration of particular rights, and the limitations on these rights. Intellectual property law is conventionally categorized according to subject matter: inventions, artistic expression, secrets, semiconductor designs, and so on. Intellectual property law regulates what people may legally *do* with these inventions, expressions and so on. The regulations regarding each subject matter area tend to form distinct bodies of law; the rules permitting reproduction without license of patented inventions and copyrighted expression are entirely independent of one another.

Generally, the action regulated by intellectual property is unauthorized reproduction. However, as indicated above, some rights go beyond this to grant a full suite of exclusive rights on a particular idea or product. Generally, it is true to say that intellectual property rights grant the holder the ability to stop others doing something (i.e., a negative right), but not necessarily a right to do it themselves (i.e., a positive right). For example, the holder of a patent on a pharmaceutical product may be able to prevent others selling it, but (in most countries) cannot sell it themselves without a separate license from a regulatory authority.

Most intellectual property rights are nothing more than the right to sue an infringer, which has the effect that people will approach the rightholder for permission to perform the acts covered by the rightholder's exclusive rights. The granting of this permission is termed licensing, and IP licenses may be used to impose conditions on the licensee, generally the payment of a fee or an undertaking not to engage in particular forms of conduct. In many jurisdictions the law places limits on what restrictions the licensor (the person granting the licence) can impose. In the European Union, for example, competition law has a strong influence on how licences are granted by large companies.

A license is 'permission' to do something, in contract form. Therefore a license is only required for activities which fall under the exclusive rights in question. The intellectual property laws of certain countries provides for certain activities which do not require any license, such as reproduction of small amounts of texts, sometimes termed fair use. Many countries' legal systems afford compulsory licenses for particular activities, especially in the area of patent law.

Many intellectual property rights are awarded by a government for a limited period of time. Such rights are justified as a reward for creating intellectual works. Economic theory typically suggests that a free market with no intellectual property rights will lead to too little production of intellectual works relative to an efficient outcome. Thus by increasing rewards for authors, inventors and other producers of intellectual capital, overall efficiency might be improved. On the other hand, intellectual property law could in some circumstances lead to increased transaction costs that outweigh these gains (see Coase's Penguin). Another consideration is that restricting the free reuse of information

and ideas will also have costs, where the use of the best available technique for a given task or the creation of a new derived work is prevented.

ARGUMENTS AGAINST THE TERM *INTELLECTUAL PROPERTY*

The term *intellectual property* is problematic because the rights conferred by IP laws are limited, in contrast with the legal rights associated with property interests in physical goods or land. Not entirely coincidentally, the presence of the word *property* in the term favours the position of proponents of the expansion of intellectual property rights, who may thereby more readily draw on the rhetoric of property itself to remove the many restrictions built into intellectual property law which would be inappropriate if applied to physical goods. For instance, most nations grant copyrights for only limited terms, and allow copyright holders to control only the duplication, and not the sale or modification of physical copies of a work.

A common argument against the term Intellectual Property is that information is fundamentally different from physical property in that a "stolen" idea or copy does not affect the original possession. Another, more specific objection to the term, held by Richard Stallman, is that the term is *confusing (http://www.gnu.org/philosophy/words-to-avoid.html#IntellectualProperty)* . It implies a non-existent similarity between copyrights, patents, trademarks, and other forms of intellectual property which makes clear thinking and discussion about various forms difficult. Furthermore, most legal systems, including that of the United States, imply that intellectual property rights are a government grant, rather than a right held by citizens.

Though it is convenient for beneficiaries to regard intellectual rights as akin to "property", most items protected by IP law are not physical objects "ownable" in the traditional sense. For example, the holder of the copyright in a book has the legal right to make and sell copies of the book, and the right to forbid others from making and selling copies of the same book. By analogy, then, he can be said to "own" the words in a similar way to which he might own the press on which they were printed, because ownership of a physical object also confers the right to forbid others from using the object.

Opponents of the term also point out that the law itself treats these rights differently than those involving physical property. To give three examples, copyright infringement is not punishable by laws against theft, but rather by an entirely different set of laws with different penalities. Patent infringement is not a criminal offense although it may subject the infringer to civil liability. Possessing stolen physical goods is a criminal offense while mere possessing of goods which infringe on copyright is not.

Others would argue that the law is simply recognising the reality of a situation. In some jurisdictions a lease of land (e.g. a flat or apartment) is regarded as intangible property in the same way that copyright is. In these cases too the law accepts that the property cannot be stolen - if someone moves into the flat and prevents you from living there

they are not regarded as 'thieves of the lease' but as 'squatters' and the law provides different remedies.

History

It is not exactly clear where the concept of intellectual property originated.

The first patent in England was granted by Henry VI in 1449 to a Flemish man a 20 year monopoly (co-incidentally, the current length of UK/EU patents is still 20 years) on the manufacture of stained glass (destined for Eton College). This was the start of a long tradition by the English Crown of the granting of "letters patent" (meaning 'open letter', as opposed to a letter under seal) which granted "monopolies" to favoured persons (or people who were prepared to pay for them). This became increasingly open to abuse as the Crown granted patents in respect of all sorts of known goods (salt, for example). After public outcry, James I was forced to revoke all existing monopolies and declare that they were only to be used for 'projects of new invention'. This was incorporated into the Statute of Monopolies 1623. In the reign of Queen Anne the rules were changed again so that a written description of the article was given.

Outside of England, patent law was the subject of legislative protection in the Venetian Statute of 1474.

Copyright was not invented until after the advent of the printing press and wider public literacy. In England the King was concerned by the unfair copying of books and used the royal prerogative to pass the Licencing Act 1662 which established a register of licensed books and required a copy to be deposited with the Stationers Company. The Statute of Anne was the first real act of copyright, and gave the author rights for a fixed period. Internationally, the Berne Convention in the late 1800's set out the scope of copyright protection and is still in force to this day.

Design rights started in England in 1787 with the Designing & Printing of Linen Act and have expanded from there.

The term *intellectual property* appears to have originated in Europe during the 19th century. French author A. Nion mentions "*propriété intellectuelle*" in his *Droits civils des auteurs, artistes et inventeurs*, published in 1846, and there may well have been earlier uses of the term.

During the period in question, there was some controversy over the nature of copyright and patent protections in Europe; those who supported unlimited copyrights frequently used the term *property* to advance that agenda, while others who supported a more limited system sometimes used the term *intellectual rights* (*droits intellectuels*).

The system currently used by much of the Western world is more in line with the second view, with limited copyrights that eventually expire. Regardless, the term *intellec-*

tual property has gained prominence throughout the world, as evidenced by the United Nations World Intellectual Property Organization (WIPO), formed in 1967.

Trends

Recently the general trend in intellectual property law has been expansion: to cover new types of subject matter such as databases, to regulate new categories of activity in respect of the subject matter already protected, to increase the duration of individual rights, and to remove restrictions and limitations on these rights.

Another effect of this trend is an increase in the term of the government-granted rights, and an expansion of the definition of "author" to include corporations as the legitimate creators and owners of works. The concept of work for hire has had the effect of treating a corporation or business owner as the legal author of works created by people while employed.

Another trend is to increase the number and type of what is claimed as intellectual property. This has resulted in increasingly broad patents and trademarks: for instance, Microsoft attempting to trademark the phrase, "Where do you want to go today?". Trade marks in EU law can now encompass smells (e.g. of cut grass for tennis balls), shapes (e.g. of a soft drinks bottle), colors (e.g. red for fizzy drinks), words (e.g. COCA-COLA) and sounds (Intel, has registered four notes). The granting of patents for life forms, software algorithms and business models stretches the initial concept of giving the inventor limited rights to exclude the use if his invention.

Some argue that these expansions harm an essential "bargain" driven between public and copyright holders: as most "new" ideas borrow from other ideas, it is thought that too many intellectual property laws will lead to a reduction the overall creative output of a society. The expansion of exclusive rights is also alleged to have led to the emergence of organizations whose business model is to frivolously sue other companies.

The electronic age has seen an increase in the attempt to use software based digital rights management tools to restrict the copying and use of digitally based works. This can have the effect of limiting fair use provisions of copyright law and even make the first sale doctrine (known in EU law as 'exhaustion of rights') moot. This would allow, in essence the creation of a book which would disintegrate after one reading. As individuals have proven adept at circumventing such measures in the past, many copyright holders have also successfully lobbied for laws such as the Digital Millennium Copyright Act, which uses criminal law to prevent any circumvention of software used to enforce digital rights management systems. Equivalent provisions, to prevent circumvention of copyright protection have existed in EU for some time, and are being expanded in, for example, Article 6 and 7 the Copyright Directive. Other examples are Article 7 of the Software Directive of 1991 (91/250/EEC), and the Conditional Access Directive of 1998 (98/84/EEC).

At the same time, the growth of the Internet, and particularly peer-to-peer file-sharing networks like Kazaa and Gnutella represents a challenge to intellectual property laws. The Recording Industry Association of America, in particular, has been on the front lines of the fight against what it terms "piracy". Though the industry has had some victories against services, including a highly publicized case against the file-sharing company Napster, the increasingly decentralized nature of these networks is making legal action more difficult.

Non-government systems of IP protection

The notion of protecting intellectual capital is much older than copyright or patent law. There have long existed socially-enforced systems for protecting intellectual capital. These include the ancient scholarly taboo against plagiarism, along with other informal systems such as the one used by clowns to protect their unique style of makeup.

On a more modern topic, intellectual property law has been brought to bear on domain names where trademark holders (in particular) have objected to third parties registering domain names which they believe should be theirs. The domain name registries, many of whom are not governmental organisations, have had to find a solution to this and therefore have dispute resolution systems which operate in parallel with national laws. The majority of the generic top level domain names (.com, .net etc.) use the ICANN model known as the Uniform Dispute Resolution Policy (UDRP). Other registries, such as the .uk registry Nominet UK have their own different systems. For example, Nominet's system is called the Dispute Resolution Service.

Economic view

Intellectual property rights such as copyrights and patents give the holder an exclusive right to sell, or license, the right to use that work. As such, the holder is the only seller in the market for that particular item of intellectual property, and the holder is often described as having a monopoly for this reason.

However, it may be the case that there are other items of intellectual property that are close substitutes. For example, the holder of publishing rights for a book may be competing with various other authors to get a book published. In such cases, economists may find that another market form, such as oligopoly or monopolistic competition better describes the workings of the market for the intellectual property. For this reason, many writers prefer that intellectual property rights are described as *exclusive rights* rather than *monopoly rights.*

If the market for the rights to some intellectual property is perfectly competitive, then the rights to that intellectual property will generally be worthless. This is because in a perfectly competitive market, sellers are *price takers* and can sell to as many people as they like at the prevailing price in the market. It costs little or nothing to grant someone

the right to use a copyrighted work or patent, so the optimal behaviour for the seller is to sell as many licenses as possible, whatever the price is, forcing the price towards zero. Thus intellectual property rights, to be valuable, must give the holder some *market power* (the ability to influence price) in the market for rights to use that intellectual property. An example may be a patent covering an idea where another idea which is in the public domain provides the same utility and no-one is likely to accidentally stumble on and use the patented idea. If someone were to re-invent the patented idea and use it unaware a patent exists, the patent holder can claim damages.

The case for intellectual property in economic theory is substantially different than the case for tangible property. Consumption of tangible property is rivalrous. For example, if one person uses a plot of land to build a home, that plot is unavailable for use by others. Without the right to exclude others from tangible resources, a tragedy of the commons can result. Intellectual property does not share this feature. For example, an indefinite number of copies can be made of a copyrighted book without interfering with the use of the book by owners of other copies. Therefore, the rationale for intellectual property rests on the incentive effects. Without intellectual property rights (or subsidies), there would be no direct financial incentive to create new inventions or works of authorship. However, as Wikipedia and Free software demonstrate, works of authorship are written without direct financial incentives. Moreover, many important works and inventions were created before copyright was invented. One might argue that much more invention occurred after patents came into existance, however, one could also argue that patents were brought into law as the power and influence of industrial interests grew.

A more elaborated view of capital suggests that the three most common property instruments applied provide exclusive rights to use different things: copyright covers creative works and expressions of ideas, patent covers ideas with industrial application and trademark covers means to uniquely identify a producer or other source of reputation. The three types of instruments have different histories, different intent, and protect (in the modern analysis that grants capital status to individual creativity, instructions, and social repute) three different kind of capital. Even if asserting that any of the three is property is acceptable, asserting that all three deserve it for the same reasons is not. Yet the most common definitions in international law confuse the three rather badly:

> "**Intellectual property** refers to creations of the mind: inventions, literary and artistic works, and symbols, names, images, and designs used in commerce." (Source: WIPO, *http://www.wipo.org/about-ip/en/*)

Those who contest the idea of IP say that this assertion is propaganda for a property view of these works or marks, and for their confusion with each other. They also prefer the older terms individual capital, instructional capital and social capital to the more modern "intellectual capital" which has an ambiguous status, even among believers in neoclassical economics. It seems no one can say exactly "what" this "IP" or "IC" "is", other than to say that they are related to each other, and that holding IC gives you the

ownership of IP, and that holding IP gives you the right to be paid. This view of political economy does nothing to suggest either is useful to society at large, or why police or court time should go into enforcing rights.

While widely accepted in Western culture, the status of IP is disputed in India, China and other developing nations. Economist Lester Thurow claims that only in nations whose culture derived from practices of Judaism, Christianity and Islam, all of which share a vision of man as "created in God's image", is the creative power of the individual assumed to be worthy of property protection. These nations have imposed the intellectual property system and benefit from it - the United States and the United Kingdom are the only two nations who consistently receive net balance of payements benefits from IP. On the other hand, if incentives do increase investments in invention and authorship, then the resulting inventions and creative works may produce net benefits to other nations--as would be suggested by the so-called law of comparative advantage.

A more recent type of IP, the protection of databases, has been introduced by the EU in 1996. This is an important right as it protects the information contained in a database against re-utilisation and extraction of a substantial part. The right, in order to come into existence, requires a substantial investment and subsists alongside copyright in the database structure.

Bibliography

- Arthur Raphael Miller, Michael H. Davis, *Intellectual Property: Patents, Trademarks, and Copyright*, West Wadsworth; 3rd edition, 2000, ISBN 0314235191 (textbook particularly covering copyright and patent law)
- Michael Perelman, *Steal This Idea: Intellectual Property Rights and the Corporate Confiscation of Creativity*, Palgrave Macmillan, 2002, ISBN 0312294085, (a critical discussion of some of the social, scientific and cultural impacts of recent intellectual property developments)
- Roger E. Schechter, John R. Thomas, *Intellectual Property: The Law of Copyrights, Patents and Trademarks*, West Wadsworth, 2003, ISBN 0314065997 (textbook)

Closed source

Closed source until a few years ago has been an integral part of commercial software development. It means that the customer will only get a binary version of the computer program they licensed and no copy of the program's source code, rendering modifications to the software practically impossible from the technical side, because the usual way to modify a program is to edit its source code and then compile it.

The *source code* in this development model is regarded a trade secret of the company, so parties that may get source code access, such as colleges, have to sign NDAs in advance.

In the 1970s, the operating system UNIX, which was available freely and with complete source code, became common in university computing centers. Users made enhancements to the operating system and applications and distributed them among themselves without restrictions.

People like Richard Stallman were used to the openness of this hacker culture, and thus it came as an unpleasant surprise when more and more skilled programmers left academia to found their own companies and market their software, no longer giving their peers source code access.

Richard Stallman saw *closed source* as a step backwards in terms of user freedom and founded the GNU project in the mid 1980s, whose GPL-licensed software may never again be released without source code availability.

Closed source still dominates commercial software development, but in the last few years through the success of open source projects like Linux, KDE, and Apache corporate thinking has undergone a transformation.

Today, some corporations have recognized that closed and open source projects can complement each other, as is evidenced for instance by Sun Microsystems' move to develop their office suite, *StarOffice*, in parallel with its *open source* incarnation, *OpenOffice.org*. This is seen as a gain for corporate image and may be a good way to attract new talent.

At times closed source code is leaked, against the desire of its writers.

Shared Source

Shared Source is a term, primarily used by Microsoft, to define a pseudo-open source form of code sharing.

With shared source, the source code to a particular piece of software is made available to reference. However, unlike most open source licenses, the authors maintain strict control over the use of that code once it has been read. For example, many shared source licenses permit only academic use of the source code, or permit reuse of the code only for non-commercial use, or permit reading but no deriving from the code base.

Proponents of shared source see it as a step forward from purely proprietary development. In the particular case of Microsoft, their shared source initiatives often permit developers to see source code they otherwise would have no access to. This permits better integration, debugging, interaction, and standardization among products.

Opponents of shared source often see it as too little, too late. Many outspoken open source advocates consider the shared source licenses as illusionary. That is, they give the impression that they are making the source code freely available, when in fact there are critical restrictions on the code's use.

The Shared Source CLI

Perhaps the most notable shared source license is that covering Rotor, the shared source implemenation of the Microsoft .NET CLI. This implemenation is freely available, including source code, as a reference guide. The license explicitly permits non-commercial use of the source code, including derived works. It explicitly forbids use of the code, or deriviatives, in any commercial software *or* open source software. (One of the provisions is that derived works must use a license that is at least as restrictive as the original shared source license.)

Software Patent

The expression **software patent** refers to a patent on software, and might be defined as a patent that has been, will be or could be granted on products or processes (including methods) which include or may include software as a significant or at least necessary part of their implementation, i.e. the form in which they are put in practice (or used) to produce the effect they intend to provide.

This is just one of many legal aspects of computing.

Definition

There is no universally accepted definition of the expression *software patent* and no legal text defines what exactly is a *software patent* is and what is not.

Software patents may however be classified in three categories: 1) patents on products or processes that may or may not include software in order to be implemented, 2) patents on products or processes that need software in order to be put into effect (along with some sort of hardware) and 3) patents that are nothing more than source code or algorithms.

These categories are arbitrary and have no legal direct value, but they may help to understand the issues at stake. Moreover, a same patent may contain several different claims, each of which belonging to a different category. So, it is actually and rigourously a classification of software patent claims rather than one of software patents, but it is quite equivalent as far as conferred protection from competition is concerned, since the claims are the most important part of a patent for determining the monopoly it confers to its owner.

Patents potentially including software

The "first" type of software patents can be defined as the patents on products or processes that may or may not include software in order to be implemented.

For instance, a (fictional) patent with a claim such as *"A high-pass filter comprising first means for converting an input analogue signal into a digital signal, second means for... and so on"* refers to a product, i.e. a filter in this case, that may or may not include software. Indeed, the filter may be implemented using either electronic *"first means for converting..."* or software *"first means"* running on a hardware support.

Patents including software

The "second" type of software patents can be defined as patents on products or processes that need software in order to be put into effect (along with some sort of hardware).

For instance, a (fictional) patent with a claim such as *"A high-pass filter comprising 1) a computer, 2) a program able to run on it and to convert an input analogue signal into a digital signal, 3)... "* refers to a product, i.e. a filter, which needs a computer and a computer program (or a software) to be implemented.

Patents on source code or algorithms

The "third" category consists in patents that contain nothing more that source code or algorithm. In other words, it could be said that this category includes methods which describe a process which can be implemented without using "forces of nature", if it is understood that the intellect is not a force of nature.

For instance, a (still-fictional) patent with a claim such as *"An algorithm which consists in taking a sequence of numbers as an input, applying to each of these numbers some kind of transformation, ..."* falls within this category.

Patentability of software

Software patents are treated differently under different jurisdictions.

In the US

In the 1950s, 1960s, and 1970s, the United States Patent and Trademark Office (PTO) did not grant a patent if the invention used a calculation made by a computer. The PTO's rationale was that patents could only be granted to processes, machines, articles of manufacture, and compositions of matter; patents could not be granted to scientific truths or mathematical expressions of it. Since the PTO viewed computer programs and inventions containing or relating to computer programs as mathematical algorithms, and

not processes or machines, they were therefore not patentable. This view was upheld by the U.S. Supreme Court in Gottschalk v. Benson (1968) and Parker v. Flook (1975).

In the 1981 case of Diamond v. Diehr, the U.S. Supreme Court ordered the PTO to grant a patent on an invention, even though there was no invention claimed besides the use of a computer program (which used well-known formulas, this was also said before the U.S. Supreme Court by the patent attorney) for calculating the time when rubber was cured and the mold could be opened. The Supreme Court stated that in this case, the invention was not merely a mathematical algorithm (which it in fact was, exactly the industrial use of it), but was a process for molding rubber, and hence ordered the PTO to patent it.

After this point, more patents on software began to be granted, albeit with conflicting and confusing results. The Federal Circuit attempted to clarify the rules; requiring that the computer program must have a practical application. However, since all software is written to perform some useful activity, many believe this to be the exception that swallows the rule.

Meanwhile, the Clinton administration pushed software patenting from the administrative agency side, by appointing Bruce Lehman as Commissioner of the Patent and Trademark Office in 1994. Unlike his predecessors, Lehman was not a patent lawyer but the chief lobbyist for the Software Publishing Industry. In 1995, the PTO established some broad guidelines for examining and issuing software patents. The PTO interpreted the courts as requiring the PTO to grant software patents for an extremely broad variety of circumstances, including those that are essentially algorithms only distantly connected to physical processes. Note, that although the US Congress has never legislated specifically that software is patentable, the broad description of patentable subject in the Patent Act of 1952 and the failure of Congress to change the law after the court decisions allowing software patents, has been interpreted as Congressional acquiescence.

Another impetus for software patenting was the growing recognition that using the copyright law to protect non-literal infringement of computer programs (rather than just piracy) was getting out-of-control. When comparing patent protection to the use of non-literal copyright infringement, many commentators argued that many protections for competitors are built into the patent system that are lacking in the copyright laws. Specifically, these commentators pointed out copyrights are not examined, but patents must first be examined to determine if the program is both novel and non-obvious; the scope of patent rights is defined by the patent claims, while the scope of non-literal copyright infringement is unclear; and the patent term of 17 or 20 years is much shorter than the copyright terms. When courts began to permit software invention to be patentable, other courts also began restricting the use of copyright law to obtain patent-like protection of software.

Those who favor software patents believe that software are inventions to the same extent as hardware and that the law should and, in practice is not able to, distinguish software inventions from hardware inventions. Proponents also argue that the patent sys-

tem rewards inventors of innovative approaches in software, and thus promote innovation. This belief is important in the US, because this is the only permitted reason for a patent to be granted according to the US Constitution. More specifically, the Constitution only permits Congress "to promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."

Opponents charge that software patents are particularly favored by lawyers, who financially benefit from patent litigation, and by some (though not all) very large software companies, who hope to use patents to prevent competitors from using the patented technology.

In Europe

The national jurisdictions relating to software patents in Europe and in the European Union are not harmonized. The EU Commission proposed a directive on the Patentability of Computer-Implemented Inventions, which was heavily amended in 2003 when reviewed at the EU Parliament. In response to this, the Commission proposed a compromise in May 2004, which opponents of software patents consider to be worse than the original proposal.

European Patent Convention

Almost all European countries are members of the European Patent Organisation but the European Patent Convention does not govern infringement and revocation proceedings before national Courts. As far as such actions are concerned, every European country may have, and indeed has, its own rules and case law.

The practice regarding software patent before the European Patent Office is however significant since the so-called "European patents" are examined and may be opposed according to the rules laid on the European Patent Convention.

Although it is widely misbelieved that software patents have been granted by the EPO only recently, thousands of patents related have been granted (rightly or wrongly) since the EPC entered into force. For instance from 1977 to 1994 only, about 11,000 software-related patents were granted by the EPO. Well-known article 52(2) only excludes for instance methods for performing mental acts, mathematical methods and programs for computer as such. Products and processes including such subject-matter and expressed in terms of their *technical features* and which provides a *technical effect* are not considered as being excluded as such (see for instance VICOM (*http://legal.european-patent-office.org/dg3/biblio/t840208ep1.htm*) decision of 1984).

Controversy

Software patents are the subject of widespread controversy.

Origin

Opponents to software patents mainly but not only comes from the free software community. Their principles are rather opposed to the underlying principles of the patent system. While the patent system is based on the assumption that a temporary monopoly is a way to encourage scientific and economic progress, their principle is that sharing is a better way to encourage scientific and economic progress. On the contrary, corporations often do not want to share their applications as they feel they must in order to protect their R&D investments.

Liberal economists oppose patent law in general.

More factually, the origin of the controversy may be traced back to patents granted in the U.S., such as the "one-click shopping" patent granted to Amazon.com, and the State Street Bank decision of 1998, according to which "everything made under the sun by man can be patented."

Issues

If all technologies are patentable, why not software?

Legalistic deduction perspective: TRIPS 27 provides that all fields of technology must be patentable. It is an open discussion whether software is regarded as part of technology.

From the instrumental perspective: This Question is a rhetoric trap that reverses the burden of proof. An application of patent law to a field has to be justified by economic evidence. Patent law is seen as an instrument of economic policy.

Are software patents economically harmful?

The following arguments are

- There is a massive cash drain to the legal system, including lawyers, courts and IPR departmens. This money is not productive. (I've heard its 2bn per year in the US. Can this be confirmed?)
- There is no evidence that software patents actually encourage innovation. Computer implemented ideas are hardly connected to any research costs. The implementation itself is usually the most difficult part and protected by the Copyright.
- Software patents actively impede innovation. A patent is a right to block others from exploiting an idea. It is not right to use the idea. Every software and every website actually infringes several patents. If these patents were enforced all internet activity could be halted by few patent holders.
- Software patents do not add to a countries competetiveness. If an idea was made in a country without software patents it can still be patented anywhere else.

Are software patents especially harmful for small companies and individual developers?

There are several reasons that contribute to the fear of small companies and individual developers that software patents are especially harmful to them.

- It is relatively expensive to obtain and enforce patents. Every company needs a defensive patent portfolio to reach a patent sharing agreement with large firms.
- Since software patents potentially span any kind of thoughts, permanent patent study is required. For ideas implemented purely in software, there are no material inputs or outputs to an implementation of an idea. Resulting in the idea being expressed in ways which don't share the same key words. Full text searches of software patent databases will not necessarily yield patents which cover the field of software programming you are interested in.

Obtaining patents

In contrast to copyright, obtaining patents is relatively more expensive. Copyright is granted automatically when publishing a work. Through the Berne Convention and TRIPs the copyright is automatically extended to all countries that are part of those treaties. There are no costs involved. In order to obtain a patent, an inventor must file an application with a patent office and a fee must be paid. This patent is only valid within the jurisdiction of the patent office. In order to obtain a worldwide protection, an inventor must apply to every patent office in the local official language and pay a fee. Additionally there are sometime barriers that make it difficult to aquire a patent. Some countries require patent applicants be natural or legal persons within the jurisdiction.

Obviously this process it lengthy and expensive. Small businesses and individual developers usually do not have the monetary resources to pay for all the fees, translations, etc. to obtain a world-wide protection. They also would have to divert important human resources for this purpose.

This makes it far more difficult for small businesses and individual developers to obtain patents than for big corporations.

Enforcement of own patents

The ownership of a patent does not prevent automatically its infringement. The ownership of a patent just allows the owner to use the legal system to obtain a remedy for the patent infringement.

In order to do so, the patent owner must first know about the infringement. To obtain such knowledge is far easier for a multi-national corporation with the presence at the market where the infringement occurs than for small businesses or invidual developers, which probably never know about such occurance outside their realm of clients.

Secondly, in order to legally enforce a patent the patent owner must hire locally registered lawyers, start proceedings in court. All of this is costly and distracts from the main business. Small bussinesses and individual developers are rarely able to spend the upfront-costs and time necessary to followed up in this way without neclecting their business. Multi-national corporations, however, have legal departments for such tasks, and have therefore an advantage persuing patent infringements over small businesses and individual developers.

Avoidance of patent infingement

Already today the European Patent Office (EPO) has granted more than 30,000 software patents. It seems very difficult for a small business or individual developer to know all those patents to avoid the usage, or to negotiate term that would allow them to use the patented technology. Even big corporations might have problems investigating if they infinging patents except they are working with mutual shared patent portfolios as describe below. In addition, the legal costs and damages that a small business or individual developer would have to pay for unintended and incidental infringement would probably cause bankrupsy in most cases. Big corporation often absorb such costs on an annual basis.

Effect of patent enforcement on small businesses and individual developers

A defence against accusations of patent infringements is not a trivial task for a small business or individual developer. If this action has to be fought against a large corporation it is also a fight against the vast resources, lawyers, and experts that can easily overwhelm the resources of a small business or individual. Apart from being a distraction from the main business, small businesses and individual developers can suffer and even be destroyed by an action which they win because their clients are likely to be affected by the uncertainty of legal action and are likely to consider switching away from products or services that potentially use patented technology. What weighs heavily on the customers is that, if the legal action is successful and is not settled to protect them, they might be the next ones being accused of patent infringement.

The difficulty and cost of defence against allegations of breach of patent creates a competitive disadvantage for small business and the individual developer, since customers have to weigh the additional risks they take by selecting a small business or individual developer instead of a big corporation which will usually be able to settle such procedures in way that protects their clients. This is a solution which small businesses and individual developers can not afford.

Sharing of patent portfolios

Large corporations are aware that building a large patent portfolio is of increasing importance. Not so much to generate licensing revenues from the patent portfolio, but to gain access to ideas owned by other corporations through a cross-licensing deal. If your corporation has a large portfolio of patents, if a corporation which operates in the

same field as you attempts to threaten your corporation with one of their patents, there is a good chance that your corporation can return the threat, solving the issue in a cross licensing deal. This, in effect, creates an exclusive club of corporations able to exploit technology.

This effect occurs more in the field of software than in mechanical or pharmecutical fields; a piece of software may contain hundreds or thousands of ideas which may be patented. In mechanical and drug fields it tends to be closer to one patent, one product. Many patents per product coupled with the abstract/ hard to search nature of software patents makes the cross-licensing protection system (described above) the dominant business method to deal with software patents.

Therefore, software patents tend to block the field of software development for small businesses and individuals. Given that small businesses and individuals count for some of the most revolutionary advances, one might argue that the US constitutional rationale for permitting the issuance of monopolies is being broken in the field of software.

This concentration of power, according to standard economic theory, will tend to increase the price of the product (computer software) whilst reducing competitive pressure for improvement.

ntellectual Property Companies

The patentability of especially software has recently created a new line of business. New companies are formed with only one business goal, to obtain patents for the pupose of collecting license fees and damages in legal proceedings. These companies have no aim to produce any products or innovated technology. Moreover, if these companies were to innovate in the field of software, their own innovation may lead them open to threat. The only income these companies generate is by "participating" in the success of other companies. Such companies are a particular threat to small businesses and individual developers, because of their relative lack of legal expertise and resources. While the profit that could be made from big corporations is certainly bigger, the risks are also higher. Big corporations will more likely fight a long fight about patent issues. Therefore small businesses and individual developers are more likely initial targets to generate enough revenue and precedent to launch large, costly cases against big companies.

Effects of patents on employee mobility

The value of employees to their employers is often their experience. A very broad patentability will lead to a situation in which most of the experience an employee gains, will be protected in some form by patents. This in turn means that the experience is not easily transferable from one company to another. Therefore the possible mobility of the employee decreases as in turn the market value of the employee. This would put the employees in a very strong dependence of their employer since only there their experience can be applied. Certainly not only patents, but all intellectual property rights play a role in this issue. In summation, however, patents can have the biggest impact, since

they respresent an exclusive monopoly to an idea. Trade secrets and copyright can be avoided while still using the obtained experience.

Other arguments

Very often, what is monopolized this way is not even a simple method but the pure idea that something could be done, whatever means are used.

Since such methods are very generic, the scope of such patents is often very wide and they are very hard to find using keyword searches and there is no classification for them, thus they are applied for in the language of some field. A computer does not even need to be mentioned in the patent, the description could even refer to traditional machines or electronic circuits; what counts is how the claims are written and if they do not describe new teachings of forces of nature, the patent can be described as *software patent*.

Since there is no standardized language enforced by the patent offices to describe pure ideas, patent search quality in this area is very low and this is not only a problem for the patent office's patent searches in the course of examination of the patent before granting it but also in private patent searches and litigation.

Software patents are very controversial. For many decades, patent offices around the world rejected most applications for software patents. In Europe, the European Patent Convention states that "programs for computers" are excluded from the patent system "as such". The meaning of "as such" in this context was clear for decades, but recently the European Patent Office spontanously (without change of the Convention or any political signal) changed it's interpretation from "as long as the program is the claim itself" to "as the text of the program". This is strongly opposed by many european software companies, developers and users.

The exclusion of software from patentability did not suit the interests of many patent professionals and certain computer manufacturers (such as IBM), which already were used to getting patent protection for their hardware but continued to seek routes to exclusive rights over algorithms and general software which they started to sell independently of the hardware. Gradually, cases began to appear in various jurisdictions (such as the United States, Japan and Australia), holding that software could be patented in various ways. The European Patent Office (responsible for granting European patents, and separate from the European Union) decided that it could grant patents on software using a politically controversial interpretation of the European Patent Convention.

Had the story remained typical of the history of intellectual property laws, the alignment of intention between key corporations (especially IBM and Microsoft) and the patent offices of the US, Europe, and Japan, would soon have lead to mandatory software patents under international law.

At the present moment, however, armed with evidence suggesting that software patents are likely to be economically harmful, coalitions of interest groups including the free

software and open source movement and software firms without large patent portfolios are attempting to reverse the trend of patent expansionism. This conflict has been played out over the EU Directive on the Patentability of Computer-Implemented Inventions.

Opposition to Software Patents

However, there remain many opponents of software patents, including an overwhelming majority of professional software developers. For example, Burton Systems Software conducted a survey of professional programmers (*http://lpf.ai.mit.edu/Whatsnew/survey.html*), and found that by a margin of 79.6% to 8.2% (10:1), computer programmers said that granting patents on computer software impedes, rather than promotes, software development (the remaining 12.2% were undecided). By 59.2% to 26.5% (2:1), most went even further, saying that software patents should be abolished outright.

Opponents of software patents argue against them for a diverse range of reasons. Here are some of the reasons opponents give for opposing software patents:

Innovation

- There is no evidence that software patents actually encourage innovation. The 1950s, 1960s, and 1970s included a large number of software innovations (*http://www.dwheeler.com/innovation*), when software patents were not permitted. These innovations can be measured both as published papers and as new kinds of products.
- Many in the computing field believe **software patents actively impede** innovation. In 1991, Microsoft's Bill Gates wrote a memo saying, "If people had understood how patents would be granted when most of today's ideas were invented and had taken out patents, the industry would be at a complete standstill today." (Mr. Gates' company now acquires a vast patent portfolio, since to do otherwise would be suicidal, and that portfolio may be helpful in preventing competition). Donald Knuth, a highly-respected computer scientist, stated that "If software patents had been commonplace in 1980, I would not have been able to create [the TeX system used by 90% of all books and journals in mathematics and physics], nor would I probably have ever thought of doing it, nor can I imagine anyone else doing so."
- Some believe that the problem besetting the software field is not a lack of innovation, but difficulty in developing the large number of desired products. The patent process **interferes with, not aids**, the development of useful products.

Economics

- Professors James Bessen and Eric Maskin, two economists at the Massachusetts Institute of Technology (MIT), have demonstrated that introducing patenting into the

software economy only has economic usefulness if a monopoly is the most useful form of software production. This is concerning, because few believe that a monopoly is truly the most useful (or desirable) form of software production. Bessen and Maskin also demonstrated a statistical **correlation between the spread of patentability in the United States and a decline in innovation in software**. In particular, between 1987 and 1994, software patents issuance rose 195%, yet real company funded R&Ds fell by 21% in these industries while rising by 25% in industries in general.

Obviousness

- They believe the standard for "obviousness" in other fields is inappropriate for software. Because software is malleable, small, incremental changes and generalization are normal and obvious to practitioners. However, the PTO normally grants patents to small, incremental changes, even if they would be obvious to practitioners. This is an error in the first place and having no real obviousness standard is especially bad when working with software because software doesn't consist of a couple of parts but by millions of lines of text each of which could infringe on a trivial patent.
- Many techniques are considered too obvious to publish by practitioners. However, a patent may be granted later by the PTO, because no paper was found by the PTO discussing the topic. Patent search is another problem with the huge number of trivial patents: You can never be sure that you licensed everything what you'd ever need to license to get the money back which you invest thru the live cycle of your software. That's legal insecurity.
- Some believe that switching from a copyright-based system to one permitting patents puts established experts at a severe disadvantage. Experts cannot patent many concepts because they are obvious (and sometimes verbally shared among peers)—yet they can be patented by novices because they are not as obvious to novices. Dan Bricklin, inventor of the spreadsheet, is a well-known proponent of this position.
- The cost structures for software development are fundamentally different. Extremely complex software systems with hundreds of thousands of parts are often built for small amounts of money compared to physical products. However, the costs of dealing with the patent system presumes that complex systems will result in large profits, on the order of those for physical products. For most software systems, this simply is not true. Also the work and cost lies in actually getting the complex system to work in every use case (debugging) and very seldom, the challenge lies in finding new algorithms.

Litigation Culture

- The **risk of a lawsuit greatly reduces the incentive to innovate** new products. This risk is exacerbated because software patent searches are prohibitively expensive and unreliable. Besides, patents may be granted to another after the software has already

been written, so even a perfect search would not prevent risks to software developers.

- Patent licenses are especially harmful to open source software / Free software, which are becoming an increasingly important type of software and in many markets are the only alternative to no software or establishing a permanent monopoly in a functional area.

"The licensing market, such as it is, seems to be defined characterized by patentees looking for infringers, rather than productive companies looking for technology." Brian Kahin.

OTHER ARGUMENTS

- Some software patents may be granted in the United States years after they were filed. According to this strategy, someone files for a patent and ensures that it is not made public by the PTO for some time through various paperwork processes, or simply words it so that it is not noticed by the community the patent would apply to. Patentees then attempt to ensure widespread use of the patented approach, e.g. by working with standards bodies and implementers to use the approach. Then, once the approach is widely used, they then announce the patent and sue all users, who will find it difficult to switch to other approaches once they are widely embedded. This practice is known as *submarine patents*. Since the U.S signed the TRIPs agreement, if a patent is to be applied for outside the U.S, the U.S application can remain submerged for up to 18 months.
- Patent licensing strongly discourages, and in some cases prohibits entry of newcomers into the software field. Large companies collect patents and attempt to force cross-licensing with others to protect themselves from software patents. But this means that small companies (SMEs), without a large body of patents to cross-license, may be forced to license from a large number of companies to develop software at all. The total of these royalties could exceed all possible benefits, permanently blocking newcomers from the software field.
- Small litigation companies (whose only contribution is to buy patents and sue other companies) can threaten large companies, even if those companies cross-license patents. Thus, even large companies can be at risk of a patent suit. However, these companies may exist solely to create patents of previously existing or obvious ideas, and litigating these patents can be more expensive than the product is worth.
- Patent examiners tend to be paid less than they could make doing other activities in software, so they tend to be less skilled. In addition, they must be generalists, so they are unlikely to be aware of well-known approaches in any particular area. However, at EPO, patent examiners are very well-paid and they could make a better job, but there are not enough of them, so there are long examination delays as well. It is said that the job is so monotonous that nobody who is really qualified would do it, so this is even more of a problem.

- Databases of prior work are inadequate for the task of determining if something has already been done before. Keyword searches are also inadequate to find prior art for generic software patents, even if internet search engines could do a quite comprehensible and fast job on worldwide text search. Unfortunately, specialized patents applications are not always made available in full text and the EPO started to give full text only to paying customers. Before, while application were available, they were not available in clear text but in graphics (pixel) format so you could not do full text searches.
- The patent process has little incentive to identify pre-existing work. The process rewards patent requesters who do poor research, since by doing poor research, they will not find preceding work that would invalidate the claim. However, since patent officers tend to be less skilled, have inadequate databases, work under significant time pressure, and must of necessity be generalists, it is difficult for them to find preceding work. These resultant patents can still be useful to patent-holders as threats, since court cases are expensive (minimum 1 to 5 million dollars) and very uncertain.
- Patent offices are notorious for granting absurd patents, yet once they are granted they can be enforced by simply the threat of an expensive lawsuit. For a non-software example, Patent 6,368,227 (*http://www.newscientist.com/news/news.jsp?id=ns99992178*) is a patent on a particular method for swinging on a child's swing, one that has no doubt been used by children for decades. In Australia, one man patented the wheel. This is not a problem because such patents are only jokes and nobody really cares about them. But the equivalent in the software patent world is a serious patent which is Intellectual property, which is vigorously defended (for 20 years)...
- The patent system diverts many able-bodied experts into processing patents instead of innovating.
- Patent litigation is extremely expensive, and owners of patents that should never have been issued can nevertheless impede innovation or cause others to pay unnecessary fees to avoid the cost of litigation. The cost of litigation in the US starts at $500.000 per side, so the trigger level were it becomes economically possibly useful to fight before a court starts at $1 million of damage. Below, litigation makes no sense and patent deals would have to be done, which is very hard if the defending party has no patent portfolio itself to have a negotiation mass.
- The term of patents (20 years in the US) inappropriately long for software; software has a very short life cycle. A single patent creates a monopoly over ideas used in generations of software. Many otherwise viable births of software projects are aborted or die young from patents granted generations ago.

Software patents tend to be opposed by individual software developers, who view software patents as a risk to their livelihood and are a high risk to SMEs.: if enough patents are granted, they will not be able to sell their software. Some large software companies also oppose patents, fearing that they will be sued for implementing obvious techni-

ques, resulting in continuous payments to avoid court costs or steep fees for court battles. Well-known opponents of software patents include Richard Stallman (author of the gcc compiler), Dan Bricklin (inventor of the spreadsheet), Donald Knuth (an expert on computer algorithms and the author of the TeX typesetting software), Hartmut Pilch of FFII, Alex Macfie (Taiwan), Eurolinux Alliance, Lawrence Lessig, Mitch Kapor, Michel Rocard (former Prime Minister of France), Adobe and Oracle.

Dealing with Software Patents

Most software development companies in the US have decided to acquire software patents, even if they oppose the granting of them. Their motives include acquiring a patent before someone else does, or forcing competitors who acquire patents on obvious approaches to cross-license with them. Often these patents are only used defensively, e.g., they are only used against someone who first sues the company. Some organizations and licenses have formalized a nonaggression policy (a policy of never pursuing or profiting from aggressive software patent suits) and/or of mutual defense (in which a pool agree to this). Such systems, however, provide little defense to individual developers or small businesses, and it is unclear if they will prevail once companies come into financial hardship, needing patent revenues to persist. Often a patent can be worked around once the patent is known, but this can be a significant hardship if there is a significant amount of data in a format requiring the use of the patented algorithm.

A recent concern is the role of patents in the standards process. Some standards bodies have no patent policy; thus, it is possible for a member to convince a standards body to make certain technologies required by a standard while at the same trying to get a patent on that technology. As a result, many standards bodies (such as W3C) are now requiring their members to promise to grant either reasonable and non-discriminatory (RAND) or even royalty-free licenses on their patented technology that is incorporated into the standard.

Groups against software patents

- Foundation for a Free Information Infrastructure (FFII)
    - Opposition by FFII to software patent legislation in Europe (*http://swpat.ffii.org/papers/eubsa-swpat0202/index.en.html*)
- Free Software Foundation: transcript and audio of Software patents – Obstacles to software development (*http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html*) which Richard Stallman gave about software patents (the audio archive linked contains two more speeches about software patents)
- Irish Free Software Organisation (IFSO) (*http://ifso.ie/*)
- Liberal economists
- competition law bodies (BEUC)

- European SME groups (UEAPAME, CEA-PME, dmmv, DIHK, WKO, ...)
- EU campaign NoEpatents (Eurolinux-alliance) (*http://www.noepatents.org*) with more than 270 000 European signatures one of the largest Internet campaigns ever.
- League for Programming Freedom
- The History of Software Patents (*http://www.bitlaw.com/software-patent/history.html*) from BitLaw.
- Sequential Innovation, Patents, and Imitation (*http://www.researchoninnovation.org/patent.pdf*) by James Bessen and Eric Maskin
- Software Patents vs. Free Software (*http://perens.com/Articles/Patents.html*) by Bruce Perens
- Report on Software Patentability by Conseil des Mines Study Group - Stimulating Innovation in the Information Society (*http://www.pro-innovation.org/rapport_brevet/brevets_plan-en.pdf*)
- SWpat information page by ESR Pollmeier (German SME), opposed to swpat (*http://www.esr-pollmeier.de/swpat/index_en.html*)
- AEL (Association Electronique Libre) Wiki Software Patent Main Project page (*http://wiki.ael.be/index.php/FightingSWPatents*)
- http://www.softwarepatents.co.uk/
- attac (Globalisation critics)
- W3C: Letter from Tim Berners-Lee to Rogan (*http://www.w3.org/2003/10/27-rogan.html*) (about Eolas Plugin Patent):

Groups in favor of software patents

- Large software and IT companies having built up a stock of software and other patents, the patent attorneys of these companies set the patent position of these companies. They also benefit from drawbacks for smaller companies.
- Patent lawyers, they provide the extremely expensive service needed for software patents.
- Patent offices, they gain money and power from software patents.
- Patent judges and patent courts also gain power because of unpredictable software patent cases.

Important Persons

Richard Stallman

Richard Matthew Stallman (**RMS**; born 16 March 1953) is the founder of the Free Software movement, the GNU project, the Free Software Foundation, and the League for Programming Freedom. He invented the concept of copyleft to protect the ideals of this movement, and enshrined this concept in the widely-used GPL (General Public License) for software.

An image of **Richard Matthew Stallman** taken from the cover of the O'Reilly book *Free as in Freedom* by Sam Williams, published in March, 2002.

He is also a notable programmer whose major accomplishments include GNU Emacs, the GNU C Compiler, and the GNU Debugger. Since the mid 1990s Stallman has relinquished most of his software engineering duties in order to focus on the advocacy of free software. His remaining development time is devoted to GNU Emacs. He is currently supported by various fellowships, maintaining a modest standard of living while discharging his duties as an itinerant evangelist and "philosopher" of free software.

Biography

Stallman was born on 16 March 1953 in Manhattan to Alice Lippman and Daniel Stallman. He is perhaps better known by his initials, "RMS". In the first edition of the *Hacker's dictionary*, he wrote, '"Richard Stallman" is just my mundane name; you can call me "rms".'

In the 1960s, with the personal computer still a decade away, Stallman's first opportunity to gain access to a computer came during his junior year at high school. Hired by the IBM New York Scientific Center, a now-defunct research facility in downtown Manhattan, Stallman spent the summer after his high-school graduation writing his first program, a preprocessor for the IBM 7094 written in the PL/I programming language. "I first wrote it in PL/I, then started over in assembler language when the PL/I program was too big to fit in the computer", he later revealed (Williams 2002, chapter 3).

After that job, Stallman held a Laboratory Assistant position in the Biology Department at Rockefeller University. Although he was already moving toward a career in mathema-

tics or physics, his analytical mind impressed the lab director so much that only a few years after Stallman had departed for college, his mother received an unexpected phone call. "It was the professor at Rockefeller", she recalled. "He wanted to know how Richard was doing. He was surprised to learn that he was working in computers. He'd always thought Richard had a great future ahead of him as a biologist." (Williams 2002, chapter 3)

In 1971, as a freshman at Harvard University, Stallman became a hacker at the MIT AI Laboratory.

Decline of the hacker culture

In the 1980s, the hacker community that dominated Stallman's life began to dissolve under the pressure of the commercialization of the software industry. In particular, a group of breakaway AI Lab hackers founded the company Symbolics, which actively attempted to recruit the rest of the AI Lab hackers in order to replace the free software in the Lab with its own proprietary software.

For two years, from 1981 to 1983, Stallman single-handedly duplicated the efforts of the Symbolics programmers to prevent them from gaining a monopoly on the Lab's computers. By that time, however, he was the last of his generation of hackers at the Lab. He was asked to sign non-disclosure agreements and perform other actions he considered betrayals of his principles, but chose instead to share his work with others in what he regarded as a classical spirit of scientific collaboration and openness.

Stallman's philosophy was that "software wants to be free": if a user or fellow hacker benefited from a particular piece of software it was the developer's right - and indeed duty - to allow them to use and improve it without artificial hindrance or restrictions on their rights to pass the original or derivative works onto others. Consequently, in January 1984, he quit his job at MIT to work full time on the GNU project, which he'd announced in September 1983. He has worked on GNU more or less full-time since then, and did not complete a doctoral degree. He has been awarded three honorary doctoral degrees.

Founding GNU

In 1985, Stallman published the GNU Manifesto, which outlined his motivation for creating a free operating system called GNU, which would be compatible with Unix. The name GNU is a recursive acronym for GNU's Not Unix. Soon after, he incorporated the non-profit Free Software Foundation (FSF) to employ free software programmers and provide a legal framework for the free software community.

In 1989 Stallman invented and popularized the concept of *copyleft*. By then, much of the GNU system had been completed, with the notable exception of a kernel. Members of the GNU project were working on a kernel called GNU Hurd, but a risky design decision proved to be a bad gamble, and development of the Hurd was slow.

In 1991, this final gap was filled by Linux, a kernel written independently of the GNU project using the GNU development tools and system libraries. The arrival of Linux, and the availability of a completely free operating system created some confusion, however, and most people now use the name Linux to refer to the whole operating system. Stallman has attempted to correct this by asking people to call the operating system "GNU/Linux".

Free software and open source

Richard Stallman's political and moral pronouncements have made him a controversial figure. Some influential programmers who agree with the concept of sharing code disagree with Stallman's moral stance, personal philosophy, or the language he uses to describe his positions. One result of these disputes was the establishment in 1998 of a new movement, the open source movement, whose aims are broadly similar, but whose proponents emphasize the technical merits of code developed in an open fashion, rather than the principles of liberty and freedom.

Few who have encountered Stallman or read his essays would deny that he is a man of deeply held (and readily expressed) convictions; this has been interpreted in both a positive and negative light. He has been the subject (some would say the instigator) of a number of widely-publicized flamewars on discussion forums such as the Linux kernel mailing list. Although occasionally for technical reasons (Tcl vs. Scheme), most of these flamewars have revolved around the use of non-free software.

Recognition

Stallman has received numerous prizes and awards for his work, amongst them:

- 1990: MacArthur Fellowship
- 1991: The Association for Computing Machinery's Grace Hopper Award for his work on the original Emacs editor
- 1996: Honorary doctorate degree from Sweden's Royal Institute of Technology
- 1998: Electronic Frontier Foundation's Pioneer award
- 1999: Yuri Rubinski Memorial Award
- 2001: Second honorary doctorate, from the University of Glasgow
- 2001: The Takeda Techno-Entrepreneurship Award for Social/Economic Well-Being (武田 究賞)
- 2002: National Academy of Engineering membership
- 2003: Third honorary doctorate, from the Vrije Universiteit Brussel

Bibliography

- Williams, Sam (2002) *Free as in Freedom: Richard Stallman's Crusade for Free Software*, O'Reilly Press ISBN 0596002874 (also available over the web under the GFDL, see link below).
- Gay, Joshua (ed) (2002): *Free Software, Free Society: Selected Essays of Richard M. Stallman.* Boston: GNU Press. ISBN 1882114981 (also available over the web, see link below).

External links

- Richard Stallman's Personal Home Page (*http://www.stallman.org*)
- *Free As In Freedom* (*http://www.oreilly.com/openbook/freedom/*), by Sam Williams, a biography of Stallman licensed under the GNU FDL
- *Free Software, Free Society* (*http://notabug.com/2002/rms-essays.pdf*), by Joshua Gay (ed), a selection of essays of Richard Stallman
- Stallman's 1986 speech in Sweden (*http://www.gnu.org/philosophy/stallman-kth.html*)
- The GNU Philosophy pages (*http://www.gnu.org/philosophy/philosophy.html*) ~50 essays, most are by RMS
- The GNU Philosophy Audio pages (*http://www.gnu.org/philosophy/audio/audio.html*) contains Ogg Vorbis recordings of 11 speeches by RMS, plus one video. (Includes RMS's talk given at ArsDigita University)

Eric S. Raymond

Eric Steven Raymond (born 4 December 1957) (often referred to by his initials, **ESR**) is the author of "The Cathedral and the Bazaar" and the present maintainer of the "Jargon File" (also known as "The New Hacker's Dictionary"). Though the Jargon File established his original reputation as a historian/anthropologist of the hacker culture, after 1997 he became a leading figure in the open source movement, and is today one of the most famous (and controversial) hackers.

Raymond is an avowed libertarian. He is known to have a strong interest in science fiction, is an enthusiastic amateur musician, and has a black belt in taekwondo. His public advocacy of Second Amendment gun rights and strong support for the 2003 Iraq War has nettled some hackers, but he seems to enjoy the controversy this engenders.

Achievements

Born in Boston, Massachusetts in 1957, Raymond lived on three continents and forgot two languages before settling in Pennsylvania in 1971. His involvement with hacker culture began in 1976, and he wrote his first open source project in 1982.

He is the author of the fetchmail POP client. He has contributed many editing modes to the EMACS editor and co-written the GNU ncurses library. He was the creator of the C implementation of the INTERCAL programming language.

Raymond coined the sentence, "Given enough eyeballs, all bugs are shallow." He credits Linus Torvalds with the inspiration for this quotation, which he dubs "Linus's law". The "mainstream" source for the quotation is his 1999 book *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastopol, California: O'Reilly & Associates; but [1] (*http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/x147.html*) archives the earliest source (1997), originally distributed freely on the Internet. In addition to this, he maintains a dozen FAQs and writes lots of essays.

After 1997 Raymond became a principal theorist in the open source movement and one of the founders of the Open Source Initiative. He also took on the role of ambassador of open source to the press, business and mainstream culture. He is a gifted speaker with the delivery (and, perhaps, ego) of a stand-up comic, and has taken his road show to more than fifteen countries on six continents. He is routinely quoted in the mainstream press, and as of 2003 has probably achieved more public visibility than almost any other hacker.

Raymond's tactics have scored a number of remarkable successes, beginning with the release of the Mozilla source code in 1998, and he is widely credited by both hackers and mainstream observers with having taken the open source mission to Wall Street more effectively than anyone before him.

Criticism

Critics accuse Raymond of hijacking the free software movement for the sake of self promotion and profit. In that context it is argued that he has often worked to undermine other leaders/speakers of the movement. His forthright rejection of the moral and ethical arguments of RMS and the Free Software Foundation in favor of a less idealistic (though arguably more pragmatic), market-friendly stance, has exacerbated some pre-existing political tensions in the community.

There has also been some acrimony between Raymond and Linux developers, after the Linux project's refusal to incorporate CML2, an alternative kernel configuration system developed by Raymond.

He has also been accused of directly selling out. He agreed to lecture at Microsoft in return for the opportunity to meet a couple of his favorite science fiction authors. In addition, he accepted millions of dollars in stock options in return for giving VA Research/VA Linux Systems credibility as their hired "moral compass".

Furthermore, his temper has also caused some tension between himself and other Open Source advocates, most famously Bruce Perens. Perens made public a private email threat he received from Raymond on the Debian mailing lists, citing safety concerns.

Raymond's claim to being a "Core Linux Developer" has drawn criticism since he has never had code accepted into Linux (the kernel), and his largest open source code contributions amount to portions of fetchmail, Ncurses, and Emacs (as well as a long list of small toy projects listed on his homepage). This lack of credentials led to a less-than-inspiring reception to his essay "Shut Up And Show Them The Code" which he levelled at Richard Stallman, the original author of Emacs, GCC, GDB, GNU Make, and many other pieces of GNU software.

Raymond addresses some of these assertions in his essay "Take My Job, Please!", where he argues that if anyone is qualified and willing to take his job and present the case for open source to the world, he would "back them to the hilt".

During the summer of 2003, Raymond expounded his opinions about politics, terrorism and the Iraq war on his blog, provoking much heated criticism. He has also been accused of modifying the Jargon file to reflect his own views about the war.

Books by Raymond

- The New Hacker's Dictionary (editor) (MIT Press, paperback ISBN 0-262-68092-0, cloth ISBN 0-262-18178-9) — printed version of the Jargon file
- The Cathedral and the Bazaar (O'Reilly; hardcover ISBN 1565927249, October 1999; paperback ISBN 0596001088, January 2001) — includes "The Cathedral and the Bazaar", "Homesteading the Noosphere", "The Magic Cauldron" and "Revenge of the Hackers"
- The Art of Unix Programming (Addison-Wesley, October 2003; paperback ISBN 0131429019)

Movies with Raymond

- Revolution OS, Linux Documentary with Eric S. Raymond on VHS/DVD

Quote

- *Anybody who has ever owned a dog who barked when strangers came near its owner's property has experienced the essential continuity between animal territoriality and human property. Our domesticated cousins of the wolf are instinctively smarter about this than a good many human political theorists.* — from "Homesteading the Noosphere"

External links

- Raymond's home page (*http://www.catb.org/~esr/*)
- A Second Look at the Cathedral and Bazaar by Nikolai Bezroukov (First Monday) (*http://www.firstmonday.org/issues/issue4_12/bezroukov/index.html*)
- The Magic Cauldron (*http://www.ora.de/catalog/cb/chapter/*), 1999 - It is a very good read on the reality of free software development. It still applies and it's nice to see that the even the optimistic IDG projections are today surpassed by far.
- Surprised by Wealth (*http://linuxtoday.com/news_story.php3?ltsn=1999-12-10-001-05-NW-LF*) - Raymond's thoughts immediately after the VA Linux initial public offering
- The Emperor Has No Clothes (*http://esr.1accesshost.com/*) - a critique of Eric S. Raymond

Linus Torvalds

Linus Benedict Torvalds (born 28 December 1969) began the development of Linux, an operating system kernel, and today acts as the project coordinator (or Benevolent Dictator for Life). Inspired by the demo-system Minix developed by Andrew Tanenbaum, he felt the need for a capable UNIX operating system that he could run on his home PC. Torvalds did the original development of the Linux kernel primarily in his own time and on his equipment.

Biography

Torvalds was born in Helsinki, the capital of Finland, as the son of Nils and Anna Torvalds. Both of his parents were campus radicals at the University of Helsinki in the 1960s, his father a Communist who in the mid-1970s spent a year studying in Moscow. This caused embarrassment to Linus at the time since other children would tease him about his father's politics.

His family belongs to the Swedish-speaking minority (roughly 6% of Finland's population). Torvalds was named after Linus Pauling. He attended the University of Helsinki from 1988 to 1996, graduating with a masters degree in computer science.

Linus Torvalds currently lives in San Jose, California with his wife Tove (six times national Karate champion in Finland), whom he first met in fall 1993, his cat Randi (short for Mithrandir, the Elvish name for Gandalf, a wizard in *The Lord of the Rings*), and his three daughters Patricia Miranda (born 5 December 1996), Daniela Yolanda (born 16 April 1998) and Celeste Amanda (born 20 November 2000). In June 2004 Linus purchased a home in Beaverton, Oregon and enrolled his children in school.

He worked for Transmeta Corporation from February 1997 until June 2003, and is now seconded to OSDL to work on the Linux kernel full-time. Although OSDL is based in Portland, Oregon, he worked from his home in San Jose.

His personal mascot is a penguin nicknamed Tux, widely adopted by the Linux community as the mascot of Linux.

Linus's law, a tenet inspired by Linus and coined by Eric S. Raymond in his paper The Cathedral and the Bazaar, is: "*Given enough eyeballs, all bugs are shallow.*" A deep bug is one which is hard to find, and with many people looking for it, the hope (and so far most experience) is that no bug will be deep. Both men share an open source philosophy, which has been in part (and implicitly) based on this belief.

Unlike many open source "evangelists", Torvalds keeps a low profile and generally refuses to comment on competing software products, such as Microsoft's commercially dominant Windows operating system. He is neutral enough to even have been criticized by the GNU project, specifically for having worked on proprietary software with Transmeta and for his use and alleged advocacy of Bitkeeper. Nevertheless, Torvalds has occasionally reacted with strong statements to what has been widely perceived as anti-Linux (and anti open source) FUD from proprietary software vendors like Microsoft or SCO.

For example, in one e-mail reaction to statements by Microsoft Senior-VP Craig Mundie, who criticized open source software for not being innovative and destructive to intellectual property, Torvalds wrote: *"I wonder if Mundie has ever heard of Sir Isaac Newton? He's not only famous for having set the foundations for classical mechanics (and the original theory of gravitation, which is what most people remember, along with the apple tree story), but he is also famous for how he acknowledged the achievement: If I have been able to see further, it was only because I stood on the shoulders of giants ... I'd rather listen to Newton than to Mundie. He may have been dead for almost three hundred years, but despite that he stinks up the room less.*"

The Linus / Linux connection

Linus Torvalds originally used the Minix OS on his system which he replaced by his own OS; he gave a working name of Linux (Linus' Minix); but thought the name to be too egotistical and planned to have it named *Freax* (a combination of "free", "freak", and the letter x). His friend Ari Lemmke encouraged Linus to upload it to a network so it could be easily downloaded. Ari gave Linus a directory called *linux* on his FTP server, as he did not like the name Freax.

In August of 1991, he publicized his creation on the USENET newsgroup comp.os.minix:

```
Message-ID: 1991Aug25.205708.9541@klaava.helsinki.fi
From: torvalds@klaava.helsinki.fi (Linus Benedict Torvalds)
To: Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system

Hello everybody out there using minix-I'm doing a (free)
operating system (just a hobby, won't be big and professional
like gnu) for 386 (486) AT clones. This has been brewing since
april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it
somewhat

Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus
```

Only about 2% of the current Linux kernel is written by Torvalds himself, though he remains the ultimate authority on what new code and innovations are incorporated into the Linux kernel (*http://virtual.finland.fi/finfo/english/torvalds.html*); other operating system aspects (both user visible and invisible) such as the X windowing system, gcc, and various package management schemes are run by others. Many Linux distributions even have their own versions of the kernel. Torvalds tends to stay out of non-kernel-related debates, even among their developers. The Linux kernel written/supervised by him, when combined with software developed by many others (mainly the GNU system) results in a so-called *Linux distribution.* Many people refer to this combination as just *Linux*, and others refer to it as "GNU/Linux."

Torvalds owns the "Linux" trademark, and monitors (*http://slashdot.org/articles/00/01/19/0828245.shtml*) use (or abuse) of it chiefly through the non-profit organization Linux International. Needless to say, 'many eyeballs make trademark abuse difficult'; he gets help on this from the entire worldwide Linux community. Due to the Open Source philosophy, Torvalds used to dislike the fact that Linux is a trademark. However, in 1995, he had to adopt the trademark, because some other man had registered Linux himself and threatened to blackmail Torvalds.

Many Linux fans tend to worship Linus as a kind of god. In his book "Just For Fun" he complains that he finds it annoying.

In Time Magazine's Person of the Century Poll, Linus was voted at #17 at the poll's close in 2000. In 2001, he shared the Takeda Award for Social/Economic Well-Being with Richard Stallman and Ken Sakamura. In 2004, he was named one of the most influential people in the world by Time Magazine.

Further reading

- Linus Torvalds, David Diamond: *Just for Fun: The Story of an Accidental Revolutionary*, New York, HarperBusiness, 2001, ISBN 0066620724

External links

- Wikiquote - Quotes by Linus Torvalds (*http://quote.wikipedia.org/wiki/Linus_Torvalds*)
- Linus' home page (*http://www.cs.helsinki.fi/~torvalds*)
- The Rampantly Unofficial Linus Torvalds FAQ (*http://catb.org/~esr/faqs/linus/*)
- Leader of the Free World - How Linus Torvalds became the benevolent dictator of Planet Linux, the biggest collaborative project in history (Wired News) (*http://www.wired.com/wired/archive/11.11/linus_pr.html*)
- Benevolent Dictator. A slightly skeptical unauthorized biography and the first ten years of Linux (Softpanorama) (*http://www.softpanorama.org/People/Torvalds/index.shtml*)
- The famous "LINUX is obsolete" thread from the comp.os.minix newsgroup (*http://www2.educ.umu.se/~bjorn/mhonarc-files/obsolete/msg00000.html*)
    - Andrew S. Tanenbaum on the origins of Linux (*http://www.cs.vu.nl/~ast/brown/*) (2004)
- Interview with Linus, March 01, 1994 (*http://www.linuxjournal.com/article.php?sid=2736*)

Tux

Tux is the official Linux mascot — a satiated, happy, chubby penguin. Tux was created by Larry Ewing in 1996. The idea of the Linux mascot being a penguin came from Linus Torvalds, the creator of the Linux kernel.

It is sometimes claimed that the name was derived from **T**orvalds **Uni**X, a name suggested by James Hughes, rather than the explanation that penguins look vaguely like they are wearing a **tux**edo.

Tux was designed for a Linux logo contest. Pictures of some of the other contestants can be found at The Linux Logo Competition site (*http://www.cs.earlham.edu/~jeremiah/linux-pix/linux-logo.html*). The winning logo was created by Larry Ewing using the GIMP (a free software graphics package) and was released by him under the following condition:

- *Permission to use and/or modify this image is granted provided you acknowledge me lewing@isc.tamu.edu and The GIMP if someone asks.* (*http://www.isc.tamu.edu/~lewing/linux/*)

According to Jeff Ayers, Linus Torvalds had a *"fixation for flightless, fat waterfowl"* and Torvalds claims to have contracted "penguinitis" after being gently nibbled by a penguin: *"Penguinitis makes you stay awake at nights just thinking about penguins and feeling great love towards them."* Torvalds' supposed illness is of course a joke, but he really was bitten by a Little Penguin on a visit to Canberra (*http://www.linux.org.au/org/penguin.phtml*). Torvalds was looking for something fun and sympathetic to associate with Linux, and a slightly fat penguin sitting down after having had a great meal perfectly fit the bill.

Tux has become an icon for the Linux and Open Source community, with one British Linux user group adopting a penguin at Bristol Zoo. He is much more famous than his big friend, GNU, a peaceful and shy gnu that represents the GNU Project.

Tux is the star of a Linux game called Tux Racer, in which the user guides Tux down a variety of different icy hills on his belly, trying to catch herring and beat the time limit.

In some Linux distributions, Tux greets the user during booting, with multi-processor systems displaying multiple tuxes.

TUX is also the name of Linux kernel-based web server, which is able to serve static web pages much faster than traditional servers like Apache HTTP Server. This piece of software is maintained by Redhat (*http://people.redhat.com/mingo/TUX-patches/*).

External links

- Linux 2.0 Penguins (*http://www.isc.tamu.edu/~lewing/linux/*) (Larry Ewing)
- A complete history of Tux (*http://www.sjbaker.org/tux/*)
- The LWN Penguin Gallery (*http://lwn.net/Gallery/*)
- Wired News story on Tux (*http://www.wired.com/news/culture/0,1284,42209,00.html*)

Eben Moglen

Eben Moglen is a professor of law and history of law at Columbia University, and serves pro bono as General Counsel for Free Software Foundation. He is noteworthy for co-writing the GNU licenses with Richard Stallman. The most famous of these licenses is the GNU General Public License.

Moglen was a law clerk to Justice Thurgood Marshall (1986-87 term).

Moglen serves as a director of PubPat

His opinion on free software is that it's a fundamental requirement for a democratic and free society in which we are surrounded by and depending on technical devices. Only if controlling these devices is open to all via free software, power can be balanced equally.

Moglen's Metaphorical Corollary to Faraday's Law is the idea that the Internet works like induction on the humans minds of the planet. Hence Moglen's phrase "Resist the resistance!".

In 2003 he received the EFF Pioneer Award.

Publications

- Anarchism Triumphant: Free Software and the Death of Copyright

External links

- Eben Moglen's webpage at Columbia University (*http://emoglen.law.columbia.edu/*)
- http://www.pubpat.org/Board.htm
- http://www3.sys-con.com/banners/linuxworld336.cfm January 19, 2004 Interview
- http://www.cabinetmagazine.org/issues/1/i_moglen_1.php Another interview, includes stuff about the Encryption Wars (and here's Page 2 (*http://www.cabinetmagazine.org/issues/1/i_moglen_2.php*))
- http://world-information.org/wio/readme/992006691/1078412091 - Interview 11/12/2003

Alan Cox

Alan Cox is a programmer heavily involved in the development of the Linux kernel. He maintained an old branch (2.2.x), and his own versions of the previous stable branch (2.4.x) (signified by an "ac" in the version, for example 2.4.3-ac1). He was commonly regarded as being the "second in command" after Linus Torvalds himself, although this has changed over time.

He was one of the people involved in AberMUD.

Cox is employed by Red Hat and lives in Swansea, Wales.

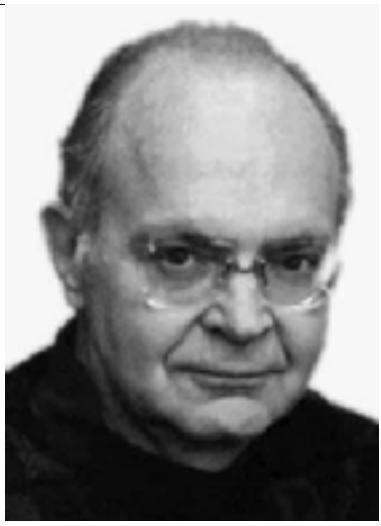
He is an ardent supporter of programming freedom, and an outspoken opponent of software patents, the DMCA and the CBDTPA. He resigned from a subgroup of Usenix in protest, and said he would not visit the United States for fear of being imprisoned after the arrest of Dmitry Sklyarov for DMCA violations.

Cox was the recipient of FSFs 2003 Award for the Advancement of Free Software at the FOSDEM conference in Brussels.

External links

- His diary (*http://www.linux.org.uk/diary/*) in Welsh
- His wife Telsa's diary (*http://www.linux.org.uk/~telsa/Diary/diary.html*)
- Interview with Alan Cox - January 15, 2002 (*http://kerneltrap.org/node/view/9*)

Donald Knuth



Donald Ervin Knuth, pronounced *ka-NOOTH* (born 10 January 1938 in Milwaukee, Wisconsin) is a foremost computer scientist and Professor Emeritus at Stanford University.

Knuth is best known as the author of the multi-volume *The Art of Computer Programming*, one of the most highly respected references in the computer science field. He practically created the field of rigorous analysis of algorithms, and made many seminal contributions to several branches of theoretical computer science. He is the creator of the T_EX typesetting system and of the Metafont font design system, and pioneered the concept of literate programming.

Knuth is considered a famous programmer, known for his geek humor: as examples, he pays a finder's fee of $2.56 for any typos/mistakes discovered in his books because *"256 pennies is one hexadecimal dollar".* (His bounty for errata in *3:16 Bible Texts Illuminated,* is, however, $3.16). Version numbers of his T_EX software approach pi, that is versions increment in the style 3, 3.1, 3.14 and so on, version numbers of Metafont approach e similarly; he once warned users of his software, *"Beware of bugs in the above code; I have only proved it correct, not tried it."* (source: *http://www-cs-faculty.stanford.edu/~knuth/faq.html*)

Knuth is the author of *3:16 Bible Texts Illuminated* (1991), ISBN 0895792524, in which he attempts to examine the Bible by a process of "stratified random sampling," namely an analysis of chapter 3, verse 16 of each book. Each verse is accompanied by a rendering in calligraphic art, contributed by a group of calligraphers under the leadership of Herman Zapf.

He received his bachelor's degree in mathematics at the Case Institute of Technology, now known as Case Western Reserve University. He earned a Ph.D. in mathematics from the California Institute of Technology in 1963. In 1968 he became a member of the faculty of Stanford University, where he was awarded the singular academic title of *Professor Emeritus of the Art of Computer Programming.* He has received various other awards including the Turing Award, the National Medal of Science, the John von Neumann Medal and the Kyoto Prize. In 2003 he was elected as a Fellow of the Royal Society.

Knuth's hobbies include music, and specifically playing the organ. He has a pipe organ installed in his home. Knuth disclaims any particular talent in the instrument, however. He does not use email, saying that he used it from about 1975 until 1 January 1990, and that was enough for one lifetime. He finds it more efficient to respond to correspondence in "batch mode", such as one day every three months, to be sent by snail mail.

He is married to Jill Knuth, who published a book on liturgy. They have two children.

Knuth published his first "scientific" article in a school magazine in 1957 under the title "Potrzebie System of Weights and Measures," part of which included defining the fundamental unit of length as the thickness of *MAD* magazine #26, and naming the fundamental unit of force "whatmeworry". *MAD* magazine bought the article and published it in the June 1957 issue.

External links

- Wikiquote - Quotations from Donald Knuth (*http://quote.wikipedia.org/wiki/Donald_Knuth*)
- The Stanford home page of Donald Knuth (*http://www-cs-faculty.stanford.edu/~knuth/*)
- Long biography of Knuth (*http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Knuth.html*)

- Donald Knuth: Leonard Euler of Computer Science (Softpanorama) (*http://www.softpanorama.org/People/Knuth/index.shtml*)

Bruce Perens

Bruce Perens is an active leader in the open source movement, with a long and distinguished record. He is a former Debian GNU/Linux Project Leader, the primary author of the Open Source Definition, a founder of Software in the Public Interest, founder of the UserLinux project, and co-founder of the Open Source Initiative. Perens also has a book series with Prentice Hall PTR called the Bruce Perens' Open Source Series.

External links

- Bruce Perens' homepage (*http://perens.com/*)

Larry Wall

Larry Wall, programmer, linguist, author, born 10 March 1949 in Duncan, British Columbia, Canada. Created 1987 the computer language Perl.

Wall is the developer of the Perl programming language. Wall is also known as the original author of the rn Usenet software, and the nearly universally used patch. He has won the IOCCC twice, and was the recipient of the first Free Software Foundations award for the Advancement of Free Software in 1998.

Beyond his technical skills, Wall is known for his wit and often ironic sense of humor which he displays in the comments to his source code or on Usenet (e.g. "We all agree on the necessity of compromise. We just can't agree on when it's necessary to compromise")

Larry Wall is a trained linguist, which helped him with his book writing, as well as with the design of Perl. He is the co-author of *Programming Perl* (often referred to as the

Camel Book), which is the definitive resource for Perl programmers. He has also edited the *Perl Cookbook*. All of the books that he has edited or co-written are published by O'Reilly.

Wall continues to oversee further development of Perl and serves as the Benevolent Dictator for Life of the Perl project.

External links

- Larry Wall's personal home page (*http://www.wall.org/~larry/*)
- Authoritative list of Larry Wall quotes (*http://www.cpan.org/misc/lwall-quotes.txt.gz*)
- Quotes by Larry Wall (*http://quote.wikipedia.org/wiki/Larry_Wall*)

Guido van Rossum

Guido van Rossum is a computer programmer who is best-known as the author of the Python programming language.

Van Rossum was born and grew up in the Netherlands. He received a master degree from the University of Amsterdam in 1982, and later worked for various research institutes, including the Dutch National Research Institute for Mathematics and Computer Science (CWI) (Amsterdam), the National Institute of Standards and Technology (NIST) (Gaithersburg, Maryland), and the Corporation for National Research Initiatives (CNRI) (Reston, Virginia). He worked on the development of the ABC programming language, a descendant of the Simula language.

Over the origin of Python, Van Rossum wrote in 1996:

> Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus). (Introduction to *Programming Python*, by Mark Lutz, published by O'Reilly)

In 1999, Van Rossum submitted a funding proposal to DARPA called *Computer Programming for Everybody*, in which he further defined his goals for Python:

- an easy and intuitive language while being just as powerful as major competitors

- open source, so anyone can contribute to its development
- code that is as understandable as plain English
- suitability for everyday tasks, allowing for short development times

many of these ambitions have since been realized. Python has grown to become a popular programming language, particularly in the Internet environment. In the Python community, Van Rossum is known as the Benevolent Dictator for Life (BDFL), meaning that he continues to oversee the Python development process, taking the ultimate decisions where necessary.

In 2002, Van Rossum received the Free Software Award of 2001 from the FSF at the FOSDEM conference in Brussels, Belgium.

External link

- Guido van Rossum's homepage (*http://www.python.org/~guido/*)
- *Computer Programming for Everybody* (*http://www.python.org/doc/essays/cp4e.html*)

Brian Behlendorf

Brian Behlendorf was a primary developer of the Apache Web server, the most popular web server software on the Internet, and a founding member of the Apache Group, which later became the Apache Software Foundation. Behlendorf served as President of the Foundation for three years, and remains on its Board of Directors.

Having grown up in Southern California near NASA's Jet Propulsion Laboratory, Behlendorf became interested in the early development of the Internet while he was a student at the University of California-Berkeley in the early '90s. In 1993, Behlendorf and Jonathan Nelson co-founded Organic, Inc., the first business dedicated to building commercial web sites. While developing the first online, for-profit, media project -- the HotWired web site for Wired Magazine -- in 1994, they realized that the most commonly used web server software at the time (developed at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign) could not handle the user registration system that the company required. So, Behlendorf patched the open-source code to support HotWired's requirements.

It turned out that Behlendorf wasn't the only one busy patching the NCSA code at the time, and he and Cliff Skolnick put together a mailing list to coordinate the work of the other programmers. By the end of February, 1995, eight core contributors to the project formed the Apache Group. Working loosely together, they eventually rewrote the entire original program as the Apache HTTP Server. In 1999, the project incorporated as the Apache Software Foundation.

Behlendorf is now the Chief Technology Officer at CollabNet, a company he co-founded in 1999 to develop tools for enabling collaborative, open-source software development.

External Links

- The Apache Software Foundation (*http://www.apache.org*)
- Personal homepage (*http://www.behlendorf.com/~brian/*)
- Organic, Inc. (*http://www.organic.com*)
- CollabNet (*http://www.collab.net/)*

Miguel de Icaza

Miguel de Icaza (born c. 1972) is a free software programmer from Mexico, best known for starting the GNOME project.

Miguel de Icaza was born in Mexico City and studied at the National Autonomous University of Mexico (UNAM). He started writing free software in 1992.

De Icaza started the GNOME project in August 1997, with Federico Mena, to create a completely free desktop environment and component model for GNU/Linux and other Unix-like operating systems. Earlier, de Icaza had worked on the Midnight Commander file manager, as well as the Linux kernel.

In 1999, de Icaza co-founded Helix Code, a GNOME-oriented free software company with Nat Friedman, and employed a large number of other GNOME hackers. In 2001, Helix Code, now renamed to Ximian, announced the Mono project, a project led by de Icaza, to implement Microsoft's new .NET development platform on Linux and Unix-like platforms. In August 2003, Ximian was acquired by Novell.

Miguel de Icaza has received the Free Software Foundation 1999 Free Software Award, the MIT Technology Review Innovator of the Year Award 1999, and was named one of Time Magazine's 100 innovators for the new century in September 2000.

External Links

- Interview with de Icaza (*http://www.linuxjournal.com/article.php?sid=6833*)
- Miguel de Icaza's blog (*http://primates.ximian.com/~miguel/all.html*)

Volker Grassmuck

Volker Grassmuck (*1961 in Hannover) is a German sociologist and media researcher.

Volker Grassmuck visited Herschelschool in Hannover, spent a year at Ridgewood High in New Jersey and finished his Abitur back at Herschelschool in 1980.

He started studying sociology in 1981 in Groningen, Nederlands at the Rijkuniversiteit Groningen but changed to Berlin in 1982. There he studied sociology, journalism, information science and psychology at the Freie Universität Berlin.

In the mid-1980s he worked alongside his studies publicist and founded the "JetSet Verlags GmbH" in 1984, which published the magazine "V max - Zeitschrift auf der Überholspur". He also worked at the local radio station "Radio 100" as editor for the radio show "Nachtflug".

Grassmuck started his academic career in 1987 with contributions to a research project and also graduated his study and started studying Japanese at the FU Berlin as well.

From 1989 he did research at the Socio-technological Research Department of Tokyo University. From 1991 he published a column in the "konpyûta kagaku" (Shujunsha) and worked as a video-editor for ABC News. 1992 found him as a newscaster at Radio Japan, NHK and worked as a freelancer for InterCommunication Magazin, NTT Shuppansha. At the university he researched networks with Dr. Kubota Akihiro.

Grassmuck returned to Berlin in 1995, starting, together with others, "mikro e.V." in 1998, a project for connecting Berlin's media cultures. He worked together with Prof. Dr. Wolfang Coy on a DFG research project on "Von der Ordnung des Wissens zur Wissensordnung digitaler Medien" ("From the system of knowledge to the knowledge system of digital media") at the Humboldt Universität.

He earned a doctorate at the FU Berlin on Japanese media history with the topic "Closed Society. Media and discursive aspects of Japan's 'three openings'". In 2000/2001 he became a replacement professor for media art at the Universität Graphik und Buchkunst in Leipzig.

Grassmuck organized the conference Wizards of OS which topics included operating systems, open sources and open contents. He holds regular lectures at congresses of the German Chaos Computer Club and is engaged in new forms of copyright like Wissensallemende or GNU.

External Links

- http://waste.informatik.hu-berlin.de/Grassmuck/ - Personal website
- http://www.mikro.org/ mikro e.V.

- http://www.mikro.org/Events/OS Wizards of OS

General Movements

Hacker

A **hacker** is anyone who enjoys the intellectual challenge of creatively overcoming or circumventing limitations, primarily in their fields of interest, namely programming or electrical engineering. As will be discussed below, there is a trend in the popular press to use the term to describe computer criminals (which some call crackers), and others, whose actions run afoul of various governments. This trend annoys some old-school computer/technology enthusiasts, although not all old-school hackers - some of them, like Steve Wozniak, hacked regardless of whether their activities were legal or illegal.

Hacker History Begins

"Hacker"

Among ham radio fans in the 1950s, *hacking* meant creatively tinkering to improve performance. It was a term borrowed from Anglo-American riding culture, where "hacking" (as opposed to fox-hunting) meant riding about informally, to no particular purpose. Compare "hacking jacket". On the U.S. East Coast in the mid-1950s, a car could be substituted for a horse, and *hacking* was a precursor to *cruising.* As the term originally developed at MIT long before computers became common; a "hack" meant a simple, but often inelegant, solution. The term ***hack*** came to refer to any clever prank perpetrated by MIT students; the perpetrator is a **hacker**. To this day the terms *hack* and *hacker* are used in that way at MIT, without necessarily referring to computers. When MIT students surreptitiously put a police car atop the dome on MIT's Building 10, that was a hack, and the students involved were therefore hackers.

In the nascent computer culture of the 1960s, the unavoidable analogy to "hacking" programs was the already-established counter-culture practice of *chopping* Harley-Davidsons in Southern California: taking them apart and "chopping" their frames, improvising to make them lower, sleeker, faster, hotter than their uncustomized "stock" originals.

Computer culture at MIT developed when members of the Tech Model Railroad Club started working with a Digital Equipment Corporation PDP-1 computer and applied local model railroad slang to computers. In modern computer culture, the label "hacker" is a compliment, indicating a skilled and clever programmer. In the media, however, it has negative connotations and has become synonymous with "software cracker".

The term **hacker** has six meanings that are in common usage:

1. Someone who knows a (sometimes specified) set of programming interfaces well enough to write novel and useful software without conscious thought on a good day.
2. Someone who (usually illegally) attempts to break into or otherwise subvert the security of a program, system or network, sometimes with malicious intent. This usage was annoying to some in the developer community who grew up with the primary meaning in sense (1), and preferred to keep it that way; they preferred the media used the term cracker. However this wound up causing even more problems as simply creating a new word did nothing to dispel misconceptions. "Black hat hacker" is a phrase that wound up with the same problems as the word "cracker".
3. Someone who gains access to systems and networks, and makes small and harmless changes. This type of vandalism is usually done for amusement or recognition. These type of hackers are called "gray hat hackers" or sometimes just "grays".
4. Someone who attempts to break into systems or networks in order to help the owners of the system by making them aware of security flaws in it. This is referred to by some as a "white hat hacker" or sneaker. Many of these people are employed by computer security companies, and are doing something completely legal; and many were formerly hackers within sense 2.
5. Someone who, through either knowledge or trial and error, makes a modification to an existing piece of software, made available to the hacker community, such that it provides a change of functionality. Such change is normally a benefit. Rather than a competition, the exchange of improvements is most often experienced as a cooperative learning effort.
6. A Reality Hacker or Urban Spelunker (origin: MIT); someone who enjoys exploring air ducts, rooftops, shafts and other hidden aspects of urban life, sometimes including pulling elaborate pranks for the enjoyment and entertainment of the community.

"Script kiddie" is reserved for a computer user of little or no skill who simply follows directions or uses a cook-book approach without fully understanding the meaning of the steps they are performing.

Note that while the term **hacker** denotes competence, the noun hack often means kludge and thus has a negative connotation while the verb hack generally shares the same competent connotations.

The hacker community (the set of people who would describe themselves as hackers, or who would be described by others as hackers) falls into at least three partially overlapping categories. The word *hacker* probably derives from the somewhat derogatory *hack*, used in the newspaper industry typically to refer to a Journalist who types his stories without checking his facts first.

HACKER: BRILLIANT PROGRAMMER

One who knows a (sometimes specified) set of programming interfaces well enough to write novel and useful software without conscious thought on a good day. This type of hacker is respected within the development community for the freedom they represent, although the term still carries some of the meaning of Hack, developing programs without adequate planning. This *zugzwang* sets freedom and the ability to be creative against methodical careful progress. Corporate programming environments typically favor only either the good hackers or the careful computer scientist.

At their best, Hackers can be very productive. The downside of Hacker productivity is generally agreed to be in maintainability, documentation, and completion. Very talented hackers may become bored with a project once they have figured out all of the hard parts, and be unwilling to finish off the *details*. This attitude can cause friction in shops where other programmers are expected to pick up the half finished work, decipher the structures and ideas, and bullet-proof the code. In other cases, where a Hacker is willing to maintain their own code, a company may be unable to find anyone else who is capable or willing to dig through code to maintain the program if the original programmer moves on to a new job.

HACKER: CRIMINALIZED BY GOVERNMENT

The popular press has been known to use the terms "hacker" and occasionally "cracker" for someone who attempts to break into or otherwise subvert the security of a system or network. Both usages are annoying to some in the developer community who grew up with the primary meaning of "hacker" in the Guru sense, and who don't see the problem solved by the invention of new and nebulous words like "cracker" or "black hat". Instead, there has been a move to define terms when describing these people.

While it is possible to use one's "hacker" skills in an illegal way, this tends to go against the loosely defined hacker ethic. One can certainly use hacking skills to commit a crime. However, this means that this particular hacker is now a criminal, sometimes called the nebulous term "cracker".

Software cracking is the process of removing any sort of software enforced protection scheme from a piece of software.

There are several recurring tools of the trade used by hackers to gain unauthorized access to computers:

- Trojan horse -- These are applications that seem to do useful work, but set up a back door so that the hacker can later return and enter the system. These include programs which mimic login screens. Viruses that fool a user into downloading and/or executing them by pretending to be useful applications are also sometimes called trojan horses.

- Snooper -- Applications that capture password and other data while it is in transit either within the computer, or over the network
- Virus -- An application that propagates itself opportunistically by waiting in the background until the user offers it a new medium to infect. The term came into usage by comparison with biological viruses, which reproduce by infecting a cell and taking advantage of its life functions. Similarly, computer viruses, unlike worms, embed themselves within files on the host system. When "infected" executables run, or sometimes when infected binary data files are read, the virus is able to spread to other binary format files on the local system, floppy disks or over the network. Viruses are often confused with worms.
- Worm -- An application that actively probes for known weaknesses across the network, then propagates itself through an exploitation of those weaknesses. The original Usenet post describing the Morris Worm described the distinction between viruses and worms thus: worms do not attach themselves to code. Popular usage appears to favour worms being more active than viruses. However, the Jargon File, as of version 4.4.1, maintains the original sense of the term. A Worm in this original sense is any independent program which reproduces itself over a network (a program reproducing itself on the local machine only repeatedly until the machine crashes is known as a wabbit). After the comparison between computer viruses and biological viruses, the obvious comparison here is to a bacterium.
- Vulnerability scanner -- A tool used to quickly check computers on a network for known weaknesses. Hackers also use port scanners. These check to see which ports on a specified computer are "open" or available to access the computer through.
- Exploit -- A prepared application that takes advantage of a known weakness
- Social engineering -- Asking someone for the password or account (possibly over a beer.) Also includes looking over someone's shoulder while they enter their password, or posing as someone else in order to get sensitive information.
- Root kit -- A toolkit for hiding the fact that a computer's security has been compromised. Root kits may include replacements for system binaries so that it becomes impossible to see applications being run by the intruder in the active process tables.
- Leet -- An English pidgin that helps to obscure hacker discussions and web sites, and paradoxically it simplifies the location of resources in public search engines for those who know the language.

HACKER: GREY HAT

1. A black-hat hacker turned white-hat. See below.
2. A white-hat hacker who uses black-hat techniques to satisfy their employers, for whom they act as white-hat.

Hacker: White hat

White hat hackers often overlap with black hat depending on your perspective. The primary difference is that a white hat hacker claim they observe the hacker ethic, a sort of golden rule of computing similar to: Do unto others as you would have them do unto you. Like black hats, white hats are often intimately familiar with the internal details of security systems, and can delve into obscure machine code when needed to find a solution to a tricky problem without requiring support from a system manufacturer.

An example of a hack: Microsoft Windows ships with the ability to use cryptographic libraries built into the operating system. When shipped overseas this feature becomes nearly useless as the operating system will refuse to load cryptographic libraries that haven't been signed by Microsoft, and Microsoft will not sign a library unless the US Government authorizes it for export. This allows the US Government to maintain some perceived level of control over the use of strong cryptography beyond its borders.

While hunting through the symbol table of a beta release of Windows, a couple of overseas hackers managed to find a second signing key in the Microsoft binaries. That is without disabling the libraries that are included with Windows (even overseas) these individuals learned of a way to trick the operating system into loading a library that hadn't been signed by Microsoft, thus enabling the functionality which had been lost to non-US users.

Whether this is good (white hat) or bad (black hat) may depend on whether you are the US Government or not, but is considered by some of the computing community to be a white hat type of activity.

How some hackers define themselves

The following is the definition given by the jargon file (a dictionary of hacker jargon) accepted by some (but not all) in the hacker community:

hacker n. [originally, someone who makes furniture with an axe]

1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.
2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming.
3. A person capable of appreciating hack value.
4. A person who is good at programming quickly.
5. An expert at a particular program, or one who frequently does work using it or on it; as in `a Unix hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.)

6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example.
7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations.
8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence `password hacker', `network hacker'. The correct term for this sense is cracker.

The term `hacker' also tends to connote membership in the global community defined by the net (see the network and Internet address). For discussion of some of the basics of this culture, see the How To Become A Hacker FAQ. It also implies that the person described is seen to subscribe to some version of the hacker ethic. It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you'll quickly be labeled bogus). See also geek, wannabe. This term seems to have been first adopted as a badge in the 1960s by the hacker culture surrounding TMRC and the MIT AI Lab. We have a report that it was used in a sense close to this entry's by teenage radio hams and electronics tinkerers in the mid-1950s.

Notable hackers

- Richard Greenblatt
- Bill Gosper
- Richard Stallman -- A hacker of the old school, Stallman walked in off the street and got a job at MIT's Artificial Intelligence Lab in 1971. Stallman is a legendary hacker, the founder of the free software movement, a MacArthur "genius grant" recipient and a programmer capable of prodigious exploits. Stallman is also the founder of the GNU project, which produced the majority of the software considered to be part of the Linux operating system.
- Ken Thompson and Dennis Ritchie -- The driving creative force behind Bell Labs' legendary computer science operating group, Ritchie and Thompson created UNIX in 1969.
- Bill Joy -- Co-founder of Sun Microsystems and author of many fundamental UNIX utilities.
- Steve Wozniak -- The co-founder of Apple Computer got his start making devices for phone phreaking.
- Linus Torvalds -- Torvalds was a computer science student at the University of Helsinki when he wrote the Linux kernel in 1991.

- Eric S. Raymond -- He is one of the founders of the Open Source Initiative. He wrote the famous text The Cathedral and the Bazaar and many other essays. He also maintains the Jargon File for the Hacker culture, which was previously maintained by Guy L. Steele, Jr..
- Larry Wall -- The creator of the Perl programming language.
- Johan "Julf" Helsingius -- Operated the world's most popular anonymous remailer, the Penet remailer (called penet.fi), until he closed up shop in September 1996.
- Tsutomu Shimomura -- Shimomura outhacked and outsmarted Kevin Mitnick, the United States's most infamous malicious cracker, in early 1994.
- Fyodor -- The author of Nmap.
- Solar Designer -- Founder of the Openwall Project.

Notable hackers who ran afoul of a government

Here are a few of the more famous hackers (many of whom have since turned to fully legal hacking):

- Dark Avenger (a pseudonym) -- Bulgarian virus writer that invented polymorphic code in 1992 as a mean to circumvent the type of pattern recognition used by Antivirus software, and nowadays also intrusion detection systems.
- Eric Corley (a.k.a Emmanuel Goldstein) -- Long standing publisher of 2600 the Hacker Quarterly. He has been part of the hacker community since the late 70's.
- John Draper (a.k.a. Captain Crunch) -- Often cited as having figured out how to make free phone calls using a plastic prize whistle he found in a cereal box. It was actually friends of his who discovered this, he just adopted the name. (See phreaking.)
- Mark Abene (a.k.a. Phiber Optik) -- Inspired thousands of teenagers around the country to "study" the internal workings of the United States's phone system. One of the founders of Masters of Deception.

Adrian Lamo -- Revised a Yahoo! news article and was prosecuted for a New York Times break-in.

- Robert Tappan Morris, Jr. -- This Cornell University graduate student unleashed the first major Internet worm in 1988.
- Kevin Mitnick -- The first hacker to have his face immortalized on an FBI "Most Wanted" poster.
- Kevin Poulsen -- In 1990 Poulsen took over all telephone lines going into Los Angeles area radio station KIIS-FM to win a call-in contest.
- Vladimir Levin -- This mathematician allegedly masterminded the Russian hacker gang that tricked Citibank's computers into spitting out $10 million.

- Xail -- At age 15, Xail hacked the nasa.gov domain. Sent to a state-run prison camp for juveniles in Erie, Pennsylvania. Released in 2002.

External links

- The Hacker Dictionary (*http://www.hacker-dictionary.com*)
- The MIT Gallery of Hacks (*http://hacks.mit.edu/*)
- Hacker News (*http://www.hackwire.com/*)
- Hacker Shirts & Stickers (*http://www.hackerstickers.com/*)
- The Jargon File (*http://www.catb.org/~esr/jargon*)
- The Hacker Emblem (*http://www.catb.org/~esr/hacker-emblem*)
- How To Become A Hacker (*http://www.catb.org/~esr/faqs/hacker-howto.html*)
- SecureRoot (*http://www.secureroot.com*)
- Open Source Initiative (*http://www.opensource.org*)
- Paul Graham's *Hackers & Painters* Essay (*http://www.paulgraham.com/hp.html*)

Hacker community

A **hacker community** is a group of programmers who share code, exchange improvements and teach one another "tricks" or better methods or writing. "Hacking" in this sense does not have anything to do with illegal computer activity; instead it connotes clever and useful solutions to legitimate computer problems.*(See: Hacker (Brilliant Programmer))*

Probably the most notable hacker community is the community of open source/free software programmers. In this community, Richard Stallman and Linus Torvalds are two of the most well-known hackers.

People contribute to such a community for various reasons, like making useful contributions where they can, wanting to replace proprietary software with open code, or being a part of a larger group.

The Internet plays a key role in hacker communities; it allows people from around the world to collaborate on a project.

In a sense, Wikipedia can be viewed as a hacker community.

Hacker culture

The **hacker culture** is the voluntary subculture which first developed in the 1960s among hackers working on early minicomputers in academic computer science environments. After

1969 it fused with the technical culture of the pioneers of the Internet, after 1980 with the culture of Unix, and after 1987 with elements of the early microcomputer hobbyists. Since the mid-1990s the hacker culture has been almost coincident with what is now called the open source movement.

History

As the above implies, it was not always appropriate to speak of a single hacker culture. Before the computing world was as networked as it is now, there were multiple independent and parallel hacker cultures, often unaware or only half-aware of each others' existence. All of these had certain important traits in common:

- placing a high value on freedom of inquiry; hostility to secrecy
- information-sharing as both an ideal and a practical strategy
- upholding the right to fork
- playfulness, taking the serious humorously and their humor seriously

These sorts of cultures were commonly found at academic settings such as college campuses. The MIT AI lab, the University of California, Berkeley and Carnegie-Mellon University were particularly well-known hotbeds of early hacker culture. They evolved in parallel, and largely unconsciously, until the Internet and other developments such as the rise of the free software movement drew together a critically large population and encouraged the spread of a conscious, common, and systematic ethos. Symptomatic of this evolution was an increasing adoption of common slang and a shared view of history, similar to the way in which other occupational groups have professionalized themselves but without the formal credentialling process characteristic of most profesional groups.

Over time, the hacker culture has tended to become more conscious, more cohesive, and better organized. The most important consciousness-raising moments have included the composition of the first Jargon File in 1973, the promulgation of the GNU Manifesto in 1985, and the publication of *The Cathedral and the Bazaar* in 1997. Correlated with this has been the gradual election of a set of shared culture heroes; first and arguably foremost Richard M. Stallman, also (in alphabetical order) Bill Joy, Eric S. Raymond, Dennis Ritchie, Ken Thompson, Linus Torvalds, and Larry Wall, among others.

The concentration of hacker culture has paralleled and partly been driven by the commoditization of computer and networking technology, and has in turn accelerated that process. In 1975 hackerdom was scattered across several different families of operating systems and disparate networks; today it is almost entirely a Unix and TCP/IP phenomenon, and is increasingly concentrated around Linux.

Artifacts and customs

The hacker culture is defined by shared work and play focused around central artifacts. Some of these artifacts are very large; the Internet itself, the World Wide Web, the GNU project, and the Linux operating system are all hacker creations, works of which the culture considers itself primary custodian. The Wikipedia itself can be considered an artifact of hacker culture.

Since 1990 the hacker culture has developed a rich range of symbols that serve as recognition symbols and reinforce its group identity. Tux, the Linux penguin, the BSD demon, and the Perl camel stand out as examples. More recently, the use of the glider sructure from Conway's Game of Life as a general Hacker Emblem has been proposed and appears to be gaining acceptance. All of these routinely adorn T-shirts, mugs, and other paraphernalia.

Notably, the hacker culture appears to have exactly one annual ceremonial day—April Fool's. There is a long tradition of perpetrating elaborate jokes, hoaxes, pranks and fake websites on this date. This is so well established that hackers look forward every year to the publication of the annual joke RFC, and one is invariably produced making this movement very closed related to activism.

External links

- A Brief History of Hackerdom (*http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/*) - more depth on the history of hackerdom
- Bruce Sterling (*http://www.egs.edu/faculty/sterling.html*) wrote The Hacker Crackdown plus a wide variety of Cyberculture texts.
- How To Become a Hacker (*http://www.catb.org/~esr/faqs/hacker-howto.html*), by Eric S. Raymond

Hacker Manifesto

The Conscience of a Hacker (a.k.a. ***The Hacker Manifesto***) is a small article written on 8 January 1986 by a hacker who went by the handle, or pseudonym, of The Mentor. It was written after the author's arrest, and first published in the underground hacker ezine Phrack in Volume One, Issue 7, Phile 3 of 10. Today is can be found on countless websites.

It is considered an important item of hacker culture, and it gives an insight into the psychology of early hackers. The Manifesto states that hackers choose to hack because it is a way for them to learn, because they are frustrated and bored in school. It also expresses the satori of a hacker realizing his potential in the realm of computers.

The article is quoted in the 1995 movie *Hackers*, although in the movie it is being read from an issue of the hacker magazine 2600, not a printout of Phrack.

External Links

- Wikisource:Hacker's Manifesto (*http://sources.wikipedia.org/wiki/Hacker%27s_Manifesto*).
- Elfqrin.com interview with The Mentor (July 31, 2000) (*http://www.elfqrin.com/docs/hakref/interviews/eq-i-mentor.html*)

Open source movement

The **open source movement** is an offshoot of the free software movement that advocates open-source software as an alternative label for free software, primarily on pragmatic rather than philosophical grounds.

The movement was founded in 1998 by John maddog Hall, Larry Augustin, Eric S. Raymond, Bruce Perens, and others. Raymond is probably the single person most identified with the movement; he was and remains its self-described principal "theorist", but does not claim to lead it in any exclusive sense. In contrast with the free software movement, which has always been essentially directed by a single figure (Richard Stallman), the open source movement is "steered" by a loose collegium of elders that includes Raymond, its other co-founders, and such notables as Linus Torvalds, Larry Wall, and Guido van Rossum.

The founders were dissatisfied with what they saw as the "confrontational attitude" of the free software movement, and favored advocating free software exclusively on the grounds of technical superiority (a claim previously made by Raymond in his essay *The Cathedral and the Bazaar*) It was hoped that "open source" and the associated propaganda would become a more persuasive argument to businesses. Raymond's comment was "If you want to change the world, you have to co-opt the people who write the big checks." (Cygnus Support had been pursuing exactly this approach for a number of years already, but not advertising it widely.)

The group adopted the Open Source Definition for open-source software, based on the Debian Free Software Guidelines. They also established the Open Source Initiative (OSI) as a steward organization for the movement. However, they were unsuccessful in their attempt to secure a trademark for "open source", to act as an imprimatur and to prevent misuse of the term. Despite this, the OSI developed considerable influence in the corporate sphere and has been able to hold abuse of the term to a tolerable minimum through vigorous jawboning. With the FSF, it has become one of the hacker community's two principal advocacy organizations.

The early period of the open-source movement coincided with and partly drove the dot-com boom of 1998-2000, and saw a large growth in the popularity of Linux and the formation of many "open-source-friendly" companies. The movement also caught the attention of the mainstream software industry, leading to open-source software offerings by established software companies such as Corel (Corel Linux), Sun Microsystems (StarOffice), and IBM (OpenAFS). By the time the dot-com boom busted in 2001, many of the early hopes of open-source advocates had already borne fruit, and the movement continued from strength to strength in the cost-cutting climate of the 2001-2003 recession.

Relations with the Free Software Movement

Since its inception, the open source movement has been a matter of controversy within the hacker community.

Stallman, speaking for the Free Software Foundation (FSF), has criticized the motivation of the open source movement. According to him, the pragmatic focus of the movement distracts users from the central moral issues and the freedoms offered by free software, blurring the distinction with semi-free or wholly proprietary software. Stallman describes the free software and the open source movements as separate "political camps" within the same free-software community, however, and says: *We disagree on the basic principles, but agree more or less on the practical recommendations. So we can and do work together on many specific projects.*

Both free-software and open-source advocates have rallied together in times of crisis, such as Microsoft's intense attacks on the GPL in 2001 and the SCO lawsuit attacking the Linux kernel in 2003. Indeed, there is not a strict division between the two movements, as many individuals identify to some extent with both groups (although some, like Stallman, espouse one of the two philosophies exclusively).

Tensions between the two communities have occasionally been exacerbated by a habit in the trade press and elsewhere of casting their differences as a personal drama between Stallman and open-source notables such as Raymond or Torvalds.

In practice, the operational definitions of free software and open-source software are the same. The lists of compliant licenses maintained by the FSF and OSI are nearly identical, differing only in corner cases such as the first version of the APSL. Adherents of the free-software and open-source movements typically have no difficulty cooperating on software projects.

Open source vs. free software thus joins the list of philosophical (and generally harmless) divisions amongst hackers, alongside the editor wars and GNOME vs KDE.

Open Source Culture?

Some in the open source movement have claimed that open source principles can be applied to technical areas other than computer software, such as digital communication protocols and data storage formats or even open source hardware (for instance the Indian development simputer). Bolder claims extend open source ideas to entirely different fields, such as the dissemination of general knowledge.

Proponents of this view have hailed the Open CourseWare project at MIT, Thacker's article on "Open Source DNA", the "Open Source Cultural Database", openwebschool, and the Wikipedia as examples of applying open source outside the realm of computer software. Skeptics have pointed out that the sharing principle predates the open source movement; for example, the free sharing of information has been institutionalized in the scientific enterprise since at least the 19th century. Raymond and other founders of the movement have sometimes publicly tried to put the brakes on speculation about applications outside of software, arguing that strong arguments for software openness should not be weakened by overreaching into areas where the story is less compelling.

The broader impacts of the open source movement, and the extent of its role in the development of new information sharing procedures, remains to be seen.

External links

- OSI's history of the open source movement (*http://www.opensource.org/docs/history.html*)
- Stallman's criticism of the open source movement (*http://www.gnu.org/philosophy/free-software-for-freedom.html*)
- MIT's OpenCourseWare project (*http://education.mit.edu/tep/11125/opencourse/*)
- Thacker on "Open Source DNA" (*http://www.mikro.org/Events/OS/text/Eugene-Thacker_OSDNA.htm*)
- McCormick on the Open Source Cultural Database (*http://www.opencritic.com/texts/CPSR_pattern.htm*)
- "Lessons from Open Source", (*http://www.firstmonday.org/issues/issue6_6/newmarch/index.html*) by Jan Shafer

IMPORTANT ORANISATIONS

GNU

GNU is a recursive acronym for "GNU's Not Unix". The **GNU project** was launched by Richard Stallman with the goal of creating a complete free operating system: the **GNU system**. Stallman requests that it be pronounced *guh-NOO* to "avoid horrible confusion" with the word "new". UNIX is a proprietary operating system that was already in widespread use; since its architecture had proven technically sound, the GNU system was designed to be compatible with it. The UNIX architecture allowed GNU to be written as individual software components: components that were already freely available, such as the TeX typesetting system and the X Window graphics system, could be adapted and reused; others would be written from scratch.

HISTORY

The project was announced to the public on September 27, 1983, on the net.unix-wizards and net.usoft newsgroups. Work on the project began in earnest on January 5, 1984, when Stallman quit his job at MIT so that they could not claim ownership and interfere with distributing GNU as free software. The original announcement was followed by Stallman's "GNU Manifesto" and other essays that laid out his motivations for the GNU project, one of which was to "bring back the cooperative spirit that prevailed in the computing community in earlier days."

To ensure that GNU software would remain free for all users "to run, copy, modify and distribute," the project would release it under a license designed to give everyone those permissions while preventing them from adding restrictions of their own. This idea, referred to as copyleft, was then embodied in the GNU General Public License (GPL).

In 1985, Stallman founded the Free Software Foundation (FSF), a tax-exempt charity, to provide logistical, legal and financial support for the GNU project. The FSF also employed programmers to contribute to GNU, though a substantial portion of development was (and continues to be) performed by volunteers. As GNU gained prominence, interested businesses began contributing to development or selling GNU software and technical support. The most prominent and successful of these was Cygnus Solutions, now part of Red Hat.

By 1990, the GNU system had an extensible text editor (Emacs), a very successful optimizing compiler (GCC), and most of the core libraries and utilities of a standard UNIX distribution. The main component still missing was the kernel.

In the GNU Manifesto, Stallman had mentioned that "an initial kernel exists but many more features are needed to emulate Unix." He was referring to TRIX, a remote procedure call kernel developed at MIT, whose authors had decided to distribute for free, and was compatible with UNIX version 7. In December 1986 work had started on modifying this kernel. However, the developers eventually decided it was unusable as a starting point, primarily because it only ran on "an obscure, expensive 68000 box" and would therefore have to be ported to other architectures before it could be used. By 1988, the Mach message-passing kernel being developed at CMU was being considered instead, although it was initially delayed while its developers removed code owned by AT&T. Initially, the kernel was to be called Alix, but developer Michael Bushnell later preferred the name Hurd, so the Alix name was moved to a subsystem and eventually dropped completely. Eventually, development of the Hurd had stalled due to technical and personality conflicts.

In 1991, Linus Torvalds wrote the UNIX-compatible Linux kernel. Although it was not originally free (as in freedom) software, in 1992 Torvalds changed the license to the GNU GPL. Linux was further developed by various programmers over the Internet. In 1992, Linux was combined with the GNU system, resulting in a fully functional free operating system. The GNU system is most commonly encountered in this form, usually referred to as a "GNU/Linux system" or a "Linux distribution." As of 2004, the Hurd is still in active development, and an experimental version (*http://www.debian.org/ports/hurd/*) of the GNU system that uses the Hurd instead of Linux is now available. There is also a project working on porting the GNU system to the kernel of FreeBSD.

It is also common to find components of GNU installed on proprietary UNIX systems, in place of the original UNIX programs. This is because many of the programs written for the GNU project have proven to be of a superior quality to the equivalent UNIX versions. Often, these components are collectively referred to as the "GNU Tools". Many GNU programs have also been ported to Microsoft Windows and Mac OS X platforms.

GNU SOFTWARE

Some of the software developed by the GNU project are:

- Bison - parser generator intended to replace yacc
- Bash - command shell
- BFD - object file library
- Classpath - libraries for Java
- DotGNU - replacement for .NET

- Emacs - extensible, self-documenting text editor
- GIMP - image-editing program
- glibc - Standard POSIX C library, plus additional functionality
- GMP - arbitrary precision numerical calculation programming library
- GNOME - graphical desktop environment
- The GNU toolchain for software development:
    - GNU Binutils - GNU Assembler, GNU Linker, and related tools
    - GNU build system - Automake, Autoconf, Libtool
    - GCC - optimizing compiler for many languages, including C, C++, Fortran, Ada, and Java.
    - GDB - debugger
- GNU MDK - a development kit for programming in MIX
- GNU Octave - a program for numerical computations similar to MATLAB
- GNU Robots - small but addictive game for computer programmers
- GNUnet - decentralized, peer-to-peer communication network designed to be resistant to censorship
- GNUstep - implementation of the OpenStep standard for a set of libraries and development tools for graphical applications
- GSL - the GNU Scientific Library
- Guile - embeddable Scheme interpreter
- Gzip - a library and program for data compression
- GNU Hurd - a microkernel-based set of servers that perform the same function as a UNIX kernel
- Maxima - a computer algebra system
- Texinfo - documentation system for producing online and printed manuals
- GNU wget - advanced file retrieval from networks and the Internet.

The GNU project also distributes and assists with the development of other packages which originated elsewhere, e.g.:

- CVS - source code control
- DDD - graphical frontend for debuggers
- eCos - small operating system for embedded devices

As of January 2004, there are a total of 260 projects under the GNU project.

External links

- Official Website: *http://www.gnu.org*
- GNU-friends, a discussion forum (*http://www.gnu-friends.org/special/about*)
- Sourceforge ports of GNU utilities for Win32 (*http://unxutils.sourceforge.net/*)
- Sourceforge like site of the GNU (*https://savannah.gnu.org/*)

Open Source Initiative

The **Open Source Initiative** is an organization dedicated to promoting open source software. It was founded in February 1998 by Bruce Perens and Eric S. Raymond.

Background

In 1997, Eric S. Raymond presented his revolutionary paper on software engineering, The Cathedral and the Bazaar, which sought to show the engineering advantages of the approach used to write the Linux kernel.

In early 1998, Netscape Communications Corporation, working with Raymond, published the source code for its flagship Netscape Communicator product as free software, due to lowering profit and hard competition with the Microsoft Internet Explorer software.

A group of people interested in free software and GNU/Linux decided to introduce a new marketing term for free software, seeking to position it as business friendly and less ideologically loaded when competing with proprietary software. This led to creating the term Open Source and a schism with Richard Stallman and his Free Software Foundation.

Successes

- The term "Open Source" achieved much press coverage from 1998 to 2000, although it was often misunderstood.
- Numerous enterprises opened to the thought of an alternative open source operating system.
- The Open Source Initiative was able to publish a number of internal Microsoft memos, the Halloween documents, that showed Microsoft was an opponent of GNU/Linux and suggested various methods of eliminate the threat of open source software. *See also* Embrace, extend and extinguish.

Present

The Open Source Initiative is still active, although not publicly visible in recent times. Its president, Eric S. Raymond, from time to time publishes comments on current community news.

The official website is *http://www.opensource.org/*

Open Source Development Network

The **Open Source Development Network** (**OSDN**) describes itself as a "news, collaboration and distribution community for IT and Open Source development, implementation and innovation." OSDN is supported by VA Software and dedicated to the Open Source Initiative. It is an attempt to bring together talent to create software that conforms to the GNU license model promoted by the Free Software Foundation.

The best known websites and projects in OSDN are:

- freshmeat.net
- geocrawler.com
- linux.com
- linuxgram.com
- newsforge.com
- slashcode.com
- slashdot.com
- sourceforge.net
- themes.org
- thinkgeek.com

OSDN dropped kuro5hin.org sometime in late 2001/early 2002 because the subject matter of the kuro5hin.org news site had become increasingly less about nerd stuff and increasingly more about politics, philosophy, the workplace, and other 'liberal arts fluff' that many engineers dislike.

OSDN can be reached unter *http://www.osdn.com/*

Free Software Foundation

Free Software Foundation (**FSF**) is a non-profit organisation founded in 1985 by Richard Stallman to support the free software movement (free as in freedom), and in particular the GNU project.

From its founding until the mid-1990s FSF's funds were mostly used to employ software developers to write free software. Since the mid- to late-1990's there are now many companies and individuals writing free software, so FSF's employees and volunteers mostly work on legal and structural issues for the free software community.

Current work of FSF

GPL Enforcement

- FSF have the resources and the will to enforce the GPL and other GNU licenses. FSF handles around 50 GPL violations per year and tries to bring the other party into compliance without involving the courts. As of January 2004, no one has yet taken FSF to court over a copyright dispute.

GNU Licenses

- The GNU GPL is the most widely used license for Free Software projects. The current version (version 2) was released in 1991 but FSF are working on a version 3. FSF have also published the GNU Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL).

Guardian of copyrights

- FSF holds the copyrights to all GNU software and some non-GNU Free Software. They require copyright assignment papers from each contributor to GNU packages so that they can defend the software in court if a dispute arises, and so that if there is a need to change the license of a work, it can be done without having to contact all contributors that have ever worked on the software.

Maintaining the Free Software Definition

- FSF maintain many of the documents that define the Free Software movement

Legal Education

- FSF hold seminars about legal aspects of using the GPL, and offers a consultancy service for lawyers.

Project Hosting

- FSF provide project hosting via their Savannah (*http://savannah.gnu.org*) website.

FSF Award for the Advancement of Free Software

- An annual award.

Structure

Membership

On 25 November 2002 the FSF launched the FSF Associate Membership program for individuals. In April 2004 they had over 2044 members. On 5 March 2003 they launched a Corporate Patronage program for commercial entities. As of April 2004, they have 45 corporate patrons.

Organizational

FSF has a board of directors with six members:

- Geoffery Knauth, Senior Software Engineer at SFA, Inc.
- Lawrence Lessig, Professor of Law at Stanford University
- Eben Moglen, Professor of Law and Legal History at Columbia University
- Henri Poole, Founder of CivicActions, a grassroots campaign technology consulting firm.
- Richard Stallman, Founder of FSF and the GNU Project and author of the GNU GPL, Versions 1 and 2
- Gerald Sussman, Professor of Computer Science at the Massachusetts Institute of Technology

Other positions held:

- Richard Stallman: President
- Bradley Kuhn: Vice President, CEO
- Eben Moglen: General Counsel
- Dan Ravicher: Senior Counsel
- Lisa "Opus" Goldstein: Business Manager
- David "Novalis" Turner: GPL Compliance Engineer
- Janet Casey, Free Software Directory maintainer
- John Sullivan, programs administrator
- Ted Teah, Copyright Assignments clerk
- Ravi Khanna
- Ted Teah

Previous employees:

- Leslie Proctor: Public Relations
- Robert J. Chassell: Founding Director and Treasurer

- Tim Ney CEO 1998-2001
- Thomas Bushnell GNU hacker, GNU Hurd
- Roland McGrath GNU hacker, GNU Libc, Make, GNU Hurd
- Leonard Tower GNU hacker
- Mike Haertel GNU hacker, diff, grep
- Pete TerMaat GNU hacker, GDB
- Phil Nelson GNU hacker
- Jay Fenlason GNU hacker, sed
- Brian Fox GNU hacker, Bash
- Noboyuki Hikichi GNU hacker
- Paul Rubin GNU hacker, cpp
- Ariel Rios GNU hacker, Guile
- there was a "Steve"
- and a "John" (he organised the digital-speech (*http://www.digitalspeech.org*) wing)

There are usually around 12 employees in the headquarters in Boston, Massachusetts. The office is managed by Bradley Kuhn.

The FSF can be reached under *http://www.fsf.org*

Free Software Foundation Europe

Founded in 2001, **Free Software Foundation Europe** (FSFeurope) is a sister organisation to the US-based Free Software Foundation. The offices of FSFeurope are in Germany where it has one full time employee and a number of part time volunteers. Its president is Georg Greve.

Most of FSFeurope's work involves the political defense of free software. This involves educating politicians about Free Software so that new laws of the digital age don't stifle it's use or development. FSFeurope is active at national, European, and world trade levels of politics. In 2003, a significant amount of it's time was devoted to the World Summit on the Information Society

FSF Europe was not the first Free Software organisation in Europe, but it aims to unite the existing organisations as well as encouraging the creation of new ones. To this end, it has created an "Official Associates" program.

The FSF Europe can be reached under *http://www.fsfeurope.org*

Free Software Foundation India

Founded in 2001, **Free Software Foundation India** (FSF-India) is a sister organisation to Free Software Foundation. FSF-India is based in Kerala. The Free Software Foundation India played an important role in the World Social Forum by providing information technology support.

In 2003, after a few meetings with Richard Stallman, the President of India announced that the IT infrastructure of India should be based on Free Software. (The President is a ceremonial position in India, but the endorsement is significant)

The FSF India can be reached under *http://www.fsf.org.in/*

Creative Commons

The **Creative Commons** is a not-for-profit organization devoted to expanding the range of creative work available for others to legally build upon and share.

Aim

Their website enables copyright holders to grant some of their rights to the public while retaining others, through a variety of licensing and contract schemes, which may include dedication to the public domain or open content licensing terms. The intention is to avoid the problems which current copyright laws create for the sharing of information.

The project provides several free licenses that copyright holders can use when they release their works on the web. They also provide RDF/XML metadata that describes the license and the work to make it easier to automatically process and locate of licensed works. They also provide a 'Founder's Copyright' contract, intended to re-create the effects of the original U.S. Copyright created by the founders of the U.S. Constitution.

History

Creative Commons was officially launched in 2001. Lawrence Lessig is the founder and chairman of Creative Commons and started the organization as an additional method of achieving the goals of his Supreme Court case, Eldred v. Ashcroft. The initial set of Creative Commons licenses was published on December 16, 2002.

LOCALIZATION

Among the Creative Commons projects, the *iCommons* (International Commons) intends to fine-tune the Creative Commons legal wording to the specifics of individual countries. This is because the main Creative Commons licenses are written with the US legal model in mind, thus the wording may not be perfect for other countries. As of April 19, 2004, the following countries and regions have joined this initiative: Australia, Brazil, Catalonia, the People's Republic of China, Croatia, Finland, France, Germany, Ireland, Italy, Japan, Jordan, the Netherlands, Spain, Taiwan, the United Kingdom.

Creatve Commons can be reached unter http://www.creativecommons.org

Important conferences and expostions

LinuxTag

LinuxTag is a Free Software expo held every summer in the town of Karlsruhe in southern Germany. It is a comparatively large expo, drawing visitors from many countries.

The LinuxTag byline is "where .COM meets .ORG", as it includes both representatives from commercial companies and from not-for-profit/community projects. The latter are given booths, free of charge, in the .org section.

LinuxTag has also been a sponsor of the several other projects, notably of Knoppix, a GNU/Linux LiveCD. A **livecd** is a CD that boots into a complete GNU/Linux environment. Knoppix is often used to showcase software running on GNU/Linux on a computer running some other operating system.

LinuxTag is sponsored by various IT-related companies and, at least in some of its iterations, by the German government.

External links

- Official LinuxTag website (*http://www.linuxtag.org*)
- The Knoppix project (*http://www.knoppix.org*)
- OpenMusic (*http://openmusic.linuxtag.org*), another LinuxTag-sponsored project. The topbar at the site provides links to (all?) other LinuxTag-sponsored projects.

Wizards of OS

The Wizards of OS is a Berlin-based conference.

The topics are the potentials of PC and internet, free communication, open cooperation in creation and collecting of knowledge and the knowledge system of digital media. The conference is interdisciplinary and wants to be a plattform for meetings of "hard" technical and "soft" cultural/social scientists.

The name is a malapropism of Wizard of Oz. The OS stands for operating system, not for open source.

WOS3

The theme of the third conference in 2004 was "The Future of the Digital Commons".

See also: *http://wizards-of-os.org*

Appendix

Authors

Following authors have written parts of the WikiReader Free Software and Free Contents. Unregistered users (i.e. IPs) are not listed.

2501, 3247, 4tilden, AHoerstemeier, Ablaubaer, Adaxl, Aka, Akl, Alex42, AlexR, Alexander.stohr, Ali-Alkohol, Anathema, Andre Engels, Andre Riemann, Andreas B.

Used articles

GNU Free Documentation Licence

Version 1.2, November 2002

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as

they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's

users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.