Intellectual Property and Open Systems

Richard O. Mason,
Edwin L. Cox School of Business,
Southern Methodist University

Abstract
Arguably, the most profound issue in designing and governing cyberspace focuses on intellectual property. Is cyberspace to be created as a commons -- a space belonging to and used freely by the community as a whole -- or, is it to be partitioned into a multitude of proprietary closed boxes that are owned and sold, distributed, modified, and used only according to the owner’s whims? This paper will discuss answers to this question.

Intellectual Property

Technically speaking intellectual property (IP) is a creative string of bits that an individual or organization wants to protect. In our new information-based economy and society the basis of that protection has come into question. Who owns IP? How are property rights -- copyright, patents, trademarks, and trade secrets -- to be determined and protected in cyberspace? Is there a difference between owning atoms and owning bits? How can fair exchanges of property – especially bits – be determined? How can peer-to-peer systems, such as pioneered by Napster, be regulated? Should, for example, cyberspace be designed to support the “trusted systems” proposed by Mark Strefik?

A general policy should be developed in response to these questions. John Perry Barlow of the Electronic Freedom Foundation once proclaimed “Copyright is dead!” – there are no IP rights in cyberspace. Few, however, will accept this extreme view. Others argue that information comes in digital containers that, although they may be leaky, should be constructed so as to provide a degree of protection. Between these extremes the new society must find the means to bring about the most efficient and just set of protections for property interests in cyberspace. One approach is to cage in or block the reproduction of information using technology. This approach stresses private property rights. Another, following Strefik’s suggestion, is to create systems that track and control the movement of information and identify the parties who receive and distribute it. Finally, there is the open systems and shareware movements that seek to share software in ways that make its internal programming visible and, even, in the case of shareware, to give it away free. In these latter movements, users are encouraged to make copies (often downloading them from the World Wide Web) and to pay a moderate fee voluntarily. (Some distributors of shareware attempt to recoup their investment by selling support services and printed documentation.) In the midst of all of these differences of opinion is an evolving set of ideas of what the concept of property really is.

The Changing Nature of the Concept of Property

The term “property” used in its widest sense denotes an exclusive relationship between a person or a group of persons (such as a corporation) and an object or a complex of objects that have material value. Changes in society have resulted in changes in the nature of this relationship and, therefore, in changes in the definition of property. Property is a concept that has a long history. The advent of the agricultural society raised questions about the ownership of land and implements as it became clear to members of society that many privileges and benefits flowed to the owner of property from its use. Moreover, by using and controlling property the owner gained power. The extent of these benefits and power were magnified considerably during the Industrial Revolution when owners gained control over the newly developed means of commerce and production. Today, as our society is emerging from the early phases of the Information Revolution, concern for the ownership and use of property is evolving once again. Historically speaking, society’s concern for property has evolved, to use Nicholas Negroponte’s apt description, from a concern first for atoms, to complex structures of atoms, to bits, and then to bits that manage and control bits. This later object – bits that control bits – is the province of software. It is also the source of enormous power and benefits.

Software as Property

Software refers to programs (sets of instructions) that control the processing preformed by computer systems.
It’s what gives hardware functionality. In “Legal Protection for Computer Software: The View from [19]79” Michael Gemignani describes, in a summary though somewhat dated form, the nature of software and the software development process:

Programs are responses to problems to be solved. First, the problem in issue must be clearly formulated. Then a solution must be outlined. To be amenable to implementation on a computer, the solution must be expressible in a precise way as a series of steps to be carried out, each step being itself clearly defined. This is usually set forth as a flowchart, a stylized diagram showing the steps of the algorithm and their relationship to one another. Once a flowchart has been constructed, it is used as a guide for expressing the algorithm in a “language” that the computer can “understand.” This “coding” of the program is almost certain to employ a “high level” computer language, such as BASIC or FORTRAN [or C++, JAVA, or HTML]. When the algorithm is “coded” in a high level language, it is called a source program. A source program may bear a striking resemblance to a set of instructions expressed in literary form. The source program is fed into the computer by means of an input device, such as a terminal or card reader [or client machine]. The source program is “translated” by the compiler, a part of an operating systems program, into machine language, a language not at all similar to ordinary speech. The program expressed in machine language is called an object program. It is the object program, which actuates the setting of switches, which enables the computer to perform the underlying algorithm and solve the problem. vi

Software produced in this manner generally falls into three categories:

1. operating systems software that provides the facilities for application software to run,
2. applications software customized to solve a specific problem or execute a particular business function, and
3. application software packages, designed to serve generic needs, for example, “MSWord,” “Excel,” “PowerPoint,” etc.

Sales of software of these three types account for more than half of the worldwide information technology industry. The growing importance of software as an economic and social force is leading to additional claims on it as property.

In software parlance “machine language” refers to the string of bits that constitute the internal programming language for a specific computer or chip. It is what actually executes on the machine. But it is dastardly hard to understand and an excruciatingly difficult medium in which to program. So the software industry created a set of higher-level languages – languages closer to natural language – in which to program applications. FORTRAN and COBOL were early such high-level languages. BASIC, PL/I, APL and C++ followed. A program called a compiler is then used to translate the higher-level language statements into machine language. As Gemignani reports, by convention, the original, higher-level program is called source code and the resulting machine language program is called the object code. vii

When you or I acquire an operating system (e.g. Windows 95, or, Windows 2000), a customized program, or an application program (e.g. MSWord) we get the object code. We usually don’t peer into this object code, muck around with it, or attempt to change it even if, in the rare case, we know how. It’s too difficult and it is likely a violation of a private property right. What some people would like access to, however, is the source code. This would allow them to create hooks for interfacing other software with it or to modify it to meet their individual requirements. There is a related economic issue involved as well. Once someone has the source code he or she can produce multiple copies and multiple versions of the object code without paying for them.

One major source of a claim that software should be treated as private property is that it is so difficult to produce. This arduous process of conceiving of a program, drawing up diagrams, writing source code, the compilation of object code and debugging it requires much in the way of time, dedication, skill and ingenuity. Many of those who endure it want to be compensated for their effort.

Frederick P. Brooks, one of computer pioneer Howard Aiken’s most able Ph.D. students who lead IBM’s infamous OS/360 operating software project during the 1960’s, likens the software production process to a “tar pit.” “No scene from prehistory is quite so vivid,” according to the computer scientist who is now at the University of North Carolina, “as that of the mortal struggles of great beasts in the tar pits. . . . . . . Large-system programming has over the past decade been such a tar pit, and many great and powerful beasts have thrashed violently in it.” viii They get thrashed because producing software entails finding and correcting numerous bugs,
extracting and analyzing information provided by others, and working under a discipline that requires producing a product that performs almost perfectly. It is a creative, sometimes joyful, process in which a programmer must draw on her ingenuity and dedication — long nights fueled by Cokes, Fritos and Twinkies — to overcome these inherent obstacles to producing a software product.

**Two Moral Claims Concerning Software as Property**

In an information society an assumed moral tension exists between

1. the demand on the part of all people to share in the ideas, knowledge, and information, including software, produced by the society, and
2. the claim of those who produce the software and related information to reap the benefits and seize the power deriving from it.

This second claim is sometimes called the “Natural Rights” argument.

Many early software producers, having emerged from Brook’s tar pit, generally sided with the second camp. This is especially true of companies who hired programmers to produce software. They wanted to recoup or enhance the return on their investment. Recent events question the validity of this assumption. Some argue that you can get the advantages of both.

The ideological justification of the claim that software is private property (horn 2 of the dilemma) is that it is seen as the sum product of the labor of human beings, who, having mixed their sweat and brains with the materials are, therefore, entitled to reap the rewards of their work. The early 18th-century philosopher, John Locke, is often quoted as the modern day developer of this point-of-view. According to Locke, when humankind first appeared on earth, it held the whole world in common. But as human beings multiplied and different civilizations arose, they began to act more individually, and they began to appropriate parts of the world and carve up the commons by applying their own human labor to it. The first “property” was created in this manner. Subsequently, common law and common sense dictated that the application of intellectual and physical labor give the producer the right to own, manage, and use the property (within restraints) and to transfer it to other parties if the producer so chooses and under conditions of the producer’s making.

A strong modern proponent of this view of private property is novelist, philosopher Ayn Rand who argues that a human being’s basic right to life requires the right to property. Since a human being must sustain life by his or her own effort, she avers, he or she must have exclusive right to the product of his or her efforts. Rand, speaking through her alter ego John Galt in Atlas Shrugged asserts:

> Just as man can’t exist without his body, so no rights can exist without the right to translate one’s rights into reality — to think, to work and to keep the results — which means: the right to property. The modern mystics of muscle who offer you the fraudulent alternative of “human rights” versus “property rights,” as if one could exist with the other, are making a last, grotesque attempt to revive the doctrine of soul versus body. Only a ghost can exist without material property; only a slave can work with no right to the product of his effort. The doctrine that “human rights” are superior to “property rights” simply means that some human beings have the right to make property out of others; since the competent have nothing to gain from the incompetent, it means the right of the incompetent to own their betters and to use them as productive cattle. Whoever regards this as human and right, has no right to the title of “human.”

The source of property rights is the law of causality. All property and all forms of wealth are produced by man’s mind and labor.

Several arguments can be mounted, however, against the natural rights claim. One is that it grants too much to the person who provided the labor. A software writer never starts entirely from scratch with no resources, no help, and no knowledge to draw on. The providers of these resources also have a claim to the resulting property. Moreover, the natural rights argument, powerful though it may be, focuses primarily on the material aspects of property — the atoms. Bits, however, have different characteristics. To begin with, bits — the elemental components of software — are ethereal, you can’t easily hold on to strings of 1’s and 0’s unless you encapsulate them in a package of some kind. But then they are still hard to contain. Bits are diffuse and tend to leak. Bits can also be moved around the globe at the speed of light. They are transportable and once moved they also stay put right where they came from. That is, bits can be shared. I can give you a sting of bits and still have them myself. Bits, importantly, can be replicated or copied easily, usually at little or no cost. They can be manipulated so as to be compressed or expanded and then reconstituted into their original or another form. Indeed bits can be...
substituted for atoms or used to control them in such a way as to make the atoms more efficient or effective.\textsuperscript{iii}

A second line of moral reasoning for treating software as property is consequentialist. One stream of this argument supports the institution of proprietary software on the basis that it is necessary to stimulate creativity, innovation and entrepreneurship. A contrary argument claims that the institution of proprietary software actually inhibits progress by stalemating the exchange of ideas. Everyone will be better off, this line of reasoning claims, if ideas are shared openly and freely.

In the public policy domain today both moral and economic arguments are being made with respect to software production and how its product is conveyed.

**Open Source Software (OSS) or Free Software versus Proprietary Software.**

In the software arena today a major battle is being fought between two camps: those who believe that software is private property (following the line of reasoning discussed above) and those who believe it is not. Some pragmatists, who believe that it is economically advantageous to share software openly even if they would prefer to own, it join the second camp.

In the more traditional, closed, private property model – John Parry Barlow calls it the “industrial” model – only a very few selected programmers are permitted to view, much less work with, the source code. All other programmers are given access only to object code in the form of an opaque, black box full of binary bits.

The contrary camp argues for open system software -- software made available in source code form at no cost to users or developers. The Linux operating system is an example. A variant of the Unix operating system, Linux has grown from a kernel developed by Finnish programmer, Linus Torvalds in 1991. Torvalds encouraged users to modify and improve his operating system as long as they agreed to freely share the changes with other users. He also continued to supervise the overall integrity of the code. Hacker Pekka Himanen explains:

\begin{quote}
[A]nyone may download Linux for free. [What] distinguishes Linux from the dominant commercial software model epitomized by Microsoft’s products is first and foremost its openness: in the same way scientific researchers allow all others in their fields to examine and use their findings, to be tested and developed further, hackers who take part in the Linux project permit all others to use, test, and develop their programs. In research, this is known as the scientific ethic. In the field of computer programming, it is called the open source model (source code being a program’s DNA, its form in the language used by programmers to develop it; without the source code, a person can use a program but is to develop it in new directions).\textsuperscript{iv}
\end{quote}

Supporters of the open source movement believe that the institution of closed, private property impedes social and economic progress, especially in the rapidly exploding field of software. They point specifically to operating systems, programs that by their nature must be shared and coordinated with others. Long-term economic efficiency, as well as concerns for justice and human rights, is best achieved, in this second camp’s view, when software is maximally open and available in all its grisly details. They argue that if programmers can read, redistribute, and modify the source code for a piece of software, then the software itself evolves and better serves all users’ needs as they adapt it, improve it and eliminate its bugs.

This philosophy has proven to be rather successful. August 2001 marks the tenth anniversary of Linux. During this time thousands of programmers worldwide have worked with and contributed to Linux and made it a robust alternative to operating systems sold by IBM, Microsoft, and Sun Microsystems. Today many of the most widely used programs on the Internet – those used for delivering e-mail, distributing discussion groups, serving domain name systems’ queries, and serving Web pages – already are open source. The use of Linux has grown from 10,000 lines of code, used by fewer than a dozen people in 1991, to more than 1.5 million lines used by over seven million users in 1998\textsuperscript{v}.

Linux is an example of software that is available under “freedom 3” in a hierarchy proposed by Richard Stallman. Setting at the “open” extreme of the open versus closed continuum, Stallman has advocated the adoption of GNU (Gnu’s Not UNIX) since the mid-1980’s through his Free Software Foundation. Stallman believes that proprietary, closed box software is the problem our new economy and society faces and counters that free software is the solution. Free software, in his view, serves to liberate people rather than dominate them. Stallman prefers the term “free software” in contrast to “open” because he believes it better conveys the moral principles involved. Liberty is the primary underlying concern, not traditional economics. Software should be treated like “free speech.” It is not “free lunch.” Stallman proposes a four level hierarchy of freedoms:

- Freedom 0. The freedom to run the program, for any purpose.
• Freedom 1. The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
• Freedom 2. The freedom to redistribute copies so you can help your neighbor.
• Freedom 3. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.\textsuperscript{xiv}

An interesting contest is playing out in the software market arena right now. Different software producers and vendors are taking different stances with respect to these freedoms, depending on their overall company philosophy and on the product and market characteristics of their particular software offerings. Microsoft Windows, Apple’s MAC-OS, and IBM’s OS/2 Warp, for example, are proprietary. Their creators, in effect, own the source code and keep it secret and closed to the outside world except to selected licensees. These vendors advocate Freedom 0. IBM, however, as will be described below, is selective as to which freedom it promotes depending on the particular piece of software involved.

Richard Stallman, is a strong proponent of all four freedoms. He maintains, moreover, that, once conveyed, these freedoms must be irrevocable, as long as they are not abused. He has coined the term “copyleft” to describe an essential rule in his philosophy: when a program is distributed to others, the distributor is prohibited from adding restrictions to the exchange transaction that would deny the recipient and others any of the four central freedoms articulated above. He views the copyleft rule as a protection and not a restriction on uses.

The moral debate in the software marketplace arena today centers on differences as to which of the four levels should emerge as the governing principle for cyberspace. Most people agree that freedom 0 is basic. We use object code programs like Windows, Power Point, SAP, or any specially devised program openly and continually to conduct our legitimate business. We do so without fear of reprisal. It’s what everyone expects. Purchasing a program from any vendor transfers private property and should permit usage consistent with freedom 0.

Challenges are mounted, however, when a user wants to exercise freedom 1. Most software providers convey only freedom 0 to their users and balk or sue if the users try to get access to source code and adapt it to their own needs. Nevertheless, some large users or selected software programmers have been granted freedom 1 access when the code programs like Windows, Apple’s MAC-OS, and IBM’s OS/2 Warp, for example, are proprietary. Their creators, in effect, own the source code and keep it secret and closed to the outside world except to selected licensees. These vendors advocate Freedom 0. IBM, however, as will be described below, is selective as to which freedom it promotes depending on the particular piece of software involved.

Richard Stallman, is a strong proponent of all four freedoms. He maintains, moreover, that, once conveyed, these freedoms must be irrevocable, as long as they are not abused. He has coined the term “copyleft” to describe an essential rule in his philosophy: when a program is distributed to others, the distributor is prohibited from adding restrictions to the exchange transaction that would deny the recipient and others any of the four central freedoms articulated above. He views the copyleft rule as a protection and not a restriction on uses.

The moral debate in the software marketplace arena today centers on differences as to which of the four levels should emerge as the governing principle for cyberspace. Most people agree that freedom 0 is basic. We use object code programs like Windows, Power Point, SAP, or any specially devised program openly and continually to conduct our legitimate business. We do so without fear of reprisal. It’s what everyone expects. Purchasing a program from any vendor transfers private property and should permit usage consistent with freedom 0.

Challenges are mounted, however, when a user wants to exercise freedom 1. Most software providers convey only freedom 0 to their users and balk or sue if the users try to get access to source code and adapt it to their own needs. Nevertheless, some large users or selected software programmers have been granted freedom 1 access when the source code owner or vendor sees a benefit from doing so.

Greater contentions arise at freedom 2. Some vendors consider it a “trespass” if a user violates the code and enters it and attempts to adapt it. A few vendors will allow users to examine the source code – like peering over a picket fence – look, don’t touch – so that they can create better interfaces to it; but, they enjoin them from altering a single line of the code. Others, like Sun Microsystems, believe that considerable advantages accrue with a certain level of kimono opening because this openness is essential to promote interoperability.\textsuperscript{xix} It also spurs technological innovation and advances the public interest. “Sun supports the principle that copyright does not protect interface specifications, which include only those elements of computer programs that are necessary for interoperability – the interchange of information between computer systems that benefits all computer users.”\textsuperscript{xvi} Sun’s policy also states that copyright protection for computer software must “carefully balance” the goals of strong property rights protection and rewards for innovation with the equally important goals of interoperability, fair competition, and open systems. Thus, Sun supports freedom 1 as long as the adaptation improves interoperability; but the company draws the line at freedom 2. It still has a strong interest in copyright protection because the company wants to recoup its substantial investment in research and development.

IBM currently takes a similar position to Sun’s with respect to its network based e-business offerings. E-business systems, in their view, are comprised of four components: clients, data, programs, and networks. They consider the client to be the general-purpose browser. The data is self-descriptive, based on XML. The programs are platform independent, based on JAVA. And the network infrastructure is based on TCP/IP. These crucial software systems the company argues should be open. In other product areas, however, IBM is quite proprietary.

Conveying freedom 2 stirs up more controversy. Traditional IT and MIS organizations found in major companies, like American Airlines, Microsoft, Texas Instruments and, to an extent, IBM, and also in many government organizations, tend to favor agreements based on pay-for-product, restricted software. For the most part, these organizations strongly support copyright and related intellectual property laws. Many small Internet-based startups, however, embrace freedom 2 with enthusiasm. They, accordingly, gravitate towards systems like Linux. This is one reason why GNU/Linux has become a reasonably popular operating system.\textsuperscript{xxv} Nevertheless, relatively few vendors accept Stallman’s freedom 3 without some reservations.

Linux is transferred to users, for the most part, under conditions of freedom 3 by means of a licensing scheme developed by Stallman called the General Public License (GPL). GPL is a “copyleft” instrument that requires that any code that is licensed under its provisions must be kept free with respect to its source free. That is, it must reside in the common. Limitations may be placed, however, on possible uses of the software. A user, for example, cannot take the public part of the code and close it to others; nor,
is a user permitted to integrate the open part with closed parts.

Linux’s market success has debunked some of the economic assumptions held by software producers and vendors whose companies convey lesser freedoms. According to Dan Kusnetzky, vice president of International Data Corporation, a computer market research firm, Linux has been the fastest-growing operating system for servers\textsuperscript{v} since 1997. His company’s research shows that in the summer of 2001 Linux held a 27 percent market share, compared with 41 percent for Microsoft’s Windows 2000, the industry leader. Novell’s Netware and Unix held about 14 percent each.\textsuperscript{vi}

IBM has taken a major strategic turn in this regard. In August 2001 IBM announced that its Websphere e-commerce software would be made available to run on a Linux mainframe. Meanwhile IBM began working with customers ranging from financial services to telecommunications to port key software applications over to computers that were running on Linux. “Linux is moving into the mainstream,” said Ross Mauri an IBM vice president. He announced that the company would spend $1 billion in 2001 developing Linux and Linux-based applications. (This is about 20\% of the company’s $5 billion R&D budget.)

All has not been rosy, however, with Linux providers. Linux vendor Eazel has closed its doors. (Ironically, some of its code is still actively in use and being updated.) Companies such as Red Hat and VA Linux Systems tried to build businesses around Linux by offering support, documentation, and compatible hardware. Most have laid off workers and scaled down their business and profit plans. Thus, the jury is still out on the economics of Linux.

**Conclusion**

Various forces with different objectives are operating actively in the intellectual property arena, struggling for position along dimensions of ownership and openness. Their positions are taken according to which level of freedom they determine is in their best economic interests or which they believe morally ought to prevail. Another way to reflect this contest is to plot products and companies on an Openness/Ownership Map. The following is a preliminary map:

<table>
<thead>
<tr>
<th>PROPRIETARY</th>
<th>PUBLIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apache</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
</tr>
<tr>
<td></td>
<td>IBM’s Linux</td>
</tr>
<tr>
<td>CLOSED</td>
<td>Microsoft Windows, Apple’s MAC-OS, IBM’s OS/2 Warp, AOL</td>
</tr>
</tbody>
</table>

Microsoft’s strategy strongly favors the Closed/Proprietary solution. Indeed, the company’s prevailing business model is based on it. Microsoft seeks to do three things: protect its dominance in end-user operating systems, keep a critical mass pool of non-Microsoft end-user applications from developing, and continue to appropriate what it calls “commodity protocols.” The company can augment these protocols, such as HTML and JavaScript, and create proprietary extensions that prevent competing application pools from forming. Consequently, Microsoft is battling mightily against GPL. If GPL prevails Microsoft cannot follow its “embrace and extend” approach since open source code cannot be appropriated and modified.

Because a GPL licensed application pool is forming around it, Linux constitutes a fundamental threat to Microsoft’s business strategy. This is why IBM’s announcement to support Linux is attracting so much attention. IBM believes that wide adoption of an open standard like Linux is a necessary condition for building the large business infrastructure needed for the future. This task is so complex that no one company can do it alone. A concerted creative effort on the part of a vast number of autonomous individual programmers and software companies is needed. Openness, then, is economically justified as well as morally.

Bill Gates, however, has drawn a new line in the sand. He now argues that customers be allowed to choose between proprietary and free software in a marketplace. In June 2001, in his typical avuncular style, he told a CNET.com reporter: “The ecosystem where you have free software and commercial software – and customers always get to decide which they use – that’s a very important and healthy ecosystem.” “We believe that there should be free software and commercial software; there should be a rich ecosystem that works around that.” Rumors are floating around Washington D.C that Microsoft is lobbying Congress to disallow the use of GPL as the license instrument for any software produced with public funds.

These movements are on a collision course. Both moral and economic forces are at work. Cyberspace will become chaotic, or stalemated, or, at best, fail to achieve its considerable promise unless some broad social resolution is reached on this crucial issue of intellectual property and open systems.

(See Appendix A for some definitions related to the open source issue.)
Appendix A

Categories of Free and Non-Free Software
Based on Richard Stallman’s Definitions

• Free Software = software that comes with the permission for anyone to use, copy, and distribute, either verbatim or with modifications, either gratis or for a fee.

• Open Source Software = a synonym, in most uses, for free software.

• Public Domain Software = software that is not copyrighted, a special case of non-copylefted free software. That is, some copies or modified versions may not be free to all.

• Copylefted Software = free software whose distribution terms do not let redistributors add any additional restrictions when they redistribute or modify the software.

• Non-copylefted Free Software = free software that comes from the author with the permission to redistribute and modify, and also to add additional restrictions to it.

• General Public License (GPL) Covered Software = a specific set of distribution terms for copylefting a program.

• The GNU System = a complete free Unix-like operating system.

• GNU Programs = equivalent to “GNU Software.”

• GNU Software = software that is released under the auspices of the GNU Project. Most GNU software is copylefted, but not all; however all GNU software is free software.

• Semi-free Software = software that is not free, but comes with permission for individuals to use, copy, distribute, and modify (including distribution of modified versions) for non-profit purposes.

• Proprietary Software = software that is not free or semi-free. Its use, redistribution or modification is either prohibited, requires permission, or heavily restricted so as to effectively make it not free.

• Freeware = commonly used for packages which permit redistribution but not modification. The source code is not made available.

• Shareware = software which comes with permission for people to redistribute copies, but requires that anyone who continues to use a copy pay a license fee. Generally the source code is not available and permission to copy without paying a license fee is not granted.

• Commercial Software = software being developed by a business which aims to make money from its use. Although most commercial software is proprietary this is not a necessary condition. Commercial free software is available as it non-commercial, non-free software.

Source: http://www.gnu.org/philosophy/categories.html
See
(4) Streffik, Mark, “Trusted Systems: Devices that enforce machine-readable rights to use the work of a musician or author may create secure ways to publish over the Internet.” *Scientific American*, March 1997 pages 78-81

See

Source code is used to generate object code (usually by means of a compiler or assembler), which is machine-readable and, in turn, used to run the programs. Object code most commonly refers to machine code that can be directly executed by the system's central processing unit (CPU).


Locke’s theory is detailed in Locke, John, *Two Treatises of Civil Government*, London: J. M. Dent & Sons. 1924 (original published in 1690)


One disadvantage of open source programs is that there is no reliable way to know how many people might be using them. In March 1998 Red Hat Software, Inc.'s Robert Young estimated that with a margin of error of +/- 25% that there were between 5,000,000 and 10,500,000 active Linux users. (http://www2.linuxjournal.com/enterprise/linuxmarket.html)

Drawing on previous studies the mean number of users was estimated to be as follows:

<table>
<thead>
<tr>
<th>Year</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>100,000</td>
</tr>
<tr>
<td>1994</td>
<td>500,000</td>
</tr>
<tr>
<td>1995</td>
<td>1,500,000</td>
</tr>
<tr>
<td>1996</td>
<td>3,500,000</td>
</tr>
<tr>
<td>March 1998</td>
<td>7,500,000</td>
</tr>
</tbody>
</table>

In an e-mail dated September 18, 2001 Young observed “For what it is worth my person opinion is that the use of Linux has continued at the annual percentage increases I saw in those” earlier studies.

Interoperability describes components of computer and communications systems that are able to function in different machine and software environments. With respect to software, interoperability occurs when programs are able to share data and resources. Microsoft Word, for example, is able to read files created by Microsoft Excel and other programs. Microsoft’s NT operating system is interoperable on Intel, DEC Alpha, and other CPUs. The SCSI standard for disk drives and other peripheral devices allows them to interoperate with different operating systems. Interoperability is an essential characteristic of large, networked systems that allow a wide variety of machines, software and databases to communicate with one another.

* Linux is a version of the UNIX System V Release 3.0 kernel developed for PCs with 80386 and higher microprocessor CPU’s. Linux is distributed free with source code through bulletin board systems (BBSs) and the Internet. Some companies distribute it as part of a commercial package with Linux-compatible utilities. The Linux kernel works with the GNU utilities developed by the Free Software Foundation.

A server is a specialized computer that is linked to other computers on a network and use to perform specific tasks requested through those computers. Part of the client-server architecture in which client machines send requests for service and the server devices perform the requested processing.