

Open Source Roadshow with AOL - Case Study: Mozilla Project

What is open source software? Why pay for software when it is free? How can teaching and research be aided by open source software? What is the technology surrounding Netscape? What innovative applications have been built, through community involvement, upon Mozilla, the code at the heart of Netscape, the first Web browser? What are the security challenges and strategies relating to open source development?

Students, faculty, staff, and visitors are invited to attend any or all of a day-long program on these topics to be held October 24, 2002, from 8 a.m. -5 p.m. in the Donaldson Brown Hotel and Conference Center, followed by an evening meeting of the [ACM Student Chapter](#). The Roadshow is being hosted by the [Internet Technology Innovation Center](#) at Virginia Tech in cooperation with [AOL](#) and [Virginia's Center for Innovative Technology](#). Join in the discussion and enjoy the demonstrations and refreshments!

Sessions will be led a team coming from Netscape and Mozilla, with support from AOL. Mitchell Baker is the general troubleshooter, spokesperson, and policy arbitrator for mozilla.org. Scott Collins has been a participant and contributor on the mozilla project since its inception. Mitch Stoltz is the lead security engineer for the Netscape client team.

Sessions will run all day on the 24th. Many are of general interest, but some will be oriented toward those with a focus on technology. Pick and choose to suit your background and schedule. For more information contact Ed Fox (x1-5113, fox@vt.edu).

The schedule is:

- DBHCC - Auditorium
 - 8:00 - 8:15 Welcome by Earving L. Blythe, Vice-President Information Technology
 - 8:15-9:15 "Securing Open Source Software: Advantages and Challenges"
Presenter: Mitchell Stoltz - Describes security implications of open source development.
 - 9:30 - 10:05 "Introduction"
Presenters: Mitchell Baker and Scott Collins - Describes the concept of Free Software/
Open Source Software.
Intended audience - everyone
 - 10:05 - 10:45 "Mozilla Project Dynamics"
Presenter: Scott Collins - Interactions in the mozilla project
Intended audience - everyone

- 11:00 - 12:15 "Security in Mozilla"
Presenter: Mitchell Stoltz - Describes the security challenges and some strategies being used to improve security.
 - 12:30 - 1:45 "Mozilla Educational Resources"
Presenter: Scott Collins - Useful to anyone who wants to teach or learn more about mozilla.
 - DBHCC - Conference Room C
 - 2 - 3:15 "Open Source Licensing"
Presenter: Mitchell Baker - Discussion of the Mozilla Public License in particular, and licensing of open source software in general, including regarding the BSD and GNU General Public License.
Intended audience - everyone The last part of this session will include a panel with local discussants:
 - Kay Heidbreder, Associate General Counsel, Virginia Tech
 - Michael Martin, Executive VP, VTIP, Inc. (<http://www.vtip.org/>)
 - Ronald D. Kriz, Associate Professor, Engineering Science and Mechanics, Virginia Tech
 - 3:30 - 5:00 "Mozilla Technologies"
Presenter: Scott Collins - A breakdown of the exploitable machinery and technologies in mozilla.
The intended audience is technical.
 - Norris 136
 - 7:00 - 9:00 ACM Student Chapter Meeting
Introduction, "The Mozilla Demos"
Presenter: Scott Collins - Q&A, and informal discussion
Intended audience - everyone
-

Some handy links include

- [Open Source Roadshow Summaries and Outline](#)
 - mozilla.org homepage
 - [Mozilla.org staff including Mitchell Baker and Scott Collins](#)
-

Special thanks go to the coordination efforts of Amy Hale, Director, University Relations, America Online, AmyHale9@aol.com, (703) 265-2690.

This page is online at <http://fox.cs.vt.edu/opensource.htm>

OpenSourceRoadshow



[HomePage](#) | [RecentChanges](#) | [Preferences](#)

[OpenSourceRoadshow](#) - Wiki site for discussion with AOL for events on open source software

For more information on using this wiki, go to the home page, using the button below, or directly to [HomePage](#).

To get discussion started, there are some separate pages created. You can add others too! The initial set includes:

- [OpenSourceRoadshowDates](#) - see dates for activities
- [OpenSourceRoadshowAgenda](#) - discussion of the agenda
- [OpenSourceRoadshowSpeakers](#) - list of people from AOL coming and speaking, with their info
- [OpenSourceRoadshowOffers](#) - offers of assistance from Virginia universities
- [OpenSourceRoadshowQandA](#) - questions for resolution
- [OpenSourceRoadshowAgendaLicensing](#) - discussion of MPL licensing

- [OpenSourceRoadshowPressRelease](#) - discussion about press release(s)

- [OpenSourceRoadshowGWU](#) - event on 10/21 at George Washington
 - [OpenSourceRoadshowGWUDiscussion](#) - discussion about event
- [OpenSourceRoadshowGMU](#) - event on 10/22 at George Mason
 - [OpenSourceRoadshowGMUDiscussion](#) - discussion about event
- [OpenSourceRoadshowVT](#) - event on 10/24 at Virginia Tech
 - [OpenSourceRoadshowVTDiscussion](#) - discussion about event

For further information contact Ed Fox, fox@vt.edu, +1-540-231-5113, home page <http://fox.cs.vt.edu>

(This page last updated 9/27/2002.)

[HomePage](#) | [RecentChanges](#) | [Preferences](#)

[Edit text of this page](#) | [View other revisions](#)

Last edited October 16, 2002 10:43 am ([diff](#))



search mozilla:

- **Roadmap**

- **Projects**

- **Coding**

- Module Owners
- Hacking
- Get the Source
- Build It

- **Testing**

- Releases
- Nightly Builds
- Report A Bug

- **Tools**

- Bugzilla
- Tinderbox
- Bonsai
- LXR

- **FAQs**

Open Source Roadshow Summaries and Outline

This document provides summaries and (some) outlines for Mozilla's presentations at the upcoming Open Source Roadshow, as well as profiles of the presenters. We expect details to change as we review, polish, and act on feedback, especially in the technology demo area. As such, this page is more of a menu than a schedule. The exact set and order of presentations will be tuned to the individual needs of the schools.

See [Ed Fox's wiki](#) for more discussion and details.

Presentation Summaries and Outlines

Introduction

Time: 1/2 hour. Presenters: Mitchell Baker, Scott Collins.

This section presents an introduction to the concept of Free Software/Open Source Software, and the mozilla project with a brief political/technical history and why the listener might be interested anything else we have to say. The intended audience is everyone. This session will be useful to anyone who wants to know more about mozilla, or who intends to see any other session.

- What is Free Software/Open Source Software

- What is this presentation?
- Who am I?
- What is mozilla? a brief history
- What is mozilla.org?
- What is mozilla University? (and why should you care). It's real-world development experience and technologies that are
 - broad
 - deep
 - relevant
 - accessible
 - and we have the materials to communicate this knowledge
- Q and A

Mozilla Project Dynamics

Time: 1/2 hour. Presenter: Scott Collins.

This section is a slice of life of the interactions in the mozilla project. It covers the activities, participants, connections, and the raw numbers that define our effort. The intended audience is everyone. This session will be useful to anyone who interested in how open source projects in general succeed and fail, and ours in particular; and to anyone interested in getting their hooks into mozilla and participating.

- it's a tight cooperative feedback loop
- design in the open (bugzilla, irc, news, and public meetings)
- bugzilla
- code and design review
- managing the source (source code control, access, branches)
- continuous integration and testing
- performance analysis
- many `eyeballs': public review of daily builds and milestones
- stable releases, freezes and milestones
- plugged into the rest of the free software/open source software community
- Q and A

The Mozilla Demos

Time: 1 hour. Presenter: Scott Collins.

This section illustrates the breadth of applications to which our technologies have been put. The intended audience is anyone generally aware of web standards and current technologies, though technical expertise is not required. This session will be useful to project planners, engineers, and anyone else who wants a loose survey of the field.

- XUL
 - demo of creating XUL
 - themes
 - remote XUL functionality
 - becomes a sidebar
 - becomes a desktop app
 - using XUL for rapid prototyping and xp delivery
- Crocodile Clips
- Komodo 2
- browser extensions
 - sidebars
 - google search
 - blog
 - gestures
 - composite
- browsers
 - mozilla
 - Chimera
 - NS7
 - Galeon
- XPCOM bindings
 - PyXPCOM (Python)
 - PIXPCOM (Perl)
 - RBXPCOM (Ruby)
- SOAP
- MathML
- SVG
- XSLT
- Q and A

Mozilla Technologies

Time: 1 hour. Presenter: Scott Collins.

This section is a breakdown of the exploitable machinery and technologies in mozilla. The intended audience is technical. This session will be useful to people who want to hack on mozilla code or UI, either as a contributor, or for their own projects (or both!).

- Gecko
 - DOM

- CSS
- XSLT
 - XPath
 - XLink
- DOM Inspector
- XUL
 - XBL
 - overlays
 - themes
- SVG
- MathML
- JavaScript
 - Venkman
- XPConnect
- XPCOM
- RDF
- security
- XPInstall
- l10n (localization), i18n (internationalization)
 - language packs
- licensing
- Bugzilla
- build tools
- userChrome/userContent
- Q and A

Mozilla Educational Resources

Time: 1 hour. Presenter: Scott Collins.

This section describes the documentation, tools, and approaches to squeeze all this real-world knowledge out of the mozilla effort and make it approachable to (almost) ordinary people. The intended audience is pretty much everyone. This session will be useful to anyone who wants to teach or learn more about mozilla. Non-technical people will get as much out of it as the technical.

- tentative courses
 - mozilla application development
 - CSS in mozilla
 - extending your mozilla application
 - XBL
 - RDF in mozilla
 - using and creating XPCOM components
 - web standards, content development and evangelism
- the real world
 - Developer Days

- using the source base for research
- lxr and cvs
 - blame
 - log
 - history
 - search
- doxygen
- Q and A

Security In Mozilla

Time: 1 hour. Presenter: Mitchell Stoltz.

This presentation will describe the security challenges we face as makers of a Web client, and some of the strategies we are using to improve our security. It will make a case for greater participation by students and faculty of computer science and engineering programs with an interest in security, and discuss the advantages of academic collaboration. Finally, it will propose some possible projects to be taken on in collaboration with universities.

Securing Open Source Software: Advantages and Challenges

Time: 1 hour. Presenter: Mitchell Stoltz.

This presentation will discuss the security implications of open source development; the caveats and pitfalls, the potential advantages, and the conditions necessary to create these advantages. This presentation will be of interest to developers of security-critical open source software, as well as anyone interested in the current debate over the security of open source.

Open Source Licensing

Time: 1 hour. Presenter: Mitchell Baker.

Open source software is governed by a license of some sort, which describes the terms under which the software can be used. [The Mozilla Public License \("MPL"\)](#) was created at the launch of the

Mozilla project, and has been widely adopted by a variety of other projects. This talk will discuss why the MPL was written, what's required of an "open source" license, how the MPL was written, how the MPL compares with two seminal open source/free software licenses (the BSD license and the [GNU General Public License](#)), acceptance of the MPL and current licensing issues. This discussion will include topics such as:

- licensing in the Open Source Software/Free Software world
- key requirements on an Open Source license
- why was the MPL created?
- creating the MPL
- major decisions in drafting the MPL
- acceptance level of the MPL
- current licensing issues

See also [this wiki page](#), and [this list of many different licenses](#).

About the Presenters

Mitchell Baker

Mitchell is the general troubleshooter, spokesperson and policy arbitrator for mozilla.org. She works extensively with companies and projects using Mozilla. This involves explaining mozilla.org processes, listening to the needs of contributors, and seeking to integrate open source development techniques with the world of commercial software development.

Mitchell also spends a lot of time driving consensus as to how mozilla.org ought to manage the project, which suggests she may have a masochistic streak. She dreams of making the Mozilla project easier to understand.

Before joining mozilla.org staff, Mitchell was the attorney at Netscape responsible for all legal issues related to product development and intellectual property protection. During that time she wrote the Netscape and Mozilla Public Licenses.

Scott Collins

Scott Collins has been a participant and contributor on the mozilla project since its inception. Some of his initial contributions to the project were chronicled in the PBS documentary "[Code Rush](#)" as Netscape undertook efforts to put its browser code into the public domain. He is currently the lead evangelist of the "[Mozilla University](#)" effort that is underway to promote the use, understanding, research and development of mozilla technologies in a variety of educational settings.

Mitch Stoltz

Mitch Stoltz is the lead security engineer and "fireman" for the Netscape client team. He works on finding and fixing security problems in Mozilla and educating developers on secure coding practices.

[Site Map](#)[Contact Us](#)[Donate](#)

Copyright © 1998-2004 The Mozilla Organization

Last modified October 15, 2002

[Document History](#)

[Edit this Page](#) (or [via CVS](#))



What is Mozilla?

The Mozilla project maintains choice and innovation on the Internet by developing the acclaimed, open source, Mozilla 1.6 web and email suite and related products and technology.

Thunderbird 0.6 is Now Available

Some of the new features include a Windows installer, Pinstripe theme for Mac OS X, new artwork, improved junk mail controls, improved mail notification in the system dock for Mac OS X, server-wide news filters and a slew of other new features. [Get more info.](#)

download: [Windows](#), [Linux](#), [Mac OS X](#)

Mozilla 1.6



- For web browsing, email, HTML editing, IRC chat, and more
- Built with your privacy and security in mind
- Stop popups and junk mail
- Open multiple web pages in the same window with tabbed browsing

Order the CD

Free Download: [Windows](#), [Mac OS X](#), [Linux](#), [More...](#)

"Best browser of 2003"
— *PC World Magazine*

"Beyond Bliss"
— *Time Magazine*

Technology Preview



Mozilla 1.7 RC 1

This is the latest preview of the Mozilla application suite, including Navigator, Messenger, and Composer.

download: [Windows](#), [Linux](#), [Mac OS X](#) or in Europe: [Windows](#), [Linux](#), [Mac OS X](#)



Firefox 0.8

Our award winning next generation browser is a sign of things to come from Mozilla. Powerful yet easy to use. Perfect for those who live on the cutting edge!

download: [Windows](#), [Linux](#), [Mac OS X](#) (or try [other download sites](#))



Thunderbird 0.6

An email and newsgroup client with powerful, new junk mail controls

download: [Windows](#), [Linux](#), [Mac OS X](#)



Camino 0.7

Recommended by c|Net for advanced security setting and privacy features
download: **Mac OS X**

Other Available Products



Bugzilla

Enterprise-grade bug tracking software [[more](#)]

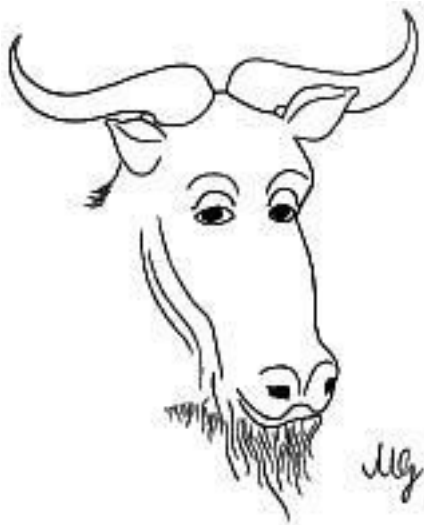
Copyright © 1998-2003 The Mozilla Organization

[Translations](#) of this page

Why Software Should Be Free

by [Richard Stallman](#)

(Version of April 24, 1992)



Introduction

The existence of software inevitably raises the question of how decisions about its use should be made. For example, suppose one individual who has a copy of a program meets another who would like a copy. It is possible for them to copy the program; who should decide whether this is done? The individuals involved? Or another party, called the "owner"?

Software developers typically consider these questions on the assumption that the criterion for the answer is to maximize developers' profits. The political power of business has led to the government adoption of both this criterion and the answer proposed by the developers: that the program has an owner, typically a corporation associated with its development.

I would like to consider the same question using a different criterion: the prosperity and freedom of the public in general.

This answer cannot be decided by current law--the law should conform to ethics, not the other way around. Nor does current practice decide this question, although it may suggest possible answers. The only way to judge is to see who is helped and who is hurt by recognizing owners of software, why, and how much. In other words, we should perform a cost-benefit analysis on behalf of society as a whole,

taking account of individual freedom as well as production of material goods.

In this essay, I will describe the effects of having owners, and show that the results are detrimental. My conclusion is that programmers have the duty to encourage others to share, redistribute, study, and improve the software we write: in other words, to write [`free" software.\(1\)](#)

How Owners Justify Their Power

Those who benefit from the current system where programs are property offer two arguments in support of their claims to own programs: the emotional argument and the economic argument.

The emotional argument goes like this: "I put my sweat, my heart, my soul into this program. It comes from *me*, it's *mine*!"

This argument does not require serious refutation. The feeling of attachment is one that programmers can cultivate when it suits them; it is not inevitable. Consider, for example, how willingly the same programmers usually sign over all rights to a large corporation for a salary; the emotional attachment mysteriously vanishes. By contrast, consider the great artists and artisans of medieval times, who didn't even sign their names to their work. To them, the name of the artist was not important. What mattered was that the work was done--and the purpose it would serve. This view prevailed for hundreds of years.

The economic argument goes like this: "I want to get rich (usually described inaccurately as `making a living'), and if you don't allow me to get rich by programming, then I won't program. Everyone else is like me, so nobody will ever program. And then you'll be stuck with no programs at all!" This threat is usually veiled as friendly advice from the wise.

I'll explain later why this threat is a bluff. First I want to address an implicit assumption that is more visible in another formulation of the argument.

This formulation starts by comparing the social utility of a proprietary program with that of no program, and then concludes that proprietary software development is, on the whole, beneficial, and should be encouraged. The fallacy here is in comparing only two outcomes--proprietary software vs. no software--and assuming there are no other possibilities.

Given a system of software copyright, software development is usually linked with the existence of an owner who controls the software's use. As long as this linkage exists, we are often faced with the choice of proprietary software or none. However, this linkage is not inherent or inevitable; it is a consequence of the specific social/legal policy decision that we are questioning: the decision to have owners. To formulate the choice as between proprietary software vs. no software is begging the question.

The Argument against Having Owners

The question at hand is, "Should development of software be linked with having owners to restrict the use of it?"

In order to decide this, we have to judge the effect on society of each of those two activities *independently*: the effect of developing the software (regardless of its terms of distribution), and the effect of restricting its use (assuming the software has been developed). If one of these activities is helpful and the other is harmful, we would be better off dropping the linkage and doing only the helpful one.

To put it another way, if restricting the distribution of a program already developed is harmful to society overall, then an ethical software developer will reject the option of doing so.

To determine the effect of restricting sharing, we need to compare the value to society of a restricted (i. e., proprietary) program with that of the same program, available to everyone. This means comparing two possible worlds.

This analysis also addresses the simple counterargument sometimes made that "the benefit to the neighbor of giving him or her a copy of a program is cancelled by the harm done to the owner." This counterargument assumes that the harm and the benefit are equal in magnitude. The analysis involves comparing these magnitudes, and shows that the benefit is much greater.

To elucidate this argument, let's apply it in another area: road construction.

It would be possible to fund the construction of all roads with tolls. This would entail having toll booths at all street corners. Such a system would provide a great incentive to improve roads. It would also have the virtue of causing the users of any given road to pay for that road. However, a toll booth is an artificial obstruction to smooth driving--artificial, because it is not a consequence of how roads or cars work.

Comparing free roads and toll roads by their usefulness, we find that (all else being equal) roads without toll booths are cheaper to construct, cheaper to run, safer, and more efficient to use.⁽²⁾ In a poor country, tolls may make the roads unavailable to many citizens. The roads without toll booths thus offer more benefit to society at less cost; they are preferable for society. Therefore, society should choose to fund roads in another way, not by means of toll booths. Use of roads, once built, should be free.

When the advocates of toll booths propose them as *merely* a way of raising funds, they distort the choice that is available. Toll booths do raise funds, but they do something else as well: in effect, they degrade the road. The toll road is not as good as the free road; giving us more or technically superior roads may not be an improvement if this means substituting toll roads for free roads.

Of course, the construction of a free road does cost money, which the public must somehow pay.

However, this does not imply the inevitability of toll booths. We who must in either case pay will get more value for our money by buying a free road.

I am not saying that a toll road is worse than no road at all. That would be true if the toll were so great that hardly anyone used the road--but this is an unlikely policy for a toll collector. However, as long as the toll booths cause significant waste and inconvenience, it is better to raise the funds in a less obstructive fashion.

To apply the same argument to software development, I will now show that having ``toll booths" for useful software programs costs society dearly: it makes the programs more expensive to construct, more expensive to distribute, and less satisfying and efficient to use. It will follow that program construction should be encouraged in some other way. Then I will go on to explain other methods of encouraging and (to the extent actually necessary) funding software development.

The Harm Done by Obstructing Software

Consider for a moment that a program has been developed, and any necessary payments for its development have been made; now society must choose either to make it proprietary or allow free sharing and use. Assume that the existence of the program and its availability is a desirable thing.[\(3\)](#)

Restrictions on the distribution and modification of the program cannot facilitate its use. They can only interfere. So the effect can only be negative. But how much? And what kind?

Three different levels of material harm come from such obstruction:

- Fewer people use the program.
- None of the users can adapt or fix the program.
- Other developers cannot learn from the program, or base new work on it.

Each level of material harm has a concomitant form of psychosocial harm. This refers to the effect that people's decisions have on their subsequent feelings, attitudes, and predispositions. These changes in people's ways of thinking will then have a further effect on their relationships with their fellow citizens, and can have material consequences.

The three levels of material harm waste part of the value that the program could contribute, but they cannot reduce it to zero. If they waste nearly all the value of the program, then writing the program harms society by at most the effort that went into writing the program. Arguably a program that is profitable to sell must provide some net direct material benefit.

However, taking account of the concomitant psychosocial harm, there is no limit to the harm that proprietary software development can do.

Obstructing Use of Programs

The first level of harm impedes the simple use of a program. A copy of a program has nearly zero marginal cost (and you can pay this cost by doing the work yourself), so in a free market, it would have nearly zero price. A license fee is a significant disincentive to use the program. If a widely-useful program is proprietary, far fewer people will use it.

It is easy to show that the total contribution of a program to society is reduced by assigning an owner to it. Each potential user of the program, faced with the need to pay to use it, may choose to pay, or may forego use of the program. When a user chooses to pay, this is a zero-sum transfer of wealth between two parties. But each time someone chooses to forego use of the program, this harms that person without benefitting anyone. The sum of negative numbers and zeros must be negative.

But this does not reduce the amount of work it takes to *develop* the program. As a result, the efficiency of the whole process, in delivered user satisfaction per hour of work, is reduced.

This reflects a crucial difference between copies of programs and cars, chairs, or sandwiches. There is no copying machine for material objects outside of science fiction. But programs are easy to copy; anyone can produce as many copies as are wanted, with very little effort. This isn't true for material objects because matter is conserved: each new copy has to be built from raw materials in the same way that the first copy was built.

With material objects, a disincentive to use them makes sense, because fewer objects bought means less raw material and work needed to make them. It's true that there is usually also a startup cost, a development cost, which is spread over the production run. But as long as the marginal cost of production is significant, adding a share of the development cost does not make a qualitative difference. And it does not require restrictions on the freedom of ordinary users.

However, imposing a price on something that would otherwise be free is a qualitative change. A centrally-imposed fee for software distribution becomes a powerful disincentive.

What's more, central production as now practiced is inefficient even as a means of delivering copies of software. This system involves enclosing physical disks or tapes in superfluous packaging, shipping large numbers of them around the world, and storing them for sale. This cost is presented as an expense of doing business; in truth, it is part of the waste caused by having owners.

Damaging Social Cohesion

Suppose that both you and your neighbor would find it useful to run a certain program. In ethical concern for your neighbor, you should feel that proper handling of the situation will enable both of you to use it. A proposal to permit only one of you to use the program, while restraining the other, is divisive; neither you nor your neighbor should find it acceptable.

Signing a typical software license agreement means betraying your neighbor: "I promise to deprive my neighbor of this program so that I can have a copy for myself." People who make such choices feel internal psychological pressure to justify them, by downgrading the importance of helping one's neighbors--thus public spirit suffers. This is psychosocial harm associated with the material harm of discouraging use of the program.

Many users unconsciously recognize the wrong of refusing to share, so they decide to ignore the licenses and laws, and share programs anyway. But they often feel guilty about doing so. They know that they must break the laws in order to be good neighbors, but they still consider the laws authoritative, and they conclude that being a good neighbor (which they are) is naughty or shameful. That is also a kind of psychosocial harm, but one can escape it by deciding that these licenses and laws have no moral force.

Programmers also suffer psychosocial harm knowing that many users will not be allowed to use their work. This leads to an attitude of cynicism or denial. A programmer may describe enthusiastically the work that he finds technically exciting; then when asked, "Will I be permitted to use it?", his face falls, and he admits the answer is no. To avoid feeling discouraged, he either ignores this fact most of the time or adopts a cynical stance designed to minimize the importance of it.

Since the age of Reagan, the greatest scarcity in the United States is not technical innovation, but rather the willingness to work together for the public good. It makes no sense to encourage the former at the expense of the latter.

Obstructing Custom Adaptation of Programs

The second level of material harm is the inability to adapt programs. The ease of modification of software is one of its great advantages over older technology. But most commercially available software isn't available for modification, even after you buy it. It's available for you to take it or leave it, as a black box--that is all.

A program that you can run consists of a series of numbers whose meaning is obscure. No one, not even a good programmer, can easily change the numbers to make the program do something different.

Programmers normally work with the "source code" for a program, which is written in a programming language such as Fortran or C. It uses names to designate the data being used and the parts of the program, and it represents operations with symbols such as '+' for addition and '-' for subtraction. It is designed to help programmers read and change programs. Here is an example; a program to calculate the distance between two points in a plane:

```
float
distance (p0, p1)
    struct point p0, p1;
```

```

{
  float xdist = p1.x - p0.x;
  float ydist = p1.y - p0.y;
  return sqrt (xdist * xdist + ydist * ydist);
}

```

Here is the same program in executable form, on the computer I normally use:

```

1314258944      -232267772      -231844864      1634862
1411907592      -231844736      2159150         1420296208
-234880989      -234879837      -234879966      -232295424
1644167167      -3214848        1090581031      1962942495
572518958       -803143692      1314803317

```

Source code is useful (at least potentially) to every user of a program. But most users are not allowed to have copies of the source code. Usually the source code for a proprietary program is kept secret by the owner, lest anybody else learn something from it. Users receive only the files of incomprehensible numbers that the computer will execute. This means that only the program's owner can change the program.

A friend once told me of working as a programmer in a bank for about six months, writing a program similar to something that was commercially available. She believed that if she could have gotten source code for that commercially available program, it could easily have been adapted to their needs. The bank was willing to pay for this, but was not permitted to--the source code was a secret. So she had to do six months of make-work, work that counts in the GNP but was actually waste.

The MIT Artificial Intelligence Lab (AI Lab) received a graphics printer as a gift from Xerox around 1977. It was run by free software to which we added many convenient features. For example, the software would notify a user immediately on completion of a print job. Whenever the printer had trouble, such as a paper jam or running out of paper, the software would immediately notify all users who had print jobs queued. These features facilitated smooth operation.

Later Xerox gave the AI Lab a newer, faster printer, one of the first laser printers. It was driven by proprietary software that ran in a separate dedicated computer, so we couldn't add any of our favorite features. We could arrange to send a notification when a print job was sent to the dedicated computer, but not when the job was actually printed (and the delay was usually considerable). There was no way to find out when the job was actually printed; you could only guess. And no one was informed when there was a paper jam, so the printer often went for an hour without being fixed.

The system programmers at the AI Lab were capable of fixing such problems, probably as capable as the original authors of the program. Xerox was uninterested in fixing them, and chose to prevent us, so we were forced to accept the problems. They were never fixed.

Most good programmers have experienced this frustration. The bank could afford to solve the problem by writing a new program from scratch, but a typical user, no matter how skilled, can only give up.

Giving up causes psychosocial harm--to the spirit of self-reliance. It is demoralizing to live in a house that you cannot rearrange to suit your needs. It leads to resignation and discouragement, which can spread to affect other aspects of one's life. People who feel this way are unhappy and do not do good work.

Imagine what it would be like if recipes were hoarded in the same fashion as software. You might say, "How do I change this recipe to take out the salt?" and the great chef would respond, "How dare you insult my recipe, the child of my brain and my palate, by trying to tamper with it? You don't have the judgment to change my recipe and make it work right!"

"But my doctor says I'm not supposed to eat salt! What can I do? Will you take out the salt for me?"

"I would be glad to do that; my fee is only \$50,000." Since the owner has a monopoly on changes, the fee tends to be large. "However, right now I don't have time. I am busy with a commission to design a new recipe for ship's biscuit for the Navy Department. I might get around to you in about two years."

Obstructing Software Development

The third level of material harm affects software development. Software development used to be an evolutionary process, where a person would take an existing program and rewrite parts of it for one new feature, and then another person would rewrite parts to add another feature; in some cases, this continued over a period of twenty years. Meanwhile, parts of the program would be "cannibalized" to form the beginnings of other programs.

The existence of owners prevents this kind of evolution, making it necessary to start from scratch when developing a program. It also prevents new practitioners from studying existing programs to learn useful techniques or even how large programs can be structured.

Owners also obstruct education. I have met bright students in computer science who have never seen the source code of a large program. They may be good at writing small programs, but they can't begin to learn the different skills of writing large ones if they can't see how others have done it.

In any intellectual field, one can reach greater heights by standing on the shoulders of others. But that is no longer generally allowed in the software field--you can only stand on the shoulders of the other people *in your own company*.

The associated psychosocial harm affects the spirit of scientific cooperation, which used to be so strong that scientists would cooperate even when their countries were at war. In this spirit, Japanese

oceanographers abandoning their lab on an island in the Pacific carefully preserved their work for the invading U.S. Marines, and left a note asking them to take good care of it.

Conflict for profit has destroyed what international conflict spared. Nowadays scientists in many fields don't publish enough in their papers to enable others to replicate the experiment. They publish only enough to let readers marvel at how much they were able to do. This is certainly true in computer science, where the source code for the programs reported on is usually secret.

It Does Not Matter How Sharing Is Restricted

I have been discussing the effects of preventing people from copying, changing, and building on a program. I have not specified how this obstruction is carried out, because that doesn't affect the conclusion. Whether it is done by copy protection, or copyright, or licenses, or encryption, or ROM cards, or hardware serial numbers, if it *succeeds* in preventing use, it does harm.

Users do consider some of these methods more obnoxious than others. I suggest that the methods most hated are those that accomplish their objective.

Software Should be Free

I have shown how ownership of a program--the power to restrict changing or copying it--is obstructive. Its negative effects are widespread and important. It follows that society shouldn't have owners for programs.

Another way to understand this is that what society needs is free software, and proprietary software is a poor substitute. Encouraging the substitute is not a rational way to get what we need.

Vaclav Havel has advised us to ``Work for something because it is good, not just because it stands a chance to succeed." A business making proprietary software stands a chance of success in its own narrow terms, but it is not what is good for society.

Why People Will Develop Software

If we eliminate copyright as a means of encouraging people to develop software, at first less software will be developed, but that software will be more useful. It is not clear whether the overall delivered user satisfaction will be less; but if it is, or if we wish to increase it anyway, there are other ways to encourage development, just as there are ways besides toll booths to raise money for streets. Before I talk about how that can be done, first I want to question how much artificial encouragement is truly necessary.

Programming is Fun

There are some lines of work that few will enter except for money; road construction, for example. There are other fields of study and art in which there is little chance to become rich, which people enter for their fascination or their perceived value to society. Examples include mathematical logic, classical music, and archaeology; and political organizing among working people. People compete, more sadly than bitterly, for the few funded positions available, none of which is funded very well. They may even pay for the chance to work in the field, if they can afford to.

Such a field can transform itself overnight if it begins to offer the possibility of getting rich. When one worker gets rich, others demand the same opportunity. Soon all may demand large sums of money for doing what they used to do for pleasure. When another couple of years go by, everyone connected with the field will deride the idea that work would be done in the field without large financial returns. They will advise social planners to ensure that these returns are possible, prescribing special privileges, powers, and monopolies as necessary to do so.

This change happened in the field of computer programming in the past decade. Fifteen years ago, there were articles on "computer addiction": users were "onlining" and had hundred-dollar-a-week habits. It was generally understood that people frequently loved programming enough to break up their marriages. Today, it is generally understood that no one would program except for a high rate of pay. People have forgotten what they knew fifteen years ago.

When it is true at a given time that most people will work in a certain field only for high pay, it need not remain true. The dynamic of change can run in reverse, if society provides an impetus. If we take away the possibility of great wealth, then after a while, when the people have readjusted their attitudes, they will once again be eager to work in the field for the joy of accomplishment.

The question, "How can we pay programmers?" becomes an easier question when we realize that it's not a matter of paying them a fortune. A mere living is easier to raise.

Funding Free Software

Institutions that pay programmers do not have to be software houses. Many other institutions already exist that can do this.

Hardware manufacturers find it essential to support software development even if they cannot control the use of the software. In 1970, much of their software was free because they did not consider restricting it. Today, their increasing willingness to join consortiums shows their realization that owning the software is not what is really important for them.

Universities conduct many programming projects. Today they often sell the results, but in the 1970s they did not. Is there any doubt that universities would develop free software if they were not allowed to sell software? These projects could be supported by the same government contracts and grants that now support proprietary software development.

It is common today for university researchers to get grants to develop a system, develop it nearly to the point of completion and call that ``finished'', and then start companies where they really finish the project and make it usable. Sometimes they declare the unfinished version ``free''; if they are thoroughly corrupt, they instead get an exclusive license from the university. This is not a secret; it is openly admitted by everyone concerned. Yet if the researchers were not exposed to the temptation to do these things, they would still do their research.

Programmers writing free software can make their living by selling services related to the software. I have been hired to port the [GNU C compiler](#) to new hardware, and to make user-interface extensions to [GNU Emacs](#). (I offer these improvements to the public once they are done.) I also teach classes for which I am paid.

I am not alone in working this way; there is now a successful, growing corporation which does no other kind of work. Several other companies also provide commercial support for the free software of the GNU system. This is the beginning of the independent software support industry--an industry that could become quite large if free software becomes prevalent. It provides users with an option generally unavailable for proprietary software, except to the very wealthy.

New institutions such as the [Free Software Foundation](#) can also fund programmers. Most of the Foundation's funds come from users buying tapes through the mail. The software on the tapes is free, which means that every user has the freedom to copy it and change it, but many nonetheless pay to get copies. (Recall that ``free software'' refers to freedom, not to price.) Some users who already have a copy order tapes as a way of making a contribution they feel we deserve. The Foundation also receives sizable donations from computer manufacturers.

The Free Software Foundation is a charity, and its income is spent on hiring as many programmers as possible. If it had been set up as a business, distributing the same free software to the public for the same fee, it would now provide a very good living for its founder.

Because the Foundation is a charity, programmers often work for the Foundation for half of what they could make elsewhere. They do this because we are free of bureaucracy, and because they feel satisfaction in knowing that their work will not be obstructed from use. Most of all, they do it because programming is fun. In addition, volunteers have written many useful programs for us. (Even technical writers have begun to volunteer.)

This confirms that programming is among the most fascinating of all fields, along with music and art. We don't have to fear that no one will want to program.

What Do Users Owe to Developers?

There is a good reason for users of software to feel a moral obligation to contribute to its support.

Developers of free software are contributing to the users' activities, and it is both fair and in the long-term interest of the users to give them funds to continue.

However, this does not apply to proprietary software developers, since obstructionism deserves a punishment rather than a reward.

We thus have a paradox: the developer of useful software is entitled to the support of the users, but any attempt to turn this moral obligation into a requirement destroys the basis for the obligation. A developer can either deserve a reward or demand it, but not both.

I believe that an ethical developer faced with this paradox must act so as to deserve the reward, but should also entreat the users for voluntary donations. Eventually the users will learn to support developers without coercion, just as they have learned to support public radio and television stations.

What Is Software Productivity?

If software were free, there would still be programmers, but perhaps fewer of them. Would this be bad for society?

Not necessarily. Today the advanced nations have fewer farmers than in 1900, but we do not think this is bad for society, because the few deliver more food to the consumers than the many used to do. We call this improved productivity. Free software would require far fewer programmers to satisfy the demand, because of increased software productivity at all levels:

- Wider use of each program that is developed.
- The ability to adapt existing programs for customization instead of starting from scratch.
- Better education of programmers.
- The elimination of duplicate development effort.

Those who object to cooperation claiming it would result in the employment of fewer programmers are actually objecting to increased productivity. Yet these people usually accept the widely-held belief that the software industry needs increased productivity. How is this?

``Software productivity" can mean two different things: the overall productivity of all software development, or the productivity of individual projects. Overall productivity is what society would like to improve, and the most straightforward way to do this is to eliminate the artificial obstacles to cooperation which reduce it. But researchers who study the field of ``software productivity" focus only on the second, limited, sense of the term, where improvement requires difficult technological advances.

Is Competition Inevitable?

Is it inevitable that people will try to compete, to surpass their rivals in society? Perhaps it is. But competition itself is not harmful; the harmful thing is *combat*.

There are many ways to compete. Competition can consist of trying to achieve ever more, to outdo what others have done. For example, in the old days, there was competition among programming wizards--competition for who could make the computer do the most amazing thing, or for who could make the shortest or fastest program for a given task. This kind of competition can benefit everyone, *as long as* the spirit of good sportsmanship is maintained.

Constructive competition is enough competition to motivate people to great efforts. A number of people are competing to be the first to have visited all the countries on Earth; some even spend fortunes trying to do this. But they do not bribe ship captains to strand their rivals on desert islands. They are content to let the best person win.

Competition becomes combat when the competitors begin trying to impede each other instead of advancing themselves--when "Let the best person win" gives way to "Let me win, best or not." Proprietary software is harmful, not because it is a form of competition, but because it is a form of combat among the citizens of our society.

Competition in business is not necessarily combat. For example, when two grocery stores compete, their entire effort is to improve their own operations, not to sabotage the rival. But this does not demonstrate a special commitment to business ethics; rather, there is little scope for combat in this line of business short of physical violence. Not all areas of business share this characteristic. Withholding information that could help everyone advance is a form of combat.

Business ideology does not prepare people to resist the temptation to combat the competition. Some forms of combat have been banned with anti-trust laws, truth in advertising laws, and so on, but rather than generalizing this to a principled rejection of combat in general, executives invent other forms of combat which are not specifically prohibited. Society's resources are squandered on the economic equivalent of factional civil war.

"Why Don't You Move to Russia?"

In the United States, any advocate of other than the most extreme form of laissez-faire selfishness has often heard this accusation. For example, it is leveled against the supporters of a national health care system, such as is found in all the other industrialized nations of the free world. It is leveled against the advocates of public support for the arts, also universal in advanced nations. The idea that citizens have any obligation to the public good is identified in America with Communism. But how similar are these ideas?

Communism as was practiced in the Soviet Union was a system of central control where all activity was regimented, supposedly for the common good, but actually for the sake of the members of the

Communist party. And where copying equipment was closely guarded to prevent illegal copying.

The American system of software copyright exercises central control over distribution of a program, and guards copying equipment with [automatic copying-protection schemes](#) to prevent illegal copying.

By contrast, I am working to build a system where people are free to decide their own actions; in particular, free to help their neighbors, and free to alter and improve the tools which they use in their daily lives. A system based on voluntary cooperation and on decentralization.

Thus, if we are to judge views by their resemblance to Russian Communism, it is the software owners who are the Communists.

The Question of Premises

I make the assumption in this paper that a user of software is no less important than an author, or even an author's employer. In other words, their interests and needs have equal weight, when we decide which course of action is best.

This premise is not universally accepted. Many maintain that an author's employer is fundamentally more important than anyone else. They say, for example, that the purpose of having owners of software is to give the author's employer the advantage he deserves--regardless of how this may affect the public.

It is no use trying to prove or disprove these premises. Proof requires shared premises. So most of what I have to say is addressed only to those who share the premises I use, or at least are interested in what their consequences are. For those who believe that the owners are more important than everyone else, this paper is simply irrelevant.

But why would a large number of Americans accept a premise that elevates certain people in importance above everyone else? Partly because of the belief that this premise is part of the legal traditions of American society. Some people feel that doubting the premise means challenging the basis of society.

It is important for these people to know that this premise is not part of our legal tradition. It never has been.

Thus, the Constitution says that the purpose of copyright is to "promote the progress of science and the useful arts." The Supreme Court has elaborated on this, stating in *Fox Film vs. Doyal* that "The sole interest of the United States and the primary object in conferring the [copyright] monopoly lie in the general benefits derived by the public from the labors of authors."

We are not required to agree with the Constitution or the Supreme Court. (At one time, they both condoned slavery.) So their positions do not disprove the owner supremacy premise. But I hope that the

awareness that this is a radical right-wing assumption rather than a traditionally recognized one will weaken its appeal.

Conclusion

We like to think that our society encourages helping your neighbor; but each time we reward someone for obstructionism, or admire them for the wealth they have gained in this way, we are sending the opposite message.

Software hoarding is one form of our general willingness to disregard the welfare of society for personal gain. We can trace this disregard from Ronald Reagan to Jim Bakker, from Ivan Boesky to Exxon, from failing banks to failing schools. We can measure it with the size of the homeless population and the prison population. The antisocial spirit feeds on itself, because the more we see that other people will not help us, the more it seems futile to help them. Thus society decays into a jungle.

If we don't want to live in a jungle, we must change our attitudes. We must start sending the message that a good citizen is one who cooperates when appropriate, not one who is successful at taking from others. I hope that the free software movement will contribute to this: at least in one area, we will replace the jungle with a more efficient system which encourages and runs on voluntary cooperation.

Footnotes

1. The word "free" in "free software" refers to freedom, not to price; the price paid for a copy of a free program may be zero, or small, or (rarely) quite large.
2. The issues of pollution and traffic congestion do not alter this conclusion. If we wish to make driving more expensive to discourage driving in general, it is disadvantageous to do this using toll booths, which contribute to both pollution and congestion. A tax on gasoline is much better. Likewise, a desire to enhance safety by limiting maximum speed is not relevant; a free-access road enhances the average speed by avoiding stops and delays, for any given speed limit.
3. One might regard a particular computer program as a harmful thing that should not be available at all, like the Lotus Marketplace database of personal information, which was withdrawn from sale due to public disapproval. Most of what I say does not apply to this case, but it makes little sense to argue for having an owner on the grounds that the owner will make the program less available. The owner will not make it *completely* unavailable, as one would wish in the case of a program whose use is considered destructive.

This essay is published in [*Free Software, Free Society: The Selected Essays of Richard M. Stallman*](#)

[Other Texts to Read](#)

Translations of this page:

[[简体中文](#) | [繁體中文](#) | [češky](#) | [Deutsch](#) | [English](#) | [Español](#) | [Français](#) | [••••](#) | [Bahasa Indonesia](#) | [Português](#) | [Suomi](#)]

Return to the [GNU Project home page](#).

Please send FSF & GNU inquiries to gnu@gnu.org. There are also [other ways to contact](#) the FSF.

Please send broken links and other corrections (or suggestions) to webmasters@gnu.org.

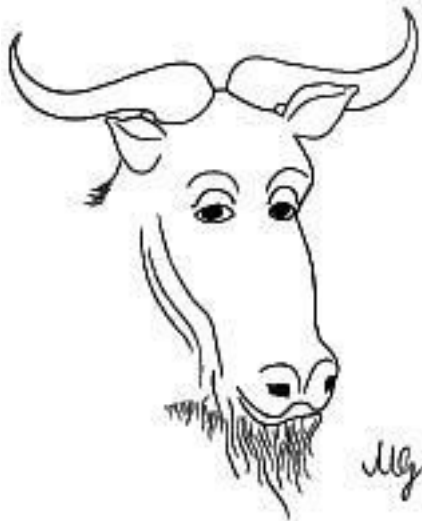
Please see the [Translations README](#) for information on coordinating and submitting translations of this article.

Copyright (C) 1998, 2001 Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA
Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

Updated: \$Date: 2004/04/09 11:49:18 \$ \$Author: the_duke \$

[Translations](#) of this page

The Free Software Definition



We maintain this free software definition to show clearly what must be true about a particular software program for it to be considered free software.

“Free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer.”

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies 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). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to [anyone anywhere](#). Being free to do these things means (among other things) that you do not have to ask or pay for permission.

You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be

required to notify anyone in particular, or in any particular way.

The freedom to use a program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job, and without being required to communicate subsequently with the developer or any other specific entity.

The freedom to redistribute copies must include binary or executable forms of the program, as well as source code, for both modified and unmodified versions. (Distributing programs in runnable form is necessary for conveniently installable free operating systems.) It is ok if there is no way to produce a binary or executable form for a certain program (since some languages don't support that feature), but you must have the freedom to redistribute such forms should you find or develop a way to make them.

In order for the freedoms to make changes, and to publish improved versions, to be meaningful, you must have access to the source code of the program. Therefore, accessibility of source code is a necessary condition for free software.

In order for these freedoms to be real, they must be irrevocable as long as you do nothing wrong; if the developer of the software has the power to revoke the license, without your doing anything to give cause, the software is not free.

However, certain kinds of rules about the manner of distributing free software are acceptable, when they don't conflict with the central freedoms. For example, copyleft (very simply stated) is the rule that when redistributing the program, you cannot add restrictions to deny other people the central freedoms. This rule does not conflict with the central freedoms; rather it protects them.

Thus, you may have paid money to get copies of free software, or you may have obtained copies at no charge. But regardless of how you got your copies, you always have the freedom to copy and change the software, even to [sell copies](#).

``Free software" does not mean ``non-commercial". A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important.

Rules about how to package a modified version are acceptable, if they don't effectively block your freedom to release modified versions. Rules that ``if you make the program available in this way, you must make it available in that way also" can be acceptable too, on the same condition. (Note that such a rule still leaves you the choice of whether to publish the program or not.) It is also acceptable for the license to require that, if you have distributed a modified version and a previous developer asks for a copy of it, you must send one.

In the GNU project, we use [``copyleft"](#) to protect these freedoms legally for everyone. But [non-copylefted free software](#) also exists. We believe there are important reasons why [it is better to use](#)

[copyleft](#), but if your program is non-copylefted free software, we can still use it.

See [Categories of Free Software](#) for a description of how "free software," "copylefted software" and other categories of software relate to each other.

Sometimes government export control regulations and trade sanctions can constrain your freedom to distribute copies of programs internationally. Software developers do not have the power to eliminate or override these restrictions, but what they can and must do is refuse to impose them as conditions of use of the program. In this way, the restrictions will not affect activities and people outside the jurisdictions of these governments.

Most free software licenses are based on copyright, and there are limits on what kinds of requirements can be imposed through copyright. If a copyright-based license respects freedom in the ways described above, it is unlikely to have some other sort of problem that we never anticipated (though this does happen occasionally). However, some free software licenses are based on contracts, and contracts can impose a much larger range of possible restrictions. That means there are many possible ways such a license could be unacceptably restrictive and non-free.

We can't possibly list all the possible contract restrictions that would be unacceptable. If a contract-based license restricts the user in an unusual way that copyright-based licenses cannot, and which isn't mentioned here as legitimate, we will have to think about it, and we will probably decide it is non-free.

When talking about free software, it is best to avoid using terms like "give away" or "for free", because those terms imply that the issue is about price, not freedom. Some common terms such as "piracy" embody opinions we hope you won't endorse. See [Confusing Words and Phrases that are Worth Avoiding](#) for a discussion of these terms. We also have a list of [translations of "free software"](#) into various languages.

Finally, note that criteria such as those stated in this free software definition require careful thought for their interpretation. To decide whether a specific software license qualifies as a free software license, we judge it based on these criteria to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, we reject it, even if we did not anticipate the issue in these criteria. Sometimes a license requirement raises an issue that calls for extensive thought, including discussions with a lawyer, before we can decide if the requirement is acceptable. When we reach a conclusion about a new issue, we often update these criteria to make it easier to see why certain licenses do or don't qualify.

If you are interested in whether a specific license qualifies as a free software license, see our [list of licenses](#). If the license you are concerned with is not listed there, you can ask us about it by sending us email at [<licensing@gnu.org>](mailto:licensing@gnu.org).

Another group has started using the term "open source" to mean something close (but not identical) to "free software". We prefer the term "free software" because, once you have heard it refers to freedom rather than price, [it calls to mind freedom](#).

[Other Texts to Read](#)

Translations of this page:

[[Česky](#) | [Dansk](#) | [Deutsch](#) | [English](#) | [Español](#) | [Français](#) | [Galego](#) | [••••](#) | [Hrvatski](#) | [Bahasa Indonesia](#) | [Italiano](#) | [日本語](#) | [_____](#) | [Magyar](#) | [Nederlands](#) | [Norsk](#) | [Polski](#) | [Português](#) | [Română](#) | [_____](#) | [Slovensko](#) | [Türkçe](#)]

Return to the [GNU Project home page](#).

Please send FSF & GNU inquiries to gnu@gnu.org. There are also [other ways to contact](#) the FSF.

Please send broken links and other corrections (or suggestions) to webmasters@gnu.org.

Please see the [Translations README](#) for information on coordinating and submitting translations of this article.

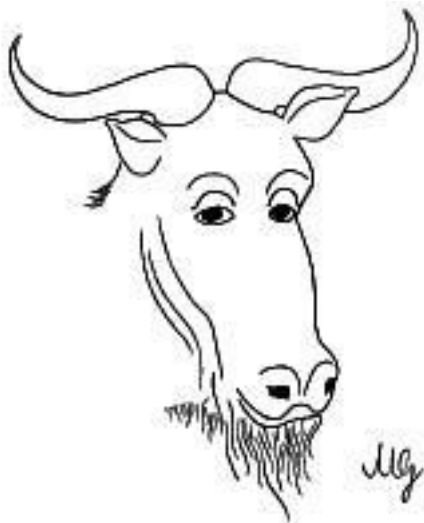
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

Updated: \$Date: 2004/04/09 09:19:00 \$ \$Author: the_duke \$

[Translations](#) of this page

Why "Free Software" is better than "Open Source"



While free software by any other name would give you the same freedom, it makes a big difference which name we use: different words *convey different ideas*.

In 1998, some of the people in the free software community began using the term "[open source software](#)" instead of "[free software](#)" to describe what they do. The term "open source" quickly became associated with a different approach, a different philosophy, different values, and even a different criterion for which licenses are acceptable. The Free Software movement and the Open Source movement are today [separate movements](#) with different views and goals, although we can and do work together on some practical projects.

The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, "Open source is a development methodology; free software is a social movement." For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution.

Relationship between the Free Software movement and Open Source movement

The Free Software movement and the Open Source movement are like two political camps within the free software community.

Radical groups in the 1960s developed a reputation for factionalism: organizations split because of

disagreements on details of strategy, and then treated each other as enemies. Or at least, such is the image people have of them, whether or not it was true.

The relationship between the Free Software movement and the Open Source movement is just the opposite of that picture. 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. We don't think of the Open Source movement as an enemy. The enemy is [proprietary software](#).

We are not against the Open Source movement, but we don't want to be lumped in with them. We acknowledge that they have contributed to our community, but we created this community, and we want people to know this. We want people to associate our achievements with our values and our philosophy, not with theirs. We want to be heard, not obscured behind a group with different views. To prevent people from thinking we are part of them, we take pains to avoid using the word ``open" to describe free software, or its contrary, ``closed", in talking about non-free software.

So please mention the Free Software movement when you talk about the work we have done, and the software we have developed--such as the [GNU/Linux](#) operating system.

Comparing the two terms

This rest of this article compares the two terms ``free software" and ``open source". It shows why the term ``open source" does not solve any problems, and in fact creates some.

Ambiguity

The term ``free software" has an ambiguity problem: an unintended meaning, ``Software you can get for zero price," fits the term just as well as the intended meaning, ``software which gives the user certain freedoms." We address this problem by publishing a [more precise definition of free software](#), but this is not a perfect solution; it cannot completely eliminate the problem. An unambiguously correct term would be better, if it didn't have other problems.

Unfortunately, all the alternatives in English have problems of their own. We've looked at many alternatives that people have suggested, but none is so clearly ``right" that switching to it would be a good idea. Every proposed replacement for ``free software" has a similar kind of semantic problem, or worse--and this includes ``open source software."

The official definition of ``open source software," as published by the Open Source Initiative, is very close to our definition of free software; however, it is a little looser in some respects, and they have accepted a few licenses that we consider unacceptably restrictive of the users. However, the obvious meaning for the expression ``open source software" is ``You can look at the source code." This is a much weaker criterion than free software; it includes free software, but also includes [semi-free](#) programs such

as Xv, and even some [proprietary](#) programs, including Qt under its original license (before the QPL).

That obvious meaning for ``open source" is not the meaning that its advocates intend. The result is that most people misunderstand what those advocates are advocating. Here is how writer Neal Stephenson defined ``open source":

Linux is ``open source" software meaning, simply, that anyone can get copies of its source code files.

I don't think he deliberately sought to reject or dispute the ``official" definition. I think he simply applied the conventions of the English language to come up with a meaning for the term. The state of Kansas published a similar definition:

Make use of open-source software (OSS). OSS is software for which the source code is freely and publicly available, though the specific licensing agreements vary as to what one is allowed to do with that code.

Of course, the open source people have tried to deal with this by publishing a precise definition for the term, just as we have done for ``free software."

But the explanation for ``free software" is simple--a person who has grasped the idea of ``free speech, not free beer" will not get it wrong again. There is no such succinct way to explain the official meaning of ``open source" and show clearly why the natural definition is the wrong one.

Fear of Freedom

The main argument for the term ``open source software" is that ``free software" makes some people uneasy. That's true: talking about freedom, about ethical issues, about responsibilities as well as convenience, is asking people to think about things they might rather ignore. This can trigger discomfort, and some people may reject the idea for that. It does not follow that society would be better off if we stop talking about these things.

Years ago, free software developers noticed this discomfort reaction, and some started exploring an approach for avoiding it. They figured that by keeping quiet about ethics and freedom, and talking only about the immediate practical benefits of certain free software, they might be able to ``sell" the software more effectively to certain users, especially business. The term ``open source" is offered as a way of doing more of this--a way to be ``more acceptable to business." The views and values of the Open Source movement stem from this decision.

This approach has proved effective, in its own terms. Today many people are switching to free software for purely practical reasons. That is good, as far as it goes, but that isn't all we need to do! Attracting users to free software is not the whole job, just the first step.

Sooner or later these users will be invited to switch back to proprietary software for some practical advantage. Countless companies seek to offer such temptation, and why would users decline? Only if they have learned to *value the freedom* free software gives them, for its own sake. It is up to us to spread this idea--and in order to do that, we have to talk about freedom. A certain amount of the "keep quiet" approach to business can be useful for the community, but we must have plenty of freedom talk too.

At present, we have plenty of "keep quiet", but not enough freedom talk. Most people involved with free software say little about freedom--usually because they seek to be "more acceptable to business." Software distributors especially show this pattern. Some [GNU/Linux](#) operating system distributions add proprietary packages to the basic free system, and they invite users to consider this an advantage, rather than a step backwards from freedom.

We are failing to keep up with the influx of free software users, failing to teach people about freedom and our community as fast as they enter it. This is why non-free software (which Qt was when it first became popular), and partially non-free operating system distributions, find such fertile ground. To stop using the word "free" now would be a mistake; we need more, not less, talk about freedom.

If those using the term "open source" draw more users into our community, that is a contribution, but the rest of us will have to work even harder to bring the issue of freedom to those users' attention. We have to say, "It's free software and it gives you freedom!"--more and louder than ever before.

Would a Trademark Help?

The advocates of "open source software" tried to make it a trademark, saying this would enable them to prevent misuse. This initiative was later dropped, the term being too descriptive to qualify as a trademark; thus, the legal status of "open source" is the same as that of "free software": there is no *legal* constraint on using it. I have heard reports of a number of companies' calling software packages "open source" even though they did not fit the official definition; I have observed some instances myself.

But would it have made a big difference to use a term that is a trademark? Not necessarily.

Companies also made announcements that give the impression that a program is "open source software" without explicitly saying so. For example, one IBM announcement, about a program that did not fit the official definition, said this:

As is common in the open source community, users of the ... technology will also be able to collaborate with IBM ...

This did not actually say that the program *was* "open source", but many readers did not notice that detail. (I should note that IBM was sincerely trying to make this program free software, and later adopted a new license which does make it free software and "open source"; but when that

announcement was made, the program did not qualify as either one.)

And here is how Cygnus Solutions, which was formed to be a free software company and subsequently branched out (so to speak) into proprietary software, advertised some proprietary software products:

Cygnus Solutions is a leader in the open source market and has just launched two products into the [GNU/]Linux marketplace.

Unlike IBM, Cygnus was not trying to make these packages free software, and the packages did not come close to qualifying. But Cygnus didn't actually say that these are ``open source software'', they just made use of the term to give careless readers that impression.

These observations suggest that a trademark would not have truly prevented the confusion that comes with the term ``open source''.

Misunderstandings(?) of ``Open Source''

The Open Source Definition is clear enough, and it is quite clear that the typical non-free program does not qualify. So you would think that ``Open Source company'' would mean one whose products are free software (or close to it), right? Alas, many companies are trying to give it a different meaning.

At the ``Open Source Developers Day'' meeting in August 1998, several of the commercial developers invited said they intend to make only a part of their work free software (or ``open source''). The focus of their business is on developing proprietary add-ons (software or [manuals](#)) to sell to the users of this free software. They ask us to regard this as legitimate, as part of our community, because some of the money is donated to free software development.

In effect, these companies seek to gain the favorable cachet of ``open source'' for their proprietary software products--even though those are not ``open source software''--because they have some relationship to free software or because the same company also maintains some free software. (One company founder said quite explicitly that they would put, into the free package they support, as little of their work as the community would stand for.)

Over the years, many companies have contributed to free software development. Some of these companies primarily developed non-free software, but the two activities were separate; thus, we could ignore their non-free products, and work with them on free software projects. Then we could honestly thank them afterward for their free software contributions, without talking about the rest of what they did.

We cannot do the same with these new companies, because they won't let us. These companies actively invite the public to lump all their activities together; they want us to regard their non-free software as favorably as we would regard a real contribution, although it is not one. They present themselves as

``open source companies," hoping that we will get a warm fuzzy feeling about them, and that we will be fuzzy-minded in applying it.

This manipulative practice would be no less harmful if it were done using the term ``free software." But companies do not seem to use the term ``free software" that way; perhaps its association with idealism makes it seem unsuitable. The term ``open source" opened the door for this.

At a trade show in late 1998, dedicated to the operating system often referred to as [``Linux"](#), the featured speaker was an executive from a prominent software company. He was probably invited on account of his company's decision to ``support" that system. Unfortunately, their form of ``support" consists of releasing non-free software that works with the system--in other words, using our community as a market but not contributing to it.

He said, ``There is no way we will make our product open source, but perhaps we will make it `internal' open source. If we allow our customer support staff to have access to the source code, they could fix bugs for the customers, and we could provide a better product and better service." (This is not an exact quote, as I did not write his words down, but it gets the gist.)

People in the audience afterward told me, ``He just doesn't get the point." But is that so? Which point did he not get?

He did not miss the point of the Open Source movement. That movement does not say users should have freedom, only that allowing more people to look at the source code and help improve it makes for faster and better development. The executive grasped that point completely; unwilling to carry out that approach in full, users included, he was considering implementing it partially, within the company.

The point that he missed is the point that ``open source" was designed not to raise: the point that users *deserve* freedom.

Spreading the idea of freedom is a big job--it needs your help. That's why we stick to the term ``free software" in the GNU Project, so we can help do that job. If you feel that freedom and community are important for their own sake--not just for the convenience they bring--please join us in using the term ``free software".

Joe Barr wrote an article called [Live and let license](#) that gives his perspective on this issue.

Lakhani and Wolf's [paper on the motivation of free software developers](#) says that a considerable fraction are motivated by the view that software should be free. This was despite the fact that they surveyed the developers on SourceForge, a site that does not support the view that this is an ethical issue.

This essay is published in [*Free Software, Free Society: The Selected Essays of Richard M. Stallman.*](#)

See also [Other Texts to Read](#)

Translations of this page:

[[•esky](#) | [English](#) | [Français](#) | [••••](#) | [Italiano](#) | [_____](#) | [Polski](#) | [Română](#) | [_____](#)]

Return to the [GNU Project home page](#).

Please send FSF & GNU inquiries to gnu@gnu.org. There are also [other ways to contact](#) the FSF.

Please send broken links and other corrections (or suggestions) to webmasters@gnu.org.

Please see the [Translations README](#) for information on coordinating and submitting translations of this article.

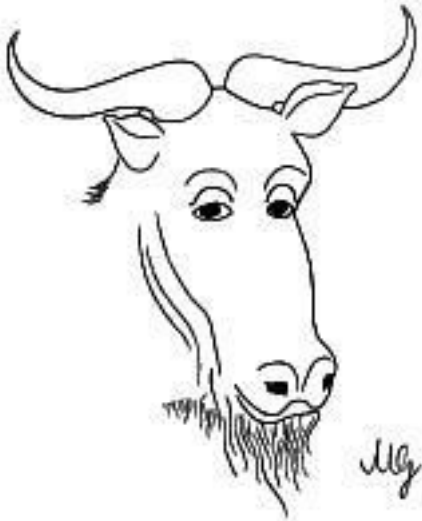
Copyright (C) 1998, 1999, 2000, 2001 Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

Updated: \$Date: 2004/04/09 13:55:51 \$ \$Author: the_duke \$

[Translations](#) of this page

Categories of Free and Non-Free Software

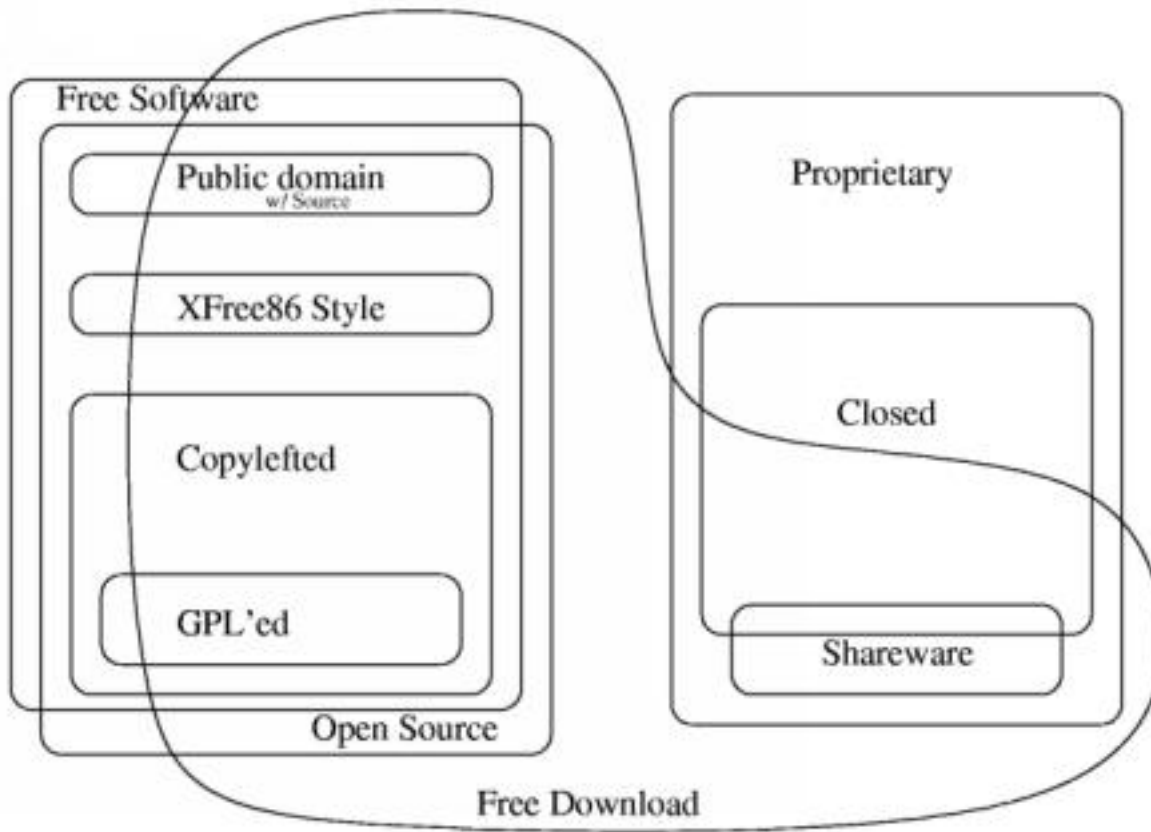


Here is a glossary of various categories of software that are often mentioned in discussions of free software. It explains which categories overlap or are part of other categories.

Table of Contents

- [Other Texts to Read](#)
- [Free software](#)
- [Open source](#)
- [Public domain software](#)
- [Copylefted software](#)
- [Non-copylefted free software](#)
- [GPL-covered software](#)
- [The GNU system](#)
- [GNU programs](#)
- [GNU software](#)
- [Semi-free software](#)
- [Proprietary software](#)
- [Shareware](#)
- [Freeware](#)
- [Commercial software](#)
- [Other Texts to Read](#)

Also note [Confusing Words which You Might Want to Avoid](#).



This diagram by Chao-

Kuei explains the different categories of software. It's available as an [XFig file](#), as a [JPEG picture](#) and as a 1.5 magnified [PNG image](#).

[Free software](#)

Free software is software that comes with permission for anyone to use, copy, and distribute, either verbatim or with modifications, either gratis or for a fee. In particular, this means that source code must be available. ``If it's not source, it's not software." This is a simplified definition; see also the [full definition](#).

We also have a list of [translations of the term "free software"](#) into various other languages.

If a program is free, then it can potentially be included in a free operating system such as GNU, or free versions of the [GNU/Linux system](#).

There are many different ways to make a program free---many questions of detail, which could be decided in more than one way and still make the program free. Some of the possible variations are described below.

Free software is a matter of freedom, not price. But proprietary software companies sometimes use the term ``free software" to refer to price. Sometimes they mean that you can obtain a binary

copy at no charge; sometimes they mean that a copy is included on a computer that you are buying. This has nothing to do with what we mean by free software in the GNU project.

Because of this potential confusion, when a software company says its product is free software, always check the actual distribution terms to see whether users really have all the freedoms that free software implies. Sometimes it really is free software; sometimes it isn't.

Many languages have two separate words for "free" as in freedom and "free" as in zero price. For example, French has "libre" and "gratuit". English has a word "gratis" that refers unambiguously to price, but no common adjective that refers unambiguously to freedom. This is unfortunate, because such a word would be useful here.

Free software is often [more reliable](#) than non-free software.

Open Source software

The term "open source" software is used by some people to mean more or less the same thing as free software. However, their criteria are somewhat less strict; they have accepted some kinds of license restrictions that we have rejected as unacceptable. We prefer the term "[free software](#)"; follow that link to see the reasons.

Public domain software

Public domain software is software that is not copyrighted. If the source code is in the public domain, that is a special case of [non-copylefted free software](#), which means that some copies or modified versions may not be free at all.

In some cases, an executable program can be in the public domain but the source code is not available. This is not free software, because free software requires accessibility of source code. Meanwhile, most free software is not in the public domain; it is copyrighted, and the copyright holders have legally given permission for everyone to use it in freedom, using a free software license.

Sometimes people use the term "public domain" in a loose fashion to mean "free" or "available gratis." However, "public domain" is a legal term and means, precisely, "not copyrighted". For clarity, we recommend using "public domain" for that meaning only, and using other terms to convey the other meanings.

Copylefted software

Copylefted software is free software whose distribution terms do not let redistributors add any additional restrictions when they redistribute or modify the software. This means that every copy of the software, even if it has been modified, must be free software.

In the GNU Project, we copyleft almost all the software we write, because our goal is to give *every* user the freedoms implied by the term "free software." See [Copylefted](#) for more explanation of how copyleft works and why we use it.

Copyleft is a general concept; to actually copyleft a program, you need to use a specific set of distribution terms. There are many possible ways to write copyleft distribution terms, so in principle there can be many copyleft free software licenses. However, in actual practice nearly all copylefted software uses the [GNU General Public License](#). Two different copyleft licenses are usually "incompatible", which means it is illegal to merge the code using one license with the code using the other license; therefore, it is good for the community if people use a single copyleft license.

Non-copylefted free software

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

If a program is free but not copylefted, then some copies or modified versions may not be free at all. A software company can compile the program, with or without modifications, and distribute the executable file as a [proprietary](#) software product.

The [X Window System](#) illustrates this. The X Consortium releases X11 with distribution terms that make it non-copylefted free software. If you wish, you can get a copy which has those distribution terms and is free. However, there are non-free versions as well, and there are popular workstations and PC graphics boards for which non-free versions are the only ones that work. If you are using this hardware, X11 is not free software for you. [The developers of X11 even made X11 non-free for a while.](#)

GPL-covered software

The [GNU GPL \(General Public License\)](#) is one specific set of distribution terms for copylefting a program. The GNU Project uses it as the distribution terms for most GNU software.

The GNU system

The [GNU system](#) is a complete free Unix-like operating system.

A Unix-like operating system consists of many programs. The GNU system includes all the GNU software, as well as many other packages such as the X Window System and TeX which are not GNU software.

We have been developing and accumulating components for the GNU system since 1984; the first test release of a "complete GNU system" was in 1996. In 2001 the GNU system with the Hurd began working reliably. In the mean time, the [GNU/Linux system](#), an offshoot of the GNU

system which uses Linux as the kernel, became a great success in the 90s.

Since the purpose of GNU is to be free, every single component in the GNU system has to be free software. They don't all have to be copylefted, however; any kind of free software is legally suitable to include if it helps meet technical goals. We can and do use non-copylefted free software such as the X Window System.

GNU programs

``GNU programs'' is equivalent to [GNU software](#). A program Foo is a GNU program if it is GNU software. We also sometimes say it is a ``GNU package''.

GNU software

[GNU software](#) is software that is released under the auspices of the [GNU Project](#). Most GNU software is [copylefted](#), but not all; however, all GNU software must be [free software](#).

If a program is GNU software, we also say that it is a GNU program.

Some GNU software is written by [staff](#) of the [Free Software Foundation](#), but most GNU software is contributed by [volunteers](#). Some contributed software is copyrighted by the Free Software Foundation; some is copyrighted by the contributors who wrote it.

Semi-free software

Semi-free software is 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. PGP is an example of a semi-free program.

Semi-free software is much better ethically than [proprietary software](#), but it still poses problems, and we cannot use it in a free operating system.

The restrictions of copyleft are designed to protect the essential freedoms for all users. For us, the only justification for any substantive restriction on using a program is to prevent other people from adding other restrictions. Semi-free programs have additional restrictions, motivated by purely selfish goals.

It is impossible to include semi-free software in a free operating system. This is because the distribution terms for the operating system as a whole are the conjunction of the distribution terms for all the programs in it. Adding one semi-free program to the system would make the system *as a whole* just semi-free. There are two reasons we do not want that to happen:

- We believe that free software should be for everyone--including businesses, not just schools and hobbyists. We want to invite business to use the whole GNU system, and

therefore we must not include a semi-free program in it.

- o Commercial distribution of free operating systems, including the [GNU/Linux system](#), is very important, and users appreciate the convenience of commercial CD-ROM distributions. Including one semi-free program in an operating system would cut off commercial CD-ROM distribution for it.

The Free Software Foundation itself is non-commercial, and therefore we would be legally permitted to use a semi-free program ``internally''. But we don't do that, because that would undermine our efforts to obtain a program which we could also include in GNU.

If there is a job that needs doing with software, then until we have a free program to do the job, the GNU system has a gap. We have to tell volunteers, ``We don't have a program yet to do this job in GNU, so we hope you will write one.'' If we ourselves used a semi-free program to do the job, that would undermine what we say; it would take away the impetus (on us, and on others who might listen to our views) to write a free replacement. So we don't do that.

Proprietary software

Proprietary software is software that is not free or semi-free. Its use, redistribution or modification is prohibited, or requires you to ask for permission, or is restricted so much that you effectively can't do it freely.

The Free Software Foundation follows the rule that we cannot install any proprietary program on our computers except temporarily for the specific purpose of writing a free replacement for that very program. Aside from that, we feel there is no possible excuse for installing a proprietary program.

For example, we felt justified in installing Unix on our computer in the 1980s, because we were using it to write a free replacement for Unix. Nowadays, since free operating systems are available, the excuse is no longer applicable; we have eliminated all our non-free operating systems, and any new computer we install must run a completely free operating system.

We don't insist that users of GNU, or contributors to GNU, have to live by this rule. It is a rule we made for ourselves. But we hope you will decide to follow it too.

Freeware

The term ``freeware'' has no clear accepted definition, but it is commonly used for packages which permit redistribution but not modification (and their source code is not available). These packages are *not* free software, so please don't use ``freeware'' to refer to free software.

Shareware

Shareware is software which comes with permission for people to redistribute copies, but says that anyone who continues to use a copy is *required* to pay a license fee.

Shareware is not free software, or even semi-free. There are two reasons it is not:

- For most shareware, source code is not available; thus, you cannot modify the program at all.
- Shareware does not come with permission to make a copy and install it without paying a license fee, not even for individuals engaging in nonprofit activity. (In practice, people often disregard the distribution terms and do this anyway, but the terms don't permit it.)

Commercial Software

Commercial software is software being developed by a business which aims to make money from the use of the software. ``Commercial" and ``proprietary" are not the same thing! Most commercial software is [proprietary](#), but there is commercial free software, and there is non-commercial non-free software.

For example, GNU Ada is always distributed under the terms of the GNU GPL, and every copy is free software; but its developers sell support contracts. When their salesmen speak to prospective customers, sometimes the customers say, ``We would feel safer with a commercial compiler." The salesmen reply, ``GNU Ada *is* a commercial compiler; it happens to be free software."

For the GNU Project, the emphasis is in the other order: the important thing is that GNU Ada is free software; whether it is commercial is not a crucial question. However, the additional development of GNU Ada that results from its being commercial it is definitely beneficial.

Please help spread the awareness that commercial free software is possible. You can do this by making an effort not to say ``commercial" when you mean ``proprietary."

Other Texts to Read

Translations of this page:

[[Català](#) | [•esky](#) | [Deutsch](#) | [English](#) | [Español](#) | [Français](#) | [Bahasa Indonesia](#) | [Italiano](#) | [日本語](#) | [Polski](#) | [Português](#) | [Slovensko](#)]

Return to the [GNU Project home page](#).

Please send FSF & GNU inquiries to gnu@gnu.org. There are also [other ways to contact](#) the FSF.

Please send broken links and other corrections (or suggestions) to webmasters@gnu.org.

Please see the [Translations README](#) for information on coordinating and submitting translations of this article.

Copyright (C) 1996, 1997, 1998, 2001 Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA
Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

Updated: \$Date: 2004/02/27 10:13:05 \$ \$Author: wkotwica \$

Translations available: [Czech](#) | [French](#) | [Japanese](#) | [Spanish](#)

Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!

David A. Wheeler

dwheeler@dwheeler.com

Revised as of December 31, 2003

This paper provides quantitative data that, in many cases, using [open source software / free software](#) is a reasonable or even superior approach to using their proprietary competition according to various measures. This paper's goal is to show that you should consider using OSS/FS when acquiring software. This paper examines [market share](#), [reliability](#), [performance](#), [scalability](#), [security](#), and [total cost of ownership](#). It also has sections on [non-quantitative issues](#), [unnecessary fears](#), [OSS/FS on the desktop](#), [usage reports](#), [other sites providing related information](#), and ends with some [conclusions](#). An [appendix](#) gives more background information about OSS/FS. You can view this paper at http://www.dwheeler.com/oss_fs_why.html (HTML format). Palm PDA users can view it in [Plucker format](#) (you will also need [Plucker](#) to read it). A short briefing based on this paper is also available in [PDF](#) and [Open Office Impress](#) formats (for the latter, use [Open Office Impress](#)). [Old archived copies](#) and a list of [changes](#) are also available.

1. Introduction

[Open Source Software / Free Software \(OSS/FS\)](#) has risen to great prominence. Briefly, OSS/FS programs are programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program (without having to pay royalties to previous developers).

This goal of this paper is to show that you should consider using OSS/FS when you're looking for software, based on quantitative measures. Some sites provide a few anecdotes on why you should use OSS/FS, but for many that's not enough information to justify using OSS/FS. Instead, this paper emphasizes *quantitative* measures (such as experiments and market studies) on why using OSS/FS products is, in many circumstances, a reasonable or even superior approach. I should note that while I find much to like about OSS/FS, I'm not a rabid advocate; I use both proprietary and OSS/FS products myself. Vendors of proprietary products often work hard to find numbers to support their claims; this page provides a useful antidote of hard figures to aid in comparing proprietary products to OSS/FS.

Note that this paper's goal is *not* to show that all OSS/FS is better than all proprietary software.

Certainly, there are many who believe this is true from [ethical, moral, or social grounds](#), users do have control and flexibility advantages when they can modify and maintain their own software, and some countries perceive political advantages to not depending on a company from another country. However, no numbers could prove the broad claim that OSS/FS is always better. Instead, I'll simply compare commonly-used OSS/FS software with commonly-used proprietary software, to show that at least in certain situations and by certain measures, some OSS/FS software is at least as good or better than its proprietary competition. Of course, some OSS/FS software is technically poor, just as some proprietary software is technically poor, and even very good software may not fit your specific needs. But although most people understand the need to compare proprietary products before using them, many people fail to even consider OSS/FS products. This paper is intended to explain why acquirers should consider OSS/FS alternatives. This paper doesn't examine transition approaches, but it's worth noting that [organizations can transition to OSS/FS in part or in stages](#), which for many is a more practical transition approach.

I'll emphasize the operating system (OS) known as GNU/Linux (which many abbreviate as ["Linux"](#)) and the Apache web server, since these are some of the most visible OSS/FS projects. I'll also primarily compare OSS/FS software to Microsoft's products (such as Windows and IIS), since Windows has a significant market share and Microsoft is one of proprietary software's strongest proponents. I'll mention Unix systems in passing as well, though the situation with Unix is more complex; today's Unix systems include many OSS/FS components or software primarily derived from OSS/FS components. Thus, comparing proprietary Unix systems to OSS/FS systems (when examined as whole systems) is often not as clear-cut. This paper uses the term "Unix-like" to mean systems intentionally similar to Unix; both Unix and GNU/Linux are "Unix-like" systems. The most recent Apple Macintosh OS (MacOS OS X) presents the same kind of complications; older versions of MacOS were wholly proprietary, but Apple's OS has been redesigned so that it's now based on a Unix system with substantial contributions from OSS/FS programs. Indeed, [Apple is now openly encouraging collaboration with OSS/FS developers](#).

This paper omits or at least tries to warn about studies funded by a product's vendor. Remember that [vendor-sponsored studies are often rigged](#) (no matter who the vendor is) to make the vendor look good instead of being fair comparisons. Indeed, after a pair of vendor-funded studies were publicly lambasted, [Forrester Research announced that it will no longer accept projects that involve paid-for, publicized product comparisons](#). This paper includes data over a series of years, not just the past year; all relevant data should be considered when making a decision, instead of arbitrarily ignoring older data. Note that the older data shows that OSS/FS has a history of many positive traits, as opposed to being a temporary phenomenon.

You can get a more detailed explanation of the terms "open source software" and "Free Software", as well as related information, from [the appendix](#) and my [list of Open Source Software / Free Software \(OSS/FS\) references at http://www.dwheeler.com/oss_fs_refs.html](#). Note that those who use the term ["open source software"](#) tend to emphasize technical advantages of such software (such as better

reliability and security), while those who use the term [“Free Software”](#) tend to emphasize freedom from control by another and/or ethical issues. The opposite of OSS/FS is “closed” or “proprietary” software. Software for which the source code that can be viewed, but cannot modified and redistributed without further limitation (e.g., “source viewable” or “open box” software, including “shared source” and “community” licenses), are not considered here since they don’t meet the definition of OSS/FS. Many OSS/FS programs are commercial programs, so don’t make the mistake of thinking OSS/FS is equivalent to “non-commercial” software (indeed, any article making this mistake should be ignored since it is obviously poorly researched). Almost no OSS/FS programs are in the “public domain” (which has a specific legal meaning), so avoid that term as well. Other alternative terms for OSS/FS software include “libre software” (where libre means free as in freedom), free-libre and open-source software (FLOS software or FLOSS), open source / Free Software (OS/FS), free / open source software (FOSS), open-source software (indeed, “open-source” is often used as a general adjective), “freed software,” and even “public service software” (since often these software projects are designed to serve the public at large). OSS/FS is not “freeware”; freeware is usually defined as proprietary software given away without cost, and does not provide any right to examine, modify, or redistribute the source code. The most popular OSS/FS license is the General Public License (GPL); all software released under the GPL is OSS/FS, but not all OSS/FS software uses the GPL; nevertheless, some people do inaccurately use the term “GPL software” when they mean OSS/FS software.

This is a large paper, with many acronyms. A few of the most common acryonyms are:

Acronym	Meaning
GNU	GNU’s Not Unix (a project to create an OSS/FS operating system)
GPL	General Public License (the most common OSS/FS license)
OS, OSes	Operating System, Operating Systems
OSS/FS	Open Source Software/Free Software

Below is data discussing [market share](#), [reliability](#), [performance](#), [scalability](#), [security](#), and [total cost of ownership](#). I close with a brief discussion of [non-quantitative issues](#), [unnecessary fears](#), [OSS/FS on the desktop](#), [usage reports](#), [other sites providing related information](#), and [conclusions](#). An [appendix](#) gives more background information about OSS/FS (definitions, motivations of developers, history, and license types).

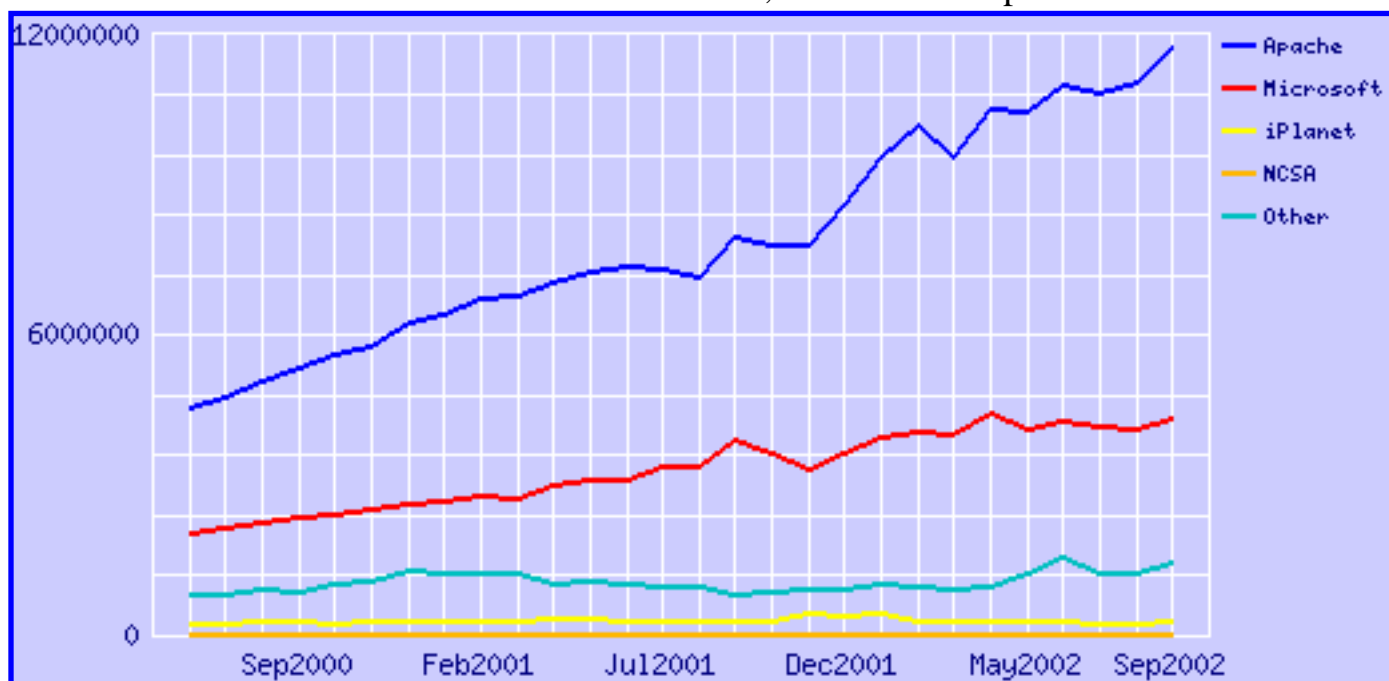
2. Market Share

Many people think that a product is only a winner if it has significant market share. This is lemming-like, but there’s some rationale for this: products with big market shares get applications, trained users, and momentum that reduces future risk. Some writers argue against OSS/FS or GNU/Linux as “not being mainstream”, but if their use is widespread then such statements reflect the past, not the present. There’s excellent evidence that OSS/FS has significant market share in numerous markets:

1. **The most popular web server has always been OSS/FS since such data have been collected. For example, Apache is currently the #1 web server with over twice the market share of its next-ranked competitor. [Netcraft's statistics on web servers](#) have consistently shown Apache (an OSS/FS web server) dominating the public Internet web server market ever since Apache grew into the #1 web server in April 1996. Before that time, the NCSA web server (Apache's ancestor) dominated the web from August 1995 through March 1996 - and it is also OSS/FS.**

Since 2000, Netcraft has been trying to separately count only “active” web sites. The problem is that many web sites have been created that are simply “placeholder” sites (i.e., their domain names have been reserved but they are not being used); such sites are termed “inactive.” Netcraft's count of only the active sites is a more relevant figure than counting all web sites, since the count of active sites shows the web server selected by those who choose to actually develop a web site. When counting active sites, Apache very well; in September 2002, Apache had 66.04% of the web server market, Microsoft had 24.18%, iPlanet had 1.57%, and Zeus had 1.34%.

Market Share for Active Web Servers, June 2000 - September 2002



Netcraft's September 2002 survey also reported on websites based on their “IP address” instead of the host name; this has the effect of removing computers used to serve multiple sites and sites with multiple names. When counting by IP address, Apache has shown a slow increase from 51% at the start of 2001 to 54%, while Microsoft has been unchanged at 35%.

Netcraft's September 2002 survey also polled all the web sites they could find (totaling 35,756,436 sites), and found that of all the sites they could find (active or not), counting by name, Apache had 59.91% of the market and Microsoft had 29.18% of the market, iPlanet had 2.08%,

and Zeus had 1.36%.

Apache's dominance in the web server market has been shown independently by [E-soft](#) - their report on web server market share published October 1, 2002 surveyed 9,045,027 web servers in September 2002 and found that Apache was #1 (66.75%), with Microsoft IIS being #2 (21.83%). E-soft also reports specifically on secure servers (web servers supporting SSL/TLS, such as e-commerce sites), and even here Apache has a commanding 51.26% market share, as compared to Microsoft's 34.85%, Netscape/iPlanet's 5.68%, and Stronghold's 2.71%. Since Stronghold is a repackaging of Apache, Apache's real market share is at least 53.97%. [Another set of independent surveys is conducted by SecuritySpace](#), with basically the same result.

Obviously these figures fluctuate monthly; see [Netcraft](#) and [E-soft](#) for the latest survey figures.

- 2. GNU/Linux is the #2 web serving OS on the public Internet (counting by physical machine), according to a study by Netcraft surveying March and June 2001.** Some of [Netcraft's](#) surveys have also included data on OSes; two 2001 surveys (their [June 2001](#) and [September 2001](#) surveys) found that GNU/Linux is the #2 OS for web servers when counting physical machines (and has been consistently gaining market share since February 1999). As Netcraft themselves point out, the usual Netcraft web server survey (discussed above) counts web server hostnames rather than physical computers, and so it doesn't measure such things as the installed hardware base. Companies can run several thousand web sites on one computer, and most of the world's web sites are located at hosting and co-location companies.

Therefore, Netcraft developed a technique that indicates the number of actual computers being used as Web servers, together with the OS and web server software used (by arranging many IP addresses to reply to Netcraft simultaneously and then analyzing the responses). This is a statistical approach, so many visits to the site are used over a month to build up sufficient certainty. In some cases, the OS detected is that of a "front" device rather than the web server actually performing the task. Still, Netcraft believes that the error margins world-wide are well within the order of plus or minus 10%, and this is in any case the best available data.

Before presenting the data, it's important to explain Netcraft's system for dating the data. Netcraft dates their information based on the web server surveys (not the publication date), and they only report OS summaries from an earlier month. Thus, the survey dated "June 2001" was published in July and covers OS survey results of March 2001, while the survey dated "September 2001" was published in October and covers the operating system survey results of June 2001.

Here's a summary of Netcraft's study results:

OS group	Percentage (March)	Percentage (June)	Composition
----------	--------------------	-------------------	-------------

Windows	49.2%	49.6%	Windows 2000, NT4, NT3, Windows 95, Windows 98
[GNU/]Linux	28.5%	29.6%	[GNU/]Linux
Solaris	7.6%	7.1%	Solaris 2, Solaris 7, Solaris 8
BSD	6.3%	6.1%	BSDI BSD/OS, FreeBSD, NetBSD, OpenBSD
Other Unix	2.4%	2.2%	AIX, Compaq Tru64, HP-UX, IRIX, SCO Unix, SunOS 4 and others
Other non-Unix	2.5%	2.4%	MacOS, NetWare, proprietary IBM OSs
Unknown	3.6%	3.0%	not identified by Netcraft OS detector

Much depends on what you want to measure. Several of the BSDs (FreeBSD, NetBSD, and OpenBSD) are OSS/FS as well; so at least a part of the 6.1% for BSD should be added to GNU/Linux's 29.6% to determine the percentage of OSS/FS OSes being used as web servers. Thus, it's likely that approximately one-third of web serving computers use OSS/FS OSes. There are also regional differences, for example, GNU/Linux leads Windows in Germany, Hungary, the Czech Republic, and Poland.

Well-known web sites using OSS/FS include [Google](#) (GNU/Linux) and [Yahoo](#) (FreeBSD).

If you really want to know about the web server market breakdown of "Unix vs. Windows," you can find that also in this study. All of the various Windows OSes are rolled into a single number (even Windows 95/98 and Windows 2000/NT4/NT3 are merged, although they are fundamentally very different systems). Merging all the Unix-like systems in a similar way produces a total of 44.8% for Unix-like systems (compared to Windows' 49.2%) in March 2001.

Note that these figures would probably be quite different if they were based on web addresses instead of physical computers; in such a case, the clear majority of web sites are hosted by Unix-like systems. As stated by Netcraft, "Although Apache running on various Unix systems runs more sites than Windows, Apache is heavily deployed at hosting companies and ISPs who strive to run as many sites as possible on one computer to save costs."

- GNU/Linux is the #1 server OS on the public Internet (counting by domain name), according to a 1999 survey of primarily European and educational sites.** The first study that I've found that examined GNU/Linux's market penetration is a survey by [Zoebelein in April 1999](#). This survey found that, of the total number of servers deployed on the Internet in 1999

(running at least ftp, news, or http (WWW)) in a database of names they used, the #1 OS was GNU/Linux (at 28.5%), with others trailing. It's important to note that this survey, which is the first one that I've found to try to answer questions of market share, used existing databases of servers from the .edu (educational domain) and the RIPE database (which covers Europe, the Middle East, parts of Asia, and parts of Africa), so this isn't really a survey of "the whole Internet" (e.g., it omits ".com" and ".net"). This is a count by domain *name* (e.g., the text name you would type into a web browser for a location) instead of by physical computer, so what it's counting is different than the Netcraft June 2001 OS study. Also, this study counted servers providing ftp and news services (not just web servers).

Here's how the various OSes fared in the study:

Operating System	Market Share	Composition
GNU/Linux	28.5%	GNU/Linux
Windows	24.4%	All Windows combined (including 95, 98, NT)
Sun	17.7%	Sun Solaris or SunOS
BSD	15.0%	BSD Family (FreeBSD, NetBSD, OpenBSD, BSDI, ...)
IRIX	5.3%	SGI IRIX

A part of the BSD family is also OSS/FS, so the OSS/FS OS total is even higher; if over 2/3 of the BSDs are OSS/FS, then the total share of OSS/FS would be about 40%. Advocates of Unix-like systems will notice that the majority (around 66%) were running Unix-like systems, while only around 24% ran a Microsoft Windows variant.

4. **GNU/Linux was the #2 server OS sold in 1999, 2000, and 2001.** According to [a June 2000 IDC survey](#) of 1999 licenses, 24% of all servers (counting both Internet and intranet servers) installed in 1999 ran GNU/Linux. Windows NT came in first with 36%; all Unices combined totaled 15%. Again, since some of the Unices are OSS/FS systems (e.g., FreeBSD, OpenBSD, and NetBSD), the number of OSS/FS systems is actually larger than the GNU/Linux figures. Note that it all depends on what you want to count; 39% of all servers installed from this survey were Unix-like (that's 24%+15%), so "Unix-like" servers were actually #1 in installed market share once you count GNU/Linux and Unix together.

IDC released a similar study on January 17, 2001 titled "[Server Operating Environments: 2000 Year in Review](#)". On the server, Windows accounted for 41% of new server OS sales in 2000, growing by 20% - but GNU/Linux accounted for 27% and grew even faster, by 24%. Other major Unices had 13%.

[IDC's 2002 report found that Linux held its own in 2001 at 25%](#). All of this is especially

intriguing since GNU/Linux had 0.5% of the market in 1995, [according to a Forbes quote of IDC](#). Data such as these (and the TCO data shown later) have inspired statements such as this one from IT-Director on November 12, 2001: [“Linux on the desktop is still too early to call, but on the server it now looks to be unstoppable.”](#)

These measures do *not* measure all server systems installed that year; some Windows systems are not paid for (they're illegally pirated), and OSS/FS OSes such as GNU/Linux and the BSDs are often downloaded and installed on multiple systems (since it's legal and free to do so).

Note that [a study published October 28, 2002 by the IT analyst company Butler Group](#) concluded that on or before 2009, Linux and Microsoft's .Net will have fully penetrated the server OS market from file and print servers through to the mainframe.

5. **GNU/Linux and Windows systems (when Windows CE and Windows XP are combined) are the leaders and essentially even in terms of developer use for future embedded projects, according to Evans Data Corporation (EDC).** [Evans Data Corp \(EDC\)'s Embedded Systems Developer Survey, fielded in July 2002](#), asked developers “For each of the following operating systems, please indicate whether you are targeting the OS on your current project or your next project.” They collected data from 444 developers. Their results: 30.2% of embedded developers use or expect to use Linux, while 16.2% say they will use Windows CE and another 14.4% say they will use Windows XP Embedded. If the two Windows systems are combined into a single value, this gives Windows Embedded operating systems a statistically insignificant edge over Embedded Linux (at 30.6% vs. 30.2%). However, Embedded Linux has nearly double the growth rate, and combining two different Windows systems into a single value is somewhat misleading. Wind River's VxWorks embedded OS, the incumbent embedded software market leader, “trails slightly behind Embedded Linux for current project use, and VxWorks' modest gain of just 2.9% for expected use in future projects drops it to a distant third place position, ending up with less than half the usage rate of the two neck-and-neck future project usage leaders (Windows Embedded and Embedded Linux).”
6. **An Evans Data survey published in November 2001 found that 48.1% of international developers and 39.6% of North Americans plan to target most of their applications to GNU/Linux. In October 2002, they found that 59% of developers expect to write Linux applications in the next year.** The [November 2001 edition of the Evans Data International Developer Survey Series](#) reported on in-depth interviews with over 400 developers representing over 70 countries, and found that when asked which OS they plan to target with most of their applications next year, 48.1% of international developers and 39.6% of North Americans stated that they plan to target most of their applications to GNU/Linux. This is surprising since only a year earlier less than a third of the international development community was writing GNU/Linux applications. The survey also found that 37.8% of the international development community and 33.7% of North American developers have already written applications for GNU/Linux, and that over half of those surveyed have enough confidence in GNU/Linux to use it for

mission-critical applications.

[Evans Data conducted a survey in October 2002](#). In this survey, they reported “Linux continues to expand its user base. 59% of survey respondents expect to write Linux applications in the next year.”

Later data seems to confirm this, for example, the Japanese [Linux white paper 2003](#) found that 49.3% of IT solution vendors support Linux in Japan.

7. **A Japanese survey found widespread use and support for GNU/Linux; overall use of GNU/Linux jumped from 35.5% in 2001 to 64.3% in 2002 of Japanese corporations, and GNU/Linux was the most popular platform for small projects.** The book [Linux White Paper 2003](#) (published by Impress Corporation) surveys the use of GNU/Linux in Japan (it is an update to an earlier book, “Linux White Paper 2001-2002”). This is written in Japanese; here is a brief summary of its contents.

The survey has two parts, user and vendor. In “Part I : User enterprise”, they surveyed 729 enterprises that use servers. In “Part II : Vendor enterprise”, they surveyed 276 vendor enterprises who supply server computers, including system integrators, software developers, IT service suppliers, and hardware resellers. The most interesting results are those that discuss the use of Linux servers in user enterprises, the support of Linux servers by vendors, and Linux server adoption in system integration projects.

First, the use of Linux servers in user enterprises:

System	2002	2001
Linux server	64.3%	35.5%
Windows 2000 Server	59.9%	37.0%
Windows NT Server	64.3%	74.2%
Commercial Unix server	37.7%	31.2%

And specifically, here’s the average use in 2002:

System	Ave. units	# samples
Linux server	13.4	N=429 (5.3 in 2001)
Windows 2000 Server	24.6	N=380
Windows NT Server	4.5	N=413
Commercial Unix server	6.9	N=233

Linux servers are the fastest growing category from last year. The average units of server per enterprise increased by 2.5-fold from 5.3 units to 13.4 units.

Second, note the support of GNU/Linux servers by vendors:

System	Year 2002 Support
Windows NT/2000 Server	66.7%
Linux server	49.3%
Commercial Unix server	38.0%

This is the rate of vendors that develop or sale products supporting Linux server; note that Linux is already a major OS when compared with its competitors. The reasons for supporting Linux server were also surveyed, which turn out to be different than the reasons in some other countries (for a contrast, see the [European FLOSS report](#)):

Increase of importance in the future	44.1%
Requirement from their customers	41.2%
Major OS in their market	38.2%
Free of licence fee	37.5%
Most reasonable OS for their purpose	36.0%
Open source	34.6%
High reliability	27.2%

Third, note the rate of Linux server adoption in system integration projects:

Project Size (Million Yen)	Linux		Win2000	Unix
	2002	2001	2002	2002
0-3	62.7%	65.7%	53.8%	15.4%
3-10	51.5%	53.7%	56.3%	37.1%
10-50	38.3%	48.9%	55.8%	55.8%
50-100	39.0%	20.0%	45.8%	74.6%
100+	24.4%	9.1%	51.1%	80.0%

Where 1 Million Yen = \$8,000 US. GNU/Linux servers are No.1 (62.5%) in small projects less than 3,000,000 Yen (\$24,000 US), and GNU/Linux has grown in larger projects more than 50,000,000 Yen (\$400,000 US) from 20.0% to 39.0%. In projects over 100,000,000 Yen (\$800,000 US), Linux is adopted by 24.4% of the projects (mainly as a substitute for proprietary Unix systems). Note that many projects (especially large ones) use multiple platforms

simultaneously, so the values need not total 100%.

8. **The European FLOSS study found significant use of OSS/FS.** The large report [Free/Libre and Open Source Software \(FLOSS\): Survey and Study](#), published in June 2002, examined many issues including the use of OSS/FS. This study found significant variance in the use of OSS/FS; 43.7% of German establishments reported using OSS/FS, 31.5% of British establishments reported using OSS/FS, while only 17.7% of Swedish establishments reported using OSS/FS. In addition, they found that OSS usage rates of larger establishments were larger than smaller establishments, and that OSS usage rates in the public sector were above average.

9. **Microsoft sponsored its own research to “prove” that GNU/Linux is not as widely used, but this research has been shown to be seriously flawed.** Microsoft sponsored a [Gartner Dataquest report](#) claiming only 8.6% of servers shipped in the U.S. during the third quarter of 2000 were Linux-based. However, it’s worth noting that Microsoft (as the research sponsor) has every incentive to create low numbers, and these numbers are quite different from IDC’s research in the same subject. IDC’s Kusnetzky commented that the likely explanation is that Gartner used a very narrow definition of “shipped”; he thought the number was “quite reasonable” if it only surveyed new servers with Linux, “But our research is that this is not how most users get their Linux. We found that just 10 to 15 percent of Linux adoption comes from pre-installed machines... for every paid copy of Linux, there is a free copy that can be replicated 15 times.” Note that it’s quite difficult to buy a new x86 computer without a Microsoft OS (Microsoft’s contracts with computer makers ensure this), but that doesn’t mean that these OSes are used. Gartner claimed that it used interviews to counter this problem, but its final research results (when compared to known facts) suggest that Gartner did not really counter this effect. For example, Gartner states that Linux shipments in the supercomputer field were zero. In fact, Linux is widely used on commodity parallel clusters at many scientific sites, including many high-profile sites. Many of these systems were assembled in-house, showing that Gartner’s method of defining a “shipment” does not appear to correlate to working installations. The Register’s article, [“No one’s using Linux”](#) (with its companion article [“90% Windows..”](#)) discusses this further. In short, Microsoft-sponsored research reported low numbers, but these numbers are quite suspect.

10. **Businesses plan to increase their use of GNU/Linux.** A Zona Research study found that over half of the large enterprise respondents expected increases of up to 25% in the number of GNU/Linux users in their firm, while nearly 20% expected increases of over 50%. In small companies, over one third felt that GNU/Linux usage would expand by 50%. The most important factors identified that drove these decisions were reliability, lower price, speed of applications, and scalability. Here are the numbers:

Expected GNU/Linux Use	Small Business	Midsize Business	Large Business	Total
50% increase	21.0%	16%	19.0%	19%
10-25% increase	30.5%	42%	56.5%	44%

No growth	45.5%	42%	24.5%	36%
Reduction	3.0%	0%	0%	1%

You can see more about this study in [“The New Religion: Linux and Open Source”](#) (ZDNet) and in InfoWorld’s February 5, 2001 article “Linux lights up enterprise: But concerns loom about OS vendor profitability.”

11. **The global top 1000 Internet Service Providers expect GNU/Linux use to increase by 154%, according to Idaya’s survey conducted January through March 2001.** A [survey](#) conducted by [Idaya](#) of the global top 1000 ISPs found that they expected GNU/Linux to grow a further 154% in 2001. Also, almost two thirds (64%) of ISPs consider the leading open source software meets the standard required for enterprise level applications, comparable with proprietary software. Idaya produces OSS/FS software, so keep that in mind as a potential bias.
12. **A 2002 European survey found that 49% of CIOs in financial services, retail, and the public sector expect to be using OSS/FS.** OpenForum Europe published in February 2002 a survey titled [Market Opportunity Analysis For Open Source Software](#). Over three months CIOs and financial directors in financial services, retail and public sector were interviewed for this survey. In this survey, 37% of the CIOs stated that they were already using OSS/FS, and 49% expected to be using OSS/FS in the future. It is quite likely that even more companies are using OSS/FS but their CIOs are not aware of it. Perceived benefits cited included decreased costs in general (54%), lower software license cost (24%), better control over development (22%), and improved security (22%).
13. **IBM found a 30% growth in the number of enterprise-level applications for GNU/Linux in the six month period ending June 2001.** At one time, it was common to claim that “Not enough applications run under GNU/Linux” for enterprise-level use. However, [IBM found there are over 2,300 GNU/Linux applications \(an increase in 30% over 6 months\)](#) available from IBM and the industry’s top independent software vendors (ISVs). A [Special report by Network Computing on Linux for the Enterprise](#) discusses some of the strengths and weaknesses of GNU/Linux, and found many positive things to say about GNU/Linux for enterprise-class applications.
14. **Morgan Stanley found significant and growing use of GNU/Linux.** [Morgan Stanley surveyed 225 CIOs on August 2002](#). Among the respondents, 29% said they owned GNU/Linux servers, 8% did not but are formally considering buying them, and 17% of the CIOs said they neither owned nor were formally considering GNU/Linux servers but that they were informally considering them. The remainder (slightly less than half, or 46%) noted they didn’t own and weren’t considering GNU/Linux. For those that have recently purchased new GNU/Linux servers, 31% were adding capacity, 31% were replacing Windows systems, 24% were replacing Unix and 14% were replacing other OSes. It’s easier to transition to GNU/Linux from Unix than from Windows, so it’s intriguing that Windows was being replaced more often than Unix. [CNet](#)

[news commented on this study with additional commentary about open source vs. Microsoft.](#)

15. **Revenue from sales of GNU/Linux-based server systems increased 90% in the fourth quarter of 2002 compared to the fourth quarter of 2001.** This 90% increase compared sharply with the 5% increase of server market revenue overall. This data was determined by Gartner Dataquest, and [reported in C|Net](#).

[Sales of GNU/Linux servers increased 63% from 2001 to 2002.](#) This is an increase from \$1.3 billion to \$2 billion, according to Gartner.

16. **A 2001 survey found that 46.6% of IT professionals were confident that their organizations could support GNU/Linux, a figure larger than any OS except Windows.** A [TechRepublic Research survey titled *Benchmarks, Trends, and Forecasts: Linux Report*](#) found that “support for Linux runs surprisingly deep” when it surveyed IT professionals and asked them how confidently their organizations could support various OSes. Given Windows’ market dominance on the desktop, it’s not surprising that most were confident that their organizations could support various versions of Windows (for Windows NT the figure was 90.6%; for Windows 2000, 81.6%). However, GNU/Linux came in third, at 46.4%; about half of those surveyed responded that their organizations were already confident in their ability to support GNU/Linux! This is especially shocking because GNU/Linux beat other well-known products with longer histories including Unix (42.1%), Novell Netware (39.5%), Sun Solaris (25.7%), and Apple (13.6%). TechRepublic suggested that there are several possible reasons for this surprisingly large result:
- GNU/Linux is considered to be a rising technology; many IT professionals are already studying it and learning how to use it, assuming that it will be a marketable skill in the near future.
 - Many IT professionals already use GNU/Linux at home, giving GNU/Linux an entree into professional organizations.
 - Since GNU/Linux is similar to Unix, IT professionals who are proficient in Unix can easily pick up GNU/Linux.

TechRepublic suggests that IT executives should inventory their staff’s skill sets, because they may discover that their organization can already support GNU/Linux if they aren’t currently using it.

17. **Sendmail, an OSS/FS program, is the leading email server.** A [survey between 2001-09-27 and 2001-10-03 by D.J. Bernstein of one million random IP addresses](#) successfully connected to 958 SMTP (email) servers (such servers are also called mail transport agents, or MTAs). Bernstein found that Unix Sendmail had the largest market share (42% of all email servers), followed by Windows Microsoft Exchange (18%), Unix qmail (17%), Windows Ipswitch IMail (6%), Unix smap (2%), UNIX Postfix (formerly VMailer, 2%) and Unix Exim (1%). Note that Bernstein implements one of Sendmail’s competitors (qmail), so he has a disincentive to identify Sendmail’s large market share. Qmail is not OSS/FS, because [derivatives of Qmail cannot be freely redistributed](#); Qmail is “source viewable,” so some people are confused into believing that

Qmail is OSS/FS. However, Sendmail, Postfix, and Exim *are* all OSS/FS. Indeed, not only is the leading program (Sendmail) OSS/FS, but that OSS/FS program has more than twice the installations of its nearest competition.

18. **A survey in the second quarter of 2000 found that 95% of all reverse-lookup domain name servers (DNS) used bind, an OSS/FS product.** The Internet is built from many mostly-invisible infrastructure components. This includes domain name servers (DNSs), which take human-readable machine names (like “yahoo.com”) and translate them into numeric addresses. Publicly accessible machines also generally support “reverse lookups”, which convert the numbers back to names; for historical reasons, this is implemented using the hidden “in-addr.arpa” domain. By surveying the in-addr domain, you can gain insight into how the whole Internet is supported. [Bill Manning has surveyed the in-addr domain](#) and found that 95% of all name servers (in 2q2000) performing this important Internet infrastructure task are some version of “bind.” This includes all of the [DNS root servers](#), which are critical for keeping the Internet functioning. Bind is an OSS/FS program.
19. **PHP is the web’s #1 Server-side Scripting Language.** PHP, a recursive acronym for “Hypertext Preprocessor”, is an open source server-side scripting language designed for creating dynamic Web pages (e.g., such as e-commerce). [As noted in a June 3, 2002 article](#), PHP recently surpassed Microsoft’s ASP to become the most popular server-side Web scripting technology on the Internet, and was used by over 24% of the sites on the Internet. Of the 37.6 million web sites surveyed worldwide, PHP is running on over 9 million sites, and over the past two years PHP has averaged a 6.5% monthly growth rate.
20. **OpenSSH is the Internet’s #1 implementation of the SSH security protocol.** The Secure Shell (SSH) protocol is widely used to securely connect to computers and control them remotely (using either a text or X-Windows graphical interface). On April 2002, a survey of 2.4 million Internet addresses found that OpenSSH, an OSS/FS implementation of SSH, was the #1 implementation, with 66.8% of the market; the proprietary “SSH” had 28.1%, Cisco had 0.4%, and others totaled 4.7%. You can see [general information about the survey](#), or the [specific SSH statistics for April 2002](#). It’s also interesting to note that OpenSSH had less than 5% of the market in the third quarter of 2000, but its use steadily grew. By the fourth quarter of 2001, over half of all users of the SSH protocol were using OpenSSH, and its market share has continued to grow since.
21. **CMP TSG/Insight found that 41% of application development tools were OSS/FS, and VARBusiness found 20% of all companies using GNU/Linux.** [VARBusiness reported in September 2003 on "The Rise of Linux"](#). In the article, it reports a finding of CMP TSG/Insight: 41% of application development tools in use were OSS/FS, second only to Microsoft (76%) and leading Oracle (35%), IBM (26%), Sun (21%), and Borland (18%). They also reported their own finding that 20% of all companies they surveyed were GNU/Linux, presumably less than that of Microsoft, but twice that of Netware and Unix. Indeed, they note that GNU/Linux has transformed "from a curiosity to a core competency."
22. **A set of 2003 Gartner studies notes that the TCO of Linux (or OSS/FS) on the Desktop**

depends on your situation. [Gartner reported that that enterprises that installed Linux on client desktops would save \\$80 in hardware acquisition costs and an average of \\$74 per user per year on office automation software](#) (assuming that StarOffice will be purchased instead of Microsoft Office). However, they also note that "lost productivity stemming from learning curves and compatibility can eat up direct-cost savings when moving to Linux on the desktop." A key issue is that many organizations have built or bought specialized applications that only run on Windows. Note that these studies primarily examine Linux vs. Windows on the client desktop, not other OSS/FS deployment options (such as moving to web-based applications using OSS/FS tools that work with any client operating system, or using OSS/FS applications on Windows). Gartner concludes that both Windows and GNU/Linux can have a lower TCO, depending on your circumstance, and that "before migrating your desktop computers to Linux, take inventory of your business applications and compare Linux to Windows in terms of total cost of ownership."

Perhaps the simplest argument that GNU/Linux will have a significant market share is that [Sun is modifying its Solaris product to run GNU/Linux applications, and IBM has already announced that GNU/Linux will be the successor of IBM's own AIX.](#)

3. Reliability

There are a lot of anecdotal stories that OSS/FS is more reliable, but finally there is quantitative data confirming that mature OSS/FS programs are more reliable:

1. **Equivalent OSS/FS applications are more reliable, according to the Fuzz study.** The paper ["Fuzz Revisited"](#) paper measured reliability by feeding programs random characters and determining which ones resisted crashing and freeze-ups. This approach is unlikely to find subtle failures, yet the study authors found that their approach still manages to find many errors in production software and is a useful tool for finding software flaws. What's more, this approach is extremely fair and can broadly applied to any program, making it possible to compare different programs fairly.

OSS/FS had higher reliability by this measure. It states in section 2.3.1 that:

It is also interesting to compare results of testing the commercial systems to the results from testing "freeware" GNU and Linux. The seven commercial systems in the 1995 study have an average failure rate of 23%, while Linux has a failure rate of 9% and the GNU utilities have a failure rate of only 6%. It is reasonable to ask why a globally scattered group of programmers, with no formal testing support or software engineering standards can produce code that is more reliable (at least, by our measure) than commercially produced code. Even if you consider only the utilities that were available from GNU or Linux, the failure rates for these two

systems are better than the other systems.

There is evidence that Windows applications have similar reliability to the proprietary Unix software (e.g., less reliable than the OSS/FS software). A later paper, [“An Empirical Study of the Robustness of Windows NT Applications Using Random Testing”](#), found that with Windows NT GUI applications, they could crash 21% of the applications they tested, hang an additional 24% of the applications, and could crash or hang *all* the tested applications when subjecting them to random Win32 messages. Thus, there’s no evidence that proprietary Windows software is more reliable than OSS/FS by this measure. Yes, Windows has progressed since that time - but so have the OSS/FS programs.

Although this experiment was done in 1995, nothing that’s happened since suggests that proprietary software has become much better than OSS/FS programs since then. Indeed, since 1995 there’s been an increased interest and participation in OSS/FS, resulting in far more “eyeballs” examining and improving the reliability of OSS/FS programs.

The fuzz paper’s authors found that proprietary software vendors generally didn’t fix the problems identified in an earlier version of their paper, and found that concerning. In contrast, [Scott Maxwell led an effort to remove every flaw identified in the OSS/FS software](#) in the 1995 fuzz paper, and eventually fixed every flaw. Thus, the OSS/FS community’s response shows why, at least in part, OSS/FS programs have such an edge in reliability; if problems are found, they’re often fixed. Even more intriguingly, the person who spearheaded ensuring that these problems were fixed wasn’t an original developer of the programs - a situation only possible with OSS/FS.

Now be careful: OSS/FS is not magic pixie dust; beta software of any kind is still buggy! However, the 1995 experiment measured mature OSS/FS to mature proprietary software, and the OSS/FS software was more reliable under this measure.

- 2. GNU/Linux is more reliable than Windows NT, according to a 10-month ZDnet experiment.** [ZDnet ran a 10-month test for reliability](#) to compare Caldera Systems OpenLinux, Red Hat Linux, and Microsoft’s Windows NT Server 4.0 with Service Pack 3. All three used identical (single-CPU) hardware, and network requests were sent to each server in parallel for standard Internet, file, and print services. The result: NT crashed an average of once every six weeks, each taking about 30 minutes to fix; that’s not bad, but neither GNU/Linux server *ever* went down. This ZDnet article also does a good job of identifying GNU/Linux weaknesses (e.g., desktop applications and massive SMP). Hopefully Windows has made improvements since this study - but the OSS/FS have certainly made improvements as well.
- 3. GNU/Linux is more reliable than Windows NT, according to a one-year Bloor Research experiment.** [Bloor Research](#) had both OSes running on relatively old Pentium machines. During the one year test, GNU/Linux crashed once due to a hardware fault (disk problems), which took 4

hours to fix, giving it a measured availability of 99.95 percent. Windows NT crashed 68 times, caused by hardware problems (disk), memory (26 times), file management (8 times), and various odd problems (33 times). All this took 65 hours to fix, giving an availability of 99.26 percent. It's intriguing that the only GNU/Linux problem and many of the Windows problems were hardware-related; it could be argued that the Windows hardware was worse, or it could be argued that GNU/Linux did a better job of avoiding and containing hardware failures. The file management failure is due to Windows, and the odd problems appear due to Windows too, indicating that GNU/Linux is far more reliable than Windows. GNet summarized this as saying "the winner here is clearly Linux."

4. **A study by Reasoning found that the Linux kernel's implementation of the TCP/IP Internet protocol stack had fewer defects than the equivalent stacks of several proprietary general-purpose operating systems, and equalled the best of the embedded operating systems.** As noted in [their press release](#) and [C|Net](#), Reasoning's study compared six implementations of TCP/IP, the fundamental protocols underlying the Internet. Besides the Linux kernel, three of the implementations were part of commercial general-purpose operating systems, and two were embedded in commercial telecommunications equipment. The Linux kernel primarily used as the kernel of a general-purpose operating system; it would be reasonable to expect that the embedded operating systems would have better reliability because of the need for reliability in that market. The study was not commissioned by any of the GNU/Linux vendors or companies who might be competing with GNU/Linux, and thus should be free of bias.

The company used automated tools to look five kinds of defects in code: Memory leaks, null pointer dereferences, bad deallocations, out of bounds array access and uninitialized variables. Reasoning found 8 defects in 81,852 lines of Linux kernel source lines of code (SLOC), resulting in a defect density rate of 0.1 defects per KSLOC. In contrast, the three proprietary general-purpose operating systems (two of them versions of Unix) had between 0.6 and 0.7 defects/KSLOC; thus the Linux kernel had a smaller defect rate than all the competing general-purpose operating systems examined. The rates of the two embedded operating systems were 0.1 and 0.3 defects/KSLOC, thus, the Linux kernel had an defect rate better than one embedded operating system, and equivalent to another.

One issue is that the tool detects issues that may not be true problems. For example, of those 8 defects, one was clearly a bug and had been separately detected and fixed by the developers, and 4 defects clearly had no effect on the running code. None of the defects found were security flaws. To counter this, they also tracked which problems were repaired by the developers of the various products. The Linux kernel did quite well by this measure as well: the Linux kernel had 1 repaired defect out of 81.9 KSLOC, while the proprietary implementations had 235 repaired defects out of 568 KSLOC. This means the Linux kernel had a repair defect rate of 0.013 defects/KSLOC, while the proprietary implementations had a repair defect rate of 0.41 defects/KSLOC.

CEO Scott Trappe explained this result by noting that the open source model encourages several

behaviors that are uncommon in the development of commercial code. First, many users don't just report bugs, as they would do with [proprietary] software, but actually track them down to their root causes and fix them. Second, many developers are reviewing each other's code, if only because it is important to understand code before it can be changed or extended. It has long been known that peer review is the most effective way to find defects. Third, the open source model seems to encourage a meritocracy, in which programmers organize themselves around a project based on their contributions. The most effective programmers write the most crucial code, review the contributions of others, and decide which of these contributions make it into the next release. Fourth, open source projects don't face the same type of resource and time pressures that [proprietary] projects do. Open source projects are rarely developed against a fixed timeline, affording more opportunity for peer review and extensive beta testing before release.

This certainly doesn't prove that OSS/FS will always be the highest quality, but it clearly shows that OSS/FS can be of high quality.

5. **A similar study by Reasoning found that the MySQL database (a leading OSS/FS database) fewer defects than a set of 200 proprietary programs used for comparison.** In a similar manner to the previous study, on December 15, 2003, [Reasoning announced its analysis results comparing MySQL with various proprietary programs](#). MySQL had found 21 software defects in 236,000 source lines of code (SLOC), producing a defect density of 0.09 defects/KSLOC. Using a set of 200 recent proprietary projects (totalling 35 million SLOC), the same tools found a defect rate of 0.57 defects/KSLOC -- over six times the error rate. Again, not all defects are found by their tool, and this certainly doesn't prove that OSS/FS will always be the highest quality, but it clearly shows that OSS/FS can be of high quality.

6. **Sites using Microsoft's IIS web serving software have over double the time offline (on average) than sites using the Apache software, according to a 3-month Swiss evaluation.** These are the results of [Syscontrol AG's analysis of website uptime \(announced February 7, 2000\)](#) They measured over 100 popular Swiss web sites over a three-month period, checking from 4 different locations every 5 minutes (it'd be interesting to see what a larger sample would find!). You can [see their report \(in German\)](#), or a [Babelfish \(machine\) translation of the report](#). Here's their set of published data on "average down-time (in hours in that month) for each type of server", plus a 3-month average that I've computed:

Downtime	Apache	Microsoft	Netscape	Other
September	5.21	10.41	3.85	8.72
October	2.66	8.39	2.80	12.05
November	1.83	14.28	3.39	6.85
Average	3.23	11.03	3.35	9.21

It's hard not to notice that Apache (the OSS web server) had the best results over the three-month average (and with better results over time, too). Indeed, Apache's worst month was better than Microsoft's best month. The difference between Netscape and Apache is statistically insignificant - but this still shows that the freely-available OSS/FS solution (Apache) has a reliability at least as good as the most reliable proprietary solution.

The report does state that this might not be solely the fault of the software's quality, and in particular it noted that several Microsoft IIS sites had short interruptions at the same time each day (suggesting regular restarts). However, this still begs the question - why did the IIS sites require so many regular restarts compared to the Apache sites? Every outage, even if preplanned, results in a service loss (and for e-commerce sites, a potential loss of sales). Presumably, IIS site owners who perform periodic restarts do so because they believe that doing so will improve their IIS systems' overall reliability. Thus, even with pre-emptive efforts to keep the IIS systems reliable, the IIS systems are less reliable than the Apache-based systems which simply do not appear to require constant restarting.

- 7. According to a separate uptime study by Netcraft, OSS/FS does very well; as of August 3, 2001, of the 50 sites with the highest uptimes, 92% use Apache and 50% run on OSS/FS OSes.** Netcraft keeps a track of the 50 often-requested sites with the longest uptimes at <http://uptime.netcraft.com>. Looking at [the August 3, 2001 uptime report](#), I found that 92% (46/50) of the sites use Apache; one site's web server was unknown, and three others were not Apache. Of those three, only one reported to be Microsoft IIS, and that one instance is suspicious because its reported OS is BSD/OS (this apparent inconsistency can be explained in many ways, e.g., perhaps there is a front-end BSD/OS system that "masks" the IIS web site, or perhaps the web server is lying about its type to confuse attackers). In this snapshot, 50% (25/50) ran on an OSS/FS OS, and only Unix-like OSes had these large uptimes (no Windows systems were reported as having the best uptimes).

As with all surveys, this one has weaknesses, as discussed in [Netcraft's Uptime FAQ](#). Their techniques for identifying web server and OSes can be fooled. Only systems for which Netcraft was sent many requests were included in the survey (so it's not "every site in the world"). Any site that is requested through the "what's that site running" query form at Netcraft.com is added to the set of sites that are routinely sampled; Netcraft doesn't routinely monitor all 22 million sites it knows of for performance reasons. Many OSes don't provide uptime information and thus can't be included; this includes AIX, AS/400, Compaq Tru64, DG/UX, MacOS, NetWare, NT3/Windows 95, NT4/Windows 98, OS/2, OS/390, SCO UNIX, Sony NEWS-OS, SunOS 4, and VM. Thus, this uptime counter can only include systems running on BSD/OS, FreeBSD (but not the default configuration in versions 3 and later), recent versions of HP-UX, IRIX, GNU/Linux 2.1 kernel and later (except on Alpha processor based systems), MacOS X, recent versions of NetBSD/OpenBSD, Solaris 2.6 and later, and Windows 2000. Note that Windows NT systems cannot be included in this survey (because their uptimes couldn't be counted). Windows 2000 systems's data are included in the source source for this survey, but they have a different

problem. Windows 2000 had little hope to be included in the August 2001 list, because the 50th system in the list had an uptime of 661 days, and Windows 2000 had only been launched about 17 months (about 510 days) earlier. Note that HP-UX, GNU/Linux (usually), Solaris and recent releases of FreeBSD cycle back to zero after 497 days, exactly as if the machine had been rebooted at that precise point. Thus it is not possible to see an HP-UX, GNU/Linux (usually), or Solaris system with an uptime measurement above 497 days, and in fact their uptimes can be misleading (they may be up for a long time, yet not show it). There is yet one other weakness: if a computer switches operating systems later, the long uptime is credited to the new OS. Still, this survey does compare Windows 2000, GNU/Linux (up to 497 days usually), FreeBSD, and several other OSes, and OSS/FS does quite well.

It could be argued that perhaps systems on the Internet that haven't been rebooted for such a long time might be insignificant, half-forgotten, systems. For example, it's possible that security patches aren't being regularly applied, so such long uptimes are not necessarily good things. However, a counter-argument is that Unix and Linux systems don't need to be rebooted as often for a security update, and this is a valuable attribute for a system to have. Even if you accepted that unproven claim, it's certainly true that there are half-forgotten Windows systems, too, and they didn't do so well. Also, only systems someone specifically asked for information about were included in the uptime survey, which would limit the number of insignificant or half-forgotten systems.

At the very least, Unix and Linux are able to quantitatively demonstrate longer uptimes than their Windows competitors can, so Unix and Linux have significantly better evidence of their reliability than Windows.

Damien Challet and Yann Le Du of the University of Oxford have written a paper titled [*Closed source versus open source in a model of software bug dynamics*](#). In this paper they develop a model of software bug dynamics where users, programmers and maintainers interact through a given program. They then analyzed the model, and found that all other things being equal (such as number of users, programmers, and quality of programmers), “debugging in open source projects is always faster than in closed source projects.”

Of course, there are many anecdotes about Windows reliability vs. Unix. For example, the [*Navy's “Smart Ship” program caused a complete failure of the USS Yorktown ship in September 1997*](#). Whistle-blower Anthony DiGiorgio stated that Windows is “the source of the Yorktown's computer problems.” Ron Redman, deputy technical director of the Fleet Introduction Division of the Aegis Program Executive Office, said “there have been numerous software failures associated with [Windows] NT aboard the Yorktown.” Redman also said “Because of politics, some things are being forced on us that without political pressure we might not do, like Windows NT... If it were up to me I probably would not have used Windows NT in this particular application. If we used Unix, we would have a system that has less of a tendency to go down.”

One problem with reliability measures is that it takes a long time to gather data on reliability in real-life circumstances. Thus, there's more data comparing older Windows editions to older GNU/Linux editions. The key is that these comparisons are fair, because they compare contemporaneous products. The available evidence suggests that OSS/FS has a significant edge in reliability, at least in many circumstances.

4. Performance

Comparing GNU/Linux and Microsoft Windows performance on equivalent hardware has a history of contentious claims and different results based on different assumptions. OSS/FS has at least shown that it's often competitive, and in many circumstances it beats the competition.

Performance benchmarks are very sensitive to the assumptions and environment, so the best benchmark is one you set up yourself to model your intended environment. Failing that, you should use unbiased measures, because it's so easy to create biased measures.

First, here are a few recent studies suggesting that some OSS/FS systems beat proprietary competitors in at least some circumstances:

1. In 2002, TPC-C database measures found that a Linux based system was faster than a Windows 2000 based system. More specifically, an HP ProLiant DL580 with 32 Intel Xeon 900MHz CPUs running Oracle 9i R2 Enterprise edition ran faster running on a stock Red Hat Linux Advanced Server than on Microsoft Windows 2000 Advanced Server. You can see the [Linux](#) and [Windows](#) reports; note that [HP did not modify the Linux kernel to get these results](#).
2. **PC Magazine's November 2001 performance tests for file servers found that Linux with Samba significantly outperformed Windows 2000.** PC Magazine's article [Performance Tests: File Server Throughput and Response Times](#) found that Linux with Samba significantly outperformed Windows 2000 Server when used as a file server for Microsoft's own network file protocols. This was true regardless of the number of simultaneous clients (they tested a range up to 30 clients), and it was true on the whole range on computers they used (Pentium II/233MHz with 128MiB RAM, Pentium III/550MHz with 256MiB RAM, and Pentium III/1GHz with 512MiB RAM, where [MiB is 2^20 bytes](#)). Indeed, as the machines became more capable the absolute difference grew more pronounced. On the fastest hardware while handling largest number of clients, GNU/Linux's throughput was about 130 MB/sec vs. Windows' 78 MB/sec (GNU/Linux was 78% faster).
3. **PC Magazine file server performance tests again in April 2002; again Linux with Samba beat Windows 2000, only now Samba surpasses Windows 2000 by about 100% and can handle 4 times as many clients.** PC Magazine published another comparison of Samba and Windows; a summary is available electronically as ["Samba runs rings around Win2000."](#) They noted that the latest Samba software now surpasses the performance of Windows 2000 by about

100 percent under benchmark tests, and found that Linux and Samba can handle four times as many client systems as Windows 2000 before performance begins to drop off. Jay White, IT manager at electronics firm BF Group, said that Samba is one of the most useful pieces of server software available for a mixed Windows and Linux environment. “Our Samba server has been online for 394 days so far. The total cost is the hardware plus 30 minutes of my time each year,” he said. Mark Twells, IT coordinator at a large education facility, said, “We run six Samba servers on a variety of hardware [and] we have around 1,000 users.

4. **In performance tests by Sys Admin magazine, GNU/Linux beat Solaris (on Intel), Windows 2000, and FreeBSD.** The article [“Which OS is Fastest for High-Performance Network Applications?”](#) in the July 2001 edition of *Sys Admin* magazine examined high-performance architectures and found that GNU/Linux beat its competition when compared with Solaris (on Intel), FreeBSD (an OSS/FS system), and Windows 2000. They intentionally ran the systems “out of the box” (untuned), except for increasing the number of simultaneous TCP/IP connections (which is necessary for testing multi-threaded and asynchronous applications). They used the latest versions of OSes and the exact same machine. They reported (by OS) the results of two different performance tests.

The FreeBSD developers complained about these tests, noting that FreeBSD by default emphasizes reliability (not speed) and that they expected anyone with a significant performance need would do some tuning first. Thus, [Sys Admin’s re-did the tests for FreeBSD after tuning FreeBSD](#). One change they made was switching to “asynchronous” mounting, which makes a system faster (though it increases the risk of data loss in a power failure) - this is the GNU/Linux default and easy to change in FreeBSD, so this was a very small and reasonable modification. However, they also made many other changes, for example, they found and compiled in 17 FreeBSD kernel patches and used various tuning commands. The other OSes weren’t given the chance to “tune” like this, so comparing untuned OSes to a tuned FreeBSD isn’t really fair.

In any case, here are their two performance tests:

1. Their “real-world” test measured how quickly large quantities of email could be sent using their email delivery server (MailEngine). Up to 100 simultaneous sends there was no difference, but as the number increased the systems began showing significant differences in their hourly email delivery speed. By 500 simultaneous sends GNU/Linux was clearly faster than all except FreeBSD-tuned, and GNU/Linux remained at the top. FreeBSD-tuned had similar performance to GNU/Linux when running 1000 or less simultaneous sends, but FreeBSD-tuned peaked around 1000-1500 simultaneous connections with a steady decline not suffered by GNU/Linux, and FreeBSD-tuned had trouble going beyond 3000 simultaneous connections. By 1500 simultaneous sends, GNU/Linux was sending 1.3 million emails/hour, while Solaris managed approximately 1 million, and Windows 2000 and FreeBSD-untuned were around 0.9 million.
2. Their “disk I/O test” created, wrote, and read back 10,000 identically-sized files in one

directory, varying the size of the file instances. Here Solaris was the slowest, with FreeBSD-untuned the second-slowest. FreeBSD-tuned, Windows 2000, and GNU/Linux had similar speeds at the smaller file sizes (in some cases FreeBSD-tuned was faster, e.g., 8k and 16k file size), but when the file sizes got to 64k to 128k the OSes began to show significant performance differences; GNU/Linux was the fastest, then Windows 2000, then FreeBSD. At 128k, FreeBSD was 16% worse than Windows 2000, and 39% worse than GNU/Linux; all were faster than FreeBSD-untuned and Solaris. When totaling these times across file sizes, the results were GNU/Linux: 542 seconds, Windows 2000: 613 seconds, FreeBSD-tuned: 630 seconds, FreeBSD-untuned: 2398 seconds, and Solaris: 3990 seconds.

5. **GNU/Linux with TUX has produced better SPEC values than Windows/IIS in several cases, even when given inferior drive configurations.** One organization that tries to develop unbiased benchmarks is the [SPEC Consortium](#), which develops and maintains a whole series of benchmarks. We can compare Microsoft Windows versus GNU/Linux by comparing SPECweb99 results (which measure web server performance) on identical hardware if both have undergone the same amount of performance optimization effort. Alas, things are not so simple; rarely are the same basic hardware platforms tested with both OSes, and even when that occurs, as of July 13, 2001 no exactly identical configurations have been tested (they differ in ways such as using a different number of hard drives, or including some faster hard drives). Using all results available by July 13, 2001, there were three hardware configurations, all from Dell, which ran both GNU/Linux (using the TUX web server/accelerator) and Windows (using IIS) on exactly the same underlying hardware. Here are the SPECweb99 results as of July 13, 2001 (larger is better), noting configuration differences:

System	Windows SPEC Result	Linux SPEC Result
Dell PowerEdge 4400/800, 2 800MHz Pentium III Xeon	1060 (IIS 5.0, 1 network controller)	2200 (TUX 1.0, 2 network controllers)
Dell PowerEdge 6400/700, 4 700MHz Pentium III Xeon	1598 (IIS 5.0, 7 9GB 10KRPM drives)	4200 (TUX 1.0, 5 9GB 10KRPM drives)
Dell PowerEdge 8450/700, 8 700MHz Pentium III Xeon	7300/NC (IIS 5.0, 1 9Gb 10KRPM and 8 16Gb 15KRPM drives) then 8001 (IIS 5.0, 7 9Gb 10KRPM and 1 18Gb 15KRPM drive)	7500 (TUX 2.0, 5 9Gb 10KRPM drives)

The first row (the PowerEdge 4400/800) doesn't really prove anything. The IIS system has lower performance, but it only had one network controller and the TUX system has two - so while the TUX system had better performance, that could simply be because it had two network connections it could use.

The second entry (the PowerEdge 6400/700) certainly suggests that GNU/Linux plus TUX really is much better - the IIS system had two more disk drives available to it (which should increase performance), but the TUX system had over twice the IIS system's performance.

The last entry for the PowerEdge 8450/700 is even more complex. First, the drives are different - the IIS systems had at least one drive that revolved more quickly than the TUX systems (which should give IIS higher performance overall, since the transfer speed is almost certainly higher). Also, there were more disk drives (which again should give IIS still higher performance). When I originally put this table together showing all data publicly available in April 2001 (covering the third quarter of 1999 through the first quarter of 2001), IIS 5.0 (on an 8-processor Dell PowerEdge 8450/700) had a SPECweb99 value of 7300. Since that time, Microsoft changed the availability of Microsoft SWC 3.0, and by SPECweb99 rules, this means that those test results are "not compliant" (NC). This is subtle; it's not that the test itself was invalid, it's that Microsoft changed what was available and used the SPEC Consortium's own rules to invalidate a test (possibly because the test results were undesirable to Microsoft). A retest then occurred, with yet another disk drive configuration, at which point IIS produced a value of 8001. However, both of these figures are on clearly better hardware - and in one circumstance the better hardware didn't do better.

Thus, in these configurations the GNU/Linux plus TUX system was given inferior hardware yet still sometimes won on performance. Since other factors may be involved, it's hard to judge - there are pathological situations where "better hardware" can have worse performance, or there may be another factor not reported that had a more significant effect. Hopefully in the future there will be many head-to-head tests in a variety of identical configurations.

Note that TUX is intended to be used as a "web accelerator" for many circumstances, where it rapidly handles simple requests and then passes more complex queries to another server (usually Apache). I've quoted the TUX figures because they're the recent performance figures I have available. As of this time I have no SPECweb99 figures or other recent performance measures for Apache on GNU/Linux, or for Apache and TUX together; I also don't have TUX reliability figures. I expect that such measures will appear in the future.

- 6. Low-level benchmarks by IBM found that GNU/Linux had better performance than Windows for pipes (an input/output mechanism), and also process and thread creation.** Ed Bradford (manager of Microsoft Premier Support for IBM Software group) published in October 2001 the study [*Pipes in Linux, Windows 2000, and Windows XP*](#). In this study he examined the the performance of pipes, a common low-level mechanism for communicating between program processes. He found the pipes in Red Hat 7.1 (with Linux kernel version 2.4.2) had a peak I/O rate of around 700 MB/sec, with a steady state at near 100 MB/sec for very large block sizes. In contrast, Windows 2000 peaked at 500 MB/sec, with a large block steady state of 80 MB/sec. Windows XP Professional (evaluation version) was especially disappointing; its peak I/O rate was only 120 MB/sec, with a stead state of 80 MB/sec, all on the same platform and all running a

GUI.

In February 2002 he published [Managing processes and threads](#), in which he compared the performance of Red Hat Linux 7.2, Windows 2000 Advanced Server ("Win2K"), and Windows XP Professional ("WinXP"), all on a Thinkpad 600X with 320MiB of memory. Linux managed to create over 10,000 threads/second, while Win2K didn't quite manage 5,000 threads/second and WinXP only created 6,000 threads/second. In process creation, Linux managed 330 processes/second, while Win2K managed less than 200 processes/second and WinXP less than 160 processes/second.

- 7. eWeek found in its tests that the OSS/FS program MySQL was quite comparable to the proprietary Oracle database program, and the pair outperformed other proprietary programs.** [eWeek Labs/PC Labs compared several database packages](#) and released the results on February 25, 2002. Comparable performance measures of database programs are actually quite rare. As they note, "database vendors routinely use no-benchmarking clauses in their license agreements to block publication of benchmarks of which they do not approve." Indeed, to their knowledge, this is the first time a computer publication has published database benchmark results tested on the same hardware since PC Magazine did so in October 1993 (almost 9 years earlier). However, they took the risk and published the results examining five server databases: IBM's DB2 7.2 with FixPack 5, Microsoft Corp.'s SQL Server 2000 Enterprise Edition with Service Pack 2, MySQL AB's MySQL 4.0.1 Max, Oracle Corp.'s Oracle9i Enterprise Edition 9.0.1.1.1, and Sybase Inc.'s ASE (Adaptive Server Enterprise) 12.5.0.1. Their goal was to create a level playing field to determine which database performed best when used with a Java-based application server.

The results? They found that overall Oracle9i and MySQL had the best performance and scalability; Oracle9i was slightly ahead of MySQL in most cases, but Oracle costs far more. "ASE, DB2, Oracle9i and MySQL finished in a dead heat up to about 550 Web users. At this point, ASE's performance leveled off at 500 pages per second, about 100 pages per second less than Oracle9i's and MySQL's leveling-off point of about 600 pages per second. DB2's performance dropped substantially, leveling off at 200 pages per second under high loads. Due to its significant JDBC (Java Database Connectivity) driver problems, Microsoft's SQL Server was limited to about 200 pages per second for the entire test."

Naturally, "Manual tuning makes a huge difference with databases - in general, our final measured throughput was twice as fast as our initial out-of-the-box test runs." In this case, they found that "SQL Server and MySQL were the easiest to tune, and Oracle9i was the most difficult because it has so many separate memory caches that can be adjusted."

MySQL also demonstrated some significant innovation. Its performance was due primarily to its "query cache", a capability not included in any other database. If the text of a query has a byte-for-byte match with a cached query, MySQL can retrieve the results directly from its cache

without compiling the query, getting locks or doing index accesses. Obviously, this technique is only effective for tables with few updates, but it certainly made an impact on this benchmark and is a helpful optimization for many situations. MySQL also supports different database engines on a table-by-table basis; no other tested database had this feature.

They also found that of the five databases they tested, only Oracle9i and MySQL were able to run their test application as originally written for 8 hours without problems. They had to work around various problems for all the others.

In this case, an OSS/FS program beat most of its proprietary competition in both performance and reliability (in terms of being able to run a correctly-written application without problems). A proprietary program (Oracle) beat it, but barely, and its competitor is far more expensive. It certainly is arguable that MySQL is (for this application) a comparable application worthy of consideration.

[MySQL AB also reports other benchmark results comparing MySQL with other products](#); however, since they are not an independent lab, I'm not highlighting their results here.

8. **In February 2003, scientists broke the Internet2 Land Speed Record using GNU/Linux.** [Scientists sent 6.7 GB of uncompressed data at 923 megabits per second in just 58 seconds from Sunnyvale, California, to Amsterdam](#) - the equivalent of four hours of DVD-quality movies, using a transfer speed 3,500 times faster than a typical household broadband connection. The team used PCs running Debian GNU/Linux in Amsterdam and Red Hat Linux in Sunnyvale, California.
9. **Benchmarks comparing Sun Solaris x86 and GNU/Linux found many similarities, but GNU/Linux had double the performance in web operations.** Tony Bourke's October 2003 evaluation [Sun Versus Linux: The x86 Smack-down](#) gave a general review comparing Sun Solaris x86 and Red Hat Linux. He found that "Performance was overall similar for most of the metrics tested, perhaps with Linux in a very slight lead. However, with the web operations test (arguably the most important and relevant), Linux is a clear winner." He found that, given the same web serving programs and configuration, GNU/Linux supported over 2000 fetches/second while Solaris x86 supported less than 1000 fetches/second.

All OSes in active development are in a constant battle for performance improvements over their rivals. The history of comparing Windows and GNU/Linux helps put this in perspective:

1. **Ziff-Davis found that GNU/Linux with Apache beat Windows NT 4.0 with IIS by 16%-50% depending on the GNU/Linux distribution.** [Ziff-Davis compared Linux and Windows NT's performance at web serving](#). They found that "Linux with Apache beats NT 4.0 with IIS, hands down. SuSE, the least effective Linux, is 16% faster than IIS, and Caldera, the leader, is 50%

faster.”

2. **[Mindcraft released a report in April 1999](#) that claimed that Microsoft Windows NT Server 4.0 is 2.5 times faster than Linux (kernel 2.2) as a File Server and 3.7 times faster as a Web Server when running on a 4-CPU SMP system.** Several people and organizations, such [Linux Weekly News \(LWN\)](#) and [Dan Kegel](#), identified serious problems with this study. An obvious issue was that NT was specially tuned by Microsoft’s NT experts, at Microsoft, while GNU/Linux was not tuned at all. Another issue is that the price/performance wasn’t considered (nor was total expenditure kept constant - for the same amount of money, the GNU/Linux system could have had better hardware). Mindcraft claimed they asked for help, but they didn’t use the documented methods for getting help nor did they purchase a support contract. Many were especially offended that even though this study was funded by Microsoft (one of the contestants) and held at their facility, neither Mindcraft’s initial announcement nor its paper made any mention of this conflict-of-interest - and it could be easily claimed that their configuration was designed to put GNU/Linux at a disadvantage. Their configuration was somewhat bizarre - it assumed all web pages were static (typical big sites tend to use many dynamically generated pages) and that there were 100 or so clients connected via 100baseT (in 1999 a more typical situation would be that most clients are using slower 28.8 or 56 Kbps modems).

Careful examination of the benchmark did find some legitimate Linux kernel problems, however. These included a TCP bug, the lack of “wake one” semantics, and SMP bottlenecks (see [Dan Kegel’s pages](#) for more information). The Linux kernel developers began working on the weaknesses identified by the benchmark.

3. **PC Week confirmed that Windows did indeed do better in this less probable configuration.** In June 30, 1999, Mindcraft released their [Open Benchmark](#) in conjunction with PC Week. While this didn’t excuse Mindcraft’s biases, it did make a convincing case that there were legitimate problems in the Linux kernel and Apache that made GNU/Linux a poorer-performing product in this somewhat improbable configuration (serving static web pages to clients with high-speed connections). Note that this configuration was considerably different than Ziff-Davis’s, so the benchmarks don’t necessarily conflict; it’s merely that different assumptions can produce different results (as I’ve already stressed).
4. **The German magazine c’t found that web sites with NT was better at static content and dual network connections, but GNU/Linux was better for sites with dynamic content and single connections.** Their article [Mixed Double: Linux and NT as Web Server on the Test Bed](#) examined Windows NT with IIS against GNU/Linux (kernel 2.2.9) with Apache on a machine with four Pentium II Xeon CPUs. They found that the performance winner depended on the situation (by now that should not be a surprise). If the web server primarily served static web pages through two high-performance network cards, NT’s performance was better. However, they also noted that in sophisticated web sites this result didn’t apply, because such sites tend to have primarily dynamic content, and that few sites had this kind of dual-network connection

(when only one network board was available, GNU/Linux generally had an edge). They concluded that “Minecraft’s result can’t be transferred to situations with mainly dynamic contents - the common case in nearly every sophisticated web site... In the web server areas most relevant for practical use, Linux and Apache are already ahead by at least one nose. If the pages don’t come directly from the system’s main memory, the situation is even reverted to favor Linux and Apache: Here, the OpenSource movement’s prime products leave their commercial competitors from Redmond way behind.” See their paper for more figures and background.

5. **Network Computing found that GNU/Linux with Samba ran at essentially the same speed as Windows for file serving.** In their article [“Is it Time for Linux”](#), Network Computing compared Red Hat Linux v5.2 running Samba 2.0.3 against Microsoft Windows NT Server Enterprise Edition on a Pentium II-based HP NetServer LPr, stressing the machine with multiple reads and writes of small, medium and large files over the course of several hours.

For file serving, they discovered only “negligible performance differences between the two for average workloads... [and] depending on the degree of tuning performed on each installation, either system could be made to surpass the other slightly in terms of file-sharing performance.” Red Hat Linux slightly outperformed NT on file writes, while NT edged out Red Hat Linux on massive reads. Note that their configuration was primarily network-limited; they stated “At no point were we able to push the CPUs much over 50-percent utilization-the single NIC, full duplex 100BASE-T environment wouldn’t allow it.”

They also noted that “examining the cost difference between the two licenses brings this testing into an entirely new light... the potential savings on licenses alone is eye-opening. For example, based on the average street price of \$30 for a Windows NT client license, 100 licenses would cost around \$3,000, plus the cost of an NT server license (around \$600). Compare this to the price of a Red Hat Linux CD, or perhaps even a free download, and the savings starts to approach the cost of a low-end workgroup server. Scale that up to a few thousand clients and you begin to see the savings skyrocket.” See this paper’s section on [total cost of ownership](#).

6. **The Linux developers’ various efforts to improve performance appear to have paid off.** In June 2000, Dell measured the various SPECweb99 values noted above.

There are other benchmarks available, but I’ve discounted them on various grounds:

1. A more recent set of articles from eWeek on June 2001, shows some eye-popping performance numbers for GNU/Linux with TUX. However, although they compare it to Microsoft IIS, they don’t include Microsoft’s SWC (Scalable Web Cache), Microsoft’s response to TUX - and omitting it makes this comparison less balanced. You can read more at [“Tux: Built for Speed”](#), [“Smart Coding pays off Big”](#), and [Kegel’s detailed remarks](#).
2. The ZDNet article [Take that! Linux beats MS in benchmark test](#), loudly trumpeted that GNU/

Linux was the May 2001 performance leader in the TPC-H decision support (database) benchmark (“100Gb” category). However, this result should not be taken very seriously; the hardware that Linux ran on was more powerful than that of the runner-up (Windows 2000). Frankly, the more surprising fact than its top score (which can be easily explained by the hardware) is its mere measurement at all with this benchmark - traditionally only Microsoft’s numbers are reported for this benchmark at this range. For more information, see [the TPC results](#).

More information on various benchmarks is available from Kegel’s [NT vs. Linux Server Benchmark Comparisons](#), [SPEC](#), and the [dmoz entry on benchmarking](#).

Remember, in benchmarking everything depends on the configuration and assumptions that you make. Many systems are constrained by network bandwidth; in such circumstances buying a faster computer won’t help at all. Even when network bandwidth isn’t the limitation, neither Windows nor GNU/Linux do well in large-scale symmetric multiprocessing (SMP) configurations; if you want 64-way CPUs with shared memory, neither are appropriate (Sun Solaris, which is not OSS/FS, does much better in this configuration). On the other hand, if you want massive distributed (non-shared) memory, GNU/Linux does quite well, since you can buy more CPUs with a given amount of money. If massive distribution can’t help you and you need very high performance, Windows isn’t even in the race; today Windows 2000 only runs on Intel x86 compatible chips, while GNU/Linux runs on much higher performance processors as well as the x86.

5. Scalability

Which brings us to the topic of scalability, a simple term with multiple meanings:

1. **GNU/Linux and NetBSD (both OSS/FS) support a wider range of hardware platforms and performance than any other OS.** Many people mean by “scalability” to answer the question, “can you use the same software system for both small and large projects?” Often the implied issue is that you’d like to start with a modest system, but have the ability to grow the system as needs demand without costly modifications. Here OSS/FS is unbeatable; because many people can identify scalability problems, and because its source code can be optimized for its platform, the scalability of many OSS/FS products is amazing. Let’s specifically look at GNU/Linux. GNU/Linux works on [PDAs](#) (including the [Agenda VR3](#)), [obsolete hardware \(so you needn’t throw the hardware away\)](#), common modern PC hardware, over a dozen different chipsets (not just Intel x86s), [mainframes](#), [massive clusters](#), and a [number of supercomputers](#). GNU/Linux can be used for massive parallel processing; a common approach for doing this is the [Beowulf architecture](#). [Sandia’s “CPlant”](#) runs on a set of systems running GNU/Linux, and it’s the forty-second most powerful computer in the world as of June 2001 (number 42 on the [TOP 500 Supercomputer list, June 2001](#)). [IBM announced in October 2002 that GNU/Linux will be the main OS for IBM’s “Blue Gene” family of supercomputers](#). IBM plans for the Blue Gene family to eventually

perform perform a quadrillion calculations per second (one petaflop). Blue Gene/L, the first member of the family due in 2004 or 2005, will contain 65,000 processors, 16 trillion bytes of memory, and be able to perform 200 trillion calculations per second. There's even a prototype implementation of GNU/Linux on a [wrist watch](#), And GNU/Linux runs on a vast number of different CPU chips, including the [x86, Intel Itanium, ARM, Alpha, IBM AS/400 \(midrange\), SPARC, MIPS, 68k, and Power PC](#). Another OS that widely scales to many other hardware platforms is [NetBSD](#).

Thus, you can buy a small GNU/Linux or NetBSD system and grow it as your needs grow; indeed, you can replace small hardware with massively parallel or extremely high-speed processors or very different CPU architectures without switching Oses. Windows CE/ME/NT scales down to small platforms, but not to large ones, and it only works on x86 systems. Many Unix systems (such as Solaris) scale well to specific large platforms but not as well to distributed or small platforms. These OSS/FS systems are some of the most scalable programs around.

- 2. OSS/FS development processes can scale to develop large software systems.** At one time it was common to ask if the OSS/FS process is “scalable,” that is, if OSS/FS processes could really develop large-scale systems. Bill Gates’ 1976 “Open Letter to Hobbyists” asked rhetorically, “Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute it for free?” He presumed these were unanswerable questions - but he was wrong. See [my reports estimating GNU/Linux's size](#). For Red Hat Linux 6.2, I found the size to be over 17 million source lines of code (SLOC). Implemented traditionally it would have taken 4,500 person-years and over \$600 million to implement this distribution. For Red Hat Linux 7.1, I found it to have over 30 million SLOC, representing 8,000 person-years or \$1 billion (a “Gigabuck”). Most developers ascribe to the design principle that components should be divided into smaller components where practical - a practice also applied to GNU/Linux - but some components aren't easily divided, and thus some components are quite large themselves (e.g., over 2 million lines of code for the kernel, mostly in device drivers). By October 2002, [Sourceforge.net announced that it had surpassed 500,000 registered users and supported almost 50,000 OSS/FS projects](#) - and a vast number of OSS/FS projects don't use SourceForge. Thus, it's no longer reasonable to argue that OSS/FS cannot scale to develop large systems -- because it clearly can.

6. Security

Quantitatively measuring security is very difficult. However, here are a few attempts to do so, and they suggest that OSS/FS is often superior to proprietary systems, at least in some cases. I'll concentrate on comparing OSS/FS to Windows systems, since as noted above other proprietary systems are increasingly including OSS/FS components (making comparisons more difficult).

- 1. J.S. Wurzler Underwriting Managers' “hacker insurance” costs 5-15% more if Windows is**

used instead of Unix or GNU/Linux for Internet operation. At least one insurance company has indicated that Windows NT is less secure than Unix or GNU/Linux systems, resulting in higher premiums for Windows-based systems. It's often difficult to find out when a company has been successfully cracked; companies often don't want to divulge such information to the public for a variety of reasons. Indeed, if consumers or business partners lost trust in a company, the resulting loss might be much greater than the original attack. However, insurance companies that insure against cracking can require that they get such information (as a condition of coverage), and can compute future premiums based on that knowledge. According to Cnet, Okemos, Mich.-based J.S. Wurzler Underwriting Managers, one of the earliest agencies to offer "hacker insurance" (and thus more likely to have historical data for premium calculation), has begun [charging its clients anywhere from 5 to 15 percent more if they use Microsoft's Windows NT software instead of Unix or GNU/Linux for their Internet operations.](#) Walter Kopf, senior vice president of underwriting, said that "We have found out that the possibility for loss is greater using the NT system." He also said the decision is based on findings from hundreds of security assessments the company has done on their small and midsize business clients over the past couple of years.

- 2. Most defaced web sites are hosted by Windows, and Windows sites are disproportionately defaced more often than explained by its market share.** Another way to look at security is to look at the OS used by defaced web sites, and compare them to their market share. A "defaced" web site is a site that has been broken into and has its content changed (usually in a fairly obvious way, since subtle modifications are often not reported). The advantage of this measure is that unlike other kinds of security break-ins (which are often "hushed up"), it's often very difficult for victims to hide the fact that they've been successfully attacked. Historically, this information was maintained by Attrition.org. A summary can be found in [James Middleton's article](#), with the actual data found in [Attrition.org's web site](#). Attrition.org's data showed that 59% of defaced systems ran Windows, 21% Linux, 8% Solaris, 6% BSD, and 6% all others in the period of August 1999 through December 2000. Thus, Windows systems have has nearly 3 times as many defacements as GNU/Linux systems. This would make sense if there were 3 times as many Windows systems, but no matter which figures you use, that's simply not true.

Of course, not all sites are broken through their web server and OS - many are broken through exposed passwords, bad web application programming, and so on. But if this is so, why is there such a big difference in the number of defacements based on the OS? No doubt some other reasons could be put forward (this data only shows a correlation not a cause), but this certainly suggests that OSS/FS can have better security.

[Attrition.org has decided to abandon keeping track of this information due to the difficulty of keeping up with the sheer volume of broken sites](#), and it appeared that tracking this information wouldn't be possible. However, [defaced.alldas.de](#) has decided to perform this valuable service. Their recent reports show that this trend has continued; on July 12, 2001, they report that 66.09% of defaced sites ran Windows, compared to 17.01% for GNU/Linux, out of 20,260 defaced

websites.

3. **The Bugtraq vulnerability database suggests that the least vulnerable OS is OSS/FS, and that all the OSS/FS OSes in its study were less vulnerable than Windows in 1999-2000, unless you counted every GNU/Linux vulnerability multiple times.** One approach to examining security is to use a vulnerability database; an analysis of one database is the [Bugtraq Vulnerability Database Statistics](#) page. As of September 17, 2000, here are the total number of vulnerabilities for some leading OSes:

OS	1997	1998	1999	2000
Debian GNU/Linux	2	2	30	20
OpenBSD	1	2	4	7
Red Hat Linux	5	10	41	40
Solaris	24	31	34	9
Windows NT/2000	4	7	99	85

You shouldn't take these numbers very seriously. Some vulnerabilities are more important than others (some may provide little if exploited or only be vulnerable in unlikely circumstances), and some vulnerabilities are being actively exploited (while others have already been fixed before exploitation). OSS/FS OSes tend to include many applications that are usually sold separately in proprietary systems (including Windows and Solaris). For example, Red Hat 7.1 includes two relational database systems, two word processors, two spreadsheet programs, two web servers, and many text editors. In addition, in the open source world, vulnerabilities are discussed publicly, so vulnerabilities may be identified for software still in development (e.g., "beta" software). Those with small market shares are likely to have less analysis. The "small market share" comment won't work with GNU/Linux, since GNU/Linux is the #1 or #2 server OS (depending on how you count them). Still, this clearly shows that the three OSS/FS OSes listed (Debian GNU/Linux, OpenBSD, and Red Hat Linux) did much better by this measure than Windows in 1999 and (so far) in 2000. Even if a bizarre GNU/Linux distribution was created explicitly to duplicate all vulnerabilities present in any major GNU/Linux distribution, this intentionally bad GNU/Linux distribution would still do better than Windows (it would have 88 vulnerabilities in 1999, vs. 99 in Windows). The best results were for OpenBSD, an OSS/FS OS that for years has been specifically focused on security. It could be argued that its smaller number of vulnerabilities is because of its rarer deployment, but the simplest explanation is that OpenBSD has focused strongly on security - and achieved it better than the rest.

This data is partly of interest because various reporters make the same mistake: counting the same vulnerability multiple times. [One journalist, Fred Moody, failed to understand his data sources](#) - he used these figures to try to show that GNU/Linux had worse security. He took these numbers and then added the GNU/Linux ones so each Linux vulnerability was counted at

least twice (once for every distribution it applied to plus one more). By using these nonsensical figures he declared that GNU/Linux was worse than anything. If you read his article, you also must read [the rebuttal by the manager of the Microsoft Focus Area at SecurityFocus](#) to understand why the journalist's article was so wrong.

In 2002, [another journalist \(James Middleton\) made the same mistake](#), apparently not learning from prior work. Middleton counted the same Linux vulnerability up to *four* times. What's bizarre is that he even reported the individual numbers showing that specific Linux systems were actually *more* secure by using Bugtraq's vulnerability list through August 2001, and somehow he didn't realize what it meant. He noted that Windows NT/2000 suffered 42 vulnerabilities, while Mandrake Linux 7.2 notched up 33 vulnerabilities, Red Hat Linux 7.0 suffered 28, Mandrake 7.1 had 27 and Debian 2.2 had 26. In short, all of the GNU/Linux distributions had significantly fewer vulnerabilities by this count. It's not fully clear what was being considered as being "in" the OS in this case, which makes a difference. There are some hints that vulnerabilities in some Windows-based products (such as Exchange) were not counted, while vulnerabilities in GNU/Linux products with the same functionality (e.g., sendmail) *were* counted. It also appears that many of the Windows attacks were more dangerous (which were often attacks that could be invoked by remote attackers and were actively exploited), as compared to the GNU/Linux ones (which were often attacks that could only be invoked by local users and were not actively exploited at the time). I would appreciate links to someone who's analyzed these issues more carefully. The funny thing is that given all these errors, the paper gives evidence that the GNU/Linux distributions were *more* secure.

The [September 30, 2002 VNUnet.com article "Honeymoon over for Linux Users"](#), claims that there are more "Linux bugs" than "Microsoft bugs." It quotes X-Force (the US-based monitoring group of security software firm Internet Security Systems), and summarizes by saying that in 2001 the centre found 149 bugs in Microsoft software compared to 309 for Linux, and in 2002 485 Linux bugs were found compared to Microsoft's 202. However, [Linux Weekly News discovered and reported serious flaws in these figures](#):

1. "Each distribution is counted independently. The same vulnerability in five distributions will count as five separate vulnerabilities. This practice drastically overstates the number of reported Linux problems.
2. Linux vulnerabilities include those in applications (i.e. PostgreSQL) which are not part of a standard Windows system.
3. Most Linux vulnerabilities are found through code audits and similar efforts; they are patched and reported before any exploits happen. Any Windows bugs found through similar audits are fixed silently and do not appear in these counts.

Indeed, assuming that the vulnerabilities were only counted three times (and thus dividing by only 3) would show Linux as having a better result, never mind the fact that there are more than 3 Linux distributions and the other factors noted by Linux Weekly News.

Indeed, as noted in Bruce Schneier's [Crypto-gram of September 15, 2000](#), vulnerabilities are affected by other things such as how many attackers exploit the vulnerability, the speed at which a fix is released by a vendor, and the speed at which they're applied by administrators. Nobody's system is invincible.

A more recent analysis by John McCormick in Tech Republic compared Windows and Linux vulnerabilities using numbers through September 2001. This is an interesting analysis, showing that although Windows NT lead in the number of vulnerabilities in 2000, using the 2001 numbers through September 2001, Windows 2000 had moved to the "middle of the pack" (with some Linux systems having more, and others having fewer, vulnerabilities). However, it appears that in these numbers, bugs in Linux applications have been counted with Linux, while bugs in Windows applications haven't - and if that's so, this isn't really a fair comparison. As noted above, typical Linux distributions bundle many applications that are separately purchased from Microsoft.

4. **Red Hat (an OSS/FS vendor) responded more rapidly than Microsoft or Sun to advisories; Sun had fewer advisories to respond to yet took the longest to respond.** Another data point is that SecurityPortal has compiled a [list of the time it takes for vendors to respond to vulnerabilities](#). They concluded that:

How did our contestants [fare]? Red Hat had the best score, with 348 recess days on 31 advisories, for an average of 11.23 days from bug to patch. Microsoft had 982 recess days on 61 advisories, averaging 16.10 days from bug to patch. Sun proved itself to be very slow, although having only 8 advisories it accumulated 716 recess days, a whopping three months to fix each bug on average.

Their table of data for 1999 is as shown:

1999 Advisory Analysis			
Vendor	Total Days, Hacker Recess	Total Advisories	Recess Days/Advisory
Red Hat	348	31	11.23
Microsoft	982	61	16.10
Sun	716	8	89.50

Clearly this table uses a different method for counting security problems than the prior table. Of the three noted here, Sun's Solaris had the fewest vulnerabilities, but it took by far the longest to fix security problems identified. Red Hat was the fastest at fixing security problems, and placed in the middle of these three in number of vulnerabilities. It's worth noting that the OpenBSD OS (which is OSS/FS) had fewer reported vulnerabilities than all of these. Clearly, having a proprietary OS doesn't mean you're more secure - Microsoft had the largest number of security advisories, by far, using either counting method.

More recent examples seem to confirm this; on September 30, 2002, [eWeek Labs' article "Open Source Quicker at Fixing Flaws"](#) listed specific examples of more rapid response. This article can be paraphrased as follows: In June 2002, a serious flaw was found in the Apache Web server; the Apache Software Foundation made a patch available two days after the Web server hole was announced. In September 2002, a flaw was announced in OpenSSL and a patch was available the same day. In contrast, a serious flaw was found in Windows XP that made it possible to delete files on a system using a URL; Microsoft quietly fixed this problem in Windows XP Service Pack 1 without notifying users of the problem. A more direct comparison can be seen in how Microsoft and the KDE Project responded to an SSL (Secure Sockets Layer) vulnerability that made the Internet Explorer and Konqueror browsers, respectively, potential tools for stealing data such as credit card information. The day the SSL vulnerability was announced, KDE provided a patch. Later that week, Microsoft posted a memo on its TechNet site basically downplaying the problem.

5. **A 2002 survey of developers found that GNU/Linux systems are relatively immune from attacks from outsiders.** Evans Data Corp.'s [Spring 2002 Linux Developer Survey](#) surveyed over 400 GNU/Linux developers, and found that Linux systems are relatively immune from attacks from outsiders. Even though computer attacks have almost doubled annually since 1988 (according to CERT), 78% of the respondents to the GNU/Linux developers survey have never experienced an unwanted intrusion and 94% have operated virus-free. Clearly, the survey shows that GNU/Linux "doesn't get broken into very often and is even less frequently targeted by viruses," according to Jeff Child (Evans Data Corp.'s Linux Analyst); and claims that "Linux systems are relatively immune from attacks from outsiders." Child notes that it's much harder to hack a knowledgeable owner's system (and most Linux developers have hands-on, technical knowledge) and that because there are fewer desktop GNU/Linux systems there are fewer viruses being created to attack GNU/Linux. The developers being surveyed attributed the low incidence of attacks to the Open Source Software (OSS) environment; "more than 84% of Linux developers believe that Linux is inherently more secure than software not created in an OSS environment," and they ranked "Linux's security roughly comparable in security to Solaris and AIX ... and above any of the Windows platforms by a significant margin."
6. **Apache has a better security record than Microsoft's IIS, as measured by reports of serious vulnerabilities.** Eweek's July 20, 2001 article ["Apache avoids most security woes"](#) examined security advisories dating back to Apache 1.0. They found that Apache's last serious security problem (one where remote attackers could run arbitrary code on the server) was announced in January 1997. A group of less serious problems (including a buffer overflow in the server's logresolve utility) was announced and fixed in January 1998 with Apache 1.2.5. In the three and a half years since then, Apache's only remote security problems have been a handful of denial-of-service and information leakage problems (where attackers can see files or directory listings they shouldn't).

In contrast, in the article [“IT bugs out over IIS security,”](#) eWeek determined that Microsoft has issued [21 security bulletins for IIS from January 2000 through June 2001](#). Determining what this number means is a little difficult, and the article doesn’t discuss these complexities, so I examined these bulletins to find their true significance. Not all of the bulletins have the same significance, so just stating that there were “21 bulletins” doesn’t give the whole picture. However, it’s clear that several of these bulletins discuss dangerous vulnerabilities that allow an external user to gain control over the system. I count 5 bulletins on such highly dangerous vulnerabilities for IIS 5.0 (in the period from January 2000 through June 2001), and prior to that time, I count 3 such bulletins for IIS 4.0 (in the period of June 1998 through December 1999). Feel free to examine the bulletins yourself; they are MS01-033, MS01-026, MS01-025, MS01-023, MS00-086, MS99-025, MS99-019, and MS99-003. The [Code Red](#) worm, for example, exploited a vast number of IIS sites through the vulnerabilities identified in the June 2001 security bulletin MS01-033.

In short, by totaling the number of reports of dangerous vulnerabilities (that allow attackers to execute arbitrary code), I find a total of 8 bulletins for IIS from June 1998 through June 2001, while Apache had zero such vulnerabilities for that time period. Apache’s last such report was in January 1998, and that one affected the log analyzer not the web server itself. As was noted above, the last such dangerous vulnerability in Apache itself was announced in January 1997.

It’s time-consuming to do this kind of analysis, so I haven’t repeated the effort more recently. However, it’s worth noting [eWeek’s April 10, 2002 article](#) noting that ten more IIS flaws have been found in IIS Server 4.0, 5.0, and 5.1, some of which would allow attackers to crash the IIS service or allow the attacker to run whatever code he chooses.

Even this doesn’t give the full story, however; a vulnerability in IIS tends to be far more dangerous than an equivalent vulnerability in Apache, because Apache wisely follows the good security practice of “least privilege.” IIS is designed so that anyone who takes over IIS can take over the *whole system*, performing actions such as reading, modifying, or erasing any file on the system. In contrast, Apache is installed with very few privileges by default, so even taking over Apache gives attackers relatively few privileges. For example, cracking Apache does not give attackers the right to modify or erase most files. This is still not good, of course, and an attacker may be able to find another vulnerability to give them unlimited access, but an Apache system presents more challenges to an attacker than IIS.

The article claims there are four reasons for Apache’s strong security, and three of these reasons are simply good security practices. Apache installs very few server extensions by default (a “minimalist” approach), all server components run as a non-privileged user (supporting “least privilege” as noted above), and all configuration settings are centralized (making it easy for administrators to know what’s going on). However, the article also claims that one of the main reasons Apache is more secure than IIS is that its “source code for core server files is well-scrutinized,” a task that is made much easier by being OSS/FS, and it could be argued that OSS/

FS encourages the other good security practices.

Simple vulnerability notice counts are an inadequate metric for security. A vendor could intentionally release fewer bulletins - but since Apache's code and its security is publicly discussed, it seems very unlikely that Apache is deliberately underreporting security vulnerabilities. Fewer vulnerability notices could result if the product isn't well scrutinized or is rarely used - but this simply isn't true for Apache. Even the trend line isn't encouraging - using the months of the bulletins (2/99, 6/99, 7/99, 11/00, three in 5/01, and 6/01), I find the time in months between new major IIS vulnerability announcements to be 4, 1, 18, 6, 0, 0, 1, and 3 as of September 2001; this compares to 12 and 44 as of September 2001 for Apache. Given these trends, it looks like IIS's security is slowly improving, but it has little likelihood of meeting Apache's security in the near future. Indeed, these vulnerability counts are corroborated by other measures such as the web site defacement rates.

The issue here isn't whether or not a given program is invincible (what nonsense!) - the issue is which is more likely to resist future attacks, based on past performance. It's clear that the OSS/FS Apache has much a better security record than the proprietary IIS, so much so that Gartner Group decided to make an unusual recommendation (described below).

7. **IIS was attacked 1,400 times more frequently than Apache in 2001, and Windows was attacked more than all versions of Unix.** SecurityFocus co-founder and CEO Arthur Wong reported an analysis of the various vulnerabilities and attacks (based on SecurityFocus's data) in the February 2002 article [RSA: Security in 2002 worse than 2001, exec says](#). IIS was attacked 17 million times, but Apache was attacked only 12,000 times. This is a stunning comparison, since there are about twice as many Apache systems on the Internet. In 2001, Windows systems were attacked 31 million times, while Unix systems were attacked 22 million times. See the article for more information.
8. **The Gartner Group is recommending that businesses switch from Microsoft IIS to Apache or iPlanet due to IIS's poor security track record, noting that enterprises had spent \$1.2 billion simply fixing Code Red (IIS-related) vulnerabilities by July 2001.** Microsoft's IIS has such a bad security record that in September 2001, [Gartner Group announced a recommendation](#) that "businesses hit by both Code Red and Nimda immediately investigate alternatives to IIS, including moving Web applications to Web server software from other vendors such as iPlanet and Apache. Although those Web servers have required some security patches, they have much better security records than IIS and are not under active attack by the vast number of virus and worm writers." Microsoft is sometimes a Gartner Group customer, so this announcement is especially surprising.

In a [background document by Gartner](#), they discuss Code Red's impacts further. By July 2001, Computer Economics (a research firm) estimated that enterprises worldwide had spent \$1.2 billion fixing vulnerabilities in their IT systems that Code Red could exploit (remember, Code

Red is designed to only attack IIS systems; systems such as Apache are immune). To be fair, Gartner correctly noted that the problem is not just that IIS has vulnerabilities; part of the problem is that enterprises using IIS are not keeping their IT security up to date, and Gartner openly wondered why this was the case. However, Gartner also asked the question, “why do Microsoft’s software products continue to provide easily exploited openings for such attacks?” This was prescient, since soon after this the “Nimba” attack surfaced which attacked IIS, Microsoft Outlook, and other Microsoft products.

A brief aside is in order here. Microsoft spokesman Jim Desler tried to counter Gartner’s recommendation, trying to label it as “extreme” and saying that “serious security vulnerabilities have been found in all Web server products and platforms.. this is an industry-wide challenge.” While true, this isn’t the whole truth. As Gartner points out, “IIS has a lot more security vulnerabilities than other products and requires more care and feeding.” It makes sense to select the product with the best security track record, even if no product has a perfect record.

9. **The majority of the most serious security problems only apply to Microsoft’s products, and not to OSS/FS products, as suggested by the CERT/CC’s “most frequent, high-impact types of security incidents and vulnerabilities” and the ICAT database.** Some security vulnerabilities are more important than others, for a variety of reasons. Thus, some analysis centers try to determine what’s “most important,” and their results suggest that OSS/FS just doesn’t have as many vulnerabilities.

The CERT Coordination Center (CERT/CC) is federally funded to study security vulnerabilities and perform related activities such as publishing security alerts. I sampled their list of [“current activity” of the most frequent, high-impact security incidents and vulnerabilities on September 24, 2001](#), and found yet more evidence that Microsoft’s products have poor security compared to others (including OSS/FS). Four of the six most important security vulnerabilities were specific to Microsoft: W32/Nimda, W32/Sircam, cache corruption on Microsoft DNS servers, and “Code Red” related activities. Only one of the six items primarily affected non-Microsoft products (a buffer overflow in telnetd); while this vulnerability is important, it’s worth noting that many open source systems (such as Red Hat 7.1) normally don’t enable this service (telnet) in the first place and thus are less likely to be vulnerable. The sixth item (“scans and probes”) is a general note that there is a great deal of scanning and probing on the Internet, and that there are many potential vulnerabilities in all systems. Thus, 4 of 6 issues are high-impact vulnerabilities are specific to Microsoft, 1 of 6 are vulnerabilities primarily affecting Unix-like systems (including OSS/FS OSes), and 1 of 6 is a general notice about scanning. Again, it’s not that OSS/FS products never have security vulnerabilities - but they seem to have fewer of them.

The [ICAT](#) system provides a searchable index and ranking for the vulnerabilities cross-references by CVE. I sampled its top ten list on December 19, 2001; this top ten list is defined by the number of requests made for a vulnerability in ICAT (and including only vulnerabilities within the last year). In this case, 8 of the top 10 vulnerabilities only affect proprietary systems (in all

cases, Windows). Only 2 of 10 affect OSS/FS systems (#6, CAN-2001-0001, a weakness in PHP-Nuke 4.4, and #8, CVE-2001-0013, a new vulnerability found in an old version of BIND - BIND 4). Obviously, by itself this doesn't prove that there are fewer serious vulnerabilities in OSS/FS programs, but it is suggestive of it.

10. **Computer viruses are overwhelmingly more prevalent on Windows than any other system.**

Virus infection has been a major cost to users of Microsoft Windows. The LoveLetter virus alone is estimated to have cost \$960 million in direct costs and \$7.7 billion in lost productivity, and the anti-virus software industry sales total nearly \$1 billion annually. Dr Nic Peeling and Dr Julian Satchell's [*Analysis of the Impact of Open Source Software*](#) includes an analysis of the various data sources for virus counts, noting the disproportionate vulnerability of Windows systems. Here is what they said:

The numbers differ in detail, but all sources agree that computer viruses are overwhelmingly more prevalent on Windows than any other system. There are about 60,000 viruses known for Windows, 40 or so for the Macintosh, about 5 for commercial Unix versions, and perhaps 40 for Linux. Most of the Windows viruses are not important, but many hundreds have caused widespread damage. Two or three of the Macintosh viruses were widespread enough to be of importance. None of the Unix or Linux viruses became widespread - most were confined to the laboratory.

Many have noted that one reason Windows is attacked more often is simply because there are so many Windows systems in use. Windows is an attractive target for virus writers simply because it is in such widespread use. For a virus to spread, it must transmit itself to other susceptible computers; on average, each infection must cause at least one more. The ubiquity of Windows machines makes it easier for this threshold to be reached.

There may be a darker reason: there are many who do not like Microsoft's business practices, and perhaps this contributes to the problem. Some of Microsoft's business practices have been proven in court to be illegal, but the U.S. government appears unwilling to effectively punish or stop those practices. Some computer literate people may be taking their frustration out on users of Microsoft's product. This is absolutely wrong, and in most countries illegal. It is extremely unethical to attack an innocent user of a Microsoft product simply because of Microsoft's policies, and I condemn such behavior. At this point, although this has been speculated many times, I have not found any evidence that this is a widespread motivator for actual attacks. On the other hand, if you are choosing products, do you really want to choose the product whom people may have a vendetta against?

However, the reasons given above don't explain the disproportionate vulnerability of Microsoft's products. A simpler explanation, and one that is easily proven, is that Microsoft has made many design choices over the years in Microsoft's products that are fundamentally less secure, and this

has made their products a much easier target than many other systems. Examples include executing start-up macros in Word, executing attachments in Outlook, and the lack of write protection on system directories in Windows 3.1/95/98. This may be because Microsoft has assumed that customers will buy their products whether or not Microsoft secures them. After all, until recently there's been little competition, so there was no need to spend money on "invisible" attributes such as security. It's also possible that Microsoft is still trying to adjust to an Internet-based world; the Internet would not have developed as it has without Unix-like systems, which have supported the Internet standards for decades, while for many years Microsoft ignored the Internet and then suddenly had to play "catch-up" in the early 1990s. Microsoft has sometimes claimed that they can't secure their products because they want to ensure that their products are "easy to use". While it's true that some security features can make a product harder to use, usually a secured product can be just as easy to use if the security features are carefully designed into the product. Besides, what's so easy to use about a system that must be reformatted and reinstalled every few months because yet another virus got in? But for whatever the reason, it's demonstrably true that Microsoft's designers have in the past made decisions that made their products' security much weaker than other systems.

In contrast, while it's possible to write a virus for OSS/FS Oses, their design makes it more difficult for viruses to spread... showing that Microsoft's design decisions were not inevitable. It appears that OSS/FS developers tend to select design choices that limit the damage of viruses, perhaps in part because their code is subject to public inspection and comment. For example, OSS/FS programs generally do not support start-up macros nor execution of mail attachments that can be controlled by attackers. Also, leading OSS/FS Oses (such as GNU/Linux and the *BSDs) have always had write protection on system directories. [Another discussion on why viruses don't seem to significantly affect OSS/FS systems is available from Roaring Penguin.](#) OSS/FS systems are *not* immune to malicious code, but they are certainly more resistant.

11. **Microsoft has had far more vulnerabilities than anyone else, according to SecurityTracker.** The paper [SecurityTracker Statistics](#) (March 2002) analyzes vulnerabilities from April 2001 through March 2002. They identified 1595 vulnerability reports, covering 1175 products from 700 vendors. Their analysis found that Microsoft had more vulnerabilities than anyone else (187, or 11.7% of all vulnerabilities), and more than four times the next vendor. The next largest were Sun (42, 2.6% of the total), HP (40, 2.5%), and IBM (40, 2.5%). Solely OSS/FS vendors did much better: the Apache Software Foundation had 13 (0.8% of the total), and Red Hat had 10 (0.6% of the total). It can be argued that Microsoft sells more kinds of software than most other vendors, but this is nevertheless an astonishingly large number of vulnerabilities. The gap between Microsoft and everyone else widened during the second half of the year, which is even scarier.
12. **According to a Network Security evaluation, an OSS/FS vulnerability scanner (Nessus) was found to be the best (most effective).** On January 8, 2001, Network Computing's article [Vulnerability Assessment Scanners](#). reported an evaluation of nine network scanning tools, most

of them proprietary. In their evaluation, Network Computing set up demonstration systems with 17 of the most common and critical vulnerabilities; they then used the various network scanning tools to see how effectively each of the tools detected these vulnerabilities. Sadly, not one product detected all vulnerabilities; the best scanner was the OSS/FS program Nessus Security Scanner, which found 15 of the 17 (which also received their top total score); the next best was a proprietary scanner which only found 13.5 out of 17.

In their words,

Some of us were a bit skeptical of the open-source Nessus project's thoroughness until [Nessus] discovered the greatest number of vulnerabilities. That's a hard fact to argue with, and we are now eating our words ... [Nessus] got the highest overall score simply because it did more things right than the other products.

I agree with the authors that ideally a network vulnerability scanner should find every well-known vulnerability, and that "even one hole is too many." Still, perfection is rare in the real world. More importantly, a vulnerability scanner should only be part of the process to secure an organization - it shouldn't be the sole activity. Still, this evaluation suggests that an organization will be *more* secure, not less secure, by using an OSS/FS program. It could be argued that this simply shows that this OSS/FS program had more functionality - not more security - but in this case, the product's sole functionality was to improve security.

One serious problem is that there are strong economic disincentives for proprietary vendors to make their software secure. For example, if vendors make their software more secure, they would often fail to be "first" in a given market; this often means that they will lose that market. Since it is extremely difficult for customers to distinguish proprietary software with strong security from those with poor security, the poor products tend to eliminate the good ones (after all, they're cheaper to develop and thus cost less). Governments have other disincentives as well. For a discussion of some of the economic disincentives for secure software, see [Why Information Security is Hard - an Economic Perspective by Ross Anderson](#) (Proceedings of the Annual Computer Security Applications Conference (ACSAC), December 2001, pp. 358-365). It's not clear that OSS/FS always avoids these disincentives, but it appears in at least some cases it does. For example, OSS/FS source code is public, so the difference in security is far more visible than in proprietary products.

One of the most dangerous security problems with proprietary software is that if intentionally malicious code is snuck into it, such code is extremely difficult to find. Few proprietary vendors have other developers examine *all* code in great detail - their testing processes are designed to catch mistakes (not malice) and often don't look at the code at all. In contrast, malicious code can be found by anyone when the source code is publicly available, and with OSS/FS, there are incentives for arbitrary people to review it (such as to add new features or perform a security review of a product they intend to use). Thus, someone inserting malicious code to an OSS/FS project runs a far greater risk of detection. Here are two examples, one confirmed, one not confirmed:

1. Some time between 1992 and 1994, Borland inserted an intentional “back door” into their database server, “InterBase”, as a secret username and fixed password. This back door allowed any local or remote user to manipulate any database object and install arbitrary programs, and in some cases could lead to controlling the machine as “root”. This vulnerability stayed in the product for at least 6 years - no one else could review the product, and Borland had no incentive to remove the vulnerability. Then Borland released its source code on July 2000 as an OSS/FS project. The “Firebird” project began working with the source code, and uncovered this serious security problem with InterBase in December 2000 (only 5 months after release). By January 2001 the CERT announced the existence of this back door as CERT advisory CA-2001-01. What’s discouraging is that the backdoor can be easily found simply by looking at an ASCII dump of the program (a common cracker trick), so it’s quite possible that this vulnerability was exploited many times in the intervening years. Once this problem was found by open source developers reviewing the code, it was patched quickly.
2. Mohammad Afroze Abdul Razzak, arrested by Mumbai (Bombay) police Oct. 2, 2001, claims that [Osama bin Laden’s Al Qaeda network were able to gain employment at Microsoft and attempted to plant “trojans, trapdoors, and bugs in Windows XP.”](#) This was reported to Ravi Visvesvaraya Prasad, a New Delhi information systems and telecommunication consultant, and then reported by the [Washington Post’s Newsbytes division](#). This claim has not been confirmed; indeed, I’m somewhat skeptical. The problem, however, is that this is impossible to disprove. Even if this particular case isn’t true, note that this threat is unfortunately a credible threat to proprietary software, because very few of its users can review the code. This is far less dangerous to OSS/FS software, due to the worldwide review that’s possible (including the ability to see the changes made in each version).

[Bruce Perens, in “Open sourcers wear the white hats”](#), makes the interesting claim that most of the people reviewing proprietary products looking for security flaws (aside from one or two paid reviewers) are “black hats,” outsiders who disassemble the code or try various types of invalid input in search of a flaw that they can exploit (and not report). There is simply little incentive, and many roadblocks, for someone to search for security flaws simply to improve someone else’s proprietary product. “Only a black hat would disassemble code to look for security flaws. You won’t get any ‘white hats’ doing this for the purpose of [just] closing the flaws.” In contrast, he thinks many open source developers *do* have such an incentive. This article slightly overstates the case; there are other incentives (such as fame) that can motivate a few people to review some other company’s proprietary product for security. Still, it has a point; even formal reviews often only look at designs (not code), proprietary code is often either unreviewed or poorly reviewed, and there are many cases (including the entire OpenBSD system) where legions of developers review open source code for security issues. As he notes, “open source has a lot of ‘white hats’ looking at the source. They often do find security bugs while working on other aspects of the code, and the bugs are reported and closed.”

One complication for OSS/FS is that often the OSS/FS business model does not easily support certain kinds of formal security evaluations, in particular a Common Criteria evaluation. But this seems to be primarily a problem of finding sponsors to fund the evaluation, rather than some inability to perform an

evaluation of OSS/FS software. [Red Hat and Oracle](#) have announced in February 2003 that they have teamed up to get Red Hat Linux Advanced Server evaluated under the Common Criteria. [IBM has separately announced that it will fund Common Criteria evaluations of GNU/Linux](#). The Cyberspace Security Policy and Research Institute (CSPRI) is leading the project [Secure Government Software](#) to advance security enhanced OSS/FS for government through research, development, and NIAP certification; they hope to complete a GNU/Linux server configuration evaluation (at EAL 2, and for a broader range of configurations), a PKI evaluation (at EAL 2), and then a strong security evaluation (EAL 4).

The “Alexis de Tocqueville Institute” (ADTI) published a white paper called [“Opening the Open Source Debate”](#) that purported to examine OSS/FS issues. Unfortunately, it makes many wrong, specious, and poorly-argued claims about OSS/FS, including some related to security. Wired (in its article [Did MS Pay for Open-Source Scare?](#)) made some startling discoveries about ADTI, and found strong circumstantial evidence that the paper was paid for by Microsoft (a prime competitor to OSS/FS), directly or indirectly: “A Microsoft spokesman confirmed that Microsoft provides funding to the Alexis de Tocqueville Institution... Microsoft did not respond to requests for comment on whether the company directly sponsored the debate paper. De Tocqueville Institute president Ken Brown and chairman Gregory Fossedal refused to comment on whether Microsoft sponsored the report.” [Politech found additional suspicious information about ADTI](#). ADTI apparently has a history of creating “independent” results that are apparently paid for by corporations (e.g., see the [Smoke Free for Health article](#) about ADTI’s pro-tobacco-lobby papers). Reputable authors clearly identify any potential conflict of interest, even if it’s incidental; ADTI did not when it developed this OSS/FS paper. Not surprisingly, the ADTI paper makes many errors and draws unwarranted conclusions. I’ll just note a few examples of the paper’s problems that aren’t as widely noted elsewhere: incorrect or incomplete quotations, rewriting web browser history, and cleverly omitting the most important data in one of their charts:

- The ADTI “quotes” me several times in the paper, but in some cases claims I said something I never said, and in others places them out of context by intentionally omitting important things that I said. ADTI originally claimed that I said that “without licensing the source code in a multilicense format, (referring to other more permissive licenses), it is impossible for GPL to work for a proprietary business model.” But I *never said this*. In fact, I specifically noted to ADTI that [Microsoft sells a GPL’ed product](#) (a fact I’d already publicly published). Instead of removing the statement, ADTI later made up a statement and claimed that I said it. What I really said was more nuanced: “without licensing the source code in a multilicense format [GPL and other licenses], the GPL does not permit certain kinds of uses in proprietary business models.” The words are similar, but this is a much narrower statement; note that many business models (including some proprietary ones!) are permitted by the GPL. ADTI also claims I said that “today I would be confident that the number [of GPL software] has probably grown to 80%,” I only said that I believed the number was probably larger than 50%, but since I couldn’t remember the exact figures offhand, I told them to examine my papers - which ADTI did not do (if they had, they’d notice that I’d recently published that [71.85% of Freshmeat’s software packages were covered by](#)

- [the GPL](#)). More intriguing are the omissions. For example, I explained to ADTI the GPL license (which they did not understand, even though they were attacking it); ADTI seems to think that the GPL requires public release of code, but it does not. The GPL only requires that those who receive the binary executable receive the source code. This is crucial, because it means you can still keep “secrets” in GPL’ed code, in spite of ADTI’s implied assertion otherwise. Besides, there’s anecdotal evidence that the government uses most GPL’ed code as-is, in which case these issues don’t apply - the GPL permits arbitrary use and redistribution of unmodified copies.
- For a second example, the ADTI paper rewrites the history of web browsers in an attempt to make its claims; it bases much on the claim that Mosaic was an open source web browser, but [it never was](#); modified versions of the Unix version could only be used non-commercially without a separate license (OSS/FS must be usable commercially), and the Mac and Windows licenses were even more restrictive. It also completely omits the heavily publicized move of Netscape to OSS/FS in 1998, clearly the most important event in web browser history relating to OSS/FS. I specifically mentioned these problems to ADTI before they published their paper, but ADTI was not willing to fix their paper to meet the facts.
 - Switching to the third example, ADTI includes a chart of showing source lines of code (SLOC) for various programs; it even references my paper [More than a Gigabuck](#) while noting that the Linux kernel is over 2 million SLOC. The same chart also reports that Windows XP is 30 million SLOC, an interesting statement since to my knowledge this value has not been made public (ADTI has *not* revealed their source, but has confirmed to me that they really meant Windows XP). But note the invalid comparison - ADTI reports on the Linux kernel (a small part of an OS), and Windows XP (a whole OS), but not on an whole OSS/FS OS. ADTI willfully ignores my paper’s abstract and main point, which reported that the whole Red Hat Linux 7.1 distribution is *also* 30 million SLOC; by omitting the most important data, ADTI gives false impressions. But these are merely the tip of the iceberg; the paper’s flaws are so numerous, and discussing the flaws in its conclusions require so much effort, that a serious rebuttal would require writing a whole separate paper.

Thus, I recommend that anyone who reads the ADTI paper also examine the detailed rebuttals available from many different sources, since these rebuttals expose the paper’s numerous flaws. Rebuttals are available from [John Viega and Bob Fleck of Secure Software](#) (Viega is a respected security expert), [Juliao Duarte](#) (Director of the Security Skill Center, Oblog Software, SA), [Roaring Penguin’s David Skoll](#) (via the Register), [Ken Ambrose](#) (via LWN), and [Leon Brooks](#). [Anthony Awtrey analyzed the changes made](#) in the published editions of the ADTI paper. In short, ADTI’s paper is a highly biased and poorly researched “report.”

There are other non-quantitative discussions on OSS/FS and security. [The October 2002 paper *Open Source Digital Forensics Tools: The Legal Argument* by Brian Carrier](#) notes that to enter scientific evidence into a United States court, a forensics tool must be reliable and relevant as determined through the “Daubert” guidelines. The paper examines then those guidelines and argues that “open source tools may more clearly and comprehensively meet the [forensics] guidelines than closed source tools.”

Many security experts have stated that OSS/FS has advantages over the security of proprietary software, including [Whitfield Diffie](#) (co-inventor of public key cryptography), Bruce Schneier (expert on cryptography and computer security), Vincent Rijmen (a developer of the Advanced Encryption Standard (AES)), [Elias Levy](#) (Aleph1, the former moderator of the popular security discussion group Bugtraq). [John Viega](#) (author of a book on secure programming), and Peter Neumann This doesn't guarantee that a particular OSS/FS program is more secure than a particular proprietary product - merely that there are some fundamental security advantages to easing public review.

In contrast, [Microsoft's Jim Allchin disclosed under oath in court testimony that some Microsoft code was so flawed it could not be safely disclosed to the public](#). Yet more recently, Microsoft announced its "Government Security Program" to allow governments to view most source code (though not all code, and they cannot change and freely redistribute the results). Indeed, Reuters reported a survey by Forrester Research Inc. that found that [most computer security experts at major companies do not think Microsoft Corporation's products are secure](#); 77% said security was a top concern when using Windows. The primary problem reported was that patches were not implemented, because "administrators lacked both the confidence that a patch won't bring down a production system and the tools and time to validate Microsoft's avalanche of patches."

Now it should be obvious from these figures that OSS/FS systems are not magically invincible from security flaws. Indeed, some have argued that making the source code available gives attackers an advantage (because they have more information to make an attack). While OSS/FS gives attackers more information, this ignores opposing forces: having the source code also gives the defenders more information (because they can also examine its original source code), and in addition, the defenders can improve the code. More importantly, the necessary information for breaking into a program is in the binary executable of the program; disassemblers and decompilers can quickly extract whatever information is needed from executables to break into a program, so hiding the source code isn't all that helpful for preventing attacks against attackers who are willing to use such programs. Again, it is *not* true that proprietary programs are always more secure, or that OSS/FS is always more secure, because there are many factors at work. For example, a well-configured and well-maintained system, of any kind, will almost always be far more secure than a poorly configured and unmaintained system of any kind. For a longer description of these issues, see [my discussion on open source and security](#) (part of my book on [writing secure software](#)). However, from these figures, it appears that OSS/FS systems are in many cases *better* - not just equal - in their resistance to attacks as compared to proprietary software.

7. Total Cost of Ownership (TCO)

Total cost of ownership (TCO) is an important measure; it doesn't matter if a product starts out cheaply if it costs you more down the line. However, TCO is extremely sensitive to the set of assumptions you make.

Indeed, whatever product you use or support, you can probably find a study to show it has the lowest TCO for some circumstance. Not surprisingly, both Microsoft and [Sun](#) provide studies showing that their products have the lowest TCO. [Xephon](#) has a study determining that mainframes are the cheapest per-user (due to centralized control) at £3450 per user per year; Centralized Unix cost £7350 per user per year, and a decentralized PC environment costs £10850 per user per year. [Xephon](#) appears to be a mainframe-based consultancy, though, and would want the results to come out this way. There are indeed situations where applying a mainframe makes sense.. but as we'll see in a moment, you can use OSS/FS in such environments too.

In short, what has a smaller TCO depends on your environment and needs. To determine TCO you must identify all the important cost drivers (the "cost model") and estimate their costs. Don't forget "hidden" costs, such as administration costs, upgrade costs, technical support, end-user operation costs, and so on. However, OSS/FS has many strong cost advantages in various categories that, in many cases, will result in its having the smallest TCO.

1. **OSS/FS costs less to initially acquire.** OSS/FS costs much less to get initially. OSS/FS isn't free in the monetary sense, because the "free" in "free software" refers to freedom, not price. This distinction is usually summarized as "free speech, not free beer". [Merrill Lynch executive Robert Lefkowitz found what may be a better way to describe it: "We like to think of it as 'free as in market.'"](#)

OSS/FS isn't cost-free, because you'll still spend money for paper documentation, support, training, system administration, and so on, just as you do with proprietary systems. In many cases, the actual programs in OSS/FS distributions can be acquired freely by downloading them ([linux.org provides some pointers on how to get distributions](#)). However, most people (especially beginners and those without high-speed Internet connections) will want to pay a small fee to a distributor for a nicely integrated package with CD-ROMs, paper documentation, and support. Even so, OSS/FS costs far less to acquire.

For example, examine the price differences when trying to configure a server, such as public web server or an intranet file and email server, in which you'd like to use C++ and an RDBMS. This is simply an example; different missions would involve different components. Using the prices from "Global Computing Supplies" (Suwanee, GA), September 2000, rounded to the nearest dollar, here is a quick summary of the purchasing costs:

	Microsoft Windows 2000	Red Hat Linux
Operating System	\$1510 (25 client)	\$29 (standard), \$76 deluxe, \$156 professional (all unlimited)
Email Server	\$1300 (10 client)	included (unlimited)
RDBMS Server	\$2100 (10 CALs)	included (unlimited)

C++ Development	\$500	included
-----------------	-------	----------

Basically, Microsoft Windows 2000 (25 client) costs \$1510; their email server Microsoft Exchange (10-client access) costs \$1300, their RDBMS server SQL Server 2000 costs \$2100 (with 10 CALs), and their C++ development suite Visual C++ 6.0 costs \$500. Red Hat Linux 6.2 (a widely-used GNU/Linux distribution) costs \$29 for standard (90 days email-based installation support), \$76 for deluxe (above plus 30 days telephone installation support), or \$156 for professional (above plus SSL support for encrypting web traffic); in all cases it includes all of these functionalities (web server, email server, database server, C++, and much more). A public web server with Windows 2000 and an RDBMS might cost \$3610 (\$1510+\$2100) vs. Red Hat Linux's \$156, while an intranet server with Windows 2000 and an email server might cost \$2810 (\$1510+\$1300) vs. Red Hat Linux's \$76.

Both packages have functionality the other doesn't have. The GNU/Linux system always comes with an unlimited number of licenses; the number of clients you'll actually use depends on your requirements. However, this certainly shows that no matter what, Microsoft's server products cost thousands of dollars more per server than the equivalent GNU/Linux system.

For another in-depth analysis comparing the initial costs GNU/Linux with Windows, see [Linux vs. Windows: The Bottom Line](#) by [Cybersource Pty Ltd](#). Here's a summary of their analysis (in 2001 U.S. dollars):

	Microsoft Solution	OSS/FS (GNU/Linux) Solution	Savings by using GNU/Linux
Company A (50 users)	\$69,987	\$80	\$69,907
Company B (100 users)	\$136,734	\$80	\$136,654
Company C (250 users)	\$282,974	\$80	\$282,894

[Consulting Times](#) found that as the number of mailboxes got large, the three-year TCO for mainframes with GNU/Linux became in many cases quite compelling. For 50,000 mailboxes, an Exchange/Intel solution cost \$5.4 million, while the Linux/IBM(G6) solution cost \$3.3 million. For 5,000 mailboxes, Exchange/Intel cost \$1.6 million, while Groupware on IFL cost \$362,890. For yet another study, see the [Cost Comparison from jimmo.com](#). Obviously, the price difference depends on exactly what functions you need for a given task, but for many common situations, GNU/Linux costs far less to acquire.

- Upgrade/maintenance costs are typically far less.** Long-term upgrade costs are far less for OSS/FS systems. For example, upgrading a Microsoft system will typically cost around half the original purchase. What's worse, you are essentially at their mercy for long-term pricing, because

there is only one supplier (see [Microsoft Turns the Screws](#)). In contrast, the GNU/Linux systems can be downloaded (free), or simply re-purchased (generally for less than \$100), and the single upgrade be used on every system. This doesn't include technical support, but the technical support can be competed (a situation that's not practical for proprietary software). An anti-trust lawyer would say that OSS/FS technical support is "contestable." In short, if you don't like your GNU/Linux supplier (e.g., they've become too costly), you can switch.

- OSS/FS does not impose license management costs and avoids nearly all licensing litigation risks.** Proprietary vendors make money from the sale of licenses, and are imposing increasingly complex mechanisms on consumers to manage these licenses. Customers who cannot later prove than they paid for every installed copy of proprietary software (e.g., due to copying by an employee or losing the license paperwork) risk stiff penalties. In short: by using proprietary software, you run the risk of having the vendor to sue you.

To counter these risks, organizations must keep careful track of license purchases. This means that organizations must impose strict software license tracking processes, purchase costly tracking programs, and pay for people to keep track of these licenses and perform occasional audits.

In contrast, there's no license management or litigation risk in using OSS/FS software. Some OSS/FS software do have legal requirements if you modify the program or embed the program in other programs, but proprietary software usually forbids modifying the program and often also imposes licensing requirements for embedding a program (e.g., royalty payments). Thus, software developers must examine what components they're employing to understand their ramifications, but this would be true for both OSS/FS and proprietary programs. See the [licensing litigation discussion](#) later in this paper for more about licensing costs and risks.

- OSS/FS can often use older hardware more efficiently than proprietary systems, yielding smaller hardware costs and sometimes eliminating the need for new hardware.** OSS/FS runs faster on faster hardware, of course, but many OSS/FS programs can use older hardware more efficiently than proprietary systems, resulting in lower hardware costs - and in some cases requiring no new costs (because "discarded" systems can suddenly be used again). For example, the [minimum requirements for Microsoft Windows 2000 Server \(according to Microsoft\)](#) are a Pentium-compatible CPU (133 MHz or higher), 128 MiB of RAM minimum (with 256MiB the "recommended minimum"), and a 2 GB hard drive with at least 1.0 GB free. According to Red Hat, Red Hat Linux 7.1 (a common distribution of GNU/Linux) requires at a minimum an i486 (Pentium-class recommended), 32MiB RAM (64MiB recommended), and 650MB hard disk space (1.2 GB recommended).

In Scientific American's August 2001 issue, the article [The Do-It-Yourself Supercomputer](#) discusses how the researchers built a powerful computing platform with many discarded computers and GNU/Linux. The result was dubbed the "Stone Soupercomputer"; by May 2001 it

contained 133 nodes, with a theoretical peak performance of 1.2 gigaflops.

- 5. When used as an application server based system, the total costs for hardware drop by orders of magnitude.** Many people make the mistake of deploying OSS/FS workstations (such as GNU/Linux or the *BSDs) the same way they would deploy Windows systems. Although it's possible, this is an unnecessarily costly approach if they're installing a set of workstations for typical productivity applications (e.g., word processing, spreadsheets, etc. for an office). For many, a better approach is to provide each user with a very old GNU/Linux-based machine which is merely a graphics display (an "X terminal"), and then run the actual applications on an "application server" that is shared by all the users. See [How to create a Linux-based network of computers for peanuts](#) for more information about this. With this application server approach, workstations can cost about \$30 each (using "obsolete" machines), a server (shared by many users) can cost about \$1000 each, and nearly all system administration is centralized (reducing administration costs). A nice side-effect of this approach is that users can use any workstation just by logging in. A more detailed discussion of this approach is given in [Paul Murphy's article, Total cost of ownership series revisited](#). This is how the City of Largo, Florida, and many other organizations use GNU/Linux.
- 6. As the number of systems and hardware performance increases, this difference in initial and upgrade costs becomes even more substantial.** As the number of servers increases, proprietary solutions become increasingly costly. First, many proprietary systems (including Microsoft) sell per-client licenses; this means that even if your hardware can support more clients, you'll must pay more to actually use the hardware you've purchased. Secondly, if you want to use more computers, you must pay for more licenses in proprietary systems. In contrast, for most GNU/Linux distributions, you can install as many copies as you like for no additional fee, and there's no performance limit built into the software. There may be a fee for additional support, but you can go to competing vendors for this support.

According to [Network World Fusion News](#), Linux is increasingly being used in healthcare, finance, banking, and retail due to its cost advantages when large numbers of identical sites and servers are built. According to their calculations for a 2,000 site deployment, SCO UnixWare would cost \$9 million, Windows would cost \$8 million, and Red Hat Linux costs \$180.

- 7. There are many other factors; their effect varies on what you're trying to do.** There are many other factors in TCO, but it's difficult to categorize their effects in general, and it's generally difficult to find justifiable numbers for these other effects. Windows advocates claim that system administrators are cheaper and easier to find than Unix/Linux administrators, while GNU/Linux and Unix advocates argue that fewer such administrators are needed (because administration is easier to automate and the systems are more reliable to start with) - and [quantitative studies are beginning to back this latter claim](#). Some GNU/Linux advocates have told me that GNU/Linux lends itself to hosting multiple services on one server in cases where Windows installations must use multiple servers. License compliance administration can be

costly for proprietary systems (e.g., time spent by staff to purchase CALS, keep track of licenses, and undergo audits) - a cost that simply isn't relevant to OSS/FS.

8. **Cybersource's 2002 study found TCO savings of 24% to 34% when using OSS/FS instead of Microsoft's proprietary approach.** [Cybersource's "Linux vs. Windows: Total Cost of Ownership Comparison"](#) modeled an organization with 250 computer-using staff, an appropriate number of workstations, servers, with Internet connectivity, an e-business system, network cabling and hardware, standard software, and salaries for IT professionals to establish and support this infrastructure and technology. Using existing hardware and infrastructure, they found a three-year savings of 34.26% (\$251,393 U.S. dollars) when using the "Linux/Open Source Solution" instead of the proprietary "Microsoft solution". When new hardware and infrastructure were purchased, the savings were 24.69%. Note that this study is a follow-on of [their earlier study](#); a [commentary is available at Linux Journal](#). It could be argued that this was merely a paper study, but they claim that they've seen significant savings in their consulting work. In any case, TCO savings have been reported by real organizations, corroborating these results, as discussed below.
9. **An Italian study in 2002 found GNU/Linux to have a TCO 34.84% less than Windows.** The [full study is in Italian](#); you can try to read an automatically-generated [translation](#).
10. **For many circumstances, the total cost savings can be substantial. For example, real-world savings exceeding \$250,000 per year were reported by 32% of the Chief Technical Officers (CTOs) surveyed in a 2001 InfoWorld survey; 60% of these CTOs saved over \$50,000 annually.** The August 27, 2001 InfoWorld (pages 49-50) reported on a survey of 40 CTOs who were members of the InfoWorld CTO network. In this survey, 32% using OSS reported savings greater than \$250,000; 12% reported savings between 100,001 and \$250,000; and 16% reported saving between \$50,001 and \$100,000. Indeed, only 8% reported annual savings less than \$10,000 (so 92% were saving \$10,000 or more annually). A chief benefit of OSS, according to 93% of the CTOs, was reduced cost of application development or acquisition; 72% said that a chief benefit was reduced development or implementation time (multiple answers were allowed). The CTOs reported using or planning to use OSS for web servers (65%), server OSes (63%), web-application servers (45%), application development testing (45%), and desktop OS (38%), among other uses. InfoWorld summarized it this way: "in early 2000, it seemed as if no one was using open-source software for business-critical tasks... a vast majority of today's corporate IT executives are now using or plan to use OSS OSes and web servers for their enterprise applications."
11. **The Robert Frances Group's July 2002 study found the TCO of GNU/Linux is roughly 40% (less than half) that of Microsoft Windows and only 14% that of Sun Microsystem's Solaris.** [The Robert Frances Group \(RFG\), in Westport, Conn., studied actual costs at production deployments of Web servers running on GNU/Linux with Apache, Microsoft Windows with IIS, and Sun Solaris with Apache at 14 Global 2000 enterprises.](#) These are *real* deployments where, if

the web server goes down, money is lost - not minor prototype sites. Their TCO analysis was based on the software purchase price, hardware purchase and maintenance prices, software maintenance and upgrade prices, and administrative costs. To make the numbers comparable, these figures were scaled to a “processing unit” able to handle 100,000 hits per day; see the study for more information. They determined that over three years a (scaled) GNU/Linux deployment cost \$74,475, a Windows deployment cost \$190,662, and a Solaris deployment cost \$534,020. Thus, the cost of running GNU/Linux is roughly 40% that of Microsoft Windows and only 14% that of Sun Microsystem’s Solaris.

This report also found that GNU/Linux and Solaris had smaller administrative costs than Windows. Although Windows system administrators cost less individually, each Linux or Solaris administrator could administrate many more machines, making Windows administration much more costly. The study also revealed that Windows administrators spent twice as much time patching systems and dealing with other security-related issues than did Solaris or GNU/Linux administrators.

RFG also examined some areas that were difficult to monetize. In the end, they concluded that “Overall, given its low cost and flexible licensing requirements, lack of proprietary vendor goals, high level of security, and general stability and usability, Linux is worth considering for most types of server deployments.”

12. **Netproject reported that the TCO with Linux on the desktop was 35% that of Microsoft Windows (a 65% savings).** [Netproject’s Cost of Ownership report](#) found a very significant savings, and it reported the following causes:
 - The elimination of license fees for both the system software and office software;
 - Elimination of vendor churn that forces unnecessary software updates;
 - Reduction in the number of software security updates;
 - No need for anti-virus software for Linux computers [anti-virus software for Linux is only needed to check for viruses that run on Microsoft PCs];
 - Reduction in the number of support staff.

13. **A majority of InternetWeek Newsbreak subscribers from companies with over \$5 million in revenues reported that OSS/FS software was costs substantially less than proprietary software.**

[A survey was by TheOpenEnterprise.com](#) (a joint editorial effort between InternetWeek.com and InformationWeek) of individuals with management responsibility for IT and software specifically in companies with over \$5 million in revenue. In this survey, Indeed, 39% said “open source/ standards-based software” costs 25% to 50% less than proprietary software, while 27% (over 1 in 4) said it’ costs 50% to 75% less. In context, it appears their phrase was intended to mean the same (or similar) thing as the term OSS/FS in this paper, since in many cases they simply use the term “open-source.” As they note, “Would your CFO react favorably to a 50-75% reduction in

software costs?”

14. **Many organizations report significant savings when using OSS/FS.** Here are a few examples of specific organizations saving money through OSS/FS:
- a. The analysis [Linux as a Replacement for Windows 2000](#) compares Red Hat Linux 7.1 to Windows 2000; in this customer's case, using Linux instead of Windows 2000 saved \$10,000. The reviewer came from a Windows/DOS background, and after performing an intensive hands-on Linux project lasting several months, determined that “you will be stunned by the bang for the buck that ... open source software offers.”
 - b. Intel's IT Vice President, Doug Busch, [reported savings of \\$200 million](#) by replacing costly Unix servers with cheaper servers running GNU/Linux.
 - c. [Amazon.com was able to cut \\$17 million in technology expenses in a single quarter](#), largely due to a switch to Linux. Amazon spent \$54 million on technology and content expenses in its third quarter (ending Sept. 30), compared with \$71 million in the year-ago quarter, and executives expected that technology costs as a portion of net sales would decrease by 20% this year.
 - d. [The city of Largo, Florida](#) reports a savings of \$1 million per year using GNU/Linux and “thin clients.”
 - e. Dell offers a savings of 21% when using GNU/Linux. Dell computer has a dedicated hosting service, such as their [D-2800 offering](#). This service offers a respectable system (Pentium 850, 256MiB, 20GB, 21GB/month bandwidth) in two configurations: Red Hat Linux 7.1 for \$189/month, and Windows 2000 for \$239/month. Thus, with identical hardware and bandwidth provision, the GNU/Linux system is 21% cheaper. This is especially interesting because Dell is not out to prove which system is better; as a business, they've just figured out competitive prices at which they can offer their services.
 - f. [An independent report in Denmark](#) concluded that if the political goals for using the Internet to improve the public sector are to be fulfilled, it would be \$500 million cheaper over the next 10 years to use OSS/FS instead of Microsoft software (my thanks to Poul-Henning Kamp, who translated the conclusions).

There are many other reports from those who have switched to OSS/FS systems; see the [usage reports section](#) for more information.

15. **Even Microsoft has admitted that its products are more costly than GNU/Linux.** For some time Microsoft has tried to convince users that its products are somehow less costly. However, as documented in [Var Business](#) and [The Register](#), Microsoft CEO Steve Ballmer in 2002 admitted that Microsoft has not “figured out how to be lower-priced than Linux. For us as a company,

we're going through a whole new world of thinking." The Register summarizes Microsoft's new approach as saying that "it costs more because it's worth more"; whether this is true is rather debatable in many cases, but at least it's a more sensible argument. However, Microsoft has gone back to trying to claim that they cost less, so the detail in this section is still needed.

16. **A Microsoft-sponsored study claims that Windows is cheaper than Linux, but this has been debunked as a general claim.** [The Microsoft-sponsored study \(available from Microsoft\)](#) compared Windows 2000 to Linux; it stated that Linux had lower TCO for webserving, and Windows 2000 had a lower TCO for network infrastructure, print serving, file serving and security applications (note: the "David Wheeler" quoted in InfoWorld is not the author of this paper). I will give credit here: unlike the Mindcraft reports sponsored by Microsoft, this TCO report clearly states that it was sponsored by Microsoft, and I appreciate that.

It's important to examine the assumptions of any TCO study, to see if its assumptions could apply to many other situations - and it is easily argued that they don't. [Joe Barr discusses some of the problems in this TCO study](#). These include assuming that the operating system is never upgraded in a 5-year period, using an older operating system Microsoft is transitioning *from*, and not using the current Enterprise license agreement (which many organizations find they must use). Costs that are not included in the study include legal advice costs (when signing large-scale agreements), purchase and maintenance of a software license inventory system (which you'll generally need even with Enterprise agreements), costs if you are audited, cost of insurance and liability incidents (if a proof of purchase is misplaced, you might need to pay the \$151,000 per-incident liability), and paying multiple times for the same product (a side-effect of many Enterprise license agreements).

Barr concludes with: "TCO is like fine wine: it doesn't travel well. What may be true in one situation is reversed in another. What gets trumpeted as a universal truth ('Windows is cheaper than Linux') may or may not be true in a specific case, but it is most certainly false when claimed universally." Since the TCO of a system depends on its application, and Microsoft as sponsor could specifically set all of the parameters, the conclusions of the report were easily predicted.

- 17.
18. **Another Microsoft-sponsored study claims that Microsoft's toolsuite with .NET is cheaper than using GNU/Linux with J2EE.** [This Giga Research study sponsored by Microsoft](#) compared the costs incurred by five large and medium-size companies that used J2EE (Java 2 Enterprise Edition) with the costs incurred by seven large and medium-size companies that used . Net applications to develop Web portal applications. For large corporations, the cost of using Microsoft products (for development and deployment plus three years of maintenance) was 28% less than for J2EE/Linux. For medium-size companies, the Microsoft products were 25% cheaper.

However, once again, the TCO values all hinge on the assumptions made. [As CIO.com points out](#), the Microsoft-based solution was cheaper primarily because the GNU/Linux systems were configured using extremely expensive proprietary products such as those from Oracle (for the database system) and BEA (for the development system).

A company can certainly choose to use these particular products when developing with GNU/Linux, but not all organizations will choose to do so. Indeed, the acronym "LAMP" (Linux, Apache, MySQL, and PHP/Python/Perl) was coined because that combination is extremely popular when creating web portal applications. MySQL and PostgreSQL are popular OSS/FS database programs; PHP, Python, and Perl are popular OSS/FS development languages (and tie easily into the rest of the development suite provided by OSS/FS operating systems). An obvious question to ask is, "Why were extremely common configurations (such as LAMP) omitted in this Microsoft-funded study?" CIO.com reports Giga's answer: "Microsoft didn't ask them [to] look at any such companies."

Again, I give credit to Giga for clearly reporting who funded the study. Indeed, if your situation closely matches Giga's study, your costs might be very similar. But it would be a mistake to conclude that different situations would necessarily have the same results.

You may also want to see [MITRE Corporation's business case study of OSS](#), which considered military systems.

Most of these items assume that users will use the software unmodified, but even if the OSS/FS software doesn't do everything required, that is not necessarily the end of the story. One of the main hallmarks of OSS/FS software is that it can be modified by users. Thus, any true TCO comparison should consider not just the products that fully meet the requirements, but the existing options that with some modifications could meet the requirements. It may be cheaper to start with an existing OSS/FS program, and improve it, than to start with a proprietary program that has all of the necessary functionality. Obviously, the total TCO including such costs varies considerably depending on the circumstances.

[Brendan Scott \(a lawyer specializing in IT and telecommunications law\) argues that the long run TCO of OSS/FS must be lower than proprietary software.](#) Scott's paper makes some interesting points, for example, "TCO is often referred to as the total cost of 'ownership' ... [but] 'ownership' of software as a concept is anathema to proprietary software, the fundamental assumptions of which revolve around ownership of the software by the vendor. ... The user [of proprietary software] will, at best, have some form of (often extremely restrictive) license. Indeed, some might argue that a significant (and often uncoded) component of the cost of 'ownership' of proprietary software is that users don't own it at all." The paper also presents arguments as to why GPL-like free software gives better TCO results than other OSS/FS licenses. Scott concludes that "Customers attempting to evaluate a free software v. proprietary solution can confine their investigation to an evaluation of the ability of the packages to meet the customer's needs, and may presume that the long run TCO will favor the free software package. Further, because the licensing costs are additional dead weight costs, a customer ought to also prefer a free

software solution with functionality shortfalls where those shortfalls can be overcome for less than the licensing cost for the proprietary solution.”

Microsoft’s first TCO study comparing Windows to Solaris (mentioned earlier) is not a useful starting point for estimating your own TCO. Their study reported the average TCO at sites using Microsoft products compared to the average TCO at sites using Sun systems, but although the Microsoft systems cost 37% less to own, the Solaris systems handled larger databases, more demanding applications, 63% more concurrent connections, and 243% more hits per day. In other words, the Microsoft systems that did less work cost less than systems that did more work. This is not a useful starting point if you’re using TCO to help determine which system to buy - to make a valid comparison by TCO, you must compare the TCOs of systems that meet your requirements. A two-part analysis by Thomas Pfau (see [part 1](#) and [part 2](#)) identifies this and many other flaws in the study.

There are some studies that emphasize Unix-like systems, not OSS/FS, which claim that there are at least some circumstances where Unix-like systems are less costly than Windows. [A Strategic Comparison of Windows vs. Unix](#) by Paul Murphy is one such paper. It appears that many of these arguments would also apply to OSS/FS systems, since many of them are Unix-like.

Again, it’s TCO that matters, not just certain cost categories. However, given these large differences, in many situations OSS/FS has a smaller TCO than proprietary systems. At one time it was claimed that OSS/FS installation took more time, but nowadays OSS/FS systems can be purchased pre-installed and automatic installers result in equivalent installation labor. Some claim that system administration costs are higher, but studies like Sun’s suggest that in many cases the system administration costs are lower, not higher, for Unix-like systems (at least Sun’s). For example, on Unix-like systems it tends to be easier to automate tasks (because you can, but do not need, to use a GUI) - thus over time many manual tasks can be automated (reducing TCO). Retraining costs can be significant - but now that GNU/Linux has modern GUI desktop environments, there’s anecdotal evidence that this cost is actually quite small (I’ve yet to see serious studies quantitatively evaluating this issue). In short, it’s often hard to show that a proprietary solution’s purported advantages really help offset their demonstrably larger costs in other categories when there’s a competing mature OSS/FS product for the given function.

Clearly, if one product is significantly more productive than another where it’s used, it’s worth paying more for it. However, it’s clear that at least for major office tasks, GNU/Linux systems are about as usable as Windows systems. For example, [one usability study comparing GNU/Linux to Microsoft Windows XP](#) found that it was almost as easy to perform most major office tasks using GNU/Linux as with Windows: “Linux users, for example, needed 44.5 minutes to perform a set of tasks, compared with 41.2 minutes required by the XP users. Furthermore, 80% of the Linux users believed that they needed only one week to become as competent with the new system as with their existing one, compared with 85% of the XP users.” [The detailed report \(in German\) is also available.](#)

Does this mean that OSS/FS always have the lowest TCO? No! As I’ve repeatedly noted, it depends on its use. But the notion that OSS/FS *always* has the larger TCO is simply wrong.

8. Non-Quantitative Issues

In fairness, I must note that not all issues can be quantitatively measured, and to many they are the most important issues. The issues most important to many include freedom, protection from license litigation, and flexibility. Another issue that's hard to measure is innovation.

1. **OSS/FS protects its users from the risks and disadvantages of single source solutions.** While “free software” advocates use the term “freedom,” and some businesses emphasize different terms such as “free market”, “multiple sources”, “alternate supply channels”, and “the necessity of multiple vendors”, the issue is the same: users do not want to be held hostage by any one vendor. Businesses often prefer to buy products in which there is a large set of competing suppliers, because it reduces their risk; they can always switch to another supplier if they're not satisfied, the supplier raises their prices substantially, or the original supplier goes out of business. This translates into an effect on the products themselves: if customers can easily choose and switch between competing products, the products' prices go down and their quality goes up. Conversely, if there is a near or real monopoly for a given product, over time the vendor will continuously raise the cost to use the product and limit its uses to those that benefit the monopolist. Users who are unwilling to leave single source solutions often pay dearly later as their single source raises their costs.

For example, many organizations have chosen to use Microsoft's products exclusively, and Microsoft is trying to exploit this through its new “Microsoft Licensing 6.0 Program.” The [TIC/Sunbelt Software Microsoft Licensing Survey Results \(covering March 2002\)](#) reports the impact on customers of this new licensing scheme. 80% had a negative view of the new licensing scheme, noting, for example, that the new costs for software assurance (25% of list for server and 29% of list for clients) are the highest in the industry. Of those who had done a cost analysis, an overwhelming 90% say their costs will increase if they migrate to 6.0, and 76% said their costs would increase from 20% to 300% from what they are paying now under their current 4.0 and 5.0 Microsoft Licensing plans. This survey found that 36% of corporate enterprises don't have the funds to upgrade to the Microsoft Licensing 6.0 Program. Half indicated that the new agreement would almost certainly delay their migration initiatives to new Microsoft client, server and Office productivity platforms, and 38% say they are actively seeking alternatives to Microsoft products. In [New Zealand a Commerce Commission Complaint](#) has been filed claiming that Microsoft's pricing regime is anti-competitive. Craig Horrocks notes that the Software Assurance approach does not assure that the purchaser receives anything for the money; it merely buys the right to upgrade to any version Microsoft releases in the covered period. Microsoft may levy further charges on a release, and the contract does not obligate Microsoft to deliver anything in the time period.

There are increasing concerns about Microsoft's latest releases of Windows. Michael Jennings

argues in [Windows XP Shows the Direction Microsoft is Going](#) that Microsoft users are increasingly incurring invasion of privacy, intentionally crippled yet necessary services, and other problems.

More generally, defining an organization's "architecture" as being whatever one vendor provides is sometimes called "[Vendor Lock-in](#)" or "[Pottersville](#)", and this "solution" is a well-known [AntiPattern](#) (an AntiPattern is a "solution" that has more problems than it solves).

Having only vendor completely control a market is dangerous from the viewpoint of costs (since the customer then has no effective control over costs), and it also raises a security problem: the *monoculture vulnerability*. In biology, it is dangerous to depend on one crop strain, because any disease can cause the whole crop to fail. Similarly, one proprietary vendor who completely controls a market creates a uniformity that is far easier to massively attack. OSS/FS programs provide an alternative implementation, and even when one dominant OSS/FS program exists, because they can be changed (because the source code is available) at least some implementations are likely to be more resistant to attack.

Historically, proprietary vendors eventually lose to vendors selling products available from multiple sources, even when their proprietary technology is (at the moment) better. Sony's Betamax format lost to VHS in the videotape market, IBM's microchannel architecture lost to ISA in the PC architecture market, and Sun's NeWS lost to X-windows in the networking graphics market, all because customers prefer the reduced risk (and eventually reduced costs) of non-proprietary products. This is sometimes called "commodification", a term disparaged by proprietary vendors and loved by users. Since users spend the money, users eventually find someone who will provide what they want, and then the other suppliers discover that they must follow or give up the market area.

With OSS/FS, users can choose between distributors, and if a supplier abandons them they can switch to another supplier. As a result, suppliers will be forced to provide good quality products and services for relatively low prices, because users can switch if they don't. Users can even band together and maintain the product themselves (this is how the Apache project was founded), making it possible for groups of users to protect themselves from abandonment.

The article [Commentary from a new user: Linux is an experience, not an operating system](#), describes freedom this way:

"As I worked in Linux... the word 'free' took on a far greater meaning. As the advocates of the Open Source and Free Software movements put it, free means freedom. Yes, as a humble user of Linux, I am experiencing freedom and pride in using a world-class operating system.

Linux is not only an operating system. It embodies a myriad of concepts about how

the world of computers and software should be. This is an operating system designed by the world, meant for the world. Everyone who is interested in Linux, can develop, share and use it. People can contribute their best in programming, documenting or in any aspect of their choice. What a novel concept!

Free in Linux spells freedom -- freedom to use Linux, freedom to use the code, freedom to tweak and improve it. Not being a programmer, I still can be happy about many things. For me, freedom has meant that my operating system is transparent, and there are no hidden codes at work in my computer. Nothing about Linux is hidden from me. ... I've gained more control over my computer for the first time in my life."

2. **OSS/FS protects its users from licensing litigation and management costs.** Proprietary vendors make money from the sale of licenses, and are imposing increasingly complex mechanisms on consumers to manage these licenses. For example, Microsoft's Windows XP requires [product activation](#) - a scheme that means that an accumulation of hardware changes requires a new activation code. A license no longer gives unlimited rights to reinstall - if you have hardware trouble, you may end up being forced to re-buy your product. Indeed, for a variety of reasons, [businesses are finding that they must buy the same proprietary software more than once](#).

Proprietary vendors also litigate against those who don't comply with their complex licensing management requirements, creating increased legal risks for users. For example, the Business Software Alliance (BSA) is a proprietary software industry organization sponsored by Microsoft, Macromedia, and Autodesk, and spends considerable time searching for and punishing companies who cannot prove they are complying. As noted in the [SF Gate \(Feb. 7, 2002\)](#), the BSA encourages disgruntled employees to call the BSA if they know of any license violations. "If the company refuses to settle or if the BSA feels the company is criminally negligent and deliberately ripping off software, the organization may decide to get a little nastier and organize a raid: The BSA makes its case in front of a federal court in the company's district and applies for a court order. If the order is granted, the BSA can legally storm the company's offices, accompanied by U.S. marshals, to search for unregistered software."

[Software Licensing by Andrew Grygus](#) discusses the risks and costs of proprietary licensing schemes in more detail. According to their article, "the maximum penalty is \$150,000 per license deficiency; typically, this is negotiated down, and a company found deficient at around \$8,000 will pay a penalty of around \$85,000 (and must buy the \$8,000 in software too)." For example, [information services for the city of Virginia Beach, VA were practically shut down for over a month](#) and 50 employees were tied up trying to put its licensing in order to answer a random audit demand by Microsoft. Eventually the city was fined \$129,000 for missing licenses the city had probably paid for but couldn't match to paperwork. [Temple University had to pay \\$100,000 to the BSA](#), in spite of strong policies forbidding unauthorized copying.

To counter these risks, organizations must keep careful track of license purchases. This means that organizations must impose strict software license tracking processes, purchase costly tracking programs, and pay for people to keep track of these licenses and perform occasional audits.

A related problem is that companies using proprietary software must, in many cases, get permission from their software vendors to sell a business unit that uses the proprietary software, or face legal action. For example, [Microsoft has filed objections to Kmart's proposed \\$8.4 million sale of Bluelight.com to United Online Inc.](#), citing software licensing as one of their concerns. Microsoft stated that "The licenses that debtors (Kmart) have of Microsoft's products are licenses of copyrighted materials and, therefore, may not be assumed or assigned with[out] Microsoft's consent." Whether or not this is a risk depends on the licensing scheme used; in many cases it appears that the legal "right of first sale" doctrine cannot be applied (for example, there are many different licensing schemes for Windows, so the same action with Windows may be legal or not depending on the licensing scheme used to acquire it).

In contrast, OSS/FS users have no fear of litigation from the use and copying of OSS/FS. Licensing issues do come up when OSS/FS software is modified and then redistributed, but to be fair, proprietary software essentially forbids this action (so it's a completely new right). Even in this circumstance, redistributing modified OSS/FS software generally requires following only a few simple rules (depending on the license), such as giving credit to previous developers and releasing modifications under the same license as the original program.

One intriguing example is [the musical instrument company Ernie Ball](#), described in *World Trade*, May 2002. A disgruntled ex-employee turned them into the Business Software Alliance (BSA); who then arranged to have them raided by armed Federal Marshals. Ernie Ball was completely shut down for a day, and then was required to not touch any data other than what is minimally needed to run their business. After the investigation was completed, Ernie Ball was found to be noncompliant by 8%; Ball argued that it was "nearly impossible to be totally compliant" by their rules, and felt that they were treated unfairly. The company ended up paying a \$90,000 settlement, \$35,000 of which were Microsoft's legal fees. Ball then decided at that moment his company would become "Microsoft free." In one year he converted to a Linux-based network and UNIX "mainframe" using Sun's StarOffice (Sun's proprietary cousin to OpenOffice); he now has no Microsoft products at all, and much of the software is OSS/FS or based on OSS/FS products.

- OSS/FS has greater flexibility.** OSS/FS users can tailor the product as necessary to meet their needs in ways not possible without source code. Users can tailor the product themselves, or hire whoever they think can solve the problem (including the original developer). Some have claimed that this creates the "danger of forking," that is, of multiple incompatible versions of a product. This is "dangerous" only to those who think competition is evil - we have multiple versions of

cars as well. And in practice, the high cost of maintaining software yourself has resulted in a process in which the change is contributed back to the community. If it's not contributed (e.g., it solves a problem that needed solving but only for a specialized situation), then it's still a win for the user - because it solved a user's problem which would have been unsolved otherwise.

For example, [in 1998 Microsoft decided against developing an Icelandic version of Windows 95](#) because the limited size of the market couldn't justify the cost. Without the source code, the Icelandic people had little recourse. However, OSS/FS programs can be modified, so Icelandic support was immediately added to them, without any need for negotiation with a vendor. Users never know when they will have a specialized need not anticipated by their vendor; being able to change the source code makes it possible to support those unanticipated needs.

4. **Many believe that there are social, moral, or ethical imperatives for using OSS/FS.** The Free Software Foundation has [a set of papers describing their philosophy, i.e., why they believe Free Software is an ethical imperatives](#). These lengthy documents explain themselves in depth, so there's little need to describe them further here.
5. **There is ample evidence that OSS/FS encourages, not quashes, innovation.** Microsoft publicly claims that OSS/FS (especially its most common license, the GPL) will eliminate innovation, but the facts undermine these claims. Most IT managers reject Microsoft's claims; in 2000 a Forrester Research study interviewed 2,500 IT managers and found that 84% of them forecast that open source software would be the spark behind major innovations throughout the industry. Indeed, when examining [the most important software innovations](#), it's quickly discovered that Microsoft invented no key innovations, nor was Microsoft the first implementor of any of them. In fact, [there is significant evidence that Microsoft is not an innovator at all](#). In contrast, many of the key innovations were OSS/FS projects. For example, [Tim Berners-Lee, inventor of the World Wide Web, stated in December 2001](#) that "A very significant factor [in widening the Web's use beyond scientific research] was that the software was all (what we now call) open source. It spread fast, and could be improved fast - and it could be installed within government and large industry without having to go through a procurement process." Note that this didn't end after the ideas were originally developed; the #1 web server in 2001 (Apache) is open source and the #2 web browser in 2001 (Netscape Navigator) is almost completely OSS/FS, ten years after the original development of the web. Indeed, recent court cases give strong evidence that the only reason the proprietary Internet Explorer was the #1 web browser was due to years of illegal use of monopoly power by Microsoft. In public, Microsoft has long asserted that OSS/FS cannot innovate, but [in February 2003 even Microsoft's Bill Gates admitted that many developers are building innovative capabilities using OSS/FS systems](#).

This history of innovation shouldn't be surprising; OSS/FS approaches are based on the scientific method, allowing anyone to make improvements or add innovative techniques and then make them immediately available to the public. [Eric Raymond has made a strong case for why](#)

[innovation is more likely, not less likely, in OSS/FS projects](#). The [Sweetcode](#) web site reports on innovative free software. Here's what Sweetcode says about their site: "Innovative means that the software reported here isn't just a clone of something else or a minor add-on to something else or a port of something else or yet another implementation of a widely recognized concept... Software reported on sweetcode should surprise you in some interesting way."

If Microsoft's proprietary approaches were better for research, then you would expect that to be documented in the research community. However, the opposite is true; the paper "[NT Religious Wars: Why Are DARPA Researchers Afraid of Windows NT?](#)" found that, in spite of strong pressure by paying customers, computer science researchers strongly resisted basing research on Windows. Reasons given were: developers believe Windows is terrible, Windows really is terrible, Microsoft's highly restrictive non-disclosure agreements are at odds with researcher agendas, and there is no clear technology transition path for OS and network research products built on Windows (since only Microsoft can distribute changes to its products). Microsoft's own secret research (later leaked as "[Halloween I](#)") found that "Research/teaching projects on top of Linux are easily 'disseminated' due to the wide availability of Linux source. In particular, this often means that new research ideas are first implemented and available on Linux before they are available / incorporated into other platforms." Stanford Law School professor Lawrence Lessig (the "special master" in Microsoft's antitrust trial) noted that "[Microsoft was using its power to protect itself against new innovation](#)" and that Microsoft's practices generally threaten technical innovation - not promote it.

Developers themselves report that OSS/FS is innovative. [According to the BCG study of OSS/FS developers](#), 61.7% of surveyed developers claimed that their OSS/FS project was either their most creative effort or was equally as creative as their most creative experience.

Given an whole site dedicated to linking to innovative OSS/FS projects, OSS/FS's demonstrated history in key innovations, Microsoft's failure to demonstrate innovation itself, reports from IT managers supporting OSS/FS, reports of dissatisfaction by researchers and others about Microsoft's proprietary approaches, and Microsoft's own research finding that new research ideas are often first implemented and available on Linux before other platforms, the claim that OSS/FS quashes innovation is demonstrably false.

While I cannot quantitatively measure these issues, these issues (especially the first four) are actually the most important issues to many.

9. Unnecessary Fears

Some avoid OSS/FS, not due to the issues noted earlier, but due to unnecessary fears of OSS/FS. Let's counter some of them:

1. **Is proprietary software fundamentally better supported than OSS/FS? No.** There are actually two kinds of support for OSS/FS: traditional paid-for support and informal community support. There are many organizations who provide traditional support for a fee; since these can be competed (an option not available for proprietary software), you can often get an excellent price for support. Again, an anti-trust lawyer would say that OSS/FS support is “contestable.” For example, many GNU/Linux distributions include installation support when you purchase their distribution, and for a fee they’ll provide additional levels of support. There are many independent organizations that provide traditional support for a fee as well. The article [‘Team’ work Pays Off for Linux](#) evaluated four different technical support services for GNU/Linux systems, and found that “responsiveness was not a problem with any of the participants” and that “No vendor failed to solve the problems we threw at it.” It’s very important to understand that OSS/FS support can be competed separately from the software product; in proprietary products, support is essentially tied to purchase of a usage license.

For example, [the Gartner Group](#) reports that “By 2005, warranties and additional maintenance for at least the 100 most-popular open-source software products will be offered by commercial software vendors, service providers, or insurance companies (0.7 probability). In the meantime, users can minimize any ‘fitness for purpose’ risks through evaluation and testing, and by only using production releases of well-known, mature products from reputable distributors.” Indeed, this prediction seems nearly certain, since it’s been happening and accelerating for years.

As an alternative, you can also get unpaid support from the general community of users and developers through newsgroups, mailing lists, web sites, and other electronic forums. While this kind of support is non-traditional, many have been very satisfied with it. Indeed, in 1997 InfoWorld awarded the “Best Technical Support” award to the “Linux User Community,” beating all proprietary software vendors’ technical support. Many believe this is a side-effect of the Internet’s pervasiveness - increasingly users and developers are directly communicating with each other and finding such approaches to be more effective than the alternatives (for more on this business philosophy, see [The Cluetrain Manifesto](#)). Using this non-traditional approach effectively for support requires following certain rules; for information on these rules, consult [‘How to ask smart questions’](#) and [How to Report Bugs Effectively](#). But note that there’s a choice; using OSS/FS does not require you to use non-traditional support (and follow its rules), so those who want guaranteed traditional support can pay for it just as they would for proprietary software.

2. **Does proprietary software give you more legal rights than OSS/FS? No.** Some have commented that “with OSS/FS you give up your right to sue if things go wrong.” The obvious retort is that essentially all proprietary software licenses *also* forbid lawsuits - so this isn’t different at all! Anyone who thinks that they can sue Microsoft or other shrink-wrap proprietary vendors when things go wrong is simply fooling themselves. In any case, most users aren’t interested in suing vendors - they want working systems. See [‘A Senior Microsoft Attorney Looks at Open-Source Licensing’](#), where Bryan Pfaffenberger argues that “With open-source

software... you are, in principle, walking into the deal with your eyes wide open. You know what you're getting, and if you don't, you can find someone who does. Open-source licenses enable the community of users to inspect the code for flaws and to trade knowledge about such flaws, which they most assuredly do. Such licenses allow users to create derivative versions of the code that repair potentially hazardous problems the author couldn't foresee. They let users determine whether the program contains adequate safeguards against safety or security risks. In contrast, the wealthy software firms pushing UCITA are asking us to buy closed-source code that may well contain flaws, and even outright hazards attributable to corporate negligence - but they won't let us see the code, let alone modify it. You don't know what you're getting." Finally, if the software goes wrong and it's very important, you can fix it yourself or pay to have it fixed; this option greatly reduces risk, and this option doesn't exist for proprietary software.

There is another legal difference that's not often mentioned. Many proprietary programs require that users permit software license audits and pay huge fees if the organization can't prove that every use is licensed. So in some cases, if you use proprietary software, the biggest legal difference is that the *vendors get to sue you*.

- 3. Does OSS/FS expose you to greater risk of abandonment? No.** Businesses go out of business, and individuals lose interest in products, in both the proprietary and OSS/FS world. A major difference, however, is that all OSS/FS programs are automatically in escrow - that is, if their original developer stops supporting the product, any person or group can step forward to support it instead. This has been repeatedly demonstrated in OSS/FS. For example, the [GIMP is a bitmapped graphical editor that was abandoned by its original developers](#) (what's worse, they abandoned it before its initial release and failed to arrange for anyone else to succeed them). Nevertheless, even in this worst-case situation, after a period of time other users came forward and continued its development. As another example, [NCSA abandoned its web server "httpd", so some of its users banded together to maintain it - its results became Apache, the world's most popular web server](#).
- 4. Is OSS/FS economically viable? Yes.** There are companies that are making money on OSS/FS, or using OSS/FS to support their money-making activities. Many papers have been written about how to make money using OSS/FS, such as [Eric S. Raymond's "The Magic Cauldron"](#) and [Donald K. Rosenberg's "How to make money with open-source software."](#) OSS/FS isn't compatible with some business models, but capitalism does not guarantee that businesses can remain unchanged in changing environments.

For example, [HP reported in January 2003 that it had annual sales of \\$2 billion linked to GNU/Linux](#). [IBM reported in 2002 that they had already made almost all of their \\$1 billion investment in Linux back in only one year](#) - i.e., as profit. [The Financial Times Story "Could Linux dethrone the software king?" from January 21, 2003](#) analyzes some of the financial issues.

[Joel Spolsky's "Strategy Letter V"](#) notes that "most of the companies spending big money to develop open source software are doing it because it's a good business strategy for them." His argument is based on microeconomics, in particular, that every product in the marketplace has substitutes and complements. A substitute is another product you might buy if the first product is too costly, while a complement is a product that you usually buy together with another product. Since demand for a product increases when the prices of its complements decrease, smart companies try to commoditize their products' complements. For example, an automobile manufacturer may invest to reduce the cost of gas refinement - because if gas is cheaper, they'll sell more cars. For many companies, such as computer hardware makers and service organizations, supporting an OSS/FS product turns a complementary product into a commodity - resulting in more sales (and money) for them.

Although many OSS/FS projects originally started with an individual working in their spare time, and there are many OSS/FS projects which can still be described that way, the "major" widely-used projects tend to no longer work that way. Instead, most major OSS/FS projects have large corporate backing with significant funds applied to them. This shift has been noted for years, and is discussed in papers such as [Brian Elliott Finley's paper *Corporate Open Source Collaboration?*](#)

Also, looking only at companies making money from OSS/FS misses critical issues, because that analysis looks only at the supply side and not the demand side. Consumers are saving lots of money and gaining many other benefits by using OSS/FS, so there is a strong economic basis for its success. Anyone who is saving money will fight to keep the savings, and it's often cheaper for consumers to work together to pay for small improvements in an OSS/FS product than to keep paying and re-paying for a proprietary product. A proprietary vendor may have trouble competing with a similar OSS/FS product, because the OSS/FS product is probably much cheaper and frees the user from control by the vendor. For many, money is still involved - but it's money saved, not money directly acquired as profit. Some OSS/FS vendors have done poorly financially - but many proprietary vendors have also done poorly too. Luckily for consumers, OSS/FS products are not tied to a particular vendor's financial situation as much as proprietary products are.

Fundamentally, software is economically different than physical goods; it is infinitely replicable, it costs essentially nothing to reproduce, and it can be developed by thousands of programmers working together with little investment (driving the per-person development costs down to very small amounts). It is also durable (in theory, it can be used forever) and nonrival (users can use the same software without interfering with each other, a situation not true of physical property). Thus, the marginal cost of deploying a copy of a software package quickly approaches zero. This explains how Microsoft got so rich so quickly (by selling a product that costs nearly nothing to replicate), and why many OSS/FS developers can afford to give software away. See ["Open Source-omics: Examining some pseudo-economic arguments about Open Source"](#) by Ganesh Prasad, which counters "several myths about the economics of Open Source." [People are already](#)

[experimenting with applying OSS/FS concepts to other intellectual works](#), and it isn't known how well OSS/FS concepts will apply to other fields. However, it is clear that making economic decisions based on analogies between software and physical objects is not sensible, because software has many economic characteristics that are different from physical objects.

5. **Will OSS/FS destroy the software industry? Won't programmers starve if many programs become OSS/FS? No.** It's certainly possible that many OSS/FS products will eliminate their proprietary competition, but that's the nature of competition. If OSS/FS approaches pose a significant threat to proprietary development approaches, then proprietary vendors must either find ways to compete or join the OSS/FS movement. No one mourns the loss of buggy whip manufacturers, who were driven out of business by a superior approach to transportation (cars). Heinlein noted that no one is guaranteed protection against change in *Life-Line* (1939): "There has grown up in the minds of certain groups in this country the notion that because a man or a corporation has made a profit out of the public for a number of years, the government and the courts are charged with the duty of guaranteeing such profit in the future, even in the face of changing circumstances and contrary public interest. This strange doctrine is not supported by statute nor common law. Neither individuals nor corporations have any right to come into court and ask that the clock of history be stopped, or turned back, for their private benefit. "

[Eric Raymond's "The Magic Cauldron"](#) describes many ways to make money with OSS/FS. One particularly interesting note is that there is evidence that 95% of all software is not developed for sale. For the vast majority of software, organizations must pay developers to create it anyway. Thus, even if OSS/FS eliminated all shrink-wrapped programs, it would only eliminate 5% of the existing software development jobs. And, since the OSS/FS programs would be less expensive, other tasks could employ developers that are currently too expensive, so widespread OSS/FS development would not harm the ability of developers to make a living.

OSS/FS doesn't require that software developers work for free; many OSS/FS products are developed or improved by employees (whose job is to do so) and/or by contract work (who contract to make specific improvements in OSS/FS products). If an organization must have a new capability added to an OSS/FS program, they must find someone to add it... and generally, that will mean paying a developer to develop the addition. The difference is that, in this model, the cost is paid for development of those specific changes to the software, and not for making copies of the software. Since copying bits is essentially a zero-cost operation today, this means that this model of payment more accurately reflects the actual costs.

Indeed, there has been a recent shift in OSS/FS away from volunteer programmers and towards paid development by experienced developers. Again, see [Ganesh Prasad's article](#) for more information. [AOL decided to spin off the Mozilla project as a separate organization; not only does the separate organization employ several full-time employees, but other organizations have worked to hire Mozilla workers](#). Fundamentally, paying software developers is similar to paying for proprietary licenses, except you only have to pay for improvements (instead of paying for

each copy), so many organizations appear to have found that it's worthwhile. There's even quantitative evidence that OSS/FS developers are experienced; the [Boston Consulting Group/OSDN Hacker Survey](#) (January 31, 2002) surveyed users of SourceForge and found that OSS/FS developers had an average age of 30 and that they averaged 11 years of programming experience.

It seems unlikely that so many developers would choose to support an approach that would destroy their own industry, and there are a large number of OSS/FS developers. On January 28, 2003, Sourceforge.net all by itself reported that it had 555,314 registered users on its OSS/FS development site, and many of the largest OSS/FS projects are *not* hosted by Sourceforge.net (including the Linux kernel, the gcc compilation system, the X-Windows GUI system, the Apache web server, the Mozilla web browser, and the Open Office document management suite). Unfortunately, there seems to be no data to determine the number of OSS/FS developers worldwide, but it is likely to be at least a million people and possibly many, many more.

OSS/FS enables inexperienced developers to gain experience and credibility, while enabling organizations to find the developers they need (and will then pay to develop more software). Often organizations will find the developers they need by looking at the OSS/FS projects they depend on (or on related projects). Thus, lead developers of an OSS/FS project are more likely to be hired by organizations when those organizations need an extension or support for that project's program. This gives both hope and incentive to inexperienced developers; if they start a new project, or visibly contribute to a project, they're more likely to be hired to do additional work. Other developers can more easily evaluate that developer's work (since the code is available for all to see), and the inexperienced developer gains experience by interacting with other developers. This isn't just speculation; one of Netscape's presenters at FOSDEM 2002 was originally a volunteer contributor to Netscape's Mozilla project; his contributions led Netscape to offer him a job (which he accepted).

Karen Shaeffer has written an interesting piece, [Prospering in the Open Source Software Era](#), which discusses what she views to be the effects of OSS/FS. For example, OSS/FS has the disruptive effect of commoditizing what used to be proprietary property and it invites innovation (as compared to proprietary software which constrained creativity). She thinks the big winners will be end users and the software developers, because "the value of software no longer resides in the code base - it resides in the developers who can quickly adapt and extend the existing open source code to enable businesses to realize their objectives concerned with emerging opportunities. This commoditization of source code represents a quantum step forward in business process efficiency - bringing the developers with the expertise into the business groups who have the innovating ideas."

- 6. Is OSS/FS compatible with Capitalism? Yes.** Years ago some tried to label OSS/FS as "communistic" or "socialistic" (i.e., anti-capitalist), but that rhetoric has failed. One article explaining why OSS/FS and capitalism are compatible is [How Does the Capitalist View Open Source?](#). This paper shows that OSS/FS is quite consistent with capitalism: it increases wealth

without violating principles of property ownership or free will. See the earlier notes on economic viability.

7. **If only OSS/FS programs exist in a software category, will that completely eliminate competition? No.** Oddly enough, OSS/FS programs sometimes compete with each other in a given functional area. The text editors emacs (primarily GNU emacs) and vi (primarily vim) have duelled for decades. Sendmail is still a popular program for delivering email, but it has competition from other OSS/FS programs such as Postfix and Exim. The desktop environments GNOME and KDE compete with each other, as do the OS kernels of Linux and the BSDs. Generally, competing OSS/FS projects must distinguish themselves from each other to succeed (e.g., through user interface philosophies, design approaches, characteristics like security, licensing strategies, and so on), but of course that's true for competing proprietary programs too. Also, competing OSS/FS programs generally try to stay compatible with each other (because their customers demand it) and sometimes even help each other with technical problems. For example, freedesktop.org provides a forum to encourage cooperation among open source desktops for the X Window System (such as KDE and GNOME), and is part of the [Free Standards Group](http://FreeStandardsGroup.org) which tries to accelerate the use and acceptance of open source technologies through the development, application and promotion of standards. In addition, even if there is one product, multiple organizations can compete for maintenance and support (e.g., GNU/Linux distributors do this). Thus, even if OSS/FS eliminates all proprietary programs in a given category, that would still not eliminate competition.
8. **Is OSS/FS a “destroyer of intellectual property”?** No. You can use OSS/FS products (e.g., a word processor) to develop private and proprietary information, and you can keep the information as confidential and proprietary as you want. What you can't do is use someone else's material in a way forbidden by law... and this is true for all software, not just OSS/FS.

One interesting case is the “General Public License” (GPL), the most common OSS/FS license. Software covered by the GPL can be modified, but any release of that modified software must include an offer for the source code under the same GPL license. Basically, the GPL creates a consortium; anyone can use the program, but you can't change the program or use its code in another program and make the results proprietary. Since the GPL is a legal document, it can be hard for some to understand. Here is one less legal summary ([posted on Slashdot](#)):

This software contains the intellectual property of several people. Intellectual property is a valuable resource, and you cannot expect to be able to use someone else's intellectual property in your own work for free. Many businesses and individuals are willing to trade their intellectual property in exchange for something of value; usually money. For example, in return for a sum of money, you might be granted the right to incorporate code from someone's software program into your own.

The developers of this software are willing to trade you the right to use their intellectual property in exchange for something of value. However, instead of money, the developers are willing to trade you the right to freely incorporate their code into your software in exchange for the right to freely incorporate your code [which incorporates their code] into theirs. This exchange is to be done by way of and under the terms of the GPL. If you do not think that this is a fair bargain, you are free to decline and to develop your own code or purchase it from someone else. You will still be allowed to use the software, which is awfully nice of the developers, since you probably didn't pay them a penny for it in the first place.

Microsoft complains that the GPL does not allow them to take such code and make changes that it can keep proprietary, but this is hypocritical. Microsoft doesn't allow others to make and distribute changes to Microsoft software *at all*, so the GPL grants far *more* rights to customers than Microsoft does.

In some cases Microsoft will release source code under its "shared source" license, but that license (which is not OSS/FS) is far more restrictive. For example, it prohibits distributing software in source or object form for commercial purposes under any circumstances. Examining Microsoft's shared source license also shows that it has even more stringent restrictions on intellectual property rights. For example, it states that "if you sue anyone over patents that you think may apply to the Software for a person's use of the Software, your license to the Software ends automatically," and "the patent rights Microsoft is licensing only apply to the Software, not to any derivatives you make." [A longer analysis of this license and the problems it causes developers is provided by Bernhard Rosenkraenzer \(bero\)](#). The FSF has also posted a press release on why they believe the [GPL protects software freedoms](#).

It's true that organizations that modify and release GPL'ed software must yield any patent and copyright rights for those additions they release, but such organizations do so voluntarily (no one can *force* anyone to modify GPL code) and with full knowledge (all GPL'ed software comes with a license clearly stating this). And such grants only apply to those modifications; organizations can hold other unrelated rights if they wish to do so, or develop their own software instead. Since organizations can't make such changes at all to proprietary software in most circumstances, and generally can't redistribute changes in the few cases where they *can* make changes, this is a fair exchange, and organizations get far more rights with the GPL than with proprietary licenses (including the "shared source" license). If organizations don't like the GPL license, they can always create their own code, which was the only option even before GPL'ed code became available.

Although the GPL is sometimes called a "virus" by proprietary vendors (particularly by Microsoft) due to the way it encourages others to also use the GPL license, it's only fair to note that many proprietary products and licenses also have virus-like effects. Many proprietary products with proprietary data formats or protocols have "network effects," that is, once many

users begin to use that product, that group puts others who don't use the same product at a disadvantage. For example, once some users pick a particular product such as a proprietary OS or word processor, it becomes increasingly difficult for other users to use a different product. Over time this enforced use of a particular proprietary product also spreads like a virus.

Certainly many technologists and companies don't think that the GPL will destroy their businesses. Many seem too busy mocking Microsoft's claims instead (for an example, see [John Lettice's June 2001 article "Gates: GPL will eat your economy, but BSD's cool"](#)). After all, [Microsoft sells a product with GPL'ed components](#), and still manages to hold intellectual property (see below).

Perhaps Microsoft means the GPL "destroys" intellectual property because the owners of competing software may be driven out of business. If so, this is hypocritical; Microsoft has driven many companies out of business, or bought them up at fractions of their original price. Indeed, sometimes the techniques that Microsoft used have later been proven in court to be illegal. In contrast, there is excellent evidence that [the GPL is on very solid legal ground](#). "Destruction" of one organization by another through legal competition is quite normal in capitalistic economies.

The GPL does not "destroy" intellectual property; instead, it creates a level playing field where people can contribute improvements voluntarily to a common project without having them "stolen" by others. You could think of the GPL as creating a consortium; no one is required to aid the consortium, but those who do must play by its rules. The various motivations for joining the consortium vary considerably (see the article [License to FUD](#)), but that's true for any other consortium too. It's understandable that Microsoft would want to take this consortium's results and take sole ownership of derivative works, but there's no reason to believe that a world where the GPL cannot be used is really in consumers' best interests.

9. **Is there really a lot of OSS/FS software? Yes.** Freshmeat.net counts over 21,000 software branches of OSS/FS software as of October 2002. Sourceforge.net hosts 55,424 OSS/FS projects all by itself (as of January 28, 2003). [The dmoz list of just OS counts 114 OSS/FS OSes](#); this includes old systems (re-enabling their support), experiments, and specialized projects. There's little reason to believe that this counts *all* OSS/FS software, but it certainly indicates there's a large amount of it. These projects vary in value and quality, of course, just as proprietary programs do, but all of these OSS/FS projects can be the basis of future work.
10. **Is having the ability to view and change source code really valuable/important for many people? Surprisingly, yes.** It's certainly true that few people need *direct* access to source code; only developers or code reviewers need the ability to access and change code. But not having access to how your computer is controlled is still a significant problem. Bob Young of Red Hat uses the analogy of [having your car's hood welded shut](#) to explain why even non-technical users need access to the source code. Here is his explanation, in his own words:

Open source gives the user the benefit of control over the technology the user is investing in... The best analogy that illustrates this benefit is with the way we buy cars. Just ask the question, “Would you buy a car with the hood welded shut?” and we all answer an emphatic “No.” So ask the follow-up question, “What do you know about modern internal-combustion engines?” and the answer for most of us is, “Not much.”

We demand the ability to open the hood of our cars because it gives us, the consumer, control over the product we’ve bought and takes it away from the vendor. We can take the car back to the dealer; if he does a good job, doesn’t overcharge us and adds the features we need, we may keep taking it back to that dealer. But if he overcharges us, won’t fix the problem we are having or refuses to install that musical horn we always wanted -- well, there are 10,000 other car-repair companies that would be happy to have our business.

In the proprietary software business, the customer has no control over the technology he is building his business around. If his vendor overcharges him, refuses to fix the bug that causes his system to crash or chooses not to introduce the feature that the customer needs, the customer has no choice. This lack of control results in high cost, low reliability and lots of frustration.

To developers, source code is critical. Source code isn’t necessary to break the security of most systems, but to really fix problems or add new features it’s quite difficult without it. Microsoft’s Bill Gates has often claimed that most developers don’t need access to OS source code, but [Graham Lea’s article “Bill Gates’ roots in the trashcans of history”](#) exposes that Gates actually extracted OS source code himself from other companies by digging through their trash cans. Mr. Gates said, “I’d skip out on athletics and go down to this computer center. We were moving ahead very rapidly: Basic, FORTRAN, LISP, PDP-10 machine language, digging out the OS listings from the trash and studying those.” If source code access isn’t needed by developers, why did *he* need it?

See also the discussion on the [greater flexibility](#) of OSS/FS.

11. **Is OSS/FS really just an anti-Microsoft campaign? No.** Certainly there are people who support OSS/FS who are also against Microsoft, but it’d be a mistake to view OSS/FS as simply anti-Microsoft. Microsoft already uses OSS/FS software in its own applications; Windows’ implementation of the basic Internet protocols (TCP/IP) was derived from OSS/FS code, and its Office suite depends on the OSS/FS compression library “zlib.” Microsoft could, at any time, release programs such as its OSes as OSS/FS, take an existing OSS/FS OS and release it, or provide applications for OSS/FS systems. There is no licensing agreement that prevents this. Indeed, OSS/FS leaders often note that they are not against Microsoft per se, just some of its

current business practices, and many have repeatedly asked Microsoft to join them (e.g., see [Free Software Leaders Stand Together](#)).

In many cases OSS/FS is developed with and for Microsoft technology. On June 21, 2002, [SourceForge listed](#) 831 projects that use Visual Basic (a Microsoft proprietary technology) and 241 using C# (a language that originated from Microsoft). [A whopping 8867 projects are listed as working in Windows](#). This strongly suggests that there are many OSS/FS developers who are not “anti-Microsoft.”

Microsoft says it's primarily opposed to the GPL, *but Microsoft sells a product with GPL'ed components*. [Microsoft's Windows Services for Unix](#) includes Interix, an environment which can run UNIX-based applications and scripts on the Window NT and Windows 2000 OSes. There's nothing wrong with this; clearly, there are a lot of Unix applications, and since Microsoft wants to sell its OSes, Microsoft decided to sell a way to run Unix applications on its own products. But many of the components of Interix are covered by the GPL, such as gcc and g++ (for compiling C and C++ programs). (Microsoft seems to keep moving information about this; [here is a stable copy](#)). The problem is not what Microsoft is doing; as far as I can tell, they're following both the letter and the spirit of the law in this product. The problem is that Microsoft says no one should use the GPL, and that no one can make money using the GPL, while simultaneously making money using the GPL. Bradley Kuhn (of the FSF) bluntly said, “It's hypocritical for them to benefit from GPL software and criticize it at the same time.” Microsoft executives are certainly aware of this use of the GPL; Microsoft Senior Vice President Craig Mundie specifically acknowledged this use of GPL software when he was questioned on it. Kelly McNeill noted this dichotomy between claims and actions in the June 22, 2001 story ["Microsoft Exposed with GPL'd Software!"](#) [A more detailed description about this use of the GPL by Microsoft is given in The Standard on June 27, 2001](#). Perhaps in the future Microsoft will try to remove many of these GPL'ed components so that this embarrassing state of affairs won't continue. But even if these components are removed in the future, this doesn't change the fact that Microsoft has managed to sell products that include GPL-covered code without losing any of its own intellectual property rights.

That being said, there are certainly many people who are encouraging specific OSS/FS products (such as Linux) so that there will be a viable competition to Microsoft, or who are using the existence of a competitor to obtain the best deal from Microsoft for their organization. This is nothing unusual - customers *want* to have competition for their business, and they usually have it in most other areas of business. Certainly there is a thriving competing market for computer hardware, which has resulted in many advantages for customers. [The New York Times'](#) position is that “More than two dozen countries - including Germany and China - have begun to encourage governmental agencies to use such “open source” software ... Government units abroad and in the United States and individual computer users should look for ways to support Linux and Linux-based products. The competition it offers helps everyone.”

12. **I've always assumed there's no free lunch; isn't there some catch?** If there is an OSS/FS product that meets your needs, there really isn't a catch. Perhaps the only catch is misunderstanding the term "free." The GPL includes this (haiku) text: "When we speak of free software, we are referring to freedom, not price." I.E., OSS/FS is not necessarily cost-free. In practice, it's still often a bargain.

Naturally, if you want services besides the software itself (such as guaranteed support, training, and so on), you must pay for those things just like you would for proprietary software. If you want to affect the future direction of the software - especially if you must have the software changed in some way to fit it to your needs - then you must invest to create those specific modifications. Typically these investments involve hiring someone to make those changes, possibly sharing the cost with others who also need the change. Note that you only need to pay to change the software - you don't need to pay for permission to use the software, or a per-copy fee, only the actual cost of the changes.

For example, when IBM wanted to join the Apache group, IBM discovered there really was no mechanism to pay in money. IBM soon realized that the primary "currency" in OSS/FS is software code, so IBM turned the money into code and all turned out very well.

This also leads to interesting effects that explains why many OSS/FS projects start small for years, then suddenly leap into a mode where they have a rapidly increasing functionality and user size. For any application, there is a minimum level of acceptable functionality; below this, there will be very few users. If that minimum level is large enough, this creates an effect similar to an "energy barrier" in physics; the barrier can be large enough that most users are not willing to pay for the initial development of the project. However, at some point, someone may decide to begin the "hopeless" project anyway. The initial work may take a while, because the initial work is large and there are few who will help. However, once a minimum level of functionality is reached, a few users will start to use it, and a few of them may be willing to help (e.g., because they want the project to succeed or because they have specialized needs). At some point in this growth, it is like passing an energy barrier; the process begins to become self-sustaining and exponentially increasing. As the functionality increases, the number of potential users begins to increase rapidly, until suddenly the project is sufficiently usable for many users. A percentage of the userbase will decide to add new features, and as the userbase grows, so do the number of developers. As this repeats, there is an explosion in the program's capabilities.

10. OSS/FS on the desktop: Client computing

OSS/FS programs have been competing for many years in the server market, and are now well-established in that market. OSS/FS programs have been competing for several years in the embedded markets, and have already begun to significantly penetrate those markets as well.

In contrast, OSS/FS programs currently have only a small client (desktop and laptop) market share. This is unsurprising; OSS/FS only began to become viable for client computing in 2002, and it takes time for any software to mature, be evaluated, and be deployed. Since OSS/FS is a brand new contender in the client market, it has only begun penetrating into that market. However, there are reasons to think that OSS/FS use on client systems will grow significantly in the future.

A few definitions are necessary first, before examining the issue in more depth. Many users' only direct experience with computers is through their desktop or laptop computers running "basic client applications" such as a web browser, email reader, word processor, spreadsheet, and presentation software (the last three together are often called an "office suite"), possibly with additional client applications, and all of these must have a graphical user interface and be supported by an underlying graphical environment. Such computers are often called "client" computers (even if they are not using the technical approach called the "client-server model"). Another term also used is the "desktop", even if the computer is not on a desk.

However, the small market share should not be surprising, because viable OSS/FS client applications only became available in 2002. As a practical matter, client systems must be compatible with the market leader, for example, the office suite must be able to read and write documents in the Microsoft Office formats. Before 2002 the available OSS/FS products could not do this well, and thus were unsuitable for most circumstances. Clearly, OSS/FS client applications cannot be considered unless they are already available.

One point less understood is that OSS/FS operating systems (like GNU/Linux) could not really compete with proprietary operating systems on the client until OSS/FS basic client applications and environment were available. Clearly, few users can even consider buying a client system without basic client applications, since that system won't meet their fundamental requirements. There have been proprietary basic client applications for GNU/Linux for several years, but they didn't really make GNU/Linux viable for client applications. The reason is that a GNU/Linux system combined with proprietary basic client applications still lacks the freedoms and low cost of purely OSS/FS systems, and the combination of GNU/Linux plus proprietary client applications has to compete with established proprietary systems which have many more applications available to them. This doesn't mean that GNU/Linux can't support proprietary programs; certainly some people *will* buy proprietary basic client applications, and many people have already decided to buy many other kinds of proprietary applications and run them on a GNU/Linux system. However, few will find that a GNU/Linux system with proprietary basic client applications has an advantage over its competition. After all, the result is still proprietary, and since there are fewer desktop applications of any kind on GNU/Linux, many capabilities have been lost, little has been gained, and the switching costs will dwarf those minute gains. There is also the problem of transition. Many organizations will find it too traumatic to immediately switch all client systems to an OSS/FS operating system; it is often much easier to slowly switch to OSS/FS basic client applications on the pre-existing proprietary operating system, and then switch operating systems once users are familiar with the basic client applications. Thus, the recent availability of OSS/FS basic client applications has suddenly made OSS/FS operating systems (like GNU/Linux) far more viable on the client.

First, let's look at the available market share figures. According to [the June 2000 IDC survey](#) of 1999 licenses for client machines, GNU/Linux had 80% as many client shipments in 1999 as Apple's MacOS (5.0% for Mac OS, 4.1% for GNU/Linux). More recent figures in 2002 suggest that GNU/Linux has [1.7%](#) of the client OS market. Clearly, the market share is small at this early stage. Obviously, while this shows that there are many users (because there are so many client systems), this is still small compared to [Microsoft's effective monopoly on the client OS market](#). [IDC reported that Windows systems \(when they are all combined\) accounted for 92% of the client operating systems sold](#).

However, there are many factors that suggest that the situation may change radically in the future: OSS/FS basic client software is now available, there's increasing evidence of their effectiveness, Microsoft is raising prices, and governments want open systems:

1. *OSS/FS basic client software is available.* Back in 1997 I forecast that GNU/Linux would be "ready for the desktop" in 2002-2003 (5 years later). My forecast appears correct; OSS/FS applications and environments matured in 2002 where they are finally functionally competitive on the client. In 2002, Mozilla finally released version 1.0 of their suite (including a web browser, email reader, and other tools), and the first reasonably usable version of Open Office, the first practically useful OSS/FS office suite, was released in 2002 as well. Desktop environments matured as well; in 2002 both the GNOME and KDE projects released capable, more mature versions of their desktop environments. In addition the WINE product (a product that allows OSS/FS systems to run Windows programs) was finally able to run Microsoft Office 97, suggesting that although WINE is still immature, it may be sufficient to run some Windows applications developed internally by some organizations.

There are other plausible alternatives for client applications as well, such as Evolution (an excellent mail reader), Abiword (a lighter-weight but less capable word processor which also released its version 1.0 in 2002), Gnumeric (a spreadsheet), and KOffice (an office suite).

However, I will emphasize Mozilla and Open Office, for two reasons. First, they also run on Microsoft Windows, which makes it much easier to transition users from competitors (this enables users to migrate a step at a time, instead of making one massive change). Second, they are full-featured, including compatibility with Microsoft's products; many users want to use fully-featured products since they don't want to switch programs just to get a certain feature. In short, it looks like there are now several OSS/FS products that have begun to rival their proprietary competitors in both usability and in the functionality that people need, including some very capable programs.

2. *There is increasing evidence of OSS/FS client software effectiveness.* The [MOXIE](#) study of January 2003 randomly acquired 100 documents from the Internet in the Microsoft Office word processor, spreadsheet, and presentation software formats. Their leading OSS/FS contender, Open Office version 1.0.1, did well; it was able to successfully use 97%, 98%, and 94% of the documents (of the respective formats). The study concluded that "the current state of

interoperability is reasonably good, although there is significant room for improvement.” Since that time, the Open Office developers have specifically worked to improve interoperability with Microsoft Office, and it’s reasonable to expect that the figures are significantly higher now.

3. *Microsoft is raising prices.* Microsoft is changing many of its practices, resulting in increasing costs to its customers. It has changed its licensing so that one copy of Windows cannot be used for both home and office. Microsoft has switched its largest customers to a subscription-based approach (called “Licensing 6”), greatly increasing the costs to its customers. [TIC/Sunbelt Software Microsoft Licensing Survey Results \(covering March 2002\)](#) reports the impact on customers of this new licensing scheme. 80% had a negative view of the new licensing scheme, noting, for example, that the new costs for software assurance (25% of list for server and 29% of list for clients) are the highest in the industry. Of those who had done a cost analysis, an overwhelming 90% say their costs will increase if they migrate to 6.0, and 76% said their costs would increase from 20% to 300% from what they are paying now under their current 4.0 and 5.0 Microsoft Licensing plans. Indeed, 38% of those surveyed said that they are actively seeking alternatives to Microsoft products. [Licensing 6.0 can also significantly harm organizations trying to sell off a part of its operations.](#) The program requires accelerated software maintenance payments when the computers that are covered under the license are sold off - but Microsoft is no longer obligated to provide maintenance even if the contract is fully paid.

[Gartner’s review of Star Office](#) (Sun’s variant of Open Office) also noted that Microsoft’s recent licensing policies may accelerate moving away from Microsoft. As Gartner notes, “This [new license program] has engendered a lot of resentment among Microsoft’s customers, and Gartner has experienced a marked increase in the number of clients inquiring about alternatives to Microsoft’s Office suite... enterprises are realizing that the majority of their users are consumers or light producers of information, and that these users do not require all of the advanced features of each new version of Office... unless Microsoft makes significant concessions in its new office licensing policies, Sun’s StarOffice will gain at least 10 percent market share at the expense of Microsoft Office by year-end 2004 (0.6 probability).” They also note that “Because of these licensing policies, by year-end 2003, more than 50 percent of enterprises will have an official strategy that mixes versions of office automation products - i.e., between multiple Microsoft Office versions or vendor products (0.7 probability).”

4. *Governments want open systems.* [A New York Times article](#) noted that “More than two dozen countries in Asia, Europe and Latin America, including China and Germany, are now encouraging their government agencies to use ‘open source’ software - developed by communities of programmers who distribute the code without charge and donate their labor to cooperatively debug, modify and otherwise improve the software.” [Sun Microsystems has announced a deal with China to provide one million Linux desktops](#), and mentioned that China “has pledged to deploy 200 million copies of open standards-based desktop software”.

Indeed, the advantages of OSS/FS to governments are clear, especially to non-U.S. governments.

No government wants their computing infrastructure controlled by one company (and outside the U.S., a foreign company at that). Jiang Guangzhi, director of a software development center in Shanghai, emphasized that the Chinese government did not want one company “to manipulate or dominate the Chinese market.” [IBM signed a Linux deal with Germany](#); Germany’s Interior Minister, Otto Schilly, said the move would help cut costs, improve security in the nation’s computer networks, and lower dependence on any one supplier. [Ralph Nader’s Consumer Project on Technology](#) gives reasons the U.S. government should encourage OSS/FS. Many countries favor or are considering favoring OSS/FS in some way, such as [Peru](#), [the UK](#), and [Taiwan](#). [An older but broad survey was published in 2001 by CNet](#).

[South Africa’s government departments are being officially encouraged to stop using \(expensive\) proprietary software, and to OSS/FS instead](#). This is according to a [January 15, 2003 announcement by Mojalefa Moseki](#), chief information officer with the State Information Technology Agency (Sita). South Africa plans to save 3 billion Rands a year (approximately \$338 million USD), increase spending on software that stays in their country, and increase programming skill inside the country. South Africa reports that its small-scale introductions have already saved them 10 million Rands (approximately \$1.1 million USD). [More information is available at Tectonic](#) (see also [South African minister outlines OSS plans](#)). [The state of Oregon is considering an OSS/FS bill as well](#). [Japan has earmarked 1 billion yen for a project to boost operating systems other than Microsoft Windows](#) - it is expected to be based on OSS/FS, particularly Linux, and both South Korea and China are coordinating with Japan on it. [In December 2003, Israel’s government suspended purchases of new versions of Microsoft office software](#) and began actively encouraging the development of an open-source alternatives (especially Open Office).

[Tony Stanco’s presentation “On Open Source Procurement Policies”](#) briefly describes why he believes governments should consider OSS/FS. [A NASA technical report](#) describes in detail an approach for NASA to release some of its software as open source software.

Indeed, so many governments have begun enacting preferences for OSS/FS or officially requiring that OSS/FS options be considered that Microsoft has sponsored an organization called the [Initiative for Software Choice](#). This organization makes many nice-sounding statements. However, many observers believe the real purpose of this organization is to prevent governments from considering the advantages or disadvantages of a software license when they procure software, to prevent governments from requiring consideration of OSS/FS products, and to encourage the use of standards that inhibit the use of OSS/FS. An opposing group, founded by Bruce Perens, is [Sincere Choice.org](#), which advocates that there be a “fair, competitive market for computer software, both proprietary and Open Source.” [Bruce Perens has published an article discussing why he believes “Software Choice” is not what it first appears to be](#).

There are some interesting hints that GNU/Linux is already starting to gain on the client. Some organizations, such as [TrustCommerce](#) and the [city of Largo, Florida](#), report that they've successfully transitioned to using Linux on the desktop.

There's already some evidence that others anticipate this; [Richard Thwaite, director of IT for Ford Europe, stated in 2001 that an open source desktop is their goal, and that they expect the industry to eventually go there](#) (he controls 33,000 desktops, so this would not be a trivial move). It could be argued that this is just a ploy for negotiation with Microsoft - but such ploys only work if they're credible.

There are other sources of information on OSS/FS or GNU/Linux for clients. [Desktoplinux.com](#) is a web site devoted to the use of GNU/Linux on the desktop; they state that "We believe Linux is ready *now* for widespread use as a desktop OS, and we have created this website to help spread the word and accelerate the transition to a more open desktop, one that offers greater freedom and choice for both personal and business users."

Bart Decrem's [Desktop Linux Technology & Market Overview](#), funded by Mitch Kapor, gives a detailed analysis and prognostication of GNU/Linux on the desktop. [Paul Murphy discusses transitioning large companies to Linux and Intel \("Lintel"\) on the desktop](#), and concludes that one of the biggest risks is trying to copy a Windows architecture instead of exploiting the different capabilities GNU/Linux offers.

Indeed, it appears that many users are considering such a transition. [ZDNet published survey results on August 22, 2002](#), which asked "Would your company switch its desktop PCs from Windows to Linux if Windows apps could run on Linux?" Of the more than 15,000 respondents, 58% said they'd switch immediately; another 25% said they'd consider dumping Windows in favor of Linux within a year. While all such surveys must be taken with a grain of salt, still, these are not the kind of responses you would see from users happy with their current situation. They also noted that ZDNet Australia found that 55% of the surveyed IT managers were considering switching from Microsoft products. Most people do not expect that this transition, if it happens, will happen quickly: it is difficult to change that many systems. But the fact that it's being considered at all is very intriguing.

11. Usage Reports

There are many reports from various users who have switched to OSS/FS; here are a sample that you may find useful. This is *not* an exhaustive list, nor can it be.

As discussed earlier, [the City of Largo, Florida](#) supports 900 city employees using GNU/Linux, saving about \$1 million a year. A [BusinessWeek online article](#) notes that Mindbridge shifted their 300-employee intranet software company from Microsoft server products and Sun Solaris to GNU/Linux; after experiencing a few minor glitches, their Chief Operating Officer and founder Scott Testa says they

now couldn't be happier, and summarizes that "...we're saving hundreds of thousands of dollars between support contracts, upgrade contracts, and hardware." [Amazon.com](#) saved millions of dollars by switching to GNU/Linux. Oracle's Chairman and CEO, Larry Ellison, said that [Oracle will switch to GNU/Linux to run the bulk of its business applications](#) no later than summer 2002, replacing three Unix servers. [A travel application service provider](#) saved \$170,000 in software costs during the first six months of using GNU/Linux (for both servers and the desktop); it also saved on hardware and reported that administration is cheaper too. [CRN's Test Center](#) found that a GNU/Linux-based network (with a server and 5 workstations) cost 93% less in software than a Windows-based network, and found it to be quite capable. The article [Linux as a Replacement for Windows 2000](#) determined that "Red Hat Linux 7.1 can be used as an alternative to Windows 2000... You will be stunned by the bang for the buck that Linux bundled free 'open source' software offers."

Educational organizations have found OSS/FS software useful. The [K12 Linux Terminal Server Project](#) has set up many computer labs in the U.S. Northwest in elementary, middle, and high schools. For example, [St. Mary's School](#) is a 450-student Pre-K through 8th grade school in Rockledge, Florida that applying GNU/Linux using their approach. Their examples show that kids don't find GNU/Linux that hard to use and quite able to support educational goals. For example, third graders put together simple web pages about their favorite Saints using a variety of OSS/FS programs: they logged into GNU/Linux systems, typed the initial content using Mozilla Composer (an OSS/FS web page editor), drew pictures of the Saints using The Gimp (an OSS/FS drawing program), and shared the results with Windows users using Samba. The page [Why should open source software be used in schools?](#) gives various examples of educational organizations who have used OSS/FS programs, as well as linking to various general documents on why educational organizations should use OSS/FS. The [letter from the Kochi Free Software Users' Group to the Government of Kerala and others](#) also summarizes some of the issues, especially why governments should specify standards (and not products) for educational use. The Faculty Senate of the University at Buffalo, State University of New York, approved [a resolution strongly supporting the use of OSS/FS instead of proprietary software](#).

Many financial organizations use OSS/FS. [Online brokerage E*Trade is moving its computer systems to IBM servers running GNU/Linux](#), citing cost savings and performance as reasons for switching to GNU/Linux (the same article also notes that clothing retailer L.L. Bean and financial services giant Salomon Smith Barney are switching to GNU/Linux as well). [Merrill Lynch](#) is switching to GNU/Linux company-wide, and are hoping to save tens of millions of dollars annually within three to five years. [Adam Wiggins reports on TrustCommerce's successful transition to Linux on the desktop](#). [An April 22, 2002 report on ZDNet, titled "More foreign banks switching to Linux"](#), stated that New Zealand's TSB bank "has become the latest institution to adopt the open-source Linux OS. According to reports, the bank is to move all its branches to the Linux platform... in Europe, BP and Banca Commerciale Italiana feature among the big companies that have moved to Linux. According to IBM, as many as 15 banks in central London are running Linux clusters." They also mentioned that "Korean Air, which now does all its ticketing on Linux, and motorhome manufacturer Winnebago, are high-profile examples." [The Federal](#)

[Aviation Air Traffic Control System Command Center in Herndon, Virginia](#) is currently installing a system to support 2,000 concurrent users on Red Hat Linux. The system, known as the National Log, will act as a central clearinghouse database for users in air traffic centers across the country.

[ComputerWorld reported in October 2002 an increasing use of GNU/Linux on Wall Street](#) - Merrill Lynch reports that a majority of new projects are interested in GNU/Linux, for example, and the article references a TowerGroup (of Needham, MA) estimate that GNU/Linux is currently deployed on 7% of all servers in North American brokerage firms. TowerGroup also forecasts that GNU/Linux use will grow at an annual rate of 22% in the securities server market between 2002 and 2005, outpacing growth in Windows 2000, NT and Unix deployments.

Some organizations are deploying GNU/Linux widely at the point of sale. Many retailer cash registers are switching to GNU/Linux, according to Information Week ("Cash Registers are Ringing up Sales with Linux" by Dan Orzech, December 4, 2000, Issue 815); on September 26, 2002, [The Economist noted that "Linux is fast catching on among retailers."](#) According to Bob Young (founder of Red Hat), [BP \(the petroleum company\) is putting 3,000 Linux servers at gas stations.](#) [Zumiez](#) is installing open-source software on the PCs at all its retail locations, and expects that this will cut its technology budget between \$250,000 and \$500,000 a year; note that this includes using Evolution for email, Mozilla for web browsing (to eliminate the need for printed brochures and training manuals), and an open source spreadsheet program. [Sherwin-Williams](#), the number one U.S. paint maker, plans to convert its computers and cash registers (not including back office support systems) in over 2,500 stores to GNU/Linux and has hired IBM to do the job; this effort involves 9,700 NetVista desktop personal computers,

OSS/FS is also prominent in Hollywood. Back in 1996, when GNU/Linux was considered by some to be a risk, [Digital Domain used GNU/Linux to generate many images in Titanic](#). After that, it burst into prominence as many others began using it, so much so that a [February 2002 article in IEEE Computer](#) stated that "it is making rapid progress toward becoming the dominant OS in ... motion pictures." "Shrek" and "Lord of the Rings" used GNU/Linux to power their server farms, and now [DreamWorks SKG has switched to using GNU/Linux exclusively on both the front and back ends for rendering its movies.](#) [Industrial Light & Magic](#) converted its workstations and renderfarm to Linux in 2001 while it was working on Star Wars Episode II. They stated that "We thought converting to Linux would be a lot harder than it was" (from their SGI IRIX machines). They also found that the Linux systems are 5 times faster than their old machines, enabling them to produce much higher quality results. They also use Python extensively (an OSS/FS language), as well as a number of in-house and proprietary tools. [Disney is also shifting to GNU/Linux](#) for film animation.

Many remote imaging systems use GNU/Linux. When a remote imaging system was placed at the North Pole, reporters noted that the Linux mascot was a penguin and announced that [Penguins invade the North Pole.](#)

There are many large-scale systems. [In October 2002, Chrysler Group announced it's using a Linux cluster computer for crash simulation testing and analysis](#) in an effort to make safer cars and trucks.

Their configuration uses 108 workstations, each with 2 processors, so the system uses 216 computers all running Red Hat Linux, and expect to improve simulation performance by 20% while saving about 40% in costs.

OSS/FS is widely used by Internet-based companies. [Google uses over 6,000 GNU/Linux servers.](#) [Yahoo! is increasing its already-massive use of OSS/FS.](#) Yahoo claims it is the “World’s most trafficked Internet destination,” justified based on Nielsen/NetRatings of August 2002. Yahoo had 201 million unique users, 93 million active registered users, over 4500 servers, and over 1.5 billion pageviews a day. Yahoo noted that OSS/FS already runs their business (e.g., Perl, Apache, FreeBSD, and gcc), and they’ve recently decided to move from their proprietary in-house languages to PHP (an OSS/FS language). [Afilias has switched the registration database for the .org Internet domain](#) from the proprietary Oracle to the OSS/FS PostgreSQL database program; .org is the fifth largest top-level domain, with more than 2.4 million registered domain names.

[Bloor Research announced in November 2002 that they believe GNU/Linux is ready to support large enterprise applications](#) (i.e., it’s “enterprise ready”). They reached this conclusion after examining its scalability, availability, reliability, security, manageability, flexibility, and server consolidation characteristics. They concluded that “Linux now scales well on Intel hardware, and by taking advantage of failover extensions from Linux distributors and Grid suppliers, high availability can be achieved. Linux is proven to be reliable, especially for dedicated applications, and its open source nature ensures that it is at least as secure as its rivals.” Only 3 years earlier Bloor had said GNU/Linux wasn’t ready.

The U.S. government has been using OSS and many have suggested more use. The (U.S.) President’s Information Technology Advisory Committee (PITAC)’s report, the [Recommendations of the Panel on Open Source Software For High End Computing](#), recommends that the U.S. “Federal government should encourage the development of open source software as an alternate path for software development for high end computing.” See the separate discussion on [MITRE Corporation’s business case study of OSS](#) (which emphasized use by the U.S. government, especially the U.S. military). The U.S. National Imagery and Mapping Agency (NIMA) National Technical Alliance, through the National Center for Applied Technology (NCAT) consortium, funded the Open Source Prototype Research (OSPR) project. Under the OSPR project ImageLinks Inc., Tybrin Inc., Kodak Inc., and Florida Institute of Technology (Florida Tech) performed evaluations of open source software development practices and demonstrated the technological advantages of Open Source Software. The [OSPR final report](#) includes those evaluations, a survey, and various related documents; these are actually rather extensive. The final report concludes:

Open Source Software development is a paradigm shift and has enormous potential for addressing government needs. Substantial technology leverage and cost savings can be achieved with this approach. The primary challenge will be in establishing an organizational structure that is able to employ OSS methodology...

The paper [Open Source and These United States](#) by C. Justin Seiferth summarizes that:

The Department of Defense can realize significant gains by the formal adoption, support and use of open licensed systems. We can lower costs and improve the quality of our systems and the speed at which they are developed. Open Licensing can improve the morale and retention of Airmen and improve our ability to defend the nation. These benefits are accessible at any point in the acquisition cycle and even benefit deployed and operational systems. Open Licensing can reduce acquisition, development, maintenance and support costs and increased interoperability among our own systems and those of our Allies.

NetAction has proposed more OSS/FS use and encouragement by the government; see [The Origins and Future of Open Source Software](#) by Nathan Newman and [The Case for Government Promotion of Open Source Software](#) by Mitch Stoltz for their arguments.

More recently, [The U.S. Department of Defense Information Systems Agency \(DISA\) has certified Linux distributor Red Hat's Advanced Server operating system as a "Common Operating Environment" \(COE\)](#), meaning the server product meets the agency's software security and interoperability specification.

U.S. state governments have widely used OSS/FS too. The Center for Digital Government's 2003 "Best of the Web" awards named the top 5 state web sites as Utah, Maine, Indiana, Washington, and Arkansas. [Four of the five winning state web sites use OSS/FS programs to implement their site](#). The only state in the top five not using OSS/FS was Washington - Microsoft's home state.

There have been some efforts to stifle government use of OSS/FS, or at least forbid use of the most common OSS/FS license, the GPL, but without much success. As described in ["Geek activism" forces Congress to reconsider Open Source](#), a letter from the U.S. Congress unrelated to OSS/FS was modified by Representative Adam Smith from Washington state. Smith's largest campaign donation source is Microsoft Corporation. The modifications added statements strongly discouraging the use of the GPL. The letter was originally signed by 67 Congressmen, but as [an Associated Press piece notes](#), "Smith's attack on open-source drew an angry response from one of the original authors of the letter, Rep. Tom Davis, R-Va., chairman of the House Government Reform subcommittee on technology and procurement policy. "We had no knowledge about that letter that twisted this position into a debate over the open source GPL issues," said Melissa Wojciak, staff director of the subcommittee. Wojciak added that Davis supports government funding of open-source projects." At the end, "Many staffers of the 67 Congressman who signed are now claiming they didn't know what they were signing and the letter has been withdrawn." [Information Week also picked up the story](#).

Such benefits have not escaped the eyes of other governments. [Germany intends to increase its use](#). The [Korean government](#) announced that it plans to buy 120,000 copies of Hancom Linux Deluxe this year,

enough to switch 23% of its installed base of Microsoft users to open source equivalents; by standardizing on GNU/Linux and HancomOffice, the Korean government expects savings of 80% compared with buying Microsoft products (HancomOffice isn't OSS/FS, but GNU/Linux is). [Taiwan is starting a national plan to jump-start the development and use of OSS/FS](#). A [Linux Journal](#) article notes many interesting international experiments and approaches, for example, Pakistan plans to install 50,000 low cost computers in schools and colleges all over Pakistan using GNU/Linux. [Finnish MPs are encouraging the use of GNU/Linux](#) in government systems. [A June 14, 2002 article in PC World](#) also lists actions various governments are taking. Statskontoret, the Swedish Agency for Public Management, has performed a feasibility study on free and open source software and came to very positive conclusions (see the report in [English](#) or [Swedish](#)).

The Interchange of Data between Administrations (IDA) programme is managed by the European Commission, with a mission to "coordinate the establishment of Trans-European telematic networks between administrations." IDA has developed [The IDA Open Source Migration Guidelines](#) to describe how to migrate from proprietary programs to OSS/FS programs. The authors state that "There are many reasons for Administrations to migrate to OSS. These include: the need for open standards for e-Government; the level of security that OSS provides; the elimination of forced change; the cost of OSS. All these benefits result in far lower [Information Technology] costs."

[As reported in the Washington Post on November 3, 2002](#), Luis Millan Vazquez de Miguel, the minister of education, science and technology in a western region of Spain called Extremadura, is heading the launch of a government campaign to convert all the area's computer systems (in government offices, businesses and homes) from the Windows operating system to GNU/Linux. Vazquez de Miguel said over 10,000 desktop machines have already been switched, with 100,000 more scheduled for conversion in the next year. The regional government paid a local company \$180,000 to create a set of freely available software, and invested in a development center that is creating customized software. "So far, the government has produced 150,000 discs with the software, and it is distributing them in schools, electronics stores, community centers and as inserts in newspapers. It has even taken out TV commercials about the benefits of free software." The Post also discussed some of the reasons some governments are turning to OSS/FS. "Among the touchiest issues that Microsoft faces outside the United States is the uneasiness some countries have expressed about allowing an American company to dominate the software industry in their country. 'Non-U.S. governments in particular view open source as a way to break the stranglehold against Microsoft. If Microsoft owns everything their countries, their own companies can't get a foothold in the software industry,' said Ted Schadler, an analyst for Forrester Research Inc." Some Spanish government systems and those belonging to the telecommunications company Telefonica recently were shifted to Linux partly because of security concerns. In Florence, legislators talked of breaking the 'the computer science subjection of the Italian state to Microsoft.' "

Munich, Germany (the third largest German city) has decided to [migrate all of its 14,000 computers in public administration to GNU/Linux and other OSS/FS office applications, dropping Microsoft's Windows in the process](#). [USA Today gives a detailed discussion of how this decision was made](#). The GNU/Linux system bid had a somewhat higher cost, although Microsoft's bid did have some serious

weaknesses: in the lower-cost bid, the Windows systems wouldn't be upgraded for 6 years (who doesn't upgrade for 6 years?) and many systems would only get the word processor Word (GNU/Linux systems typically come with complete office application suites at no additional cost).

Indeed, according to Robert Kramer of CompTIA (Computer Technology Industry Association), [political leaders everywhere from California to Zambia are considering legislating a preference for Open Source software use](#); he counted at least 70 active proposals for software procurement policies that prefer OSS/FS in 24 countries as of October 2002. There are certainly debates on the value of OSS/FS preferences (even a few OSS/FS advocates like Bruce Perens don't support mandating a government preference for OSS/FS), but clearly this demonstrates significant positive interest in OSS/FS from various governments.

In 2002 an independent study was published by the European Commission. Titled "[Pooling Open Source Software](#)", and financed by the Commission's Interchange of Data between Administrations (IDA) programme, it recommends creating a clearinghouse to which administrations could "donate" software for re-use. This facility would concentrate on applications specific to the needs of the public sector. More specifically, the study suggests that software developed for and owned by public administrations should be issued under an open source license, and states that sharing software developed for administrations could lead to across-the-board improvements in efficiency of the European public sector.

[In October 2002, the European Commission awarded Netproject a pilot contract](#) valued at EUR250,000 to examine deployment of OSS/FS in government departments.

Peru is even contemplating [passing a law requiring the use of OSS/FS for public administration \(government\)](#); rationale for doing so, besides saving money, include supporting "Free access to public information by the citizen, Permanence of public data, and the Security of the State and citizens." Dr. Edgar David Villanueva Nuñez (a Peruvian Congressman) has written an interesting letter supporting this law. [Marc Hedlund written has a brief description of the letter](#); an English translation is available (from [GNU in Peru](#), [UK's "The Register"](#), and [Linux Today](#)); there is a longer discussion of this [available at Slashdot](#). Whether or not this law passes, it is an interesting development.

On October 10, 2002, the [Danish Board of Technology released a report](#) about the economic potential in using Open Source software in the public administration. The report showed a potential savings of 3.7 billion Danish Kroners (500 million Euros) over four years. A pilot project in the Hanstholm municipality determined that switching the office suite from Microsoft Office to OpenOffice.org and StarOffice did not increase their number of problems and that each user only needed 1 to 1.5 hours of training to learn the new office suite. The municipality will now use OpenOffice.org and StarOffice on all workplaces (200 in all) and will save 300,000 Danish Kroners (about 40,000 Euros) each year in license fees. They will still use Microsoft Windows as their OS. [You may want to see the Danish government's report on OSS/FS](#).

There have been many discussions about the advantages of OSS/FS in less developed countries. Heinz and Heinz argue in their paper [Proprietary Software and Less-Developed Countries - The Argentine Case](#) that the way proprietary software is brought to market has deep and perverse negative consequences regarding the chances of growth for less developed countries. Danny Yee's [Free Software as Appropriate Technology](#) argues that Free Software is an appropriate technology for developing countries, using simple but clear analogies. [Free as in Education: Significance of the Free/Libre and Open Source Software for Developing Countries](#), commissioned by the Finnish Ministry for Foreign Affairs, examines the significance of OSS/FS and related concepts; their [FLOSS for Development](#) website identifies other analyses of OSS/FS to support their goal, "To find out if and how Free/Libre and Open Source software is useful for developing countries in their efforts to achieve overall development, including bridging the digital divide."

[Bloomberg's January 14, 2003 article "Microsoft Has New Plan to Share Code With Government"](#) announces that Microsoft Corporation "will expand sharing of the code underlying its Windows programs to help governments and agencies such as Russia and the North Atlantic Treaty Organization (NATO) improve computer security." It notes that "Microsoft is facing competition from the Linux operating system, which lets customers view and modify its source code. In the government sector in particular, Microsoft has lost contracts to Linux, analysts said. More than 20 countries are looking at legislative proposals that mandate considering or using Linux in government computers... [and Microsoft has] begun to make the code available to governments, as well as key customers and partners, in an effort to compete with Linux."

[Librarians](#) have also found many advantages to OSS/FS.

One interesting usage story is the story of [James Burgett's Alameda County Computer Resource Center](#), one of the largest non-profit computer recycling centers in the United States. Its plant processes 200 tons of equipment a month in its 38,000-square-foot warehouse. It has given thousands of refurbished computers to disadvantaged people all over the world, including as human rights organizations in Guatemala, the hard-up Russian space program, schools, and orphanages. All of the machines have GNU/Linux installed on them.

Indeed, for well-established products like GNU/Linux, very strong cases can be made for considering them. On October 18, 2002, [Forrester Research reported that "Linux is now ready for prime time."](#) They stated that "CIOs have many new reasons to be confident that they'll get quality Linux support from their largest application vendors and systems integrators," referencing Amazon, Oracle, Sun, and IBM, among others who have made commitments that increase confidence that GNU/Linux is ready for deployment.

Indeed, these uses are becoming so widespread that [Microsoft admits that OSS/FS competition may force Microsoft to lower its prices](#), at least in the server market. Microsoft noted this in its 10-Q quarterly filing, stating that "To the extent the open source model gains increasing market acceptance,

sales of the company's products may decline, the company may have to reduce the prices it charges for its products, and revenues and operating margins may consequently decline.”

Summaries of government use in various countries are available from [Infoworld](#) and [IDG](#).

Several organizations collect reports of OSS/FS use, and these might be useful sources for more information. [Linux International](#) has a set of [Linux case studies/success stories](#). Mandrakesoft maintains [a site recording the experiences of business users of the Mandrake distribution](#). [Red Hat provides some similar information](#). Opensource.org includes some [case studies](#).

The Dravis Group LLC published in April 2003 [Open Source Software: Case Studies Examining its Use](#), examining several specific use cases in depth. Their study of several different organizations deploying OSS/FS concluded the following:

1. Cost is a significant factor driving adoption of open source software.
2. Control and flexibility are considered benefits as well.
3. Implementation of open solutions is evolutionary, not revolutionary.
4. Open source extends across the entire software stack.
5. Product support is not a significant concern.
6. Open source is not a magic solution.
7. Open standards may be more important than open source.

12. Other Information

Here are some other related information sources:

1. There are several general information sites about OSS/FS or Unix that might be of interest, such as the [Free Software Foundation \(FSF\)](#), the [Open Source Initiative website](#), and the [Linux.org site](#). George Mason University's Exploring and Collecting History Online (ECHO) project has a useful collection in its material on [A Free and Open History of Free and Open Source Software](#). An older paper is [John Kirch's paper, Microsoft Windows NT Server 4.0 versus UNIX](#). ([also archived at the Internet Archives](#)). The paper [Our Open Source / Free Software Future: It's Just a Matter of Time](#) argues that within the next few years, the standard de-facto OS that nearly everyone uses, as well as much of the commodity software in widespread use, will be OSS/FS. The book [The Cathedral and the Bazaar](#) by Eric Raymond examines OSS/FS development processes and issues. A useful collection of many OSS/FS writings, including the essay *The Cathedral and the Bazaar*, is in the [Open Source Reader](#). Peter Wayner's book [Free For All: How Linux and the Free Software Movement Undercut the High-tech Titans](#) describes the history and rise of OSS/FS, and includes interviews with many key leaders; the book can be either downloaded electronically without fee or purchased as a hardcover book. Ganesh C. Prasad has

published [The Practical Manager's Guide to Linux](#). [Dan Kegel's "The Case for Linux in Universities"](#) discusses why students need exposure to GNU/Linux at universities (and thus why universities should support and encourage this). You can see a collection of general information about OSS/FS at [my web page listing OSS/FS references](#).

2. MITRE Corporation has examined the application of OSS/FS to military systems. Their July 2001 report, [A Business Case Study of Open Source Software](#), concludes that "open source methods and products are well worth considering seriously in a wide range of government applications, particularly if they are applied with care and a solid understanding of the risks they entail. OSS encourages significant software development and code re-use, can provide important economic benefits, and has the potential for especially large direct and indirect cost savings for military systems that require large deployments of costly software products." They also recommend following the following steps to determine whether to use OSS or proprietary products: assess the supporting OSS developer community, examine the market, conduct a specific analysis of benefits and risks, compare the long-term costs, and choose your strategy. MITRE has received a Leadership Award from the non-profit Potomac Forum for showing that OSS can provide substantial advantages over proprietary software, particularly when reliability and long-term support are key requirements.

More recently, as noted in the [Washington Post article *Open-source Fight Flares at Pentagon*](#), "Microsoft Corp. is aggressively lobbying the Pentagon to squelch its growing use of freely distributed computer software and switch to proprietary systems such as those sold by the software giant, according to officials familiar with the campaign... But the effort may have backfired; [a second MITRE Corporation report prepared for the Department of Defense \(DoD\) Defense Information Systems Agency \(DISA\)](#) (originally dated May 10, 2002, and publicly released on October 28, 2002) concluded that OSS/FS use is widespread and should be expanded. This 2002 MITRE report concluded that "banning [OSS/FS] would have immediate, broad, and strongly negative impacts on the ability of many sensitive and security-focused DoD groups to defend against cyberattacks." The report also found that the GPL so dominates in DoD applications that a ban on just the GPL would have the same strongly negative impacts as banning all OSS/FS. MITRE noted that OSS/FS "plays a far more critical role in the DoD than has been generally recognized." In a two-week survey period MITRE identified a total of 115 FOSS applications and 251 examples of their use. MITRE concluded that "Neither the survey nor the analysis supports the premise that banning or seriously restricting [OSS/FS] would benefit DoD security or defensive capabilities. To the contrary, the combination of an ambiguous status and largely ungrounded fears that it cannot be used with other types of software are keeping [OSS/FS] from reaching optimal levels of use." In short, MITRE found that OSS/FS is widely used, and should be even more widely used. On May 28, 2003, [the DoD issued a formal memo placing OSS/FS on a level playing field with proprietary software](#), without imposing any additional barriers.

The Post article also noted that "at the Census Bureau, programmers used open-source software to launch a Web site for obtaining federal statistics for \$47,000, bureau officials said. It would

have cost \$358,000 if proprietary software were used.”

3. The European [Free/Libre and Open Source Software \(FLOSS\): Survey and Study](#) is a large multi-part report examining OSS/FS from a number of different vantage points. The report is divided into the following (besides its summary and raw data):
 - Part I: Use of Open Source Software in Firms and Public Institutions,
 - Part II: Firms’ Open Source Activities: Motivations and Policy Implications
 - Part II B: Open Source Software in the Public Sector: Policy within the European Union
 - Part III: Basics of Open Source Software Markets and Business Models
 - Part IV: Survey of Developers
 - Part V: Source Code Survey
4. Microsoft has been trying to claim that open source is somehow dangerous, and indeed is its leading critic, yet the Wall Street Journal’s Lee Gomes found that “Microsoft Uses Open-Source Code Despite Denying Use of such Software.” Here are some interesting quotes from his article:

... But Microsoft’s statements Friday suggest the company has itself been taking advantage of the very technology it has insisted would bring dire consequences to others. “I am appalled at the way Microsoft bashes open source on the one hand, while depending on it for its business on the other,” said Marshall Kirk McKusick, a leader of the FreeBSD development team.

More recently Microsoft has targeted the GPL license rather than all OSS/FS licenses, claiming that the GPL is somehow anti-commercial. But this claim lacks evidence, given the many commercial companies (e.g., IBM, Sun, and Red Hat) who are using the GPL. Also, see this paper’s earlier note that [Microsoft itself makes money by selling a product with GPL’ed components](#). The same article closes with this statement:

In its campaign against open-source, Microsoft has been unable to come up with examples of companies being harmed by it. One reason, said Eric von Hippel, a Massachusetts Institute of Technology professor who heads up a research effort in the field, is that virtually all the available evidence suggests that open source is “a huge advantage” to companies. “They are able to build on a common standard that is not owned by anyone,” he said. “With Windows, Microsoft owns them.”

Other related articles include [Bruce Peren’s comments](#), [Ganesh Prasad’s *How Does the Capitalist View Open Source?*](#), and the open letter [Free Software Leaders Stand Together](#).

5. Indeed, many who have analyzed general information technology (IT) trends or Microsoft’s actions have concluded that strongly depending on Microsoft’s products is now a dangerous strategy. [2003 And Beyond](#) by Andrew Grygus examines the IT industry from a small business point of view, and identifies a large number of dangers from depending on a Microsoft-based infrastructure. Fundamentally, Microsoft is working hard to increase customer dependency, and charges exorbitantly once the customer cannot practically switch.
6. Microsoft inadvertently advocated OSS/FS in leaked documents called the [”Halloween”](#)

[documents](#). The original first two Halloween documents found that OSS/FS was far more effective than they wished to admit. [Halloween 7](#) gives results of one of their surveys, again, with many positive comments about OSS/FS.

7. Another leaked internal Microsoft document is [Converting a UNIX .COM Site to Windows](#) (by David Brooks). This document describes lessons learned when converting Hotmail from the OSS/FS FreeBSD to Microsoft Windows after Microsoft purchased Hotmail, including advantages and disadvantages of each approach, and ends up identifying a large number of advantages of their competition. For example, it noted that “entrepreneurs in the startup world are generally familiar with one version of UNIX (usually through college education), and training in one easily converts to another.” [An article in The Register](#) summarizes many of the advantages of the Unix approach given in the paper.
8. Several documents were written to counter Microsoft’s statements such as those in Microsoft’s “Linux Myths”. This includes [LWN’s response](#) and [Jamin Philip Gray’s response](#), and the [FUD-counter site](#). The [shared source](#) page argues that Microsoft’s “shared source” idea is inferior to open source. [Richard Stallman’s The GNU GPL and the American Way](#) counters the amusing claim by Microsoft that the GPL was “un-American.” The letter [Free Software Leaders Stand Together](#) argues against the statements by Craig Mundie. You can find many general sites about Microsoft, including [Cloweth’s site](#).
9. In a story full of ironies, [Microsoft and Unisys teamed up in a well-funded marketing campaign against Unix](#), in part to try to revive Unisys’ sagging sales of Windows-based products. The 18-month, \$25 million campaign, dubbed “We have the Way Out,” specifically attacked the Unix offerings of Sun, IBM, and Hewlett-Packard, but since the major OSS/FS Oses are Unix or Unix-like, it attacks them as well. In a delicious irony, it was revealed that [the anti-Unix campaign website is powered by Unix software](#) - in this case, FreeBSD (an OSS/FS version of Unix) and the OSS/FS Web server Apache. Once this was publicly revealed, Microsoft and Unisys quickly switched to a Windows-based system.. and then [the website failed to operate at all for several days](#). If *that* wasn’t enough, [Andrew Orlovski reported in The Register](#) a further analysis of this website, noting that port 3306 was open on their website - a port primarily used by MySQL and Postgres. In other words, it appears that their anti-Unix site was still using OSS/FS software (not Microsoft’s own database) that is primarily deployed on Unix-like systems. Even their original imagery turns out to have had serious problems; the campaign’s original graphic showed a floor almost wholly covered in mauve paint (Sun Microsystem’s color), and the alternative offered was to jump through a window. [Many literate readers will recognize this symbol \(the act of throwing out through, or of being thrown out of, a window\) as defenestration, a way of killing rulers and also a popular way of inviting kings to commit suicide in 17th century Europe](#). In other words, this imagery suggests that you should use the window[s] to commit suicide (!). [Leon Brooks then analyzed the site further](#) - and found that the “way out” site used JSP (a technology fathered by Sun, Unix specialists). He also found that the site violated many standards; the site’s content failed the W3C validation suites (Microsoft is a member of the W3C), and uses a Windows-only character set that is not only non-standard, but actively conflicts with an important international standard (and ironically one which Microsoft is actively promoting). If using only Windows is so

wonderful, why can't the advocacy site conform to international standards? The real problem here, of course, is that trying to convince people that Unix is to be avoided at all costs - while using Unix and then having serious problems when trying to use an alternative - is both ironic and somewhat hypocritical.

10. [“How Big Blue Fell For Linux”](#) is an article on how IBM transitioned to becoming a major backer. IBM announced that it planned to invest \$1 Billion in GNU/Linux in 2001 *all by itself* (see the [IBM annual report](#)). In 2002 [IBM reported that they had already made almost all of the money back](#); I and others are a little skeptical of these claims, but it's clear that IBM has significantly invested in GNU/Linux and seem to be pleased with the results (for an example, see their [Linux-only mainframe](#)). This is not just a friendly gesture, of course; companies like [IBM view OSS/FS software as a competitive advantage](#), because OSS/FS frees them from control by another organization, and it also enables customers to switch to IBM products and services (who were formerly locked into competitor's products). Thankfully, this is a good deal for consumers too. In 2002, IBM had [250 employees working full time to improve Linux](#).
11. For a scientifically unworthy but really funny look at what people who *use* the various OSes say, take a look at the [Operating System Sucks-Rules-O-Meter](#). It counts how many web pages make statements like “Linux rocks”. It's really barely an opinion poll, but if nothing else it's great for a laugh.
12. There have been several academic studies of OSS/FS. For example, [“A Framework for Open Source Projects” \(a Master Thesis in Computer Science by Gregor J. Rothfuss\)](#) describes a framework for describing Open Source projects, introducing notions of actors, roles, areas, processes and tools, and depicts their interrelationships. The goal was to provide a conceptual foundation and a help for organizing and managing Open Source projects.
13. Several studies examine developers (instead of the programs they write), including [“A Quantitative Profile of a Community of Open Source Linux Developers”](#), [Herman, Hertel and Niedner's study \(based on questionnaires\)](#), and the [Who Is Doing It \(WIDI\)](#) study. The European [Free/Libre and Open Source Software Survey \(FLOSS\)](#) has a large amount of information on developers. The paper [Two Case Studies of Open Source Software Development: Apache and Mozilla](#) examines two major open source projects, the Apache web server and the Mozilla browser, and using archives (such as source code change history and problem reports) they quantify aspects of developer participation, core team size, code ownership, productivity, defect density, and problem resolution intervals for these projects. The [Boston Consulting Group/OSDN Hacker Survey](#) (release 0.73, July 21, 2002) made some interesting observations by sampling SourceForge users. For example, it gives evidence that open source developers can be divided into four groups (based on their motivations for writing OSS/FS software):
 - a. Believers (19%): believe source code should be open.
 - b. Learning and Fun (29%): for non-work needs and intellectual stimulation.
 - c. Hobbyists (27%): need the code for a non-work reason.
 - d. Professionals (25%): for work needs and professional status.

Journalists sometimes like to romanticize OSS/FS developers as being mostly teenage boys with little experience, but the survey didn't support that view. The study found that the open source

developers surveyed are mostly experienced professionals, having an average of 11 years of programming experience; the average age was 28.

14. If you determine that you wish to start an OSS/FS project, there are some documents available to aid you. This includes the [Free Software Project Management HOWTO](#) and [Software Release Practice HOWTO](#). You should also read [The Cathedral and the Bazaar](#).
15. Other evaluations include the [Gartner Group](#) and [GNet](#) evaluations.

For general information on OSS/FS, see my [list of Open Source Software / Free Software \(OSS/FS\) references at http://www.dwheeler.com/oss_fs_refs.html](#)

13. Conclusions

OSS/FS has significant [market share](#) in many markets, is often the most [reliable software](#), and in many cases has the best [performance](#). OSS/FS [scales](#), both in problem size and project size. OSS/FS software often has far better [security](#), perhaps due to the possibility of worldwide review. [Total cost of ownership](#) for OSS/FS is often far less than proprietary software, especially as the number of platforms increases. These statements are not merely opinions; these effects can be shown *quantitatively*, using a wide variety of measures. This doesn't even consider [other issues that are hard to measure](#), such as freedom from control by a single source, freedom from licensing management (with its accompanying risk of audit and litigation), [Organizations can transition to OSS/FS in part or in stages](#), which for many is a far more practical transition approach.

Realizing these potential OSS/FS benefits may require approaching problems in a different way. This might include using thin clients, deploying a solution by adding a feature to an OSS/FS product, and understanding the differences between the proprietary and OSS/FS models. Acquisition processes may need to change to include specifically identifying OSS/FS alternatives, since simply putting out a "request for proposal" may not yield all the viable candidates. OSS/FS products are not the best technical choice in absolutely all cases, of course; even organizations which strongly prefer OSS/FS generally have some sort of waiver process for proprietary programs. However, it's clear that considering OSS/FS alternatives can be beneficial.

Of course, before deploying any program you need to evaluate how well it meets your needs, and some organizations do not know how to evaluate OSS/FS programs. If this describes your circumstance, you may wish to look at the companion articles [How to Evaluate OSS/FS Programs](#) and the [Generally Recognized as Mature \(GRAM\) list](#).

OSS/FS options should be carefully considered any time software or computer hardware is needed. Organizations should ensure that their policies encourage, and not discourage, examining OSS/FS approaches when they need software.

Appendix A. About Open Source Software / Free Software (OSS/FS)

This appendix gives more information about open source software / free software (OSS/FS): definitions (of source code, free software, open source software, and various movements), motivations of developers, history, license types, management approaches, and forking.

A.1 Definitions

There are official definitions for the terms “Free Software” (as the term is used in this text) and “open source software”. However, understanding a few fundamentals about computer software is necessary before these definitions make sense. Software developers create computer programs by writing text, called “source code,” in a specialized language. This source code is often mechanically translated into a format that the computer can run. As long as the program doesn’t need to be changed (say, to support new requirements or be used on a newer computer), users don’t necessarily need the source code. However, changing what the program does usually requires possession and permission to change the source code. In other words, whoever legally controls the source code controls what the program can and cannot do. Users without source code often cannot have the program changed to do what they want or have it ported to a different kind of computer.

The next two sections give the official definitions of Free Software and Open Source Software (though in practice, the two definitions are essentially the same thing).

A.1.1 Definition of Free Software

OSS/FS programs have existed since digital computers were invented, but beginning in the 1980s, people began to try capture the concept in words. The two main definitions used are the “free software definition” (for free software) and the “open source definition” (for open source software). Software meeting one definition usually meets the other as well. Since the term “free software” came first, we’ll examine its definition first.

The [Free Software Definition](#) is published by Richard Stallman’s Free Software Foundation. Here is the key text of that definition:

“Free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer.” Free software is a matter of the users’ freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies 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). Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission. You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way. The freedom to use a program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job, and without being required to communicate subsequently with the developer or any other specific entity.

The text defining “free software” is actually much longer, explaining further the approach. It notes that “Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important.”

A.1.2 The Open Source Definition

Open source software is officially defined by the [open source definition](#):

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there

must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. The License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. No provision of the license may be predicated on any individual technology or style of interface.

A.1.3 Open Source Movement and Free Software Movement

As a practical matter, the definitions given above for free software and open source software are essentially the same. Software meeting the criteria for one generally end up meeting the other definition as well; indeed, those who established the term “open source” describe their approach as marketing approach to Free Software. However, to some people, the *connotations* and motives are different between the two terms.

Some people who prefer to use the term “free software” intend to emphasize that software should always meet such criteria for ethical, moral, or social reasons, emphasizing that these should be the rights of every software user. Such people may identify themselves as members of the “free software movement”. Richard Stallman is a leader of this group; his arguments are given in his article [Why “Free Software” is better than “Open Source”](#)

Some people are not persuaded by these arguments, or may believe the arguments but do not think that they are effective arguments for convincing others. Instead, they prefer to argue the value of OSS/FS on other grounds, such as cost, security, or reliability. Many of these people will prefer to use the term “open source software”, and some may identify themselves as part of the “open source movement”. Eric Raymond was one of the original instigators of the name “open source” and is widely regarded as a leader of this group.

Is the “free software movement” a subset of the “open source movement”? That depends on how the “open source movement” is defined. If the “open source movement” is a general term describing anyone who supports OSS or FS for whatever reason, then the “free software movement” is indeed a subset of the “open source movement”. However, some leaders of the open source movement (such as Eric Raymond) specifically recommend *not* discussing user freedoms, and since this is the central principle of the free software movement, the two movements are considered separate groups by many.

The [Free/Libre and Open Source Software Survey \(FLOSS\)](#), part IV, summarizes a survey of OSS/FS developers (primarily European developers), and specifically examined these terms. In this study, 48.0% identified themselves as part of the “Free Software”, community, 32.6% identified themselves as part of the “open source” community, and 13.4% stated that they did not care. A slight majority (52.9%)

claimed that the movements different in principle, but the work is the same, while 29.7% argued that the movements were fundamentally different, and 17.3% do not care at all about the differences. After examining the data, the surveyers determined that OSS/FS developers could be divided into six groups:

1. developers who assign themselves to the Free Software community and who see fundamental differences between the two communities (18%).
2. developers who consider themselves as part of the Open Source community and who perceive fundamental differences between the two communities (9%).
3. developers who assign themselves to the Free Software community and who perceive only principle differences between the two communities, but consider work in the two communities the same (26%).
4. developers who assign themselves to the Open Source community and see principle, but no fundamental differences between the two communities (17%).
5. developers who assign themselves to either the Free Software or the Open Source Software community, but are not bothered by differences between the two communities (9%).
6. developers who do not care to which community they belong (20%).

This difference in terminology and motivation can make it more difficult for authors of articles on OSS/FS (like this one). The motivations of the different movements may be different, but since practice the developers usually work together, it's very useful to have a common term that covers all groups. Some authors choose to use one of the terms (such as OSS). Other authors use some other term merging the two motivations, but as of this time there is no single merged term used by everyone. This article uses the merged term OSS/FS.

A.2 Motivations

This leads to a more general and oft-asked question: "Why do developers contribute to OSS/FS projects?" The short answer is that there are many different motivations.

The [Boston Consulting Group/OSDN Hacker Survey](#) (release 0.73, July 21, 2002) made some interesting observations by sampling SourceForge users. The top motivations given for participating in OSS/FS development were as follows:

1. intellectually stimulating (44.9%)
2. improves skill (41.3%)
3. work functionality (33.8%)
4. code should be open (33.1%)
5. non-work functionality (29.7%)
6. obligation from use (28.5%)

By examining these motivations, they concluded that open source developers could be divided into four groups (based on their primary motivations for writing OSS/FS software):

- a. Believers (19%): believe source code should be open.
- b. Learning and Fun (29%): for non-work needs and intellectual stimulation.
- c. Hobbyists (27%): need the code for a non-work reason.
- d. Professionals (25%): for work needs and professional status.

Part IV of the [Free/Libre and Open Source Software Survey \(FLOSS\)](#), mentioned above, also examined individual developer motivations, and found a variety of motivations.

Many businesses contribute to OSS/FS development, and their motivations also vary. Many companies develop OSS/FS to sell support - by giving away the product, they expect to get far more support contracts. [Joel Spolsky's "Strategy Letter V"](#) notes that "most of the companies spending big money to develop open source software are doing it because it's a good business strategy for them." His argument is based on microeconomics, in particular, that every product in the marketplace has substitutes and complements. A substitute is another product you might buy if the first product is too costly, while a complement is a product that you usually buy together with another product. Since demand for a product increases when the prices of its complements decrease, smart companies try to commoditize their products' complements. For many companies, supporting an OSS/FS product turns a complementary product into a commodity, resulting in more sales (and money) for them.

One widely-read essay discussing commercial motivations is Eric Raymond's [The Magic Cauldron](#). The European [Free/Libre and Open Source Software \(FLOSS\): Survey and Study](#) has additional statistics on the motivations of individuals and corporations who develop OSS/FS.

A.3 History

In the early days of computing (approximately 1945 to 1975), computer programs were often shared among developers, just as OSS/FS practitioners do now. An important during this time period was the ARPAnet (the early form of the Internet). Another critical development was the operating system Unix, developed by AT&T researchers, and distributed as source code (with modification rights) for a nominal fee. Indeed, the interfaces for Unix eventually became the basis of the POSIX suite of standards. However, as years progressed, and especially in the 1970s and 1980s, software developers increasingly closed off their software source code from users. This included the Unix system itself; many had grown accustomed to the freedom of having the Unix source code, but AT&T suddenly increased fees and limited distribution, making it impossible for many users to change the software they used and share those modifications with others.

Richard Stallman, a researcher at the MIT Artificial Intelligence Lab, found this closing of software source code intolerable. In 1984 he started the GNU project to develop a complete Unix-like operating system which would be Free Software (free as in freedom, not as in price, as described above). In 1985, Stallman established the Free Software Foundation (FSF) to work to preserve, protect and promote Free

Software; the FSF then became the primary organizational sponsor of the GNU Project. The GNU project developed many important software programs, including the GNU C compiler (gcc) and the text editor emacs. A major legal innovation by Stallman was the GNU General Public Licence (GPL), a widely popular OSS/FS software license. However, the GNU project was stymied in its efforts to develop the “kernel” of the operating system. The GNU project was following the advice of academics to use a “microkernel architecture,” and was finding it difficult to develop a strong kernel using this architecture. Without a kernel, the GNU project could not fulfill their goal.

Meanwhile, the University of California at Berkeley had had a long relationship with AT&T’s Unix operating system, and Berkeley had ended up rewriting many Unix components. Keith Bostic solicited many people to rewrite the remaining key utilities from scratch, and eventually managed to create a nearly-complete system whose source code could be freely released to the public without restriction. The omissions were quickly filled, and soon a number of operating systems were developed based on this effort. Unfortunately, these operating systems were held under a cloud of concern from lawsuits and counter-lawsuits for a number of years. Another issue was that since the BSD licenses permitted companies to take the code and make it proprietary, companies such as Sun and BSDI did so - continuously siphoning developers from the openly sharable code, and often not contributing back to the publicly available code. Finally, the projects that developed these operating systems tended to be small groups of people who gained a reputation for rarely accepting the contributions by others (this reputation is unfair, but nevertheless the perception did become widespread). The descendants of this effort include the capable operating systems NetBSD, OpenBSD, and FreeBSD, as a group called the *BSDs. However, while they are both used and respected, and proprietary variants of these (such as Apple Mac OS X) are thriving, another OSS/FS effort quickly gained the limelight and much more market share.

In 1991, Linus Torvalds began developing a small operating system kernel called “Linux”, at first primarily for learning about the Intel 80386 chip. Unlike the BSD efforts, Torvalds eventually settled on the GPL license, which forced competing companies working on the kernel code to work together. Advocates of the *BSDs dispute that this is an advantage, but even today, major Linux distributions hire key kernel developers to work together on common code, in contrast to the corresponding commercial companies to the *BSDs which often do not share their improvements to a common program. Torvalds made a number of design decisions that in retrospect were remarkably wise: using a traditional monolithic kernel design (instead of the “microkernel approach” that slowed the GNU project), using the the Intel 386 line as the primary focus, working to support user requests (such as “dual booting”), and supporting hardware that was technically poor but widely used. And finally, Torvalds stumbled into a development process rather different from traditional approaches by exploiting the Internet. Torvalds’ new process looked rather different than more traditional approaches. He publicly released new versions extremely often (sometimes more than once a day, allowing quick identification when regressions occurred), and he quickly delegated areas to a large group of developers (instead of sticking to a very small number of developers). Instead of depending on rigid standards, rapid feedback on small increments and Darwinian competition were used to increase quality.

When the Linux kernel was combined with the already-developed GNU operating system components and some components from other places (such as from the BSD systems), the resulting operating system

was surprisingly stable and capable. Such systems were called GNU/Linux systems or simply Linux systems. Note that there is a common misconception in the media that needs to be countered here: Linus Torvalds never developed the so-called “Linux operating system”. Torvalds was the lead developer of the Linux kernel, but the kernel is only one of many pieces of an operating system; most of the GNU/Linux operating system was developed by the GNU project and by other related projects.

In 1996, Eric Raymond realized that Torvalds had stumbled upon a whole new style of development, combining the sharing possibilities of OSS/FS with the speed of the Internet into a new development process. His essay [The Cathedral and the Bazaar](#) identifies that process, in a way that others could try to emulate the approach. The essay was highly influential, and in particular convinced Netscape to switch to an OSS/FS approach for its next generation web browser (the road for Netscape was bumpy, but ultimately successful).

In spring of 1997, a group of leaders in the Free Software community gathered, including Eric Raymond, Tim O’Reilly, and Larry Wall. They were concerned that the term “Free Software” was too confusing and unhelpful (for example, many incorrectly thought that the issue was having no cost). The group coined the term “open source” as an alternative term, and Bruce Perens developed the initial version of the “open source definition” to define the term. The term “open source” is now very widely used, but not universally so; Richard Stallman (head of the FSF) never accepted it, and even Bruce Perens switched back to using the term “Free Software” because Perens felt that there needed to be more emphasis on user freedom.

Major Unix server applications (such as the OSS/FS Apache web server) were easily moved to GNU/Linux or the *BSDs, since they all essentially implemented the POSIX standards. As a result, GNU/Linux and the *BSDs rapidly gained significant market share in the server market. A number of major initiatives began to fill in gaps to create completely OSS/FS modern operating systems, including graphical toolkits, desktop environments, and major desktop applications. In 2002, the first user-ready versions of capable and critical desktop applications (Mozilla for web browsing and Open Office for an office suite) were announced.

You can learn more about the history of OSS/FS from material such as [Open Sources: Voices from the Open Source Revolution](#) and *Free for All: How Linux and the Free Software Movement Undercut the High-Tech Titans* by Peter Wayner,

A.4 Licenses

There are dozens of OSS/FS licenses, but the vast majority of OSS/FS software uses one of the four major licenses: the GNU General Public License (GPL), the GNU Lesser (or Library) General Public License (LGPL), the MIT (aka X11) license, and the BSD-new license. Indeed the Open Source Initiative refers to these four licenses as the [classic open source licenses](#). The GPL and LGPL are termed “copylefting” licenses, that is, these licenses are designed to prevent the code from becoming

proprietary. Here is a short description of these licenses:

1. The GPL allows anyone to use the program and modify it, but prevents code from becoming proprietary once distributed and it also forbids proprietary programs from “linking” to it.
2. The MIT and BSD-new licenses let anyone do almost anything with the code except sue the authors. One minor complication: there are actually two “BSD” licenses, sometimes called “BSD-old” and “BSD-new”; new programs should use BSD-new instead of BSD-old.
3. The LGPL is a compromise between the GPL and the MIT/BSD-new approaches, and was originally intended for code libraries. Like the GPL, LGPL-licensed software cannot be changed and made proprietary, but the LGPL does permit proprietary programs to link to the library, like the MIT/BSD-new licenses.

Note that all of these licenses (the GPL, MIT, BSD-new, and LGPL) permit the commercial sale and the commercial use of the software, and many such programs as sold and used that way. See [Perens' paper](#) for more information comparing these licenses.

The most popular OSS/FS license by far is the GPL. For example, Freshmeat.net reported on April 4, 2002 that 71.85% of the 25,286 software branches (packages) it tracked are GPL-licensed (the next two most popular were LGPL, 4.47%, and the BSD licenses, 4.17%). Sourceforge.net reported on April 4, 2002 that the GPL accounted for 73% of the 23,651 “open source” projects it hosted (next most popular were the LGPL, 10%, and the BSD licenses, 7%). In my paper [More than a Gigabuck: Estimating GNU/Linux's Size](#), I found that Red Hat Linux, one of the most popular GNU/Linux distributions, had over 30 million physical source lines of code in version 7.1, and that 50.36% of the lines of code were licensed solely under the GPL (the next most common were the MIT license, 8.28%, and the LGPL, 7.64%). If you consider the lines that are dual licensed (licensed under both the GPL and another license, allowing users and developers to pick the license to use), the total lines of code under the GPL accounts for 55.3% of the total. [My paper on GPL compatibility discusses these figures further](#), and discusses why, if you choose to develop OSS/FS code, you should strongly consider using a licensing approach that is compatible with the GPL.

A.5 Management Approaches

There is no single approach to managing an OSS/FS project, just as there is no single approach to managing proprietary projects. Management approaches are strongly influenced by the size and scope of the project, as well as the leadership styles of those managing the project.

[The Cathedral and the Bazaar](#) argues for a particular style of development, termed the “bazaar” style. In this approach, there are a large number of small, incremental releases, and a large number of developers can send in patches for proposed improvements. The releases need to compile and run (to some extent), so that developers can test and improve them. Not all OSS/FS projects work this way, but many do.

It is useful to examine the management approaches of successful projects to identify approaches that may work elsewhere. Here are a few:

1. *Linux kernel*. The Linux kernel's development process is based on a hierarchy of four levels: ordinary developers, maintainers, trusted lieutenants, and the benevolent dictator. Ordinary developers can propose changes, but usually they submit their proposals to a maintainer of a particular component of the kernel; the maintainers then send their sets up to a trusted lieutenants, who then sends it up to the benevolent dictator (currently Linus Torvalds). At each stage testing can take place. The benevolent dictator writes code and issues general direction, but his primary job is to be the integrator and arbiter of changes. Development releases are made often; after the development has stabilized, a "stable" branch is created with a separate maintainer of the branch. Linux distributions then take the stable branch, test it further, and select the "best" version of the stable branch.
2. *Apache*. The Apache web server project, in contrast, is run by a group. At the top is the "Apache HTTP Server Project Management Committee (PMC)" a group of volunteers who are responsible for managing the Apache HTTP Server Project. Membership in the Apache PMC is by invitation only and must be approved by consensus of the active Apache PMC members. Membership can be revoked by a unanimous vote of all the active PMC members other than the member in question. Most changes are approved by consensus.

An action item requiring consensus approval must receive at least 3 binding +1 votes and no vetos (a "-1" vote). An action item requiring majority approval must receive at least 3 binding +1 votes and more +1 votes than -1 votes (i.e., a majority with a minimum quorum of three positive votes).

Ideas must be review-then-commit; patches can be commit-then-review. With a commit-then-review process, they trust that the developer doing the commit has a high degree of confidence in the change. Doubtful changes, new features, and large-scale overhauls need to be discussed before being committed to a repository.

See the [Apache Voting Rules](#) for more detailed information.

3. *Perl*. Perl was originally developed by Larry Wall, but he no longer wishes to have to always have the job of integrating patches. Thus, there is a notional "patch pumpkin" that must be acquired to change Perl. In Moody's *Rebel Code*, Wall explains that "we have essentially a chief integrator who is called the pumpkin holder." Moody adds that this "integration involves taking the approved patches and adding them into the main Perl source code." Larry Wall, as original developer, can veto any change. [More information about the patch pumpkin \(as it has currently evolved\) is available from perl.com.](#)
4. *Sourceforge-based Applications*. Many OSS/FS projects are supported by SourceForge, which includes the CVS tool for configuration management. Typically, those who have write access to the repository simply make their updates; others who do not have such access post their requests or patches to the bug tracking database (or mailing list) and ask one of those with write access to

include it. There are typically only a few people with direct write access, so conflicts are rare and CVS supports resolving the occasional conflict.

A.6 Forking

A *fork* is a competing project based on a version of the pre-existing project's source code. All OSS/FS projects can be "forked"; the ability to create a fork is fundamental to the definition of OSS/FS.

Simply creating or releasing a variant of a project's code does not normally create a fork. Indeed, releasing variants for experimentation is considered normal in a typical OSS/FS development process. Many OSS/FS projects (such as the Linux kernel development project) intentionally have "fly-offs" in which different developers implement different approaches; the results are compared and the "winning" approach is accepted by the project. These "fly-offs" are often discussed in evolutionary terms, in which the "winning mutation" is accepted into the project and the alternatives are abandoned as "evolutionary dead ends". Since all parties intend for the "best" approach to be accepted by the project, and for the other approaches to be abandoned, these are not forks. What is different about a fork is intent: the person(s) creating the fork intend for the fork to replace or compete with the original project they are forking.

Creating a fork is a major and emotional event in the OSS/FS community. It is similar to a call for a "vote of no confidence" in a parliament, or a call for a labor strike in a labor dispute. Those creating the fork are essentially stating that they believe the project's current leadership is ineffective, and are asking developers to vote against the project leadership by abandoning the original project and switching to their fork. Those who are creating the fork must argue why other developers should support their fork; common reasons given include a belief that changes are not being accepted fast enough, that changes are happening too quickly for users to absorb them, that the project governance is too closed to outsiders, that the licensing approach is hampering development, or that the project's technical direction is fundamentally incorrect.

Most attempts to create forks are ignored, for there must be a strong reason for developers to consider switching to a competing project. Developers usually resist supporting OSS/FS forks: they divide effort that would be more effective when combined, they make support and further development more difficult, and they require developers to discuss project governance rather than improving the project's products. Developers can attempt to support both projects, but this is usually impractical over time as the projects diverge. Eric Raymond, in *Homesteading the Noosphere*, argues that a prime motivation in OSS/FS development is reputation gain through the use of a gift culture, and that forking significantly interferes with this motivation.

Some historical examples of major forks may help give perspective, showing that often forks "lose" while other times they "win" against the original project:

1. *glibc* vs. *libc*. When the Linux kernel was first being developed, the kernel developers took the FSF's GNU C library (now called *glibc*) and created their own fork of it (called *libc*). Both were

licensed under the LGPL. At the time, the Linux kernel developers thought that the FSF's development process for the C library was too slow and not responding to their needs. Thus, they [created a forked version of GNU libc version 1.07.4 \(which had been released February 17, 1994\)](#). In this case, however, the original GNU C library project (led by the FSF) surpassed the forked project over time. Over the next few years the original glibc increasingly offered far better standards conformance, multithreading, higher performance, and more features than the forked libc project. [Elliot Lee briefly describes this history](#). In this case, the fork was abandoned after several years; in 1997 through 1998 nearly all GNU/Linux systems switched from libc back to glibc.

2. *gcc vs. egcs*. The GNU Compiler Collection (gcc) is a collection of important compilers, including a C++ compiler; the main compilers are licensed under the GPL. In 1997, there were disagreements over the development approach and slow development speed of gcc. In particular, many were dissatisfied with the FSF-appointed gcc maintainer, who was very slow to accept changes. Cygnus (headed by Michael Tiemann) decided to create a fork of the project named egcs, and invited others to join. Egcs worked at an accelerated pace, and soon surpassed the original gcc project. In April 1997 the rift was healed; the FSF agreed to switch to using the egcs code for gcc, and the egcs project agreed to dissolve itself and take over the original gcc project. In this case, the fork ended with the forking project's results "taking over" the original project.

Too many forks can be a serious problem for all of the related projects. In fact, one of the main reasons that Unix systems lost significant market share compared to Windows was because of the excessive number of Unix forks. Bob Young states this quite clearly in this essay "Giving it Away", and also suggests why this is unlikely to be a problem in copylefted OSS/FS software:

The primary difference between [GNU/Linux and Unix] is that Unix is just another proprietary binary-only ... OS [operating system]. The problem with a proprietary binary-only OS that is available from multiple suppliers is that those suppliers have short-term marketing pressures to keep whatever innovations they make to the OS to themselves for the benefit of their customers exclusively. Over time these "proprietary innovations" to each version of the Unix OS cause the various Unices to differ substantially from each other. This occurs when the other vendors do not have access to the source code of the innovation and the license the Unix vendors use prohibit the use of that innovation even if everyone else involved in Unix wanted to use the same innovation. In Linux the pressures are the reverse. If one Linux supplier adopts an innovation that becomes popular in the market, the other Linux vendors will immediately adopt that innovation. This is because they have access to the source code of that innovation and it comes under a license that allows them to use it.

Note that the copylefting licenses (such as the GPL and LGPL) permit forks, but greatly reduce any monetary incentive to create a fork. Thus, the project's software licensing approach impacts the likelihood of its forking.

The ability to create a fork is important in OSS/FS development, for the same reason that the ability to call for a vote of no confidence or a labor strike is important. Fundamentally, the ability to create a fork forces project leaders to pay attention to their constituencies. Even if an OSS/FS project completely dominates its market niche, there is always a potential competitor to that project: a fork of the project. Often, the threat of a fork is enough to cause project leaders to pay attention to some issues they had ignored before, should those issues actually be important. In the end, forking is an escape valve that allows those who are dissatisfied with the project's current leadership to show whether or not their alternative is better.

About the Author

David A. Wheeler is an expert in computer security and has a long history of working with large and high-risk software systems. His books include [*Software Inspection: An Industry Best Practice*](#) (published by IEEE CS Press), [*Ada 95: The Lovelace Tutorial*](#) (published by Springer-Verlag), and the [*Secure Programming for Linux and Unix HOWTO*](#). Articles he's written include [*More than a Gigabuck: Estimating GNU/Linux's Size*](#) and [*The Most Important Software Innovations*](#). Mr.

Wheeler's web site is at <http://www.dwheeler.com>; you may contact him at dwheeler@dwheeler.com, but you may not send him spam (he reserves the right to charge fees to those who send him spam).



You may reprint this article (unchanged) an unlimited number of times and distribute local electronic copies (e. g., inside an organization) as long as no fee is involved. You may not “mirror” this document to the public Internet or other public electronic distribution systems; mirrors interfere with ensuring that readers can immediately find and get the current version of this document. Copies clearly identified as old versions, not included in normal searches as current Internet data, and for which there is no charge (direct or indirect) to access them are usually fine; examples of acceptable copies are Google caches and the Internet archive's copies. Please contact David A. Wheeler for clarifications or if you'd like to translate this article into another (human) language; I would love to see more freely-available translations of this document, and I will help you coordinate with others who may be translating the document into that language. Trademarks are registered by various organizations, for example, Linux(r) is a trademark of Linus Torvalds. This is a personal essay and not endorsed by David A. Wheeler's employer. This article is a research article, not software nor a software manual.



Sat, May
15th

[home](#) | [browse](#) | [articles](#) | [contact](#) | [chat](#) | [submit](#) |
[faq](#) | [newsletter](#) | [about](#) | [stats](#) | [scoop](#)

15:35
PDT

Search for _____ in _____ Section

[login](#) «
[register](#) «
[recover password](#) «

[Introduction to Multicasting](#)

by [Jonathan Day](#), in [Tutorials](#) - Saturday, May 15th 2004 00:00 PDT

Multicasting is the ability to transmit a single stream to multiple subscribers at the same time. Unlike conventional streaming, it does not need one stream per recipient. Instead, there is one stream on any one segment of the network on which there is a subscriber. It is the task of the routers to track subscriptions and to create copies only on an as-needed basis. Unlike broadcasting, segments on which there are no subscribers do not receive the stream.

[\[#2 comments | read more\]](#)

[Skippy 0.4.1](#)

by [Hyriand](#) - Saturday, May 15th 2004 13:52 PDT

About: A full screen pager for X11, not entirely unlike expocity and Apple's Expose. It arranges snapshots of all windows on the current desktop, allowing you to easily switch between applications. It doesn't require a specific window manager, but requires NetWM compliance to work.

Take me to a [random project](#).

Username

Password

Remember me







Please note that you must have **cookies** enabled in your browser to be able to log in.

Advertising

[Editorials](#)

- [Xtreme Programming and Open Source Software Development \(May 1st\)](#)
- [Gna!, A New Host for Libre Software Development \(Feb](#)

Changes: Several segmentation faults were fixed. Support was added for window managers that still use the GNOME WM Spec (most notably, WindowMaker when compiled with GNOME support). Some pretty serious logic errors were fixed. The root pixmap is now properly tiled if it's smaller than the screen size.







Categories	Focus	License	URLs
Desktop Environment	Major bugfixes	GNU General Public License (GPL)	     

Impostor 0.8.5

by [vitezg](#) - Saturday, May 15th 2004 13:50 PDT

About: Impostor makes writing Web applications easier. It saves the whole internal state of your applications. It does so by starting a new instance of your application for every new session, instead of starting a new one for every request. Impostor works with the Python programming language and the Apache 2.x Web server.

Changes: A cookie-handling problem was worked around. The HTTP content-type was made changeable.

Categories	Focus	License	URLs
Internet :: WWW/ HTTP :: Dynamic Content	N/A	GNU General Public License (GPL)	     

11th)

- [Providing Good Feedback for Bug Reporters \(Jan 24th\)](#)

[freshmeat](#)

- [freshmeat for the world \(Apr 1st\)](#)

- [freshmeat XML-RPC API available \(Dec 16th\)](#)

- [freshmeat feature round-up \(Nov 24th\)](#)

[Category Reviews](#)

- [Spam Filters \(Aug 23rd\)](#)

- [GUI Toolkits for The X Window System \(Jul 27th\)](#)

- [Fractal Software \(Apr 26th\)](#)

[Book Reviews](#)

- [AI Application Programming \(Nov 15th\)](#)

- [Bioinformatics: Sequence and Genome Analysis \(Jun 7th\)](#)

- [Peopleware: Productive Projects and Teams \(Dec 21st\)](#)

[Project Reviews](#)

- [Irrlicht \(May 8th\)](#)

- [TWiki \(Feb 14th\)](#)

- [BitIBee: IRC vs. Instant Messengers \(Jan 31st\)](#)

[Tutorials](#)

- [Introduction to Multicasting \(May 15th\)](#)

- [Configuring an Open Source Mail Gateway \(May 31st\)](#)

- [Generating PHP Database Access Layers \(May 17th\)](#)

[Security](#)

- [SuSE: New mc packages fix](#)







Dodgin' Diamond 2 0.1

by [Juan J. Martínez](#) - Saturday, May 15th 2004

13:49 PDT

About: Dodgin' Diamond 2 is a little shoot-'em-up arcade game for one or two players. It aims to be an "old school" arcade game with low resolution graphics, top-down scrolling action, energy based gameplay, and different weapons with several levels of power.

Changes: Artwork and new graphics were added. The gameplay has been improved, and some bugs were fixed.

Categories	Focus	License	URLs
Games/	N/A	GNU	  
Entertainment ::		General	  
Arcade		Public License	
		(GPL)	

Botan 1.3.13 (Development)

by [Jack Lloyd](#) - Saturday, May 15th 2004 13:47

PDT

About: Botan is a library of cryptographic algorithms written in C++. It includes a wide selection of block and stream ciphers, public key algorithms, hash functions, and message authentication codes. It has an easy-to-use filter interface and supports many common industry standards, including X.509v3.

Changes: The compilation bug in 1.3.12 was fixed, as was a minor Mac OS X installation problem. Support for Cygwin builds has been improved significantly. The configuration script is somewhat smarter about picking which modules to compile.

[local privilege escalation \(May 14th\)](#)

- [Debian: New mah-jong packages fix denial of service \(May 13th\)](#)

- [Red Hat: Updated ipsec-tools package fixes vulnerabilities in ISAKMP daemon \(May 12th\)](#)

[Themes](#)

- [Learning From Kaleidoscope \(Oct 18th\)](#)

- [A Plea for Clear Theme Copyrights \(Nov 2nd\)](#)

- [The Antidesktop \(Oct 12th\)](#)

[Saturday](#)

- [Skippy 0.4.1](#)

- [Impostor 0.8.5](#)

- [Dodgin' Diamond 2 0.1](#)

- [Botan 1.3.13 \(Development\)](#)

- [sign 1.0.4](#)

- [DjVuLibre 3.5.13](#)

- [E-Keyboard 1.1](#)

- [IzPack 3.5.3 \(Stable\)](#)

- [Sophster 0.9.5-r15](#)

- [When 1.0.9](#)

- [LBreakout 2.5-beta5](#)

- [GenDoc 1.0 beta4](#)

- [Cyberduck 2.3](#)

- [Soma suite 1.2](#)

- [System Rescue CD 0.2.13](#)

- [IMMS Magical Favorites Collector 1.1](#)

- [svgalib 1.9.19](#)

- [Q-Midi 1.15](#)







- [dillo Web browser 0.8.1](#)

- [IaraJS 1.1.1](#)

- [LNKSRC 0.0.4 \(Alpha\)](#)

- [XStream 1.0](#)

- [TCB::Internal 1.02](#)

Categories	Focus	License	URLs
Security :: Cryptography Software Development :: Libraries	Minor bugfixes	BSD License (revised)	     






[sign 1.0.4](#)

by [Alex Pankratov](#) - Saturday, May 15th 2004

13:45 PDT

About: sign is a file signing and signature verification utility. It implements gzip-style command line syntax and OpenSSH-style key-based authentication. It is small, fast, and is meant to facilitate the use of authenticated file hashing for online distributed material.

Changes: The code was reworked a bit to build cleanly on OpenBSD.

Categories	Focus	License	URLs
Security System :: Archiving	Minor feature enhancements	BSD License (revised)	    

[DjVuLibre 3.5.13](#)

by [Leon Bottou](#) - Saturday, May 15th 2004 13:43

PDT

About: DjVu is a Web-centric format and software platform for distributing documents and images. DjVu content downloads and displays faster than competing formats. DjVu images can be smoothly zoomed and panned. DjVuLibre is an open source implementation of DjVu, including viewers, browser plugins,








- [TCB::System::Obsolete 0.99](#)
- [Host Grapher 2 \(II\) <<](#)
- [phpFormGenerator v2.08 Beta \(Stable\)](#)
- [BitTorrent 3.4.2 \(Stable\)](#)
- [Tile Driller 2.0](#)
- [VSOFont 1.8](#)
- [UPPAAL Timed Automata Parser Library 0.50](#)
- [EPS Graphics2D 0.8.2](#)

Friday

- [p0rn-comfort 0.0.2](#)
- [RealmForge Game Development Framework](#)
- [RealmForge Game Development Framework v0.1](#)
- [NetSQUID 1.0 \(Stable\)](#)
- [TCB::SysLoads 0.99](#)
- [TCB::Help 0.99](#)
- [TCB::Loan 0.99](#)
- [TCB::Equipment 0.99](#)
- [TCB::System 0.99.01](#)
- [OGRE 0.14.0](#)
- [Secret Squirrel 0.8](#)
- [jEdit 4.2pre13 \(Development\)](#)
- [Key Netplug 1.2.7](#)
- [GXPARSE gxparse-sf-alpha-0_4](#)
- [Harminv 1.0](#)
- [HA/FST - Free Solaris High Availability Clustering 1.9.2 \(Stable\)](#)
- [Skippy 0.4.0](#)
- [Slimpy 0.7a](#)
- [Spey 0.2](#)
- [Lost Labyrinth 0.9.3](#)
- [phpSavant 1.5](#)
- [NK Recnik 0.0.6](#)
- [Lintouch 1.0.1 \(Stable\)](#)
- [pam_envfeed 0.3](#)

decoders, simple encoders, and utilities.

Changes: The DjVu viewer, "djview", has gained a full screen option and navigation history buttons. The encoder "cjb2" can read TIFF files as well as PBM files. The DjVu XML tools are now installed by default. French and Japanese translations have been added. DjVuLibre now works under OS X, but still requires the X11 server.

Categories	Focus	License	URLs
Internet	Minor feature	GNU	 
Multimedia ::	enhancements	General	 
Graphics		Public	 
Multimedia ::		License	
Graphics :: Viewers		(GPL)	

E-Keyboard 1.1

by [Michal Veselenyi](#) - Saturday, May 15th 2004

13:40 PDT

About: E-keyboard is an Enlightenment epplet for switching between different keymaps. It is a graphical frontend to the setxkbmap. The actual keymap is shown using a flag and two or three letters (such as GB or SVK). Flags are mapped at a 3D flag-like (or board-like) superquadratic, and some lighting and Phong shading are applied to make it look much nicer. Keymaps can be cycled with the mouse buttons or wheel.

Changes: A COPYING file was added. Some weird debug messages were removed. The Polish flag was updated. The icon was updated.

Categories	Focus	License	URLs
------------	-------	---------	------

- [FCG FTP Client 0.0.7](#)
- [nanoweb 2.2.3](#)
- [Snowbox 0.9](#)
- [FDDA 0.9](#)
- [Yam 0.2](#)
- [pFuel 0.00.03](#)
- [RSS Orbit 1.2](#)
- [icel.tcl 002](#)
- [kimono 0.2.5b](#)
- [Adium X 0.56](#)
- [Friki 2.1.1](#)
- [ProfileManager 0.9](#)
- [TMMS 0.1.0](#)
- [Diqt 1.1.0](#)
- [Sender Policy Framework Library 0.25 beta](#)
- [Cgiapp.class.php 1.2](#)
- [pasmal 1.1](#)
- [tex2im 1.8](#)
- [ploticus 2.21](#)
- [AdvanceMAME 0.82.0](#)
- [SchoolTool Milestone 5](#)
- [Rad Inks Network Utilities 1.11 \(FTP Applet\)](#)
- [the breve simulation environment 1.9.1](#)
- [POPsearch 0.0.3](#)
- [The Frink Language 2004-05-14](#)
- [FindBugs 0.7.3](#)
- [mSTRIP 0.4.0](#)
- [Thomer's Music Vault 0.6.1-beta4](#)
- [quanekeo 1.0.0](#)
- [vSQLmail 0.1](#)
- [Simplebackup 1.2.0 RC2 \(Release Candidates\)](#)
- [tclwebtest 1.0](#)
- [Hydra Backup System 0.1-7](#)
- [Bugxter 3.0.0](#)
- [H-Sphere 2.4 patch 1](#)

[Desktop](#)Minor feature
enhancements[GNU](#)[Environment ::](#)[General](#)[Window](#)[Public](#)[Managers ::](#)[License](#)[Enlightenment](#)[\(GPL\)](#)[Desktop](#)[Environment ::](#)[Window](#)[Managers ::](#)[Enlightenment ::](#)[Epplets](#)[System :: Systems](#)[Administration](#)

IzPack 3.5.3 (Stable)

by [Julien Ponge](#) - Saturday, May 15th 2004 13:33

PDT

About: IzPack is a powerful Java installer builder. It is able to create lightweight and modular installers. You have the choice of the installer panels you want to use (some can do the same job, so that you can select the one you prefer), and you even have the choice of the kind of installer that you want to use. IzPack doesn't use any portion of native code, it is designed to be fully independent from the operating system that runs it. It is very easy for the end user with a properly installed JVM to use an installer made with IzPack, since a single "java -jar installer.jar" will launch it.

Changes: This new release contains several fixes for the German translation as well as for the TargetPanel. The other fixes include the Web kunststoff installers creation process, as well as the readme text displayed in IzPack's own installer.

Categories	Focus	License	URLs
------------	-------	---------	------

- [Project Steve Guttenberg 1.8.2](#)
- [lustre 1.2.1](#)
- [NullLogic Groupware 1.3.13 \(Development\)](#)
- [pxlib 0.2.8](#)
- [NSD 2.1.0](#)
- [kamix 0.2.1](#)
- [Paypal Shopping Cart 2.9](#)
- [Gkrellfah2 0.9.4](#)
- [ipcalc 0.37](#)
- [MerlinWork 0.11.2](#)
- [LDPC/LDGM 1.5](#)
- [MySQL Backup Pro 1.0.6-4](#)
- [Jajuk 0.2](#)
- [eyeD3 0.6.1](#)

Thursday

- [cc65 2.10.1](#)
- [QDBM: Quick DataBase Manager 1.8.11](#)
- [mod_spin 0.9.7](#)
- [Ethereal 0.10.4](#)
- [Meeting Room Booking System 1.2-pre3 release candidate](#)
- [bond - building object network databases 2.0.2](#)
- [Cactus 4.0 beta 14](#)
- [PhiloLogic 2.9pre4](#)
- [Infinity Plugin 0.5.3](#)
- [quepasa 0.0.1](#)
- [Plesk Server Administrator 7.0.2 \(7.0.x\)](#)
- [Chess Training Tools 1.2.3](#)
- [WebInvoicer 0.5](#)
- [GD-Sharp 1.0.2](#)
- [RemoteWAP 0.2](#)
- [Listserv2Mbox 1.2](#)
- [TinyButStrong 1.96](#)
- [Request Tracker 3.0.10](#)

[Software Development System :: Installation/Setup System :: Software Distribution](#)

Major bugfixes

[GNU General Public License \(GPL\)](#)



[Sophster 0.9.5-r15](#)

by [Steven Schaefer](#) - Saturday, May 15th 2004

13:31 PDT

About: Sophster is an HTTP server that provides access to the Linux server it's running on. It requires authentication and then takes on the user and group ID of the credentials. File and folder access is then limited by the user and group ID. It features file upload/download and [de]compression, file/folder delete, move, copy, and rename, basic text editing, creation of files and folders, and the ability to set permissions and user/group ownership of files/folders.

Changes: A routine for changing the last accessed/modified times of files and folders has been updated. Multiple selections can be made from the directory listing when making the time change. Directories can be recursively changed. However, changing last accessed and last modified are now separate requests. The routine for changing permissions and ownership has not changed, but the way you make the request has.

Categories	Focus	License	URLs
Communications :: File Sharing	Minor feature enhancements	Other/Proprietary License	
Internet :: WWW/HTTP			
Internet :: WWW/HTTP :: Dynamic Content			

(Release)

- [NSA Security-enhanced Linux 2004051217](#)
- [vnc2swf 0.4.1](#)
- [media-box 0.26](#)
- [DSPAM 3.0.0.beta.2](#)
- (Development)
- [GWhere 0.1.4](#)
- [GNU-Monitor 0.0.8](#)
- [X Automation Tools 0.96](#)
- [jaron 2.1](#)
- [htop 0.2](#)
- [Marzipan 0.01](#)
- [Workrave 1.6.1](#)
- [Reby 0.5.5](#)
- [getmail 3.2.4 \(Current\)](#)
- [DarkerIRC 1.1-pre1](#)
- [Dnsmasq 2.8](#)
- [xterm patch #188](#)
- [UCARP 0.8](#)
- [IFSgr 1.0](#)
- [PHPX 3.3.3 \(PHPX 3.x\)](#)
- [Symbion Daemon Tool 0.9.4](#)
- [OpenGestion 2.3 \(0405W2\)](#)
- [Daimonin BETA2-0.96](#)
- [The Tamber Project 1.9.10 \(Numenor\)](#)
- [Webmatic 1.82.5](#)
- [NoteBookManager 0.7.2](#)
- [LEAF Bering-uClibc 2.2-beta2](#)
- [TEA for Linux 0.5.0](#)
- [dbstep 0.3.1](#)
- [NoMachine NX 1.3.2 R 25/7](#)
- [Atom.NET 0.4.1](#)
- [Chemistry Development Kit 20040513 \(Stable\)](#)
- [Astaro Security Linux 5.007 \(Stable 5.x\)](#)
- [Sendmail 8.13.0.Beta2 \(Development\)](#)
- [Kaffeine Media Player 0.4.3b](#)








When 1.0.9

by [Ben Crowell](#) - Saturday, May 15th 2004 13:29

PDT

About: When is an extremely simple personal calendar program, aimed at the Unix geek who wants something minimalistic. It can keep track of things you need to do on particular dates. It's a very short and simple program, so you can easily tinker with it yourself. It doesn't depend on any libraries, so it's easy to install. You should be able to install it on any system where Perl is available, even if you don't have privileges for installing libraries. Its file format is a simple text file, which you can edit in your favorite editor.

Changes: The program now prints the time as well as the date. The variable "ampm" in the preferences file can be used to select 12-hour or 24-hour time.

Categories	Focus	License	URLs
Office/Business :: Scheduling	Minor feature enhancements	Perl License	    









LBreakout 2.5-beta5

by [kulkanie](#) - Saturday, May 15th 2004 13:25 PDT

About: LBreakout is a breakout game with nice effects, graphics, and sounds. It's got a menu to configure a lot of things and a high score chart. It can be played either by mouse or keyboard. New levels can also be created, and most other aspects of the game can be customized.

- [DSPAM phpControlCenter 1.08](#)
- [Hastymail 1.0](#)
- [sord 0.4.3 \(Development\)](#)
- [AIOCP 1.2.041](#)
- [DKMS 1.10](#)
- [Mathomatic 11.1b](#)
- [Yawiki PHP 0.8.4 alpha](#)
- [Dolda Webmail 0.2](#)
- [pop3spam 0.6](#)
- [OpenVIDIA 0.0.2](#)
- [The Manhattan Virtual Classroom 2.4](#)
- [PayStream 0.1.3 \(PayStream_B\)](#)
- [RIV2ASCII Conversion 0.1](#)
- [Sophster 0.9.5-r12](#)
- [tcptrack 1.1.1 \(Stable\)](#)
- [Jmol 10pre9 \(Development\)](#)
- [JCGrid 0.01](#)
- [Servlet Window Toolkit 1.1](#)
- [The bitap library 0.99](#)
- [pFuel 0.00.02](#)
- [E-GADS! 2.2.0](#)
- [Propagast 1.0](#)
- [Recovery Is Possible! 8.5 \(Stable\)](#)
- [1060 NetKernel Standard Edition 1.2.1](#)
- [JCache Open Source 1.0 alpha 6](#)
- [Monetra 4.0](#)
- [Hosting Backup 0.2.0](#)
- [Oracle OCI tracer 0.1](#)
- [dc_qt 0.1.1](#)
- [Libfilth 0.3](#)
- [Peephole 1.2](#)
- [Yet Another antiVirus Recipe 2.1.1.3](#)
- [Mount ISO image 0.8](#)
- [jlife 0.7](#)

Changes: A game may now be continued later. Pressing the middle mouse button (or a special key) will bring all balls to a high speed, which saves a lot of time if the level is nearly cleared. Completing a level by clearing all bricks will give 5000 points extra as an incentive to clear a level even though warp might be available already. A number of smaller bugfixes were made.

Categories	Focus	License	URLs
Games/Entertainment	Minor feature enhancements	GNU General Public License (GPL)	     
Games/Entertainment :: Arcade			
Games/Entertainment :: Simulation			



GenDoc 1.0 beta4

by [alexandre flament](#) - Saturday, May 15th 2004

13:24 PDT







About: GenDoc (formerly GenDiapo) is an XML Editor based on a existing project, MerlotXML. It can use two kinds of plugins (DTD and/or action). The DTD plugin can be used to customize the editor for a DTD, and an action plugin can be used to publish documents in HTML or PDF format. The editor is composed of 3 views: tree view, attribute view for current element, and a "styled view". The aim of styled view is to show the document with a visual aspect.

Changes: GenDoc now supports JDK 1.4. A couple bugs were fixed. In the styled view, characters no longer disappear when they are in focus.

- [Visitors Web Log Analyzer 0.2a](#)
- [qlife 0.7](#)
- [BBclone 0.4.1](#)
- [antinat 0.71](#)
- [Cut The Crap Software 2004-05-13](#)
- [Ruby/CorporateTime 0.2.1](#)
- [asm2class 0.0.8](#)
- [Nameko 0.5.3](#)
- [Damn Small Linux 0.7](#)
- [Linux-HA 1.3.0 \(Development\)](#)
- [pam_envfeed 0.1](#)
- [LibSysCTr 0.1](#)
- [Torque Network Library 1.2.0](#)
- [lisp-cgi-utils 0.2](#)
- [gbgraph2 0.3](#)

Wednesday

- [AntiExploit 1.2](#)
- [ripMIME 1.3.1.2](#)
- [mod_log_sql 1.98 \(Development\)](#)
- [Copper 2004 2.5 \(Corporate\)](#)
- [jSyncManager 2.0 \(jSerial API\)](#)
- [gimp-print 5.0.0-alpha3 \(Development\)](#)
- [ZAP Multiplayer 1.2.0](#)
- [The WebDruid 0.5.3](#)
- [Varuna Converter 0.4](#)
- [Groups, Algorithms, and Programming 4.4.3](#)
- [Javadoc Search 1.0.2](#)
- [XScreenSaver 4.16](#)
- [FinchTV 1.1.0](#)
- [Vehicle Maintenance Tracker 0.5](#)
- [LinuxStok 0.005](#)
- [Python Browsing Probe 0.2.0](#)
- [Driver On Demand Alpha 1](#)
- [Exiv2 0.3](#)

Categories	Focus	License	URLs
Text Processing :: Markup :: XML Utilities	Major bugfixes	GNU General Public License (GPL)	     



Cyberduck 2.3

by [dkocher](#) - Saturday, May 15th 2004 13:17 PDT

About: Cyberduck is an SFTP (SSH Secure File Transfer) and FTP browser for the Mac. It has been built from the ground up with usability in mind, having the same consistent graphical user interface for both SFTP and FTP browsing. Multiple connections are supported. Drag and drop is supported consequently for transferring files between server and client. A transfer queue keeps track of the pending file transfers. A simple bookmark manager ensures manageability. Core system technologies such as the Keychain and Rendezvous are supported. It has been translated into numerous languages including Japanese, Chinese, Korean, French, German, Italian, Portuguese, Spanish, and Dutch.

Changes: A Finnish localization was added. A bug that caused bookmarks to get lost upon moving items in the bookmark drawer was fixed. A bug which caused not all available Rendezvous services to get listed was fixed. The "type ahead" feature of the browser was removed, as it was responsible for frequent crashes.

Categories	Focus	License	URLs
------------	-------	---------	------

- [OOoPHP 0.4](#)
- [xmlBlaster 0.903](#)
- [IMAP Migration Tool 1.0](#)
- [monit 4.3](#)
- [IEncy 0.10](#)
- [Yaboo 0.5](#)
- [Xeukleides 1.0.0](#)
- [Eukleides 1.0.0](#)
- [Vexi Beta 1](#)
- [PolarViewer 0.8](#)
- [Xchangeboard 1.5 Final](#)
- [PunJab 0.3](#)
- [Kile 1.6.3 \(Stable\)](#)
- [Reverse Remote Shell 1.50](#)

Advertising

SecurityFocus

- [News: Sasser suspect has fans](#)
- [News: New flaw takes WiFi off the air](#)
- [News: Spam fighters infiltrate spam clubs](#)
- [News: Symantec fights auto-responder menace](#)
- [Infocus: Automating Windows Patch Mngt: Part III](#)
- [Infocus: Common Security Vulnerabilities in e-commerce Systems](#)
- [Infocus: Protecting Road Warriors: Managing Security for Mobile Users \(Part One\)](#)
- [Infocus: Solaris 10 Security](#)
- [Columnists: Secure by Default](#)
- [Columnists: WiFi High Crimes](#)

NewsForge

- [Billing error overcharges some Mandrakesoft customers by \\$1000s](#)

[Internet](#)
[Internet :: File](#)
[Transfer Protocol](#)
[\(FTP\)](#)
[Security](#)

Minor
 bugfixes

[GNU](#)
[General](#)
[Public](#)
[License](#)
[\(GPL\)](#)



[Soma suite 1.2](#)

by [Bakunin - Andrea Marchesi\[.\]](#) - Saturday, May 15th 2004 13:15 PDT

About: Soma is a suite of programs that let you play and schedule audio files from the Web. It supports extra utilities using run-time loadable modules and includes a broadcasting scheduler, a tool to control it via TCP/IP, and an utility to check configuration file syntax.

Categories	Focus	License	URLs
Multimedia :: Sound/Audio	Minor feature enhancements	GNU General Public License (GPL)	

[System Rescue CD 0.2.13](#)

by [François Dupoux](#) - Saturday, May 15th 2004 13:11 PDT

About: SystemRescueCd is a Linux system available from a bootable CDROM that provides an easy way to perform administrative tasks on your computer, such as creating and editing the partitions of the hard disk or backing up data. It contains a lot of system utilities (such as parted, partimage, and fstools), and basic programs (such as editors, midnight commander, and network tools). It also includes QtParted, a Partition Magic clone that makes

- [Two Debian project members killed in automobile accident](#)
- [You wanted two columns, you got them](#)
- [Anti-Microsoft protest planned for EP session May 17](#)
- [Seven open source business strategies for competitive advantage](#)
- [Linux Advisory Watch - May 14, 2004](#)
- [Will EJB 3.0 turn J2EE thinking on its head?](#)
- [File alteration monitoring techniques under Linux](#)
- [How AMD can cut into Intel's market dominance](#)
- [Accelerated Linux training](#)

[Slashdot](#)

- [Cryptic Code Stumps Experts](#)
- [Novell Sued Microsoft Through Caldera?](#)
- [Updated Schedule for U.S. Biometric Passports](#)
- [George Gilder on Telecommunications Policy](#)
- [Metal Velcro](#)
- [New Wave Of File-Sharing Embraces Secrecy](#)
- [Apple Files Patent for Translucent Windows](#)
- [Microsoft Blames Anti-trust Legal Fees for Price Increases](#)
- [Indie Game Jam 2004 Recounted](#)
- [Mirror.ac.uk to Scale Back Operations](#)







[ThinkGeek: What's New](#)

- [PC Mods: ThermalTake XRay](#)

editing partitions easy with its Qt graphical user interface. This CDROM aims to be very easy to use and accessible to everybody.

Changes: The kernel was updated to Linux 2.4.26 (patched with SATA support).

FrameBuffer support was improved, and should work with Dell laptops. parted was updated to 1.6.11, QtParted to 0.4.4, ntfsplogs to 1.9.2, and ClamAv to 0.70. aget (download manager), iftop (network administration tool), zile (tiny emacs editor clone), and par2cmdline (Parity Archive Volume Set v2) were added. SCSI hardware autodetection was improved. Many minor updates were made.

Categories	Focus	License	URLs
System ::	Major	GNU	  
Archiving :: Backup	bugfixes	General	  
System :: Operating		Public	
System :: Linux		License	
Distributions :: CD-Based		(GPL)	
System :: Recovery Tools			

IMMS Magical Favorites Collector 1.1







by [James 'Pug' Jones](#) - Saturday, May 15th 2004 12:46 PDT

About: IMMS Magical Favorites Collector uses IMMS collected information to find your favorite songs, and then do the hard work of compiling a CD/DVD/MP3 player full of your favorites. It can provide a top 10 songs linked to a directory of your choice, or your favorite 700 MB of songs for an MP3 CD, etc. It preserves the pathnames of files, so that the resulting collection will reflect the main collection.

- [Computing: SnapStream FireFly PC Remote](#)
- [PC Mods: Antec Super LanBoy PC Case](#)
- [PC Mods: Lian-Li PC-V1000/2000](#)
- [Cube Goodies: PixelBlocks](#)
- [Caffeine: Case o' Bawls](#)
- [Cube Goodies: Desktop Personal Air Conditioner](#)
- [Cube Goodies: Luminglass](#)
- [Gadgets: USB Devil Duckie Drive](#)
- [Cube Goodies: PixelBlocks](#)

Advertising

Changes: Some compilation and dependency errors have been resolved.






Categories	Focus	License	URLs
Multimedia :: Sound/Audio	Minor bugfixes	GNU General Public License	  
Multimedia :: Sound/Audio :: CD Audio :: CD Writing		License (GPL)	  

[svglib 1.9.19](#)

by [Matan](#) - Saturday, May 15th 2004 12:45 PDT

About: svglib is a low-level graphics library that provides VGA and SVGA modes in a console. It is not intended as an alternative to X for apps, but rather a set of tools for things like VGA games, image viewing in modes that X cannot support, etc.

Changes: The changes were mainly in the build system. Support for some new graphics chipsets was also added.

Categories	Focus	License	URLs
Software Development :: Libraries	Minor feature enhancements	Freely Distributable	 
			  








[Q-Midi 1.15](#)

by [Albert Graef](#) - Saturday, May 15th 2004 12:32 PDT

About: Q-Midi is a MIDI interface module which allows you to write MIDI applications in the Q programming language. It runs on top of Game's MidiShare package. Most basic MidiShare functionality is available, including

timing functions for realtime programming and MIDI file access. A sample MIDI player application is included (which requires Tcl/Tk).

Changes: This release adds a new midicc application and support for the new msAlsaSeq driver on Linux.

Categories	Focus	License	URLs
Multimedia ::	Minor feature	GNU	 
Sound/Audio ::	enhancements	General	 
MIDI		Public	 
Software		License	
Development ::		(GPL)	
Libraries			



dillo Web browser 0.8.1

by [Jorge](#) - Saturday, May 15th 2004 12:27 PDT

About: Dillo Web browser is a very fast, extremely small Web browser that's completely written in C. The source and binary are less than 400 kilobytes each. It is a graphical browser built upon GTK+, and it renders a good subset of HTML, excluding frames, JavaScript, and JVM support.

Changes: This version comes with an improved parser. The block/inline container model information is now an essential part of the parser. This improves both bug detection in the bug-meter and the rendering of HTML pages. Dillo now is able to detect and offer a download dialog when a clicked URL is not viewable. Several glitches of the previous release were also fixed. This version is no longer considered alpha, but very stable beta.

Categories	Focus	License	URLs
------------	-------	---------	------

[Internet :: WWW/
HTTP](#)
[Internet :: WWW/
HTTP :: Browsers
Utilities](#)

Major feature
enhancements

[GNU
General
Public
License
\(GPL\)](#)



IaraJS 1.1.1

by [Cárlisson Galdino](#) - Saturday, May 15th 2004

12:05 PDT

About: IaraJS is a set of JavaScript templates for the construction of Web sites, using JavaScript components.

Changes: A bug was fixed in iSpeedButton. Multi-selection support has been added to iSelect. iPoem has been improved. New images (made with free software) has replaced the previous images in iEmergentNews.





Categories	Focus	License	URLs
Internet :: WWW/ HTTP	Minor bugfixes	GNU General Public License (GPL)	
Internet :: WWW/ HTTP :: Dynamic Content			

LNKSRC 0.0.4 (Alpha)

by [LiNkCr](#) - Saturday, May 15th 2004 04:53 PDT

About: LNKSRC is a new packaging system for Linux. It installs and removes generic source tarballs and slackware (tgz) packages. It keep track of your installed programs and libraries.

Changes: This version contains code clean-ups and some bug fixes.

Focus	License	URLs
Minor bugfixes	GNU General Public License (GPL)	   





XStream 1.0

by [Joe Walnes](#) - Saturday, May 15th 2004 04:52

PDT

About: XStream is a simple library to serialize objects to XML and back again. It is easy to use, does not require the definition of mappings, is fast, and integrates with other XML APIs. It features customizable conversion strategies.

Changes: Initial release.

Focus	License	URLs
Initial freshmeat announcement	BSD License (revised)	   

TCB::Internal 1.02

by [Tim Skirvin](#) - Saturday, May 15th 2004 04:52

PDT

About: This pseudo-module is used by TCB web scripts to offer a consistent user interface to the outside world.

Changes: Initial release.

Categories	Focus	License	URLs
------------	-------	---------	------

[Internet :: WWW/](#)
[HTTP :: Dynamic](#)
[Content :: CGI](#)
[Tools/Libraries](#)
[Internet :: WWW/](#)
[HTTP :: Site](#)
[Management](#)
[Software](#)
[Development ::](#)
[Libraries :: Perl](#)
[Modules](#)

Initial
freshmeat
announcement

[BSD](#)
[License](#)
[\(revised\)](#)









TCB::System::Obsolete 0.99

by [Tim Skirvin](#) - Saturday, May 15th 2004 04:52

PDT

About: TCB::System::Obsolete is a set of tables for TCB::System that are obsolete and no longer used, but might as well still be accessible.

Changes: Initial public release.

Categories	Focus	License	URLs
Internet :: WWW/	Initial	BSD	 
HTTP :: Dynamic	freshmeat	License	 
Content :: CGI	announcement	(revised)	 
Tools/Libraries			
Software			
Development ::			
Libraries :: Perl			
Modules			

Host Grapher 2 (II)







by [Joao Carvalho](#) - Saturday, May 15th 2004

04:47 PDT

About: Host Grapher is a very simple collection

of Perl scripts that provide graphical display of CPU, memory, and process information for a system. There are clients for Windows, Linux, FreeBSD, and Tru64. No socket will be opened on the client, nor will SNMP be used for obtaining the data.

Changes: The system now uses plugins. Individual passwords can be assigned to different hosts. This version supports Linux and FreeBSD, with support for other operating systems to follow.

Categories	Focus	License	URLs
<u>System ::</u> <u>Monitoring</u>	Major feature enhancements	<u>GNU</u> <u>General</u> <u>Public</u> <u>License</u> (GPL)	     

[\[yesterday's page\]](#) [\[today's page\]](#) [\[the rest of today\]](#)

Tip: Tired of seeing a lot of software not of the slightest interest to you? Get a [freshmeat user account](#) with full filtering capabilities. Best of all: It's **free!**

© Copyright 2004 [OSDN](#) Open Source Development Network, All Rights Reserved.

About [freshmeat.net](#) • About [OSDN](#) • [Privacy Statement](#) • [Terms of Use](#) • [Advertise](#) • [Self Serve Ad System](#) • [Contact Us](#) • [Revision](#): v2.5.99


[my sf.net](#)
[software map](#)
[donate to sf.net](#)
[about sf.net](#)
[Login via SSL](#)
[New User via SSL](#)

Search

SF.net Subscription

- [Subscribe Now](#)
- [Manage Subscription](#)
- [Advanced Search](#)
- [Direct Download](#)
- [Priority Tech Support](#)
- [Project Monitoring](#)

SF.net Resources

- [Site Docs](#)
- [Site Status \(05/03\)](#)
- [Site Map](#)
- [SF.net Supporters](#)
- [Compile Farm](#)
- [Foundries](#)
- [Project Help Wanted](#)
- [New Releases](#)
- [Get Support](#)

Site Sponsors


[Linux Resources](#)

SourceForge.net is the world's largest Open Source software development website, with the largest repository of Open Source code and applications available on the Internet. SourceForge.net provides free services to Open Source developers.




Project of the Month

Every month the SF.net team picks one outstanding project to highlight the software and personality that drive the Open Source Community!

Project of the Month for May, 2004:
[eGroupWare](#)

Be a Power User!

 Become a [SF.net Subscriber](#). For only \$39/year you'll get a host of premium services and functionality, including advanced searching, priority tech support, project monitoring and [more](#). And for a limited time, you'll receive a free [ThinkGeek T-Shirt](#) of your choice with your [subscription](#). [Find out more](#).

Two ways to give back to SF.net

SourceForge.net Statistics

Hosted Projects: **81,203**
 Registered Users: **847,981**

SourceForge.net Newsletter

Email Address:

HTML Text

Latest News

[jMax 4.1.0 released](#)

p_tisserand - 2004-05-07 05:59 - [jMax](#)
 jMax 4.1.0 have been released. jMax is a visual programming environment for building interactive real-time musical and multimedia applications. This release is the first release with ASIO support for Windows. There is also a lot of bug fixes.

[\[Read More/Comment\]](#)

[XDoclet v1.2.1 released](#)

stevensa - 2004-05-07 05:57 - [XDoclet](#)
 XDoclet is a java code generator producing source code and other files (e.g. XML configuration files) from custom javadoc @tags in your code, enabling attribute-oriented programming. v1.2.1 is mainly a bug fix

Most Active

- 1 Compiere ERP + CRM Business Solution
- 2 Gaim
- 3 phpMyAdmin
- 4 PearPC - PowerPC Emulator
- 5 Azureus - BitTorrent Client
- 6 WinMerge
- 7 eGroupWare: Enterprise Collaboration
- 8 POPFile - Automatic Email Classification
- 9 FCKeditor
- 10 FileZilla

[More Activity >>](#)

Top Downloads

- 1 eMule
- 2 BitTorrent
- 3 DC++
- 4 Azureus - BitTorrent Client
- 5 phpBB
- 6 CDex
- 7 VirtualDub
- 8 PearPC - PowerPC Emulator
- 9 AMSN
- 10 eMule Plus

[More Statistics >>](#)

SF.net Services



Do you love SourceForge.net and what we do for the Open Source community? There are two ways to support the site.

1) **Donate!** Join the growing number of users making contributions to enhance SF.net's feature set and functionality. If SF.net has helped you in any way, consider [giving back](#) to the site so we can continue to enhance the features/mirrors/servers critical to the Open Source Community.

2) **Support Your Favorite Project!** Do you use [GAIM](#) daily? Does your website depend on [AWStats](#)? If you see a "donate" link on a project's summary page, and you use that project regularly, consider supporting that project so that the project admins can spend more time doing what they do best - developing great Open Source software. Besides doing a good deed for the community, you'll also be listed on the project's contributors page.

Enterprise

SourceForge for Corporate Use



SourceForge Enterprise Edition brings the power of SourceForge's award-winning software collaboration environment behind your firewall. With major feature extensions, third party SCM and IDE support, real-time project tracking and reporting, and enterprise grade security and stability, SourceForge Enterprise Edition helps project managers and distributed development teams produce better results in less time.

release, plus a couple of new modules have been added to support the Spring framework and OpenEJB application server.

[\[Read More/Comment\]](#)

eGroupWare 1.0 RC5 released

reinerj - 2004-05-04 06:49 - [eGroupWare: Enterprise Collaboration](#)

eGroupWare: Enterprise Collaboration

eGroupWare is the May, 2004 Project of the Month. eGroupWare is a multi-user, web-based groupware suite developed on a custom set of PHP-based APIs. Currently available modules include: email, addressbook, calendar, infolog (notes, to-do's, phone calls), content management, forum, bookmarks, wiki. eGroupWare RC5 is the next step to the final 1.0 release. Many people wait for the upcoming 1.0. The developers work hard to fix the last bugs. We hope to bring out final a RC6 in next 3 weeks and 10 days later the 1.0 release. We sure that eGroupWare RC5 is stable and recommend to update to this release.

(1 Comment) [\[Read More/Comment\]](#)

JGraph Paris (v3.4) released

alderg - 2004-05-04 06:46 - [JGraph Diagram Component](#)

JGraph Diagram Component

This release can handle overlapping edges, and has static inner handles for better subclassing. Among other minor API changes some control methods were moved to the handles. JGraph is the most powerful, lightweight, feature-rich, and thoroughly documented open-source graph component available for Java. It is accompanied by JGraphpad, the first free diagram editor for Java that offers XML, Drag and Drop and much more!

(1 Comment) [\[Read More/Comment\]](#)

MediaWiki 1.2.5 released

vibber - 2004-05-04 06:45 - [MediaWiki](#)

MediaWiki is the collaborative editing software that runs Wikipedia, the free encyclopedia, and other projects. It's designed to handle a large number of users and pages without imposing too rigid a structure or workflow.

[\[Read More/Comment\]](#)

- [Tech Jobs](#)
- [Online Books](#)
- [Comparison Shop](#)
- [Partner Product Offers](#)
- [Get Broadband](#)

Sponsored Content

[learn more >>](#)

- [STAF V2.6.2 is now available](#) 2004-05-04 06:45
- [SOURCEFORGE.NET UPDATE - 2004-04-22 EDITION](#) 2004-04-23 16:56
- [Effective PR for Open Source projects \(2004-04-24\)](#) 2004-04-23 09:49
- [ClamWin 0.30 released](#) 2004-04-23 04:43
- [ClamAV port for Cygwin environment available](#) 2004-04-23 04:42

[\[News archive\]](#)

Powered by [SourceForge\(tm\)](#) collaborative software development tools from VA Software

© Copyright 2004 - [OSDN](#) Open Source Development Network, All Rights Reserved

[About SourceForge.net](#) • [About OSDN](#) • [Privacy Statement](#) • [Terms of Use](#) • [Advertise](#) • [Self Serve Ad System](#) • [Get Support](#)



[Site Index](#) | [Trademarks/Graphics](#) | [F.A.Q.](#) | [Search](#) | ["Open Source" Swag](#)

:: [OSI News](#) - May 15 2004 ::

- [Weather.com Moves to Open Source.](#)
- [BayStar Capital Want Shares Cashed Due to Breach](#)
- [BayStar Has Now Confirmed MS Ties.](#)
- [Business Week Confirms MS Orchestrated Hedge Fund Investment in SCO](#)
- [Business Week Confirms MS Orchestrated Hedge Fund Investment in SCO](#)
- [MS & SCO May be under SEC investigation.](#)
- [Computer Associates Denies Having Bought SCO License.](#)
- [Halloween X: Follow The Money -UPDATE-](#)
- [CIO Magazine: The Myths of Open Source](#)
- [Five Newly Approved Licenses](#)
- [OSI News Weblog Launch](#)

[XML](#)

:: **Where to get Open Source News** ::

[OSDir.com](#)
[NewsForge](#)
[Linux Weekly News \(LWN\)](#)

:: **Quick Links** ::

- * [The Open Source Definition](#)
- * [Open Source Licenses](#)
- * [Open Source Awards](#)
- * [The Halloween Docs](#)
- * [O.S. for Customers, Business, and Programmers](#)
- * [Announcements Mailing List](#)
- * [Site Index](#)
- * [Graphics/Trademarks](#)



[Site Index](#) | [Trademarks/Graphics](#) | [F.A.Q.](#) | [Search](#) | ["Open Source" Swag](#)

::: Advocacy Index :::

*[List of Recommended Reading](#)

*[The Case for Open Source: For Business](#)

*[The Case for Open Source: For Customers](#)

*[The Case for Open Source: For Hackers](#)

*[Case Studies and Press Coverage](#)

*[Frequently Asked Questions](#)

*[Jobs for Hackers: Yes, You Can Eat Open Source](#)

*[Software Secrets: Do They Help or Hurt?](#)

*[Why "Free](#)

The Open Source Case for Customers

Beyond all the reliability and quality gains [we've discussed elsewhere](#), the open-source model has one overwhelming advantage for the software customer: you aren't a prisoner.

Because you can get access to source, you can survive the collapse of your vendor. You're no longer totally at the mercy of unfixed bugs. You're not shackled to every strategic decision your vendor makes. And if your vendor's support fees become exorbitant, you can buy support from elsewhere.

For this reason alone, every software customer should absolutely demand open source and refuse to deal with software vendors who close and shroud their code. It's a matter of controlling your own destiny.

(And yes, we'll say the M-word ... don't you *want* to be out from under Microsoft's thumb?)

You Are Your Developers' Customer!

This customer case for open source may apply even if the software you're concerned about was developed internally and never for sale on the open market.

We hear of one case, for example, in which a few employees at a large Internet-equipment manufacture developed a distributed printing spooler that is very

["Software" Is Too Ambiguous](#)

[*Shared Source: A Dangerous Virus](#)

important for company use, but completely unrelated to the company's normal expertise or line of business. Now ... what happens when those employees leave?

Under the closed model, this large company would be stuck ... with decaying software or an expensive retraining job.

But now imagine that the company released the spooler as open source and helped it find an interest community on the Internet. Now, when the developers leave, someone else might step in at no monetary cost to take over the software. At the very least, there's a known pool of people with an interest from which the company might hire replacements.

Open source empowers the customer, even when the producer and customer are part of the same firm.

Freedom from Legal Entanglements

Using most commercial software involves software licenses, and tracking software copies and usage. This demands record keeping, and legal exposure. Both raise costs. Thus, juggling software licenses and copies is a source of costs to businesses, and legal risk to businesses and individuals.

In many (most? all?) businesses, such tracking is imperfect, sometimes intentionally, usually not. Any such imperfection exposes the guilty party to legal actions (fines, litigation, arrest) due to breaking laws and violating copyrights; an intellectual property quagmire.

Most/all open source software can be freely copied and used. There are no licenses to track and thus no related costs, or legal risks.

Standard Objections

There are a couple of standard customer objections to the open-source model that deserve to be exploded. We

cover these on the [Frequently Asked Questions](#) list.

You probably also want to look at the [business case for open source](#) for discussion of the reliability gains from this model.

Copyright © 2004 by the [Open Source Initiative](#)
Technical questions about the website go to Steve M.:
[webmaster at opensource.org](mailto:webmaster@opensource.org) / **Policy questions** about
open source go to the [Board of Directors](#).

The contents of this website are licensed under the [Open Software License 2.0](#) or [Academic Free License 2.0](#)

[OSI is a registered non-profit with 501\(c\)\(3\) status.](#)
[Contact](#) our [Board](#) for further donation information.

Open Source Software Institute

[Home](#) | [Programs](#) | [Members](#) | [Press and Media](#) | [FAQ](#) | [References/Links](#) | [OSSI-News](#) | [Federal Government](#) | [State and Local](#) | [Academics](#) | [Contacts](#) |

Home

Saturday, 15 May 2004

Forbes and the Open Source
Software Institute



2004 Cornerstone
Sponsor



[Corporate Members](#)
[Government Members](#)
[Academic Members](#)

Host University



University of Southern Mississippi
Hattiesburg, MS

Welcome & Acknowledgement

What is OSSI? The Open-Source Software Institute (OSSI) is a non-profit (501 c 6) organization comprised of corporate, government and academic representatives whose mission is to promote the development and implementation of open-source software solutions within U.S. Federal and State government agencies and academic entities.

[OSSI Position Paper](#)

BE SURE TO FOLLOW OUR DAILY NEWS SITES:

[OSSI-News.org](#)

[GovernmentForge.org](#)

OpenSSL Project Update

On Monday, May 10, 2004, the National Institute of Standards and Technology (NIST) posted a notice that the AES, DES, 3DES, DSA and SHA-1 algorithms for OpenSSL have been validated.

The validation notices can be found at the following NIST sites:

[Advanced Encryption Standard \(AES\) Algorithm: Certification #146](#)

[Data Encryption Standard \(DES\) Validated Implementations: Cert #258](#)

[Triple Data Encryption Algorithm \(TDEA, a.k.a. "Triple DES"\): Cert #256](#)

[Digital Signature Algorithm \(DSA\) Validation System: Cert #108](#)

[Secure Hash Algorithm \(SHS\) Validation System: Cert #235](#)

Successful validation of these algorithms does NOT mean that OpenSSL has received FIPS 140-2 validation, yet. The overall FIPS 140-2 validation effort for OpenSSL is still in process. Additional updates will be posted on the OSSI web site, www.oss-institute.org.

NIST validation of these algorithms does, however, signify a major milestone in our efforts to secure the FIPS 140-2 validation for OpenSSL.

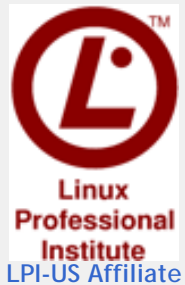
Please post any questions that you might have to questions@oss-institute.org.



Technical Office
PO Box 5022
Hattiesburg, MS 39402
Voice: 601.266.6034
Fax: 601.266.6071

Administrative Office
PO Box 547
Oxford, MS 38655
Voice: 662.236.1794
Fax: 662.236.9892

OGIP
Open Government
Interoperability Standard
**Open Government
Interoperability Standard**



OSSI was established to serve as a forum for the exchange of information and promotion of ideals embodied in open-source software.

At OSSI, we prefer to fly Southwest

Download a PDF of this announcement [here](#).

Open Source Software Institute Becomes the LPI-US Affiliate

The Linux Professional Institute (LPI), (www.lpi.org), the premier international professional certification program for the Linux community, has affiliated with Open Source Software Institute (OSSI) (www.oss-institute.org) to increase exposure and participation of Linux by corporate, government and academic environments across the United States.

[Press Release Announcing the Affiliation](#)

Open Source Software: Perspectives for Development

December 16, 2003

by Paul Dravis
The Dravis Group

The report intends to assist decision makers, globally, in better understanding Open Source software when assessing this technology option. Presented are initiatives by governments around the world, a selection of private sector uses of Open Source, support by commercial technology providers, a set of case studies in developing countries, along with a brief status of the legal landscape.

[The Dravis Group's PDF](#)

NIST/FIPS 140-2

December 8, 2003

[OpenSSL FIPS 140 FAQ](#)

[NIST/FIPS 140-2 Certification for OpenSSL](#)

[See PRESS RELEASE](#)

[NIST FIPS 140-2 Pre-Validation List](#)

[Information Assurance Ramifications of Using OpenSSL in the Department of Defense Computing Environment](#)

Project Leopard

Open Source Software Institute Releases Components to eGovernment Web Services Platform; Initiates Working Group for Open Government Interoperability Standards.

[Click here for HTML](#)
[Click here for PDF](#)





Cooperative Research and Development Agreement

NAVOCEANO CRADA
Executive Summary

NAVOCEANO CRADA Overview



OSSI exists to serve as:

An Open-Source ADVOCATE OSSI serves as a collective resource and venue for the promotion, development and implementation of open-source software solutions between corporate, government and academic entities.

A Research and Development FACILITATOR OSSI serves as a facilitator and coordinator of R&D projects, systems and application evaluation and development and academic studies regarding technical, policy, economic/market and legal aspects of open-source software.

An Open-Source POLICY CONSORTIUM OSSI serves as a forum for working committees of industry, governmental and academic representatives to address open-source issues in regards to government/industry standards, academic research, economic/market and legislative policy.



Interested in joining OSSI? ... [click here](#)
E-mail: Questions@oss-institute.org
[Contact Information](#)

Copyright 2004, Open Source Software
Institute.



Home

Hot Topics

- :: Active Directory
- :: Exchange & Outlook
- :: Migration
- :: Mobile & Wireless
- :: Networking
- :: Scripting
- :: Security
- :: Small Business Center
- :: SQL Server Magazine
- :: Storage
- :: Web Administration
- :: **All Articles by Subject**

Recent Articles

Publications Archive

- :: Windows & .NET Magazine
- :: SQL Server Magazine
- :: Exchange & Outlook Administrator
- :: Windows Scripting Solutions
- :: Security Administrator
- :: Email Newsletters
- :: **All Publications**

Forums

- :: General - Windows
- :: Windows OSs
- :: Windows Server Systems
- :: Other Computing - Windows
- :: Development
- :: SQL Server_
- :: General - SQL_
- :: SQL Server Magazine Web Site Redesign_
- :: Server 2000 / 7.0_
- :: SQL Server Yukon Beta_
- :: SQL Server 6.x_
- :: CertTutor_
- :: General - CertTutor_
- :: MCSE Exams_
- :: The Real World_
- :: Other Certifications and Technologies_

Events

- :: Web Seminars
- :: Roadshows
- :: Conferences
- :: **All Events**

My Account

[Advanced Search](#)

LOG ON

Driving the Enterprise Network :: NT :: 2000 :: XP :: 2003

[Email this Article](#) [Printer-Friendly](#) [RSS](#) [Subscribe to Windows & .NET Magazine RSS feed](#)

[April 3, 1999]

Mozilla evangelist: Open source Netscape a complete failure



Paul Thurrott
InstantDoc #18599
Paul Thurrott's WinInfo

Mozilla evangelist and Netscape engineer Jamie Zawinski resigned from the open source project "Mozilla.org" this week, citing the project's complete failure to deliver a usable product. Zawinski, who spearheaded the project during its first year, also threw some flak at Netscape purchaser America Online (AOL), which he describes as contrary to the goals of the Internet and open source software. He also blames Microsoft for destroying Netscape's greatest product, Navigator, and ruining the market for Web browsers.

"For whatever reason, [Mozilla] was not adopted by the outside," he wrote in his resignation letter. "In my humble but correct opinion, we should have shipped Netscape Navigator 5.0 no later than six months after the source code was released. But we couldn't figure out a way to make that happen. I accept my share of responsibility for this, and consider this a personal failure."

"Here we are, a year later. And we haven't even shipped a beta yet," he continues. Actually, it's been about 15 months since the Mozilla project was started. "The fact is, there has been very little contribution from people who don't work for Netscape, making the distinction (between Netscape and Mozilla)

Top 20 Viewed Articles

- 1 [How can I uninstall the Microsoft Java Virtual Machine \(JVM\) from Windows XP?](#)
- 2 [What You Need to Know About Windows Update Services](#)
- 3 [Microsoft: No XP SP2 for Pirated Copies](#)
- 4 [WinInfo Short Takes: Week of May 17](#)
- 5 [Update: Problems with Microsoft's Patch MS04-011](#)

[More Top Viewed Articles](#)



somewhat academic."

"And so I'm giving up," he concludes. "The Mozilla project has become too depressing, and too painful, for me to continue working on."

Though Zawinski is careful to mention that the problems with Mozilla are not endemic to open source software in general, one has to wonder. Aside from Linux, Mozilla is the biggest and most well-known open source project there is, and it has the distinction of being able to affect users of almost every computer system out there. The complete lack of enthusiasm for continuing Netscape's legacy is perhaps telling. In any event, Zawinski's resignation letter is a fascinating take on the failure of the most well publicized attempt to prevent Microsoft from owning a market. Check it out at [Jamie Zawinski's home page](#).

Post a Comment

We want to hear what you have to say about this article!



We review comments before posting them, so it may take a day or so for your comments to appear. Your email is only used if our editors need to contact you. It is not used or stored for any other purpose, nor posted with your comments.

Sponsored Links

- [Free Message-level Exchange Recovery White Paper](#)
- [Try Diskeeper® FREE—get optimum performance on PCs and servers!](#)
- [Get more from mobile apps -----> MSS Smart IP® FREE for 30 days!](#)
- [Spam threatens your network. Learn how to protect your company](#)
- [Free certification exam on Oracle 10g! Click here.](#)
- [Limited-time Microsoft® Windows Server™ 2003 software offer.](#)
- [Download the ultimate companion to Windows Terminal Services-FREE](#)

Featured Links

- [Get 2 free sample issues of Windows & .NET Magazine](#)
- [Microsoft Webcasts: essential guidance, industry experts](#)
- [NEW Exchange 2003 Seminar Series: Sign Up for Free](#)
- [Sample two issues of SQL Server Magazine - click here!](#)

Our Other Websites: [CertTutor](#) | [Connected Home](#) | [JSI FAQ](#) | [IT Library/eBooks](#) | [SuperSite](#) | [Windows FAQ](#) | [WinInfo News](#)

[Home](#) | [Subscribe / Register](#) | [About Us](#) | [Contact Us / Customer Service](#) | [Affiliates / Licensing](#) | [Press Room](#) | [Media Kit](#) | [RSS](#)

Windows & .NET Magazine Network is a Division of Penton Media Inc.
Copyright © 2004 Penton Media, Inc., All rights reserved. [Legal](#) | [Privacy Policy](#)

