The Sourceware Operating System Proposal

Revision 1.8

Larry McVoy
and a cast of thousands, see acknowledgments

lm@sun.com
+1-415-336-7627
Sun Microsystems Computer Corporation[1]

ABSTRACT

This document describes a proposal to provide a source form, royalty free Unix as an evolution of the COSE effort, as a means of unifying the Unix desktop market, and as an application deployment platform, with a focus on running all applications, including those from other operating systems, such as DOS and Windows 3.1. This effort is intended to provide substance to the many Unix unification and standard agreements that exist today. Significant effort has been made to address the concerns of the major Unix vendors, the Unix customer base, the DOS customer base, the Windows 3.1 customer base, the educational and research community, and the development community.

1. Introduction

This document is an assessment of the condition of Unix, and a proposal to improve the condition of Unix on the desktop. To get a quick reading, the reader may scan for the high lighted bars; they are a summary of the key points of each section.

The organization of the document is background on the state of Unix, background on the efforts to fix Unix, a digression on why bother with fixing Unix, a suggestion for how to start Unix on the path to healthiness, more details on the health plan, details on managing the resulting system, alternatives to this plan, questions and answers, and finally, acknowledgments.

[1] Neither Larry McVoy nor the views in this document are necessarily representing the views of Sun Microsystems or Sun's affiliates. Sun's tolerance in this matter is gratefully acknowledged.

2. Unix in trouble

Unix needs our help because Unix is dieing. Unix is no longer even close to competitive.

- Unix development in the industry is in bad shape. The major Unix vendors currently spend about $1,000,000,000 a year on Unix "development." A great deal of this expenditure is redundant over the set of vendors. Microsoft spends much less than this and produces a better product.[2]
- The Unix solution is fragmented. The major Unix vendors each provide their own version of Unix, resulting in at least ten major Unix systems[3] all competing for about three percent of the computer market. Microsoft has one implementation each of DOS, Windows, and Windows/NT. Windows/NT system runs on multiple platforms and compatibility is

[2] Their product is better in terms of ease of use, installation, and administration.

guaranteed by virtue of one source base.

- Unix is too expensive. Licensing ranges from $20 to $100 per seat, with vendor mark up for their costly "value add" resulting in customer seat costs of $600 - $3000. Microsoft sells Windows/NT for about $150.
- Unix has become stagnant. Unix has ceased to be the platform of choice for the development and deployment of innovative technology. A great deal of the early development of Unix was done by researchers because of Unix's ready accessibility. As time has gone on, it has become more and more difficult for research organizations to acquire source. Microsoft is planning on releasing Windows/NT to universities in an attempt to leverage from the same sources of innovation.
- Unix has become large and complex. Obsession with the wrong sorts of compatibility (i.e., unused features) has lead to a bloated, hard to maintain, Unix source base. Microsoft pushes DOS and Windows which essentially amount to a program loader and a user interface. All other applications are just that, applications.
- Unix has become too acrimonious. Rather than working together to grow the Unix market place, vendors become "religious" about their particular version of Unix. No two vendors can agree on a version, each thinks their version is better when in fact, they are all 98% identical in terms of the programming interfaces that are actually used. Microsoft is the only winner when the Unix vendors continue to disagree.
- Vendor consortiums and standards agreements for Unix are not working. While Unix vendors agree to agree, Microsoft wins by having only one source base. Microsoft will be compatible with itself, but Sun is unlikely to be compatible with other Unix vendors so long as there are multiple implementations of the same ideas.

[3] SCO, BSD, SunOS 4.x, SunOS 5.x, Unixware, HP-UX, AIX, Ultrix, OSF/1, and IRIX.

3. What is being done?

The Unix problems are not being addressed. The vendors think "standards" are the answers. The programmers think "free" software is the answer. The customers think NT is the answer.

There are two efforts under way to address Unix's problems. The vendors have one answer and the developers have another. The proposal we will make may be viewed as a merging of these two answers, preserving the best features of each.

3.1. The vendor answer

The vendors periodically respond to the problems with Unix by signing "standardization" agreements. This has been going on for years. There are several standards that all promise the same thing: POSIX, XPG, SVID, and now COSE. The customers are not buying it because they have seen it all before.

One effect of the customers lack of interest is increased desperation on the part of the vendors. It is becoming increasingly obvious that the Unix community has either to shape up and become competitive in the computer industry or eventually they will die. If it can happen to IBM, it can certainly happen to Unix. Since that means that a large number of Unix employees would be out of work, perhaps permanently, the executives are much more cooperative than ever before. All they want is a reasonable plan.

3.2. The developer answer

Unix's traditional research and development community is reluctant to develop proprietary code since vendors typically try to draw value from it by locking it up. It is pointless to explain to the vendor that software that is widely available and royalty free is more useful and valuable to the end user than proprietary software.

More and more traditional Unix developers have given up on the proprietary Unix implementations and are focusing their energy on sourceware. The results of their labors are apparent in the form of GNU software, Sun's

RPC system, the Linux operating system, the X11 window system, the free Berkeley based Unix operating system, and the Cygnus Support organization. For those that are unfamiliar with these projects, a brief description follows.

- GNU software
  The Free Software Foundation is the oldest and most well known advocate of publicly available software. The FSF covers their software with a special copyright, called the *GNU General Public License*, which is commonly referred to as the *Copyleft*. The license is designed such that the software will always be freely available, in source or binary form, to anyone that wants it.[4] Well known FSF projects include the GNU C compiler called GCC, a popular editor called Emacs, a multitude of programmer utilities (mostly reimplementations of the Unix utilities), etc.
- Sun's RPC system
  As part of the NFS effort, Sun developed the remote procedure call (RPC) system. This system has several shortcomings, but is the most widely used RPC system in the world. The reason? Sun gave it away in source form, with a license that states "users may copy or modify Sun RPC without charge."
- Linux operating system
  Linux is a small, essentially POSIX compliant, operating system for Intel i386 (and above) platforms, written from scratch by a 22 year old Finnish student named Linus Torvalds. It is extremely popular, largely because it is "copylefted." It is claimed the Linux usage runs into the millions of seats. There are device drivers for almost all popular devices (Linux is the only Unix that supports some PC devices), many features found only in mature operating system (stacking file systems), etc. Linux is feature rich but lacks the quality and stability of a commercial product.
- X11 window system
  The MIT X Window system is the de facto standard windowing environment on Unix. It is free source code, although it is not copylefted. It is wildly successful, not because of technical superiority, but because all of the Unix vendors have compatible versions. Most vendors are finding that the best way to stay compatible is to simply take what MIT delivers and ship it. The fact that users and vendors are continually contributing their changes back to the free X source base is a good rebuttal to the claim that free software stagnates.
- Free BSD operating systems
  There are several derivatives of the Berkeley Net/2 release.[5] They started with 386BSD, put together by Bill Jolitz and friends, and have moved forward into NetBSD and FreeBSD. Development here has not been as active as Linux because of (a) the fear of USL lawsuits and (b) acrimony between competing Unix egos.
- Cygnus Support
  Cygnus is the first example of a successful, for profit, business based solely on sourceware. Their business is to provide support. Their revenue stream is substantial and is doubling each year. They have about 50 employees and support development tools, primarily the GNU GCC C/C++ compiler and debugger. They were recently written up as one of the "coolest companies" by Fortune Magazine[6].

4. Why bother to fix Unix?

Unix is a mature platform, with a mature group of programmers. Fixing Unix is easier than living with NT.

If Unix is so hard to develop, so fragmented, so expensive, so acrimonious, and so feature laden it is about to die, why bother attempting to save it?

There are three reasons: consideration of the alternatives to Unix, the observation that Unix provides the best existing basis for a desktop OS today, and the existence of a small, but talented, pool of Unix hackers.

[4] The copyleft is fairly large and complicated, contact the author or any FTP site for a copy to learn more.

[5] This is the release that is supposedly AT&T code free.

[6] Perhaps because they are profitable.

4.1. Alternatives to Unix

Microsoft is the only other real alternative. If we could be guaranteed a cheap, reliable, extendible, full source operating system from Microsoft, the game would be over. There are problems with this, however.

- Windows/NT is not a complete answer. It needs a lot of work, work that has already taken place in many of the Unix systems: high speed networking, real internet support, downsizing of legacy systems, and multiple operating system emulation are all examples of problems that some version of Unix has already solved.
- Unix is a better OS base for future work than Windows/NT. For all its bloat, Unix is more salvageable than NT.
- Microsoft can be trusted only to serve their own interest in dominating the industry. They are a money making machine and do not focus on the interests of anyone other than Microsoft. IBM was the last monopoly and we have seen that that model does not work.
- Microsoft is back pedaling from the source access commitment. They said they would do it but have not yet delivered.
- Microsoft is not going to license its technology cheaply.

4.2. Unix on its merits

Customers do not care what OS they run. They care what applications they run. They couldn't care less about the operating system so long as it is cheap, works, is small, fast, and runs their applications from their other operating systems such as DOS, Windows, SCO.

If the customers do not care which OS they run, then why Unix? Because Unix works. There is a modified version of Unix called NetBSD[7] that runs SCO 286 & 386, Interactive, Cubix, Linux, Unixware, DOS, Windows, and native applications. This could be, from a feature point of view, the ultimate desktop system.

Many operating systems have promised features such as this. Only Unix, and in particular, a free version of Unix, has ever delivered. Only Unix has the suitable framework on which to build multiple, coexisting, interoperating, cooperating applications that were written and compiled for entirely different operating environments. It is certainly true that DOS has a larger market share. If Unix can run DOS applications next to Windows applications next to Unix applications, does that not speak well for Unix as a technically and financially superior operating system?

Finally, researchers and developers need an OS in which to try out new ideas. Traditionally, that OS has been Unix because it provides the best basis for forward thinking work. There is still much work to be done in the areas of high performance networking, clusters, multi media, security, mobile computing, personal assistants, etc. This work is far easier and will be accomplished far faster if the core OS is readily available.

4.3. The hackers

There is a small but crucial group of people that specifically want Unix. These people are the "hackers" and should not be discounted. Almost every good feature in computer operating systems today, including most features in DOS, Windows, and Windows/NT, came from the mind of one hacker or another. Typically, the work was not commissioned by a company, it was done as a research project and then productized. Without these people, we make no forward progress. These are the sort of people that made NetBSD run all of those different applications. This sort of talent, energy, and enthusiasm can not be bought.

The hacker community, which includes universities, research laboratories, and many people at private corporations, is interested in the best, most widely used, free software they can get. Software source code is to a hacker as speech is to other people. Since sourceware is the only software that is exchanged in source (as well as binary) form, hackers are typically uninterested in expending their considerable talents on anything other than sourceware.

[7] NetBSD is a Berkeley Unix derivative. This version has been modified by Terry Lambert for SCO, Interactive, Unixware compatibility.

For the business community to dismiss the hackers as “nuts” is a disaster. The hackers are the people that build things of value in the Unix world. To dismiss the hackers is to dismiss products that produce financial reward. It is far better to figure out a way to allow the business world and the hacker world to coexist and benefit one another. If businesses want the best technology, then businesses should move towards sourceware.

5. How do we fix Unix?

Unix needs two things: to be much smaller and focussed on running as many applications as possible; and to be available in source form.

5.1. Smaller Unix

Unix has always tried to be all things to all people. This has lead to many features that, while appropriate for servers, have no place in a desktop system. The server Unix system probably needs to remain the large, complicated system that it is today. The desktop does not need to be that complicated, in fact, DOS, Windows, and Linux are all examples of systems that provide similar function with far fewer concepts, interfaces, programs, and resource usage.

We have known that Unix is too big and complicated for years but have failed to rectify the situation. The reason is a false fear of compatibility problems. The focus has traditionally been on maintaining interfaces without any justification for the existence of the interfaces. We need to focus on running applications and only running applications. We need to further focus on compatibility for naive applications only, not for applications that grovel about in /dev/kmem or use some ancient ioctl(). Most of those applications are provided as part of the base system and, as such, need not be concerned with compatibility. It is only third party applications that we need be concerned with.

The interesting thing about compatibility is that most applications use a tiny percentage of the interfaces available to them. The reason is portability - application vendors use the least number of interfaces possible so that they have the least amount of work to do to port their application.

We believe that most Unix applications can survive with a smaller set of interfaces, and we intend to prove it. The algorithm for making Unix smaller is: if you don’t know why it is there, then take it out. Every interface, program, configuration file, literally everything, is subject to the rule: if you don’t know why it is there then take it out. Put it back if an important application actually needs it.

It is relatively easy to see what interfaces are important: the test is to take a new, smaller, kernel and library set and boot it on an old system. If the old binaries don’t break, you have a pretty good start. This system was successfully used to test compatibility when I added POSIX conformance to SunOS 4.1; I regularly booted a 4.1 kernel on a 4.0 system for testing purposes.

5.2. Sourceware

The availability of Unix source will grow the market and attract the best people to Unix development.

Sourceware is software that is either copylefted or is at least freely redistributable. Sourceware is not really zero cost software, it is accessible software. The revenue comes from support, from system integration, from having a large company standing behind the software and shipping it. Many companies ship sourceware today in the form of the TCP/IP networking code from Berkeley, the MIT X window system (including Sun’s OpenLook extensions), the GNU C/C++ compiler, etc. All of these companies are making money on sourceware. Frequently, the sourceware is in better shape than the proprietary software.

Businesses are frequently confused when facing sourceware for the first time. It is difficult for a business to understand how they make money from something that is free.

The answer is support. Customers do not care that much about who owns the software, they care who is going to support the software.

It is taken as a given that there will be problems with software and someone needs to be constantly available to address those problems. The best software in the world is worthless to 99% of the customers unless that software is supported. Hackers make up less than 1% of the customer base and only a hacker will be interested in unsupported software.

Cygnus Support is an example of a company that is working using this model. They are four years old, doubling their revenue stream each year, and they only do sourceware.

The point is that free software is not really free. The company that supports the software in effect owns the software. Customers like this because there is a built in accountability feedback loop: if the supporting company becomes unfocussed or unresponsive to customer needs, the customer may go elsewhere for their support. The effect is that companies that provide real value are successful and companies that provide real marketing but no substance are not successful.

From a customer point of view, they have peace of mind in that they know that they can choose to (a) fix the problems themselves, (b) contract with someone else to fix the problems, or (c) count on the vendor to fix the problem. In practice, only the last choice is common. The freedom of choice, however, is viewed positively by all customers.

Many customers (including Sun's largest customer) insist on source access. NASA coined the phrase: "if it isn't source, it isn't software." It is possible that, in the future, a system will not be considered viable unless source is available.

Sourceware is also used to spawn new projects and products. An example is Sun's X terminal project; the project uses SunOS 4.x, heavily stripped down, as the base environment to run the X server. They are currently interested in selling the system to a far east company but are bogged down in licensing issues. Everyone agrees that sourceware would have greatly simplified the situation and would not have resulted in less revenue for Sun.

Finally, sourceware is the best way to get the best people working on a system. Almost all of the Unix systems programmers have realized that they are being paid to reimplement the same features and interfaces over and over. The Unix professional is not happy with this, she would rather spend time doing something new. Consequently, a system that is freely available for anyone to use is a far more attractive prospect than a proprietary system.

6. The sourceware deal

The deal has to contain contain components for Sun as well as Novell.

The key components to moving Unix towards a sourceware base are managing the copyright issues with Sun and Novell, and managing the resulting system as it develops. The issues are complex. What follows is the best attempt at managing these issues that we have arrived at to date.

Both Sun and Novell have things they own and things they want. In addition, there are customer desires which are what really drive the deal.

6.1. What Sun wants

- A royalty free operating system. Sun wants this so badly that they are currently spending roughly the same amount as the Unix royalty stream to fund development of a royalty free operating system called Spring.
- Anything that pushes SPARC.
- A way to expand the low end Unix market. Sun would like to see the low end go up by a factor of 10-100 over the next five years.
- An embeddable / romable version of SunOS that could leverage SPARC into the embedded controller market.

6.2. What Novell wants

- Anything that pushes Netware servers and Netware applications.
- All the Unix lawsuits to simply go away.
- A desktop operating system that runs more applications than Windows/NT or Chicago

and includes Netware client side hooks by default.
- Distribution rights to the Intel version of the desktop operating system.
- Funds to help offset the purchase price of Unix.

6.3. What Sun and Novell both want

- A unified Unix marketplace.
- Lower development costs.
- An alternative to NT.
- A low cost volume Unix platform.

6.4. What the customer wants

- The operating system to be essentially free excluding media costs. For a desktop this means it can not exceed $50.
- Applications. The ideal OS will run Intel binaries for all of the major environments: DOS, Windows, and SCO.
- A small footprint operating system. For a laptop (or a desktop) this means that the OS and the window system use up about 2MB of RAM and 20 MB of disk at most, for normal operation (no window system). This is obviously doable, Linux installs and runs (albeit slowly) on a 2MB laptop.
- Plug and play system administration. First, fix it so that there is the bare minimum to do. Second, make it 100% consistent across all of the platforms. A common customer quote: "If Sun had put SunOS 4.x on every platform and then added WABI, Sun would have been a serious threat to Microsoft." Part of the reasons was a reasonable system administration model in SunOS. The point is that **sameness** is valuable.

6.5. What Sun / Novell have to do

Since Novell owns the rights to most of the original Unix source base, Novell is the only company that can set it free. Since Sun stands to benefit substantially, Sun has to help out.

6.5.1. What Sun has to do

The proposal is as follows. Sun adds Netware client side support to Solaris 1. Sun adds WABI to Solaris 1. Sun then donates the resulting system to a new copyright which allows the system to be freely redistributed. Ideally, this copyright would be similar to the GNU copyleft in that it preserves source accessibility forever, but different from the copyleft in that it is more focussed on the business needs of software systems. In addition Sun makes a one time buy out payment to Novell covering Unix royalties for both Solaris 1.x and Solaris 2.x. This payment would be roughly equivalently to what Sun would have had to pay in royalties over the next two years. The point being, by that time Sun will have moved to a royalty free source base anyway. Sun is in essence donating Solaris 1.x and WABI to the public domain.

6.5.2. What Novell has to do

Novell agrees to donate the resulting code contained in Solaris 1 to a new copyright that allows free redistribution in source form. Novell also agrees to donate whatever Netware client code they feel is necessary to talk to a Netware server. Finally, we would like access to Terry Lambert's NetBSD work on Unix inter-version binary compatibility.

In addition, if Novell wishes to do so, and it makes sense, Novell could contract with either Cygnus, BSDI, or Sun to produce a version of Solaris 1 that ran on Intel, complete with DOS, Windows, SCO, and Unixware compatibility. Novell's main asset appears to be its distribution and support networks. A package containing a royalty free Intel compatible version of Solaris, along with additional bundled proprietary application software, is something they could make money from distributing. Lastly, Sun might do well to consider using Novell's impressive distribution channels for SPARC software, and maybe even hardware.

6.5.3. The lawsuits

There are several lawsuits surrounding the Berkeley version of Unix. One possible settlement might be to offer a similar deal to Berkeley. If CSRG, or some other group, were to add

Netware client support to 4.4 BSD, then Novell would agree to let go of the Berkeley code. The number lines of code in question are so minute that it is not worth fighting over. The more we fight, the more Microsoft wins.

6.5.4. Results of this agreement

The main point of all of this is to get a solid, free, platform independent operating system out in the market place as soon as possible. Customers win because they get the same administration model and can run a multitude of applications. Vendors win because they sell more of their primary money maker: hardware in Sun's case, Netware in Novell's case, and support in Cygnus' case.

It is worthwhile noting that the deal is structured to encourage Unix unification. If Solaris[8] is the only solid operating system which is free, other vendors will be tempted to just use it. Novell will be able to manage the Unix unification efforts because Novell did not donate all the Unix code, just the code that was included in Solaris. There are substantial portions of the old System V based Unix that are not part of Solaris. Novell can use the licensing terms to ensure that any other vendor that might want to join in do so on a equitable basis.

7. Managing the resulting system

A common concern with free software is management. How are things to be managed in such a way as to prevent things from becoming chaotic?

The issues can be separated into several parts: copyright, branding, product demonstration, support and coordination, development, and architectural direction.

7.1. Copyright

Ideally, the copyright would be signed over to some non-profit organization. We would suggest considering either Usenix, the Free Software Foundation, or the University of Colorado at Boulder. The point is to make sure that there is an organization that can prevent a single misguided vendor attempting to monopolize the code.

7.2. Branding

X/Open is the obvious choice for handling standards compliance, verification, and branding issues, given their history and the recent noises from Novell.

7.3. Product demonstration

The networking community has demonstrated interoperability for years at the Interop conference. It is possible, even likely, that Interop could be persuaded to expand their conference to include demonstrations of application binary portability. Vendors could show up with their hardware running free Unix and customers could shove in a diskette and try out the Windows emulator. There are obvious security issues but those can be worked out.

7.4. Support and coordination

This is an area where Cygnus shines. Working with a distributed set of developers, Cygnus has turned the GNU C/C++ compilation system into a product quality system capable of running on roughly 60 different systems, and with any system being able to target code for all other systems. Cygnus has unique qualifications, credibility, and ability to provide support for a free operating system. They have worked with vendors as a behind the scenes support organization as well as provided direct customer support. Multiple vendors depend on Cygnus for their compilation tools. Cygnus has successfully maintained a vendor neutral position that allows all vendors to peacefully and profitably coexist.

7.5. Development

One of the biggest problems with Unix has been “what to fix?” Unix vendors have lost touch with their customer base. The Cygnus model is probably worthwhile emulating. Cygnus rarely does development for free. If it is worth doing, it is worth paying for, is their motto. If a customer wants some problem fixed, the

[8] The name does not have to be Solaris. Novell and Sun can work out an acceptable name to both parties. Put it all together and call it System V release 4.3. Or FreeOS. Or SrcOS.

customer can contract with a vendor to provide the fix. The result must go back into the free software base so that the customer does not have to pay for the same fix on a different vendor's platform. This provides a built in feedback loop that corrects mistakes: if the customer thinks the problem is small and the vendor shows up with a $5,000,000 proposal, the customer can "educate" the vendor that that is not quite what was expected. There is no such feedback loop today, and vendors typically take on projects that are far larger than the customer wants.

7.6. Architectural direction

Traditionally, this is where Unix development has floundered. Unix developers have large egos along with their large talent. Free software is a big step towards fixing the ego problem. A friendly architecture board could go farther. A good architectural process will naturally tend to match up the senior people with the new people, the idea people with the implementors. A good architectural process is positive for all involved. I know of no such process today, but I am trying to invent one.

The purpose of the architectural board is to maintain and encourage the development of the basic OS frameworks such as system administration, installation, networking, naming, processes, files, etc. The board, as proposed here, is made up of volunteers. Anyone may be an "architect." All that it means is that you may aid in the design of a framework and review someone else's implementation and offer your opinion on that implementation.

The board is made up of senior people from the industry. There are some requirements that must be fulfilled before you may sit on the board:

- You must be actively writing software. Theoreticians make poor architects.
- You must have used the system in order to review it.
- In order to offer a dissenting opinion, i.e., a strongly negative opinion that would discourage vendors from supporting the work, you must have first written a published position paper on the framework in question. For example, I know about clusters and have a great deal of opinions about how a general purpose cluster should be built. I can not reject someone else's implementation or ideas just because I have an opinion. I have to first write down a refereed, published paper on the topic.

The idea is that the people who really think they know the answer will write it down. Sometimes, that is all they can do. Then it is up to someone else to come along and implement the idea. Because someone wrote down the idea, the implementor will now have someone to talk over the implementation as it is being done. This sets the tone of the interaction in a positive manner, rather than the current antagonistic interactions that are all too common today. In addition, forcing the architects to do real work, use the systems they review, and write down what they think, will remove the blowhards from the architectural process.

8. Alternative to the Sun / Novell deal

It is worth noting that there is an alternative, should Sun and Novell be unable to cooperate. That alternative is Linux, the copylefted PC Unix that is rapidly gathering steam. Because the system is copylefted, there is none of the usual acrimony and empire building that has accompanied Unix like the plague. While Linux is far less stable than a mature system, such as Solaris 1.x, it has the obvious merit of no politics. At the time of this writing, NASA-Ames was seriously considering using Linux on all of their systems, even though this involves an extensive porting effort. There are other institutions that are considering Linux as well.

Linux is a win on the political front and a lose on the maturity front. Should the various companies be able to agree to copyleft Solaris 1.x, then Solaris 1.x is definitely a better base. On the other hand, if the politics kill such an idea, most software developers will simply give up on Sun and SPARC and focus instead on Linux and Intel. Sun has most to lose here.

9. Questions and answers

Q: How do you guarantee that stuff flows back into the free source base?

A: Make the source be copylefted or have a similar copyright.

Q: How do vendors provide proprietary value-add?

A: By bundling proprietary software mainly purchased from third parties. Kernel value-add is possible using loadable modules.

Q: How do you guarantee that the system is reliable? Are not most sourceware packages relatively unstable?

A: GCC is heavily used and quite stable. So is X11. Systems that people want to depend on will be supported by major vendors.

Q: What happens if the vendors can not agree to this proposal?

A: Then instead of dealing with sourceware, software that can be merged with their systems, the vendors will be stuck with whatever Microsoft does.

Q: What about administration? Is not the primary advantage of DOS its ease of use?

A: Yes, system administration, especially installation, packages, and network administration of 100's of machines, is an important area to fix. One advantage of free software is that it only has to be fixed once and then everyone will have the solution.

Q: In order to do ports, vendors will have to provide the porting team with hardware documentation? Vendors typically do not like giving this information away. What do you do about that?

A: This is changing. The Linux crowd will not support undocumented hardware. Linux has had such an effect on hardware sales that the hardware people are reversing their position and providing documentation. Sun is reasonable on this front, they provide OEM documentation that is sufficient to do a port, Wind River has done it.

10. Summary

Microsoft is a big threat to all Unix vendors. We have a chance of beating Microsoft by changing the rules, by insisting that basic enabling software be freely available. This will do nothing but improve Unix platform sales while at the same time devaluing Microsoft's cash cows: DOS and Windows.

The business model for software, particularly enabling software such as operating systems and windowing systems, is changing. The customers and the developers are no longer willing tolerate anything other than high quality, well supported operating systems at low cost. People have become so impatient with the vendors that they are reinventing their own versions of what was once only available from the vendor.

Free software could make the Unix market a vibrant thriving market once again. The current model of incompatible proprietary versions of Unix is clearly not working, millions of marketing dollars notwithstanding.

We should move forward towards the sourceware model. There is much work to be done; Solaris 1.x on Intel, system administration, small footprint desktops, embedded Solaris, etc., etc. Rather than arguing amongst ourselves, lets work out an agreement now and move forward. To delay is to give Microsoft time enough to own the entire market.

11. Acknowledgments

This document is the result of many people's energy and time. I acted as a sounding board for the concerns of all of the people listed here. Without these people taking their time, this document would contain little substance. My hat is off to all of you. Please let me know about misspellings, omissions, etc.

Note: the list here is long and represents the list of people that offered opinions, advice, and criticism. The appearance of a name or organization here does not imply that that person or organization is in complete agreement with the proposal. The appearance simply means that that person or organization was involved in the discussions at some point and provided insight that aided in the development of this proposal.

Sun Microsystems

Andy Bechtolsheim, Paul Borrill, Greg Blanck, Howard Chartok, Steve Chessin, Dave Ditzel, Bob Graham, Gordon Irlam, Bill Joy, Scott McNealy, Kevin Melia, Neal Nuckolls, Alex Osadzinski, Ken Okin, Raj P-XXX-spelling, Bill Raduchel, Joe Roebuck, Chet Silvestri, Jeff Spirer, Hal Stern, Stuart Taylor, Nancy Turbe, Curt Wozniak.

Cygnus Support

Michael Tiemann, David Henkel-Wallace, John Gilmore, Sean Fagan, and the rest of the wonderful Cygnus staff.

Novell/Univel/USL

Ransom Love, Terry Lambert, Drew Major, Ray Noorda-XXX-via-Drew, Chris Torkildson, Mark Epstein, Kanwal Rekhi.

University of California, Berkeley

Kirk McKusick, Keith Bostic.

386/Net/Free BSD Community

Chris Demetriou, Theo Deraadt, Adam Glass, Charles Hannum, Bill Jolitz, etc.

Hewlett-Packard

Carl Staelin, Michael Mahon.

International Business Machines

Kavaliya Dixit-XXX-contact-him, Phil Hester-XXX-contact-him

Usenix Association

Ellie Young, Tom Christianson, Evi Nemeth

Silicon Graphics

Greg Chesson, Vernon Shriver

University of Guelph, Canada

Rick Macklim

Berkeley Software Design

Mike Karels-XXX-contact-him, Rob Kolstad

Mentat

Bruce Carneal

Lehman Brothers

Viktor Dukhovni, John Ionnidis

Morgan Stanley

Marc Donner

Bell Laboratories

Dennis Ritchie

Lawrence Berkeley Labs

Van Jacobson-XXX-contact-him, Craig Leres, Chris Torek.

Digital Equipment

Jeff Mogul

Verity

Philippe Courtot

Institute for Defense Analysis Supercomputing Research Center

Ron Minnich

Kaleida

J. T. Conklin

Oracle

Forest Howard

Transarc

Mike Kazar

Auspex

Guy Harris