FREE SOFTWARE: OPEN SOURCE OR OPEN WOUND?

Bruce Johnson, SGI
Orlando, Florida

ABSTRACT

From out of nowhere, the open-source movement is sweeping through the simulation and training industry with all the temperance of a locomotive. Take the Linux® kernel for example. It seemed only a year ago that Linux was considered no more than a hacker's toy or a graduate student's project. But today, Linux is endorsed by nearly every major computer hardware vendor and is found in a wide variety of applications from Web servers to full flight simulators (FFSs).

While known most for the Linux kernel, the open-source movement also includes many other software products, such as real-time operating systems, compilers, libraries, development tools and Web HTTP servers. In fact, the majority of HTTP servers on the Web today are powered by free software.

Yet there are still a lot more questions than answers when dealing with open-source software. As a vendor, how do I differentiate my product? As a software developer, how do I license my product and protect my intellectual property? As an end user, how do I service and maintain my systems? This paper looks beyond the hype and addresses these and other difficult questions that are associated with open source software.

In addition, also addresses both positive and negative common misconceptions about open source software. Finally, it gives examples of how open-source software is being used in simulation and training systems today and provides some lessons learned for those not yet following the open source herd.

BIOGRAPHY

Mr. Bruce Johnson is a member of the technical staff at SGI in its Orlando, Florida facility. For the past five years he has been responsible for applying real-time technology to the simulation and training industry. Before joining SGI, Mr. Johnson spent more than 10 years working for both simulation integrators and computer system manufacturers. He has authored several papers for other I/ITSEC programs dealing with computer system architecture and operating system technologies. He holds a Bachelor of Science degree in Computer Engineering from the University of Florida.

FREE SOFTWARE:
OPEN SOURCE OR OPEN WOUND?

Bruce Johnson, SGI
Orlando, Florida

INTRODUCTION

This paper explores the benefits and pitfalls of using open-source software in the simulation and training industry. It is clear that Internet-based industries, such as Internet service providers (ISPs), can quickly benefit from open-source (since much of the software originates from that community). It is much less certain whether the simulation and training industry can equally benefit.

To that end, this paper evaluates business and technical issues associated with using and/or developing with open-source software. It examines some existing open-source projects and activities occurring in the simulation and training industry and gleans lessons from these projects that can be applied to other development work and activity.

Although this is not a Linux paper (the open-source movement is larger than Linux), it will briefly compare features of Linux with those found in other operating systems. It also addresses some of the issues associated with using Linux in a real-time training device.

A concerted effort has been made not to present material in this paper that is readily available on the Internet. Therefore, many references will be made to Web sites that contain information that supplements the material presented herein.

As a final note to the introduction, it is important to remember that the open source community moves quickly. Although it could be argued that it does not always move in the correct or even a constant direction, there is no argument that there is constant change. Change occurs so quickly, in fact, that certain points in this paper will likely be outdated by the time it is published. Necessary additions will be provided in the paper presentation.

BRIEF HISTORY OF OPEN SOURCE

The beginning of the open-source movement is credited to Richard Stallman. His Free Software Foundation (FSF, www.fsf.org) and GNU project (GNU is a recursive acronym for GNU's not UNIX®) first championed the idea of free source code. In 1984, Stallman quit his job at MIT and began developing GNU Emacs – originally designed as the first step in developing a completely free operating system and development environment.

Stallman is important to the open source movement for two reasons. First, his GNU software is used extensively in the development of virtually all of today's open source software. The GNU compilers and development tools are the basic building blocks for Linux. Second, Stallman is the heart and passion of the free software movement. It was Stallman who first delivered the idea that software (and source code) should be free – not free as in gratis but free as in unfettered.

This is a commonly misunderstood principle of the open source movement – that it believes that all software should be given away freely. According to Stallman, free software has nothing to do with price, but rather it means for a user to have the freedom to:

- Run the program
- Modify the program
- Redistribute copies of the program
- Distribute modified versions of the program

Stallman continues to be the movement's hard-liner, often criticizing other open-source software licenses for their deviation from the GNU General Purpose License (GPL). Stallman is even critical of term "open-source," feeling that it also compromises the principles of free software.

LINUX

While the roots of open source are from Stallman's GNU project, the movement today is better known for the work and vision of Linus Torvalds. Torvalds is the father of the UNIX-like operating system Linux. Torvalds began working on Linux while a graduate student at the University of Helsinki in Finland. He made available the very first Linux source code on the Internet in late 1991.

Torvalds began with the basic building blocks already provided by Stallman's FSF – most notably the GNU compiler and development environment. Linux is not a port of UNIX, but rather a completely new operating system that was written almost from scratch (Torvalds reused some code and ideas from Minix, a tiny UNIX-like operating system). Almost from the beginning, Torvalds solicited the participation of other software developers via the Internet, and the Linux code began to quickly take shape.

It is often said that Torvalds' greatest contribution to open source is not as the constructor of the Linux kernel itself, but rather the facilitator of the Linux development process. By opening up the software development process to the Internet, Torvalds was able to reap support from some of the greatest kernel developers in the world – and not only a few of them. One example of how widespread the Linux development has come is the sheer size of some Linux mailing lists. The Linux-kernel mailing list, for example, is estimated at more than 20,000 subscribers [1].

From the beginning, and to this day, Torvalds has controlled every enhancement, patch, and release of Linux. Although the Linux software development process is open, no one claims that design decisions are made by committee.

THE CATHEDRAL AND THE BAZAAR

In 1997 another heavyweight of the open-source movement, Eric Raymond, shook up the Internet world with the writing of his treatise on open source entitled, "The Cathedral and the Bazaar" (www.tuxedo.org/~esr/writings/cathedral-bazaar). In this "Internet-published" paper, Raymond outlines his perception of how and why the Linux development model works. This publication also presents a business-case argument for open-source software – comparing traditional software development models to that of the Linux open-source development model. It is this paper that helped influence Netscape to release its browser code to the open-source community in January 1998.

The open source label is credited to a strategy session held in February 1998, and attended by some of the movement's most prominent figureheads, including John `Maddog' Hall of Linux International and Eric Raymond of the Open Source Initiative. The main idea behind adopting the open-source label instead of the free-software label was because the former term was perceived as being much less threatening to business.

OPEN SOURCE SQUABBLING

It seems that the one constant in the open-source movement is the squabbling. There is no shortage of controversial issues or powerful egos in the open-source community; verbal jousting and flaming in both public newsgroups and in the press are commonplace. The squabbling concerns external issues (sometimes seen as a "suits versus the hackers" mentality) as well as internal ones. Some even argue that this squabbling in public forums is one of the things that makes the open-source movement great – even the infighting is open. Even usage of the term open source is controversial – who owns the trademark and how the term legally can be used.

Many sources provide a more detailed history of the open-source software movement. The book "Open Sources: Voices from the Open Source Revolution," a collection of essays from many of the open-source pioneers, provides insightful detail about the beginnings of the open-source movement. Also, the following Web sites contain more details about open-source history: www.opensource.org and www.fsf.org.

OPEN-SOURCE LICENSES

One would think that the issue of licensing software that is free would be a simple one, but to the contrary it has become a very complicated and controversial subject. An entire paper in itself could be devoted to this subject alone. The GNU General Public License (GPL), defined by the Free Software Foundation, is considered either the most open or the most restrictive of open-source licenses, depending on one's perspective. It is the most open because it requires that all software added to GPL-licensed software be released in source form under the original GPL license. This forbids a company or individual from modifying (or incorporating) GPL-licensed code and releasing it only in binary form.

Conversely, this openness is viewed by many as too restrictive – because it requires that new code that touches the GPL code be released as open source. Understandably, this severely restricts intellectual property rights of software developers. Partially for this reason, the FSF also has a "less open" license – the GNU Lesser General Public License (LGPL). Unlike GPL licensed code, an LGPL licensed library can be linked into nonfree programs.

In addition to open-source licenses from FSF, other licenses are generally viewed as approved for use in releasing open-source software (although not necessarily approved by all). They include the following:

- BSD license
- X Consortium license
- Any number of derived licenses

Derived licenses are those that have slight modifications to one of the aforementioned open-source licenses. As an example, SGI released its GLX library code under a license derived from the X Consortium License – it included a few added terms and clarifications (see www.sgi.com/software/opensource/glx/license.html). These additions were included to allow other parties to distribute binary-only drivers built on the open-source framework. SGI viewed this as critical because most 3D-hardware vendors (including SGI) are not willing to expose the source code of their drivers. The GLX license does meet the criteria for inclusion into the Xfree86 code base (more on this in ensuing paragraphs).

Here are a few Internet Web sites that can be used to better understand the issues of open-source licensing:

- **www.opensource.org** – Open Source Initiative Web site
- **www.mozilla.org/NPL/FAQ.html** – Netscape's Licensing FAQ Web site
- **www.fsf.org** – Free Software Foundation's Web site

OPEN SOURCE OPERATING SYSTEMS

As mentioned previously, although Linux is certainly the most prolific open-source operating system, it is not the only one. In fact, it is not even the only UNIX-like, open-source operating system available. FreeBSD (www.freebsd.org) has been available as open source software since early 1993. Although some claim there are technical advantages of FreeBSD over Linux, most FreeBSD users have chosen it over Linux for its less restrictive (and perhaps less open) licensing.

The open-software movement has also spread into the real-time and embedded operating system space with products such as eCOS (embedded Cygnus Operating System – (www.cygnus.com/ecos/) and RTEMS (Real-Time Executive for Multiprocessor Systems – (www.oarcorp.com). To date neither one of these embedded operating systems has seen near the success of Linux, and their use in the simulation and training industry is very limited.

AN OPEN-SOURCE OPERATING SYSTEM DISTRIBUTION

There is more to an operating system than simply a kernel. More often than not, a reference to Linux actually means the Linux kernel as distributed with a myriad of other pieces of open-source software. A Linux operating system distribution typically contains hundreds of software components from many different sources integrated into a single operating system (OS) distribution. Figure 1 shows the various types of software components that are added to a kernel release in order to create a complete distribution. Clearly, when one claims to have installed Linux on a computer,

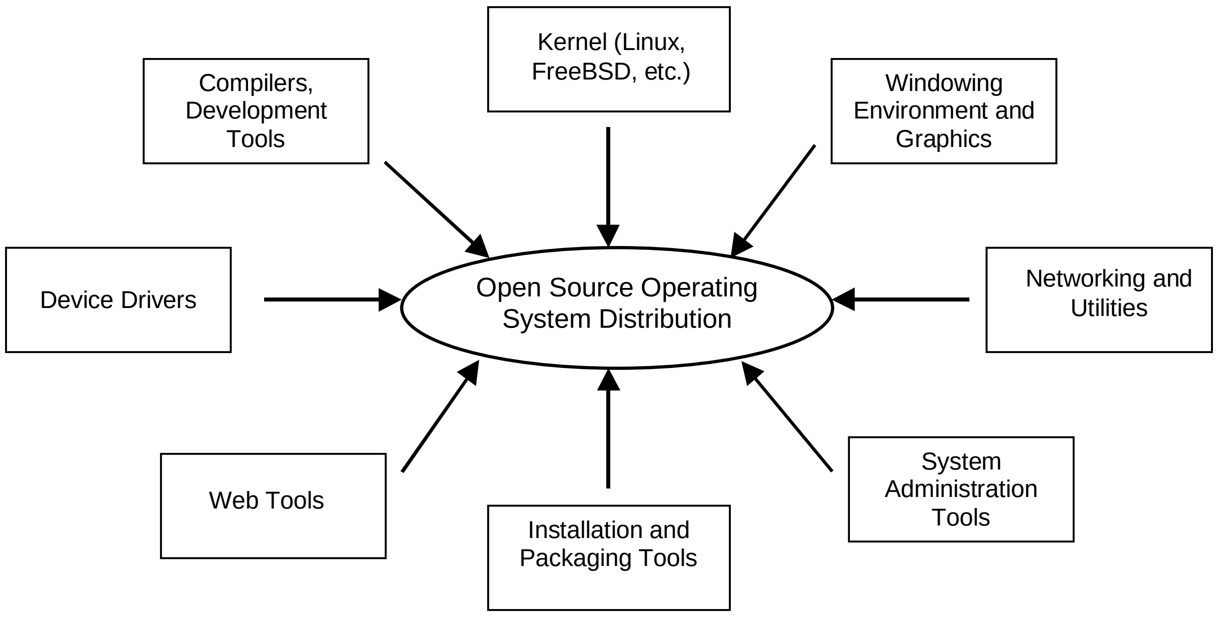
Kernel (Linux, FreeBSD, etc.)
Compilers, Development Tools
Windowing Environment and Graphics
Device Drivers
Open Source Operating System Distribution
Networking and Utilities
Web Tools
Installation and Packaging Tools
System Administration Tools

Figure 1 – Components of an Open-source Operating System Distribution

they are more correctly referring to Linux plus hundreds of other pieces of software. The top five Linux OS distributions today are generally thought to be:

- Red Hat Linux (over 50% of the market)
- Debian GNU/Linux
- Caldera OpenLinux
- Slackware
- S.u.S.E.

A complete list of distributions can be found at www.linuxhq.com/dist-index.html.

A COMPLETE OPERATING SYSTEM SOLUTION

In the same way that a Linux distribution is made up of much more than a kernel, an operating system solution is made up of much more than the software itself. As can be seen in Figure 2, an operating system typically requires many levels of support, training, and documentation. Many of these OS requirements are now provided by some of the Linux distribution vendors as well as mainstream computer vendors such as IBM, HP, and SGI. The fact that Linux has not historically been provided as a complete solution from a single vendor has bothered businesses and independent software vendors (ISVs) alike. The next few sections examine what the open-source community and traditional computer vendors are doing to address this issue.

STANDARDIZATION

The computer industry (and much of the simulation and training industry) seems to have a love-hate relationship with industry standards. Standards are generally viewed as good, unless of course, they get in the way of what is determined to be a more important agenda (such as a lower price or shorter time to market).

The past 15 years, have seen no shortage of computer industry standards, although there may have been a shortage of successful industry standards. Ideally, industry standards have the following characteristics:

- Quickly adopted
- Universally accepted
- Free from proprietary influence
- Beneficial to the community

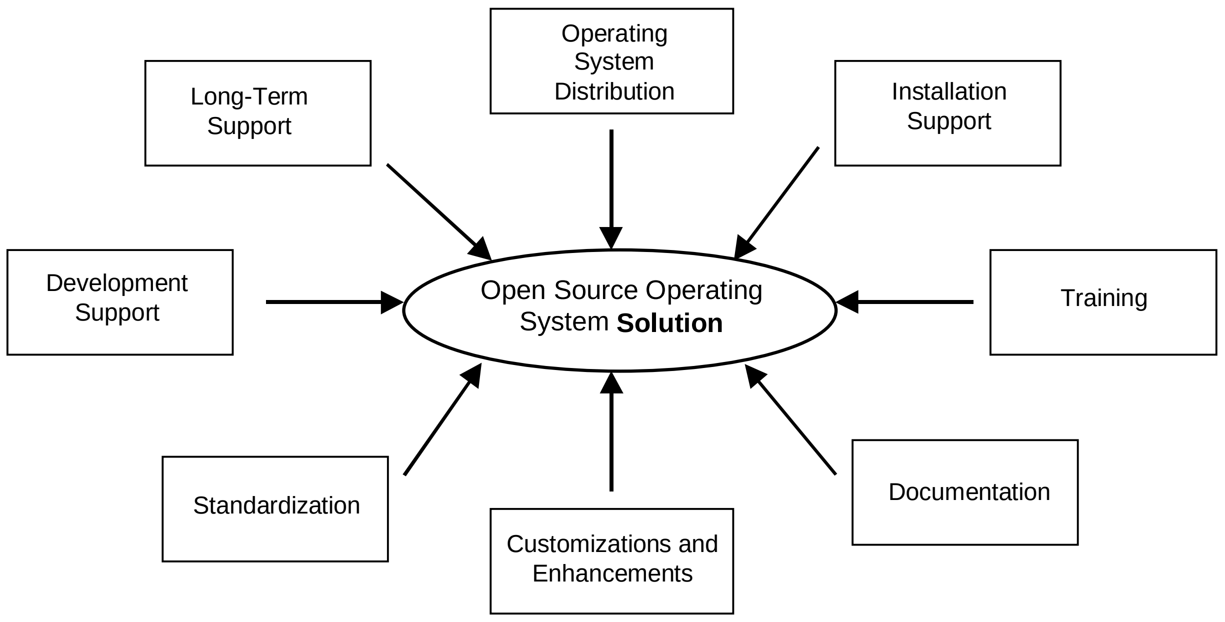
Operating System Distribution
Long-Term Support
Installation Support
Development Support
Open Source Operating System **Solution**
Training
Standardization
Customizations and Enhancements
Documentation

Figure 2 – Components of an Open-source Operating System Solution

In reality, however, it is difficult to achieve and maintain all of these goals. For example, the computer industry was quick to adopt many of Microsoft's APIs (e.g., Win32, Direct3D, COM/DCOM, etc.) as industry standards even though they are proprietary.

STANDARDIZATION OF LINUX

There are some common misconceptions about Linux and standards. For example, contrary to popular belief, there is no current Linux distribution that conforms to the POSIX standard (POSIX in this context meaning conformance to the NIST FIPS 151-2 test suite). At least one version of Linux was certified under the 1995 POSIX NIST certification tests, but the Linux kernel has evolved considerably since that time (the certification was under Linux version 1.2.13 while the current Linux version is 2.2). See Ian Nandhra's article at http://lwn.net/lwn/980611/ standardseditorial.html for a firsthand view of the work required to make Linux pass the POSIX standardization test suite (Nandhra was involved in the aforementioned Linux POSIX certification effort).

Linux does contain most of the POSIX application programming interfaces (APIs), but strict conformance to any standard has never been a goal of Linux. In fact, many would argue that conformance to industry standards is even undesirable as it inhibits change and may require licensing fees (e.g., for The Open Group's (TOG) UNIX98 branding).

Without strict conformance to standards, what will keep open-source software from suffering the same splintering fate that befell previous UNIX-based operating systems? Additionally, software vendors need some form of standards to ensure that products they port to Linux will continue to work through various revisions of the operating system. As Nandhra has said:

"Change. Linux thrives on it, users, buyers, and ISVs usually hate it."

One potential solution may come from the work of the Linux Standards Base (LSB), whose mission statement can be found at www.linuxbase.org:

"The goal of the Linux Standard Base (LSB) is to develop and promote a set of standards that will increase compatibility among Linux distributions and enable software applications to run on any compliant Linux system. In addition, the LSB will help coordinate efforts to recruit software vendors to port and write products for Linux."

As can be seen from its mission statement, the LSB is not so much interested in applying existing industry standards (such as IEEE POSIX and TOG UNIX98 standards) to the Linux code base as they are interested in standardizing Linux versions and distributions to one another. Whether the LSB (or any other standardization effort born of the open-source community) can truly deliver on its goals is yet to be seen. Many users, buyers, and ISVs are waiting eagerly.

DOCUMENTATION

Open-source software does not suffer from a lack of documentation. It suffers from a lack of organized, high-quality, consistent, up-to-date documentation. The historic sources of open-source documentation have been:

- The source code itself
- The Internet (i.e., newsgroups, mail aliases, and Web pages)
- Commercially available books

Because open-source software was born of the Internet (and continues to thrive there), an amazing amount of documentation can be found therein. The quality problem remains, however. As even Richard Stallman admits:

"The biggest deficiency in our free operating systems is not the software – it is in the lack of good free manuals that we can include in our systems."

The open source community is attempting to deal with some of this deficiency through projects such as the Linux Documentation Project (http://metalab.unc.edu/LDP). Yet it suffers from one of the eternal laws of software engineering – it is less gratifying and often less respected to document than it is to develop. Nevertheless, some significant progress is being made. The Linux Documentation Project actually covers more than just the base Linux kernel – it includes other open source software (e.g., Xfree86, GNU tools, Web software, etc.). From the LDP Manifesto:

"The Linux Documentation Project is working on developing good, reliable docs for the Linux operating system. The overall goal of the LDP is to collaborate in taking care of all of the issues of Linux documentation, ranging from online docs (man pages, texinfo docs, and so on) to printed manuals covering topics such as installing, using, and running Linux."

Documentation is also available, for a fee, from Linux distribution vendors and will no doubt be provided from traditional hardware vendors as they begin to support Linux-based products.

TRAINING AND CERTIFICATION

There is certainly no shortage of professional training available for the most widely used open-source software. More recently an option is high-quality, Web-based training offered by companies such as O'Reilly and Associates.

Of greater concern of late is the development of a Linux training and certification program similar to the Microsoft certification programs. To this end, some Linux distributors have endorsed a nonprofit organization called the Linux Professional Institute (LPI), which is developing a Linux certification program. Initially, LPI's certification program is aimed at providing a certification program that covers skills needed by low-level Linux system administrators. The plan is for this level to cover two aspects: general Linux system administration and distribution-specific material. After these certification programs are developed, it is likely that LPI will provide a service to approve the courseware that vendors will provide for Linux training.

One current controversy on the subject of Linux training, however, is the fact that independent of LPI, Red Hat Linux has already begun its own training and certification program. In the future, computer vendors that offer and support Linux on their platforms (such as HP and SGI) will probably also offer Linux training and certification.

SUPPORT

Once again, the open-source community points to the Internet (i.e., newsgroups, mail lists, etc.) as its primary source of support. In fact, open-source proponents often point to this support structure as one of the strengths of open source software. While there is some merit to this, most would acknowledge that there is plenty of need for more formal support alternatives.

Although the support aspect of open source is often criticized, the support picture is changing. Not only can full support be purchased from many major Linux distribution suppliers, but also, full Linux support is now being offered by many traditional hardware vendors such as IBM, HP, and

SGI. Some have even begun to provide full 24x7 support, the "Holy Grail" of support.

Additionally, there are companies whose prime business is providing technical support for Linux and Linux-based products (e.g., Linuxcare www.linuxcare.com).

EVALUATING OPEN SOURCE FOR USE IN SIMULATION AND TRAINING APPLICATIONS

An incredible amount of information (plus ubiquitous opinions) exists on the benefits of open-source software. Interestingly enough, however, what one group might view as an inherent benefit of open source (such as the do-it-yourself flexibility), another group views as a weakness. The ensuing paragraphs outline some potential benefits of utilizing open-source software specifically in the simulation and training industry. First, the issue of using open-source software in the general sense will be discussed, then more specifically the issue of using Linux.

Unlike other industries, the simulation and training industry is uniquely positioned to take advantage of the open source movement. Here is a brief list of some reasons why this is true:

1. Integrators largely develop their own software instead of having to rely upon third-party applications. A lack of third-party applications has hindered the growth of Linux in some cases.

2. A large percentage of existing simulation and training applications are UNIX-based. Migration from a UNIX operating system to Linux is generally a relatively simple task (i.e., when compared to porting from UNIX to Windows NT®).

3. According to Eric Raymond, an important requirement of the open-source development model is that developers need to have an "itch" to work on their software. Because simulation and training applications are among the most interesting in the world, the "itch" is relatively easy to acquire and maintain.

4. The simulation and training industry is not as tied to constant upgrades as are other industries (e.g., Internet-based industries). This freedom permits integrators to snapshot existing good distributions of open source without needing to rely on bleeding-edge releases of software.

5. There is considerable redundancy and overlap in the software developed for simulators and training devices.

6. Simulation and training applications often require specific small changes to software products provided by vendors – changes such as adding operating system functionality or graphics capabilities. Having the access to change and customization that open source provides is very valuable.

For some good reference documents that address the pros and cons of using Linux in more general cases see [2] and [3].

EVALUATING LINUX FOR USE IN SIMULATION AND TRAINING APPLICATIONS

Linux is not new in the simulation and training industry. It is clear that there are applications, such as ModSAF/JointSAF, that have been taking advantage of the benefits of Linux for a number of years. What is new about Linux is its usage in higher-end applications such as host computing and real-time 3D graphics. The next few paragraphs evaluate some requirements needed by Linux in order to extend its usage into these applications areas.

Symmetric Multi-Processing

With the most current release of Linux (version 2.2 native, version 6.0 from Red Hat), the kernel scales very well up to four processors. Although scalability is limited in the current release, definitive plans are in the works to provide a bit more. Quoting directly from Linus Torvalds on this subject:

> "Symmetric Multi-Processing (SMP) is one area that will be developed. The 2.2 Linux kernel will handle four processors pretty well, and we'll develop it up to eight or sixteen processors. The support for more than four processors is already there, but not really. If you have more than four processors now, it is like throwing money at a dead horse [1]."

Conversely, other UNIX-based operating systems, such as SGI IRIX® or Sun Solaris®, scale to 64 CPUs or more. Although 8 to 16 CPUs covers the vast majority of simulation applications, there are still a number of high-end applications that require more.

Real-time Capabilities

Linux has never purported to be a real-time operating system, but it does have some real-time features that can increase the determinism of most applications. However, native Linux (as opposed to RT-Linux) contains fewer real-time features than real-time UNIX-based operating systems such as SGI IRIX or Concurrent PowerMax OS® (see Table 1). Within this table, the term Linux refers to version 2.2 of the operating system. For a complete definition of the terms used in this table, refer to [4].

3D-Graphics Capabilities

Hardware-accelerated OpenGL® is now well supported on all significant commercial operating systems – except Linux. At the time of this writing, hardware-accelerated 3D-graphics capabilities on Linux are rather weak. However, as is typical of the open-source community, the situation is changing fast. A 3D-graphics implementation under Linux is typically made up of the components shown in Figure 3. A very simplified description of the layered libraries in Figure 3 follows.

XFree86 is an open-source implementation of The X/Window System® that runs primarily on Linux-based Intel® x86 platforms. XFree86 is developed and maintained by the XFree86 Project, Inc., a nonprofit organization. XFree86 is a very high-quality X implementation, and it ships with nearly every Linux distribution. For more information about Xfree86 see its Web site: www.xfree86.org.

GLX is the "glue" that permits an OpenGL application to coordinate with the resources of the X/Window System. It arbitrates between 2D and 3D rendering operations, coordinates OpenGL and X resources, and provides X protocol extensions for displaying OpenGL across a network connection [5].

OpenGL is the leading 3D-graphics API for developing high-end graphics applications. It is a vendor-neutral industry standard controlled by an independent consortium. OpenGL provides the interface that a programmer will use to render 3D-graphics for an application (see www.opengl.org for more information). Mesa is an OpenGL-like API available as open source and distributed under the GNU LGPL (see www.mesa3d.org for more information). Much more of the technical details about implementing OpenGL and GLX under Linux can be found in a paper presented by SGI and Precision Insight, Inc. at LinuxExpo in May 1999 [5].

Real-time Feature	**Linux**	**Native UNIX**	**Real-time UNIX**
Preemptive, priority-based multitasking	Yes	Yes	Yes
Nondegrading, real-time priorities	Yes	No	Yes
Processor isolation/process binding	No	No	Yes
Locking virtual memory	Yes	Yes	Yes
Posix 1003.1b support	Some	No	Yes
Asynchronous I/O	Yes	No	Yes
Guaranteed real-time response	No	No	Yes

Table 1 – Real-Time Features of Linux

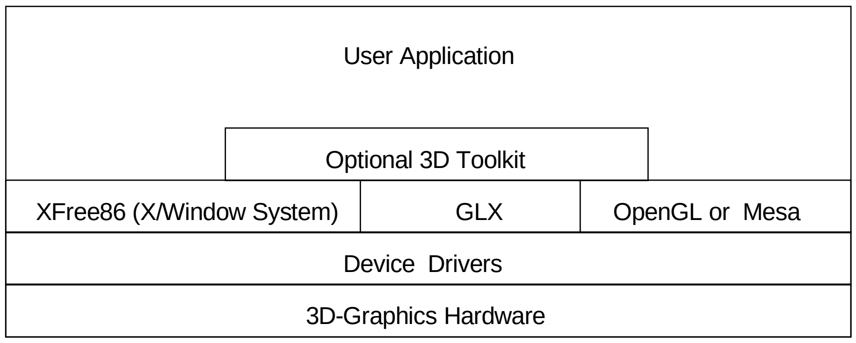

Figure 3 - Software Components of Accelerated 3D-Graphics on Linux

Currently, only a few implementations of hardware-accelerated 3D-graphics exist under Linux: 3Dfx's Mesa-on-Glide and early releases of Mesa/GLX/OpenGL libraries for some NVIDIA RIVA and Matrox G200 graphics chipsets. Glide is 3Dfx's proprietary 3D library. Although 3D performance of these implementations currently lags behind their Windows® counterparts, it is anticipated that this will change soon.

EXAMPLE CASE: RELEASING OPEN-SOURCE SOFTWARE

SGI has not been a longtime proponent of open source. However, over the past nine months SGI has become more committed to the open-source community by releasing some very powerful software to open source. Among the software that SGI has released is the GLX source code.

Figure 3 shows the relationship of GLX to the other components of an open-source, 3D-graphics solution under Linux. GLX is critical to high-quality OpenGL-accelerated 3D graphics support under Linux.

Releasing GLX code has been a catalyst for the development of 3D-graphics software in the Linux space. From the time that SGI first placed the GLX source code on its FTP server to the time that libraries first appeared (as beta) in a Linux GLX/Mesa/XFree86 implementation was less than four months.

EXAMPLE CASE: DEVELOPING OPEN SOURCE

While not designed to the level of fidelity required for a training device, the development of the FlightGear Flight Simulator (www.flightgear.org) is still an interesting case study of open source. FlightGear software is developed using the Linux development model (i.e., over the Internet) and is released under the GNU GPL. The current simulation models (e.g., flight dynamics, engine models, etc.) and graphics are somewhat primitive by training simulator standards; nevertheless the results are impressive. Perhaps most impressive, is how quickly new software is being added and how quickly it gets ported to new hardware. This achievement is of course a testament to the Linux development model.

EXAMPLE CASE: USING LINUX ON A FLIGHT SIMULATOR

A case of using open-source software in flight simulators that has received some attention in the Linux press is the work being done by Opinicus for Northwest Airlines. Opinicus is contracted by Northwest to upgrade 23 of its aging flight simulators with newer hardware and software. In some cases, this means replacing older computer systems such as Compaq (formerly DEC) VAX systems with Pentium®II-based systems running Linux[6]. To replace host-computer software that runs at 30 hertz, Opinicus uses an unmodified standard distribution of Linux. In these cases it is not unusual for them to receive frame jitter of up to seven milliseconds, but their simulation models are designed to tolerate jitter of that magnitude. For simulation models that require hard real-time capabilities, such as 2000-5000 hertz control loading models, they use RT-Linux – a hard real-time patch overlay to standard Linux available as open source from www.rtlinux.org/~rtlinux/.

CONCLUSION

This paper does not either endorse or condemn the use of open source and Linux. Clearly using open-source software in the right circumstances can greatly benefit users, developers, and ISVs. One thing is for certain, there will be seen a great deal more of Linux and other open-source software in the simulation and training industry over the next several years. Michael Tiemann of Cygnus provides some insightful thoughts to close with:

"Open Source is all well and good for the hacker … but there's a gap between what hackers can do with open-source software and what regular users can do [1]."

The extent to which the industry is successful in closing this hacker-user gap will determine the success of open source over the long term.

REFERENCES

[1] DiBona, Chris, et al., 1999. "Open Sources: Voices from the Open Source Revolution," O'Reilly and Associates, Inc.

[2] Prasad, Ganash, 1999. "The Practical Manager's Guide to Linux," Internet Article: www.osopinion.com/Opinions

[3] "Is It Time for Linux", Network Computing Online, May 31, 1999.

[4] Johnson, Bruce, 1998. "Operating Systems for Training Devices: Does It Make a Difference?," I/ITSEC 1998 Proceedings.

[5] Leech, Jon, et al., 1999. "Accelerated OpenGL for Linux and Xfree86," May Linux Expo Proceedings.

[6] Orenstein, David, 1999. "Linux Takes Flight on Northwest Simulators," May 24, 1999. Computerworld