From

The open source movement, exemplified by the growing acceptance of Linux, is finding its way not only into corporate environments but also into a home near you. For some time now, high-end applications such as software development, computer-aided design and manufacturing, and heavy computational applications have been implemented using Linux and generic PC hardware.

Now, Linux and open source software are making inroads at the other end of the computing spectrum. TiVo, the first commercially available digital video recorder (DVR), provides an example of how embedded devices are increasingly powerful enough to support Linux as an operating system—providing a great deal of leverage to system developers.

Server Room to Living Room

JIM BARTON

How open source and TiVo became a perfect match

From Server Room to Living Room

A BRIEF HISTORY OF OPEN SOURCE

To many people, the open source movement is a recent phenomenon, springing into consciousness in the late 1990s with the creation of Netscape Navigator and the rise of the Linux operating system. The true beginnings of the open source movement, however, can be traced back to the mid-1980s.

At that time, the computer industry, as well as academia, had become enchanted with the Unix operating system[1] and its variants. Computer manufacturers realized that the era of each manufacturer providing a proprietary operating system for its hardware was drawing to a close; while this strategy locked customers to a particular manufacturer, it also limited the ability to acquire new customers and cost a great deal to support.

Unix was originally developed at Bell Laboratories as a reaction to the large, complex, non-portable operating systems of the early 1970s. AT&T did not see a significant opportunity in licensing or supporting an operating system; instead, it provided Unix source code for a nominal license fee and small per-unit royalties. A number of academic efforts sprang up to take advantage of this opportunity and extend the original sources with new and interesting features.

The most famous of these efforts is the Berkeley Software Distribution (BSD) of Unix, which pioneered features such as paging and the "sockets" network abstraction. BSD Unix lives on today, but its descendants are better known, among them: FreeBSD, NetBSD, OpenBSD, and BSDi. This line of Unix development began in 1974,[2] and the BSD developers created many features of modern Unix-based operating systems. In fact, for many years BSD versions of Unix were considered far superior to AT&T Unix in features, performance, and reliability (and many would argue BSD Unix is still the best). This is largely a result of the open and collaborative nature of BSD development at a time when Unix was a little-noticed sideline within the vast halls of AT&T.

In the mid-1980s, every computer manufacturer either provided or planned to provide a Unix-based operating system for its computers. Each company had chosen a particular version of Unix to start with, and then added various proprietary features. Although all of these operating systems claimed to be Unix, software written on one version was often not portable to the other versions.

At this same time, AT&T was breaking up into separate companies in response to a U.S. government antitrust suit, spinning off the telephone operating companies and an unregulated subsidiary. The company was trying to use its wealth of internally developed technologies as a lever to create new revenue streams. Within AT&T there was growing realization that selling and supporting Unix might be a significant new source of revenue. The company began to promote its own version of Unix as the standard for the computing industry and to enforce its intellectual property rights around the Unix trademark and source code.

The prospect of AT&T "taking control" of Unix and then potentially limiting developers in what they could see or change in the software was one of the main forces behind the earliest open source initiatives.

THE GNU PROJECT AND UNIX STANDARDIZATION

Many believed this effort by AT&T was bad news for the software community. Among them was Richard Stallman, a programmer at MIT, who in 1984 founded the GNU Project ("GNU Is Not Unix") as an effort to rewrite Unix as "free" software—that is, software not controlled by any one person and available for anybody to use and modify as desired.

This led to the creation in 1985 of the Free Software Foundation (FSF), the main funding and advocacy organization for the GNU Project. The Free Software Foundation established the GNU General Public License (GPL), the most important open source license in use today.

While the GNU Project was getting started, other efforts began to at least standardize the definition of Unix to enhance software portability. One such effort was the formation of the IEEE 1003 standards committee, which would go on to produce the set of standards known as

the Portable Operating System Interface (POSIX). Another effort of note was the creation of the X/Open organization, a commercial venture oriented toward application portability, testing, and branding of Unix-compatible operating systems.

THE GREAT SCHISM

Continuing its aggressive push with Unix, AT&T signed an agreement with Sun Microsystems in 1987 to work jointly on creating a version of AT&T's System V Unix, which would become the de-facto standard for Unix. This version would run first and best on Sun's Sparc microprocessor architecture, which frightened all other Unix-based manufacturers in the computer industry. They saw themselves at a disadvantage to Sun, and Sun enhanced this fear through aggressive marketing of the joint development.

These manufacturers, led by Digital Equipment Corporation (DEC) and Hewlett-Packard (HP), arranged a meeting at DEC's Western Research Lab in Palo Alto, California. The lab was on Hamilton Avenue, so the group of companies that met that first time came to be known as the Hamilton Group.[3] In reaction to the Sun/AT&T alliance, many of these manufacturers came together to create the Open Software Foundation (OSF), which was given the task of creating an "alternative" version of Unix to compete with AT&T.

In response, AT&T formed Unix International, recruiting many computer vendors to back System V Unix. Thus began the "Unix wars." Meanwhile, quietly working in the background, the GNU Project began producing a number of useful tools and utilities, mostly used in academia and research.

THE ENEMY IS REALLY MICROSOFT

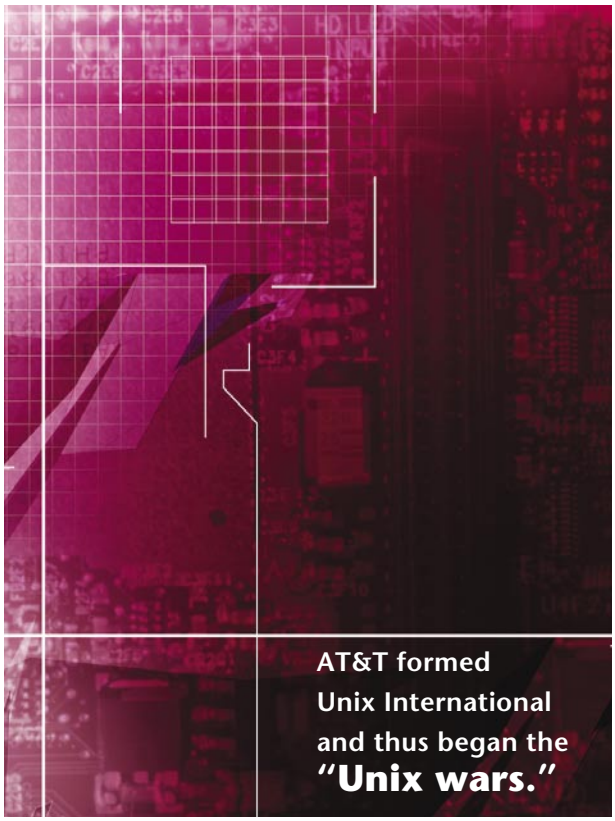
At the beginning of the 1990s, in the midst of the Unix wars, those in the Unix industry began to realize that Microsoft was beginning to dominate the PC segment of the computer industry, and that the company might well move from there into other parts of the industry.

This resulted in several infamous counter-efforts. One spectacular failure was the "ACE" initiative, created to go up against both Microsoft and Intel. It promoted the Open Software Foundation's Unix clone, OSF/1, running on the MIPS microprocessor.[4]

Another counter-effort was the creation of Unix Systems Laboratories by AT&T, with an investment from Novell, as a stand-alone Unix technology company. As this venture sputtered, AT&T sold its share to Novell, which was looking for an industrial-strength operating system to complement Novell Netware for the PC.

In 1995, Novell gave the Unix brand to X/Open and sold the software technology business to the Santa Cruz Operation (SCO). A year later, the Open Software Foundation and X/Open merged to form The Open Group, which continues the focus of both companies to develop an open standard for Unix-based systems that anyone can implement.

Meanwhile, the GNU Project continued to expand its software base and improve the quality of its offerings.



Within academia, the GNU Compiler Collection (GCC) was viewed as ideal for creating cross-platform software. In many ways the compiler suite began to exceed the quality of commercial offerings.

In 1991, Linus Torvalds began developing the Linux kernel. By choosing to use GNU Project software to build and extend this work, he created a fruitful collaboration that enhanced the quality of both offerings over time.

LINUX RISING

Throughout the mid-1990s, development continued apace on Linux, with a number of programmers adding vast contributions to the operating system and

From Server Room to Living Room

its infrastructure. In 1995, the first commercial Linux distributions combined the Linux kernel with the GCC tool suite and a full suite of Unix-like tools and utilities, creating the first credible Unix replacement.

In the late 1990s, GNU/Linux came into its own as a commercial-class operating system environment. Still-famous companies entered the GNU/Linux distribution business: Red Hat, VA Linux, Slackware, and others. Unfortunately, the GNU General Public License constrained the business models that were available to these companies. Although many had highly successful public stock offerings, they collapsed after being unable to generate self-funding cash flows. Red Hat remains the standard bearer for commercial Linux distribution and seems to be making a successful business of distributing and supporting open source software.

THE UNIX LANDSCAPE TODAY

The legacy of Unix continues to cause controversy. Companies such as IBM, Hewlett-Packard, Sun, and SGI now support Linux distributions for their computers, usually alongside their own proprietary Unix offerings.

GNU/Linux is increasingly being used as a server operating system, reducing overall costs while allowing the use of generic PC-based hardware rather than more expensive proprietary hardware designs.

Many embedded applications that incorporate Linux already exist or are about to come to market. The TiVo Client Device (TCD), a set-top box that digitally records television programs, was among the earliest of these systems. Sony and a number of other Japanese consumer electronics manufacturers recently announced their support of Linux and their commitment to making it ubiquitous in consumer electronics applications.

The legacy of Unix, however, continues to cause controversy. SCO, facing declining sales of its proprietary offerings based on the original Unix derivation, sued IBM in March, claiming that IBM had "tainted" the Linux source base by incorporating patented technology derived from the IBM Unix-based proprietary offering, called AIX. This spat further heated up with Novell's announcement that it still controlled much of the intellectual property incorporated into Unix. In addition to Greg Lehey's analysis of the SCO action that appears in this issue of *Queue* (see page 56), Eric Raymond, formerly of Netscape and now president of the Open Source Initiative, has written an extensive commentary on the action and the history of Unix in general that debunks SCO's claims.[5] In June, SCO "terminated" IBM's license to the Unix software in AIX, although IBM publicly stated that SCO has no ability to do so.

Whatever the outcome of this suit, it seems to be a rear-guard—and probably hopeless—action by SCO. For some time after filing suit, SCO was publishing its own distribution of Linux, which we must assume contained its intellectual property. Because distributing software under the GNU General Public License requires the publisher to acknowledge that it has no rights in the published source, or that it is forgoing those rights, SCO would seem to have forfeited any claims it might make. At the very least, it will be entertaining to see how this drama unfolds, although it is not likely to have a significant effect on the ongoing adoption of Linux in many areas.

A BIT OF TIVO HISTORY

The original TiVo product, as proposed by Mike Ramsay and me in 1997, was for a home network-based multimedia server, streaming content to thin clients throughout the home. To build such a product requires a solid software foundation. The home environment may be one of the most mission-critical applications conceivable. From a consumer's perspective, the device must operate flawlessly, be reliable and robust, and handle power failure gracefully.

At that time, I had become familiar with Linux through a number of avenues. At Silicon Graphics (SGI), I was the executive sponsor of an effort to port Linux to the SGI Indy workstation, based on the MIPS R5000. I was using several early Linux distributions for personal experimental and development work.

This experience led me to believe that Linux would

serve TiVo well as the operating system foundation. It was based on well-tested APIs, included a solid disk management system, a large base of support software and tools, and virtual memory and paging. It also provided access to and control of the software source code. With my operating system development background, I felt comfortable that we could handle any minor modifications or bug fixes needed to achieve the product goals. I also had a great deal of experience at Hewlett-Packard and SGI with molding Unix-like operating systems to handle real-time computing tasks. I felt that Linux could be similarly enhanced to provide the real-time performance needed to support consumer-quality television viewing.[6]

At the time, however, open source software was viewed with suspicion. People often asserted that open source software could never be as reliable as proprietary software. Software released under the GPL was looked upon with particular disfavor, because many people assumed that such software contaminated everything it touched and would disallow proprietary developments. Careful reading of the GPL convinced me that these fears were unfounded and that Linux could give us a powerful development advantage while allowing the protection of our intellectual property.

Early in the existence of TiVo, we realized that the home networking bandwidth available to support a client-server multimedia environment was not going to be easily available in the foreseeable future. We decided to collapse our design into a single product, the TiVo Client Device. The TCD takes the form of a television set-top box about the size of a video tape recorder. It contains a number of off-the-shelf electronic components, including a microprocessor, memory, modem, MPEG2 real-time encoder and decoder, and most significantly, a large-capacity PC OEM disk drive. Using patented and patent-pending technologies, the TCD has the ability to encode and store a television program on the disk drive in real time, while in parallel retrieving a program from the disk drive and outputting it as a standard television signal. This early digital video recorder could thus record one program while playing back another program that had been recorded earlier. In fact, the program being played back might even be the one being recorded, only shifted in time. This provides some of the more well-known features of a DVR—the ability to "pause" live television and to skip through television commercials.

To make the TCD reliable and easy to use, we needed to provide it with information about television programs and schedules, enabling the consumer to peruse the available programming and choose programs to be saved for later viewing. This program-guide data is automatically downloaded from our central servers during a daily telephone call and provides up to a 14-day window into future programming.

Early in the design of the TCD, we realized that delivering the simplicity and ease of use we desired would require a complex, interlocking set of capabilities. Maintaining a television-like viewing experience meant that video playback must never stop, and playback had to be perfect at all times, with no glitches or audio/video synchronization problems. Capture and storage of programs had to be similarly perfect, with no dropped frames and accurate tuning and timing. Searching the television guide or scheduling recordings must not interfere with these or other operations, such as the daily phone call to the TiVo servers. This implied the use of a preemptible multitasking operating system with both realtime and background processing abilities.

Our earlier focus on Linux turned out to be prescient when applied to the TCD. Here was an operating system that was well-developed in multitasking, while optimizing hardware resource usage. In addition, the availability of virtual memory made the software far easier to debug

From Server Room to Living Room

and validate, and paging allowed us to have a great deal of functionality available in our limited memory footprint of 16 megabytes.

For cost reasons, we used the IBM PowerPC 403GCX processor in the first TCD. With an external clock of 27 megahertz (clock doubled on chip), this processor was anemic by current standards, but quite powerful by embedded-system standards of the mid-1990s. It provided a simple memory management unit (MMU), allowing us to run Linux, as well as providing a great deal of embedded-system debugging support. This processor turned out to be ideal for our needs.

Linux posed some problems, however. A PowerPC port of Linux was available, but it was not mature and did not support the IBM 400 series microprocessors. It was also based on a "development" version of the Linux kernel (an odd-numbered version 2.1) rather than on a "stable" version of Linux. But it was enough. With a few months of effort, we ported Linux to this processor and were able to boot the first prototype TCD in the lab.

There was yet more work to do. We needed to modify the kernel to allow direct I/O (bypassing the file system buffer cache) for moving audio and video data into and out of user space. We had to change the buffer cache and paging system to interoperate properly with these changes. We needed to add extensive logging of kernel operation to help track down and eliminate realtime performance and scheduling problems. This was in addition to many minor changes to enhance functionality and improve performance.

Fortunately, we were able to perform most application development on Linux-based PCs. This was a great advantage, because the full Linux development environment was available and we could perform extensive testing quickly. On the last day of March 1999, we shipped the first TiVo boxes to retailers, the first commercial DVR ever shipped in the world. We then published our Linux kernel sources with our changes, many of which have been incorporated into new versions of the Linux kernel.

OPERATION OF THE TIVO SERVICE

To provide all of its functionality, the TiVo box requires the TiVo Service, which operates on a large number of Red Hat Linux-based PC servers. Conceptually, the service maintains a large database of information destined for download to TCDs when they contact the service.

This information is composed of many different elements:

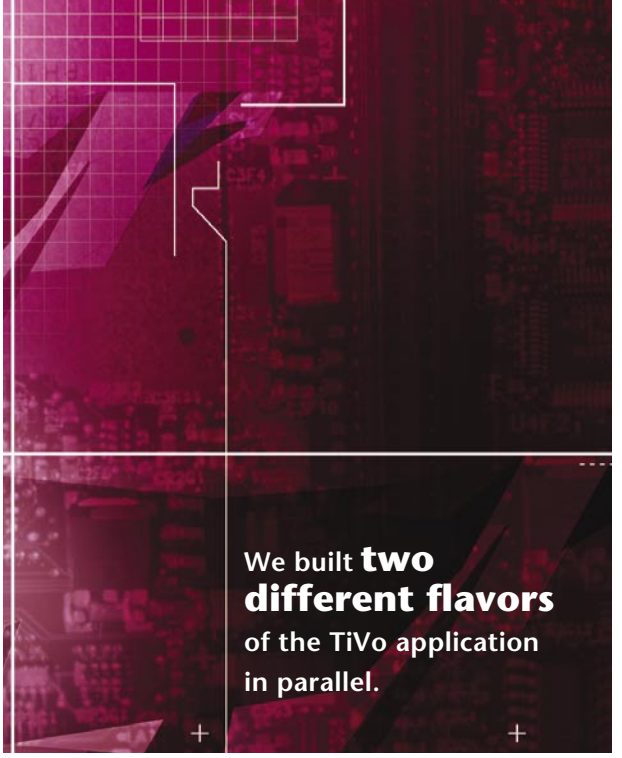
- Fourteen days of program guide information.
- Original software releases for the various TiVo DVR devices that have been produced over time, as well as upgrade releases.
- TiVo Showcases, which are customized interactive promotions for networks, programs, or products. Because the TCD allows viewers to skip quickly through regular television advertisements, we developed the TiVo Showcase technology as another avenue for advertisers to reach consumers. Viewers may choose to enter a Showcase where they are presented with in-depth information about a product, program, or movie. These viewers are truly "qualified," because they make the choice to view the material, increasing their value to advertisers.
- Capture requests, which are instructions to the DVR to record particular programs or video segments. These may be made directly available to the viewer or integrated into Showcases or other promotions.
- Messages from TiVo concerning aspects of the service or other information.
- Security keys and related information.

The service also maintains a record of every TCD manufactured, along with its related security keys. This allows the service to authenticate a TCD and determine what level of service to provide.

Each night, this large database is processed to produce customized packages of data for each zip code in the country or perhaps other subsets of the active TCD universe. This minimizes the amount of data downloaded to each TCD, a desirable feature because in general TiVo pays the transport costs for the data. This processing is performed on a parallel array of Linux-based servers; the

result is tens of thousands of customized packages.

These packages are then copied onto another parallel array of Linux servers called distribution servers. Their job is to field connections from TCDs (which may be via either broadband connections or dial-up modems) to determine if those TCDs are provided service and to download all the appropriate information for that TCD.



At the time of this connection, a TCD may also upload a file containing anonymous viewing information about what was watched, as well as when and how the various trick-play modes were used. This data is uploaded to a set of backhaul servers running Linux, where additional steps are taken to guarantee that the data is anonymous. For example, the names of the files used to store this data are chosen at random and the file creation times zeroed out. Periodically, the logs from randomly chosen subsets of the distributors are copied into a single directory on another server where the file times are zeroed again. The logs are then parsed and aggregated into a database of viewing information.[7]

OPEN SOURCE AT TIVO INC.

TiVo uses open source products and technologies in many different ways, both in operations and in the TiVo DVR itself.

GPL-based software. Fundamental to both the TiVo Client Device and TiVo Service is the Linux operating system. Within the TiVo Service, we use standard Linux distributions, typically from Red Hat.

The client software distribution is a custom configuration built by TiVo. It is important to strictly control the size and contents of the distribution, as it needs to fit into a system with limited disk space and memory. TiVo also needs to ensure that all software is verified and validated. Thus, we begin with a "bare" Linux 2.4 kernel and populate the user environment with only those commands and utilities necessary for operation.

We also rely on GNU libc (glibc) and related libraries for standard C library functionality. We use the Bourne Again Shell (bash shell) from the Free Software Foundation for many scripting functions on the client device.

Public domain software. This is software that has been made available for any use, with no restrictions. Many public domain packages are available, the most notable of which are the X Window System and BSD operating system. Such packages may have some license limitations; for example, BSD simply requires that the University of California at Berkeley be acknowledged as a copyright holder in the software. At TiVo the main public domain tool we use is the Tcl scripting language. TiVo developed a proprietary extension to Tcl for manipulating the database and media file system on the TCD.

Software development. TiVo performs all development for the TiVo Service and TiVo Client Device on Red Hat Linux-based PCs. Using a cross-compilation environment built on the GCC compiler suite and GNU make, a developer can build software for any of the currently supported microprocessor types (PowerPC, MIPS, and x86) and client devices (Series 1, Series 2, DirecTV with TiVo, etc.). Software built for a TCD may then be downloaded into a particular client device for testing. Interactive debugging is supported using gdb on the workstation and remote gdb debugging on the TCD. Generally, the TiVo application is written in C++.

We build two different flavors of the TiVo application in parallel: a debug version and a release version. We attempt to have as little source difference as possible for each version and rely on macros to turn on or off embedded debugging aids. For example, we make liberal use of assertion checking during debugging, compiling out such assertions for the release version. Because of the complexity of the application, we also embed an extensive in-line logging facility that allows developers to perform detailed analyses.

Red Hat Linux is also used for build and release management. Using an array of Linux-based PCs, TiVo performs complete software builds for each client device type continuously during the day as new changes are made. As of this writing, seven different releases are under active development, each built with separate debug and release versions for the Series 2 TCD and a Red Hat 7.x-based PC. The source trees for these releases range in size from

From Server Room to Living Room

650,000 lines of code to nearly 1 million lines, including GPL and public domain sources. These builds are made available to developers immediately or as regular packages after daily automated regression testing. We use the GCC tool-chain as a cross-compilation environment for all software and take great advantage of its many features and enhancements.

All software under active development is maintained using the commercial Perforce version and configuration management system, a cross-platform proprietary software product. Although we began development six years ago using the GPL-based Concurrent Versioning System (CVS), the challenges of source control and configuration management in the TiVo environment soon dictated the use of a more robust and externally supported solution. The actual source trees are maintained on a Red Hat Linux-based server.

We use the Red Hat Package Manager (RPM) for packaging software builds for distribution, download, and installation. Perl and Python are used in development as scripting and macro-programming languages. Tcl is used in some cases, as is the Tk windowing package for the construction of various development support tools.

As is usual in most software development environments, many programmers rely on the GPL-based emacs text editor, whereas another large group relies on the "charityware" vim text editor.

MANAGING OPEN SOURCE

The use of open source in combination with proprietary software requires attention to how the software is combined and maintained. In general, all open source software in use at TiVo, with the exception of public domain software, falls under the GNU Public Library License. GPL-based software is maintained under separate source directories with the COPYING file prominent in the base directory for the software. Except for the libraries, the open source software we use is in the form of utilities (e.g., the bash shell, the Linux modutils package, or syslogd) or as part of the software build chain (such as GNU make). Over time, the ratio of open source software lines in use versus proprietary software has declined. This is a result of both the increasing amount of proprietary software and replacement of open source software with proprietary software that is better tuned to our needs.

TiVo uses only those libraries released under the GNU Lesser General Public License (LGPL). This allows the direct linking of proprietary software with those libraries. For example, we rely on the glibc for standard C and C++ library routines.

HARDWARE DRIVER SUPPORT

The TiVo Client Device is of necessity a closed system. As a service provider, we must prevent theft of service, so TiVo pays a great deal of attention to security of the device and resistance to hacking. Additionally, we sell the TCD at a price that provides a net margin to retailers, but no profit to us. Our profits come from providing service to each device over time, rather than from up-front costs.

For this reason, it is important that the devices not be re-purposed and that our software not be replaced with other implementations of DVR functionality. Thus, hardware drivers are kept as proprietary sources and are built as loadable modules. Our interpretation of the GPL is that such use is permitted; we are careful to separate these modules from the Linux kernel sources to avoid any confusion. In fact, as has been demonstrated by the very active TiVo hacking community, it is possible to take our published sources, build a Linux kernel, and load it on some versions of the TCD. The kernel will run properly, loading the proprietary modules and executing the TiVo application, thus staying within the GPL spirit.

This use is somewhat controversial. Advocates of the GPL and the Free Software Foundation interpret the GPL more stringently to disallow the use of proprietary modules. On the other hand, Linus Torvalds has stated that proprietary loadable modules are acceptable. Regardless, the use of proprietary modules has contributed greatly to the explosive growth in the use of Linux and GNU software in general, and is key to the use of Linux in embedded systems. Had this ability not been available,

proprietary operating systems would certainly continue to be the norm for most commercial embedded applications.

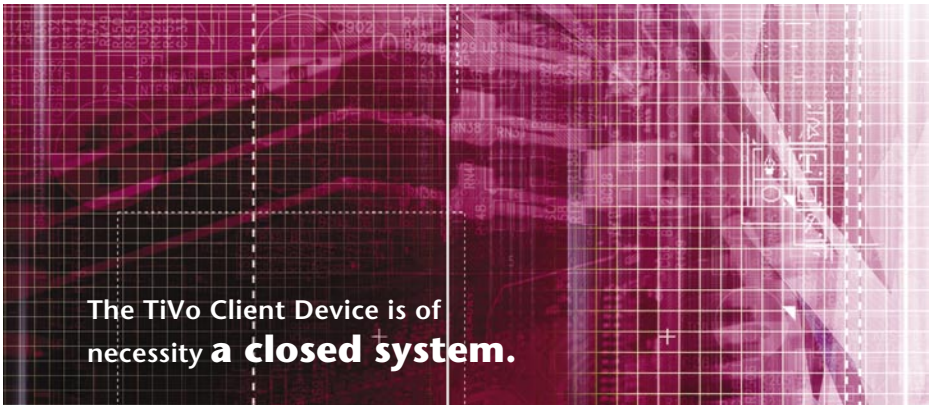
GPL software is supposed to be "free, as in free speech not free beer" (attributed to GNU Project and FSF founder Richard Stallman). When such speech is not a modification of another's speech, it should also be recognized that the freedom not to speak must be preserved as a corollary.

MAKING SOURCES AVAILABLE TO SUBSCRIBERS

Under terms of the GPL, if we incorporate GPL-based sources into commercial products, purchasers of such products must have access to incorporated GPL sources.

TiVo complies with these terms by publishing on its Web site (http://www.tivo.com/linux) the GPL and public domain sources used in each release, along with the tools used to build those sources. The intention is that any interested party would be able to re-create the derived binary versions included in the device using those sources. No proprietary software is included in these sources.

During development, TiVo may modify these sources. These modifications are included in the sources made available on the Web site, again under provisions of the GPL. From time to time, TiVo also makes some of its proprietary developments available under the GPL or as public domain sources for the benefit of the developer community. For example, TiVo has released a software development kit as part of its Home Media Option (HMO), a software application for the TCD that enables the playback of music or digital photographs across a home network from a personal computer, allows transfer of programs among TCDs in the home, and provides the ability to schedule the recording of programs from a remote Web browser. This kit documents our network protocols and provides sample code for a server application (see http://www.tivo.com/developer).



At TiVo we believe that making our modifications and enhancements to these sources generally available is part of our contribution back to the open source community and is an acknowledgment of the value that open source has brought to our products. As we neither sell nor distribute the software used in development and in the TiVo Service itself, we do not publish the underlying GPL-based or public domain sources.

TRACKING NEW VERSIONS

Because of the closed nature of the TiVo Client Device, there is little need or desire for TiVo to keep up with new versions of the open sources incorporated into the device. In fact, once a particular component has been debugged and verified, we want to avoid any changes to that component, which would require re-verification of its operation. Indeed, many of the sources in use are those first incorporated in the first TCD software release.

There are a few notable exceptions to this philosophy. First, the Linux kernel is a rapidly evolving software project, with performance enhancements, new capabilities, and bug fixes being added continuously. A great deal of hardware driver work is also under way, and at TiVo we wish to take advantage of new hardware and better performance as appropriate. This makes it worthwhile for us to track the kernel closely. Thus, while a particular kernel version will be selected, tested, and frozen early in the software development cycle, new software releases from TiVo will typically incorporate the latest kernel changes.

Secondly, the GCC tool suite is a critical component of TiVo's software development environment. Clearly, it is important that we verify and validate the operation of the GCC tools before using them, a tremendous undertaking. This means that we carefully choose a particular version of the tool suite to work with and validate it by building all current versions and releases under development followed by extensive testing of the resulting binaries. Because of the great effort involved in this testing, we continue to use the same tools for a considerable time period. Thus, TiVo has moved to a new GCC version only once in its existence. Similarly, we rely a great deal on the operation of the glibc family of libraries, and we typically move to new libraries coincident with a move to new GCC tools.

From Server Room to Living Room

CLOSING REMARKS

The evolution of open source has been remarkable since Richard Stallman proposed the concept. Today, many open source projects are recognized as high-quality undertakings, often exceeding the quality of commercial offerings while providing greater performance and more features. This comes naturally from the open nature of the source code, where many developers can study the code, find and fix flaws, and validate the security and proper operation of the code for production use.

Open source has been beneficial to TiVo and will continue to be in the foreseeable future. Careful management of our sources to abide by the terms of the GNU General Public License while protecting our proprietary developments is a small price to pay for this benefit. Q

REFERENCES AND FOOTNOTES

1. The Unix trademark is registered to and owned by The Open Group.
2. "Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable," Kirk McKusick, http://www.oreilly.com/catalog/opensources/book/kirkmck.html.
3. Yes, I was at this first meeting, representing SGI. I was also present when the Hamilton Group traveled to New York to meet with AT&T and hear how AT&T would work with the remainder of the computer industry. The arrogance shown by AT&T at that meeting toward the computer manufacturers assured the creation of an alternative effort.
4. "Endianess" of microprocessors continues to cause a great chasm in the computing world. The MIPS microprocessor architecture supports both little-endian and big-endian computing paradigms. Because DEC and Compaq led the ACE initiative, the group focus was on the little-endian paradigm. This led to the formation of an opposing effort, the Apache group (think "big Indian"), which promoted a big-endian alternative, based on System V Unix.
5. "OSI Position Paper on the SCO-vs.-IBM Complaint," Eric Raymond and Rob Landley, http://www.opensource.org/sco-vs-ibm.html.
6. I am sometimes asked why we didn't use a BSD-based operating system. The BSD license would allow us to keep any modifications to the source proprietary. BSD was more oriented toward server applications, however, and the development community around Linux was very active and growing. This meant that we could look forward to aggressive driver support for new devices and capabilities, as well as ongoing improvements to the kernel. BSD-based operating systems were mature and on a much slower development cycle. The advantages of using Linux outweighed any loss of proprietary opportunities.
7. A full description of this process may be found in a whitepaper TiVo submitted to the Federal Trade Commission describing TiVo's privacy management process: http://www.tivo.com/pdfs/ftc_letter.pdf.

LOVE IT, HATE IT? LET US KNOW:
queue-ed@acm.org or
www.acmqueue.org/forums

JIM BARTON is co-founder, chief technical officer, and senior vice president of research and development at TiVo Inc. He is responsible for all product development,and his area of emphasis is on software and digital video streaming technologies behind the TiVo Service. Prior to co-founding TiVo, Barton was president and CEO of Network Age Software, a company he founded to develop software products targeted at managed electronic distribution. He also held executive positions with SGI, where he worked to develop the only large-scale interactive television system to be put into operational service. He served as CTO of Interactive Digital Solutions, a joint venture of Silicon Graphics and AT&T Network Systems to develop interactive television systems. Barton held positions at Hewlett-Packard and Bell Laboratories in operating system and networking technology and product development. He has a bachelor's degree in electrical engineering and a master's degree in computer science from the University of Colorado at Boulder.