Advanced search

Linux Journal Issue #100/August 2002

Archive Index

Advanced search

Linux Timeline

LWN

LJ Staff

Issue #100, August 2002

100 of the most significant events in Linux history.

As part of our 100th issue celebration, we present 100 of the most significant events in Linux history. As shown in the timeline, the first issue of *Linux Journal* coincided with the release of Linux 1.0. Ever since, the fortunes of our magazine have followed those of Linux at large.

It's been a wild eight years, filled with a variety of exciting events. Choosing only 100 was a difficult task, and certainly some readers will be quick to point out events they would have chosen that we did not, but the following manages to maintain the roller-coaster ride that is Linux history.

We would like to recognize our indebtedness to Rebecca Sobol and Jonathan Corbet at Linux Weekly News, for allowing us to borrow heavily from the timeline featured on their site and for their accurate and gracious historical editing.

August 1991

“Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-) Linus (PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(."

September 1991

Linux version 0.01 is released and put on the Net.

April 1992

The first Linux newsgroup, comp.os.linux, is proposed and started by Ari Lemmke.

October 1992

Peter MacDonald announces SLS, the first standalone Linux install. At least 10MB of space on disk was recommended.

June 1993

Slackware, by Patrick Volkerding, becomes the first commercial standalone distribution and quickly becomes popular within the Linux community.

August 1993

Matt Welsh's *Linux Installation and Getting Started*, version 1 is released. This is the first book on Linux.

March 1994

The first issue of *Linux Journal* is published. This issue featured an interview with Linus Torvalds and articles written by Phil Hughes, Robert "Bob" Young, Michael K. Johnson, Arnold Robbins, Matt Welsh, Ian A. Murdock, Frank B. Brokken, K. Kubat, Micahel Kraehe and Bernie Thompson. Advertisers in the premier issue include Algorithms Inc., Amtec Engineering, Basmark, Fintronic (later became VA Research, VA Linux Systems, then...), Infomagic, Prime Time Freeware, Promox, Signum Support, SSC, Trans Ameritech, USENIX, Windsor Tech and Yggdrasil.

Linux 1.0 is released.

June 1994

While at a conference in New Orleans, Jon "maddog" Hall persuades Linus to port Linux to DEC's 64-bit Alpha computer processor chip. Less than two weeks later, maddog had also persuaded DEC to fund the project. An Alpha workstation was immediately sent to Linus. "Digital [DEC] and the Linux community formed the first truly successful venture of suits and Linux geeks working together", said maddog.

Linux International, a nonprofit vendor organization, is founded by Jon "maddog" Hall. Linux International goes on to become a major contributor to the success of Linux, helping corporations and others work toward the promotion of the Linux operating system.

August 1994

Linux trademark dispute: is Linux trademarked? William R. Della Croce, Jr. files for the trademark "Linux" on August 15, 1994, and it is registered in September. Della Croce has no known involvement in the Linux community yet sends letters out to prominent Linux companies demanding money for use of the trademark "Linux". A lawsuit is filed in 1996 against Della Croce. Plai.pngfs in the suit include Linus Torvalds; Specialized Systems Consultants, Inc. (publishers of *Linux Journal*); Yggdrasil Computing, Inc.; Linux International; and WorkGroup Solutions (also known as LinuxMall). The plaintiffs prevail, and in 1997 announce the matter as settled by the assignment of the mark to Linus Torvalds on behalf of all Petitioners and Linux users.

September 1994

Linux is first mentioned in the mainstream press. *Wired* magazine features an article titled "Kernel Kid", by Seth Rosenthal. He writes: "So, is Linus going to become the Bill Gates of Finland? Maybe not. He claims to be 'by no means a good student' and is in no hurry to graduate since 'Linux has taken a lot of time from my studies, and I like the work I have at the University which keeps me alive.'"

Randolph Bentson reports on the world's first vendor-supported Linux device driver in *Linux Journal*. Cyclades gave him a multiport serial card in exchange for developing a Linux driver for it.

December 1994

A major tradeshow and conference take notice of Linux. Open Systems World features a Linux track, hosted by *Linux Journal*. Two days of seminars include Eric Youngdale, Donald Becker, Dirk Hohndel, Phil Hughes, Michael K. Johnson and David Wexelblat as speakers.

April 1995

Linux Expo, the first Linux-specific tradeshow and conference series, launches, thanks to the folks at North Carolina State University and in particular, Donnie

Barnes. Speakers include Marc Ewing, Rik Faith and Michael K. Johnson, among others. Linux Expo snowballs and becomes the most popular and well-attended annual Linux show for the next several years (after three years Red Hat takes over organization and becomes the major sponsor). The price for entry into the exhibit hall and a pass to the conferences? $4.

January 1997

First “Linux virus” discovered. Called Bliss, it actually works on any UNIX-like OS and offers a helpful—“bliss-uninfect-files-please” command-line option. Alan Cox points out that Bliss “does not circumvent the security of the system, it relies on people with privilege to do something dumb” and reminds users to install digitally signed software from trustworthy sites only and to check signatures before installing.

“In fact it's probably easier to write a virus for Linux because it's open source and the code is available. So we will be seeing more Linux viruses as the OS becomes more common and popular.” —Wishful thinking from McAfee

January 1998

Linux Weekly News begins publication with Jonathan Corbet and Elizabeth Coolbaugh as founders. The very first issue, dated January 22, was just a tiny hint of what *LWN* was to become.

Netscape announces that they will release the source to their browser under a free software license. This almost certainly remains one of the most important events of the year; it opened a lot of eyes to what Linux and free software could provide.

Red Hat Advanced Development Labs (RHAD) is founded. It has since become one of the higher-profile places where people are paid to develop free software and an important component of the GNOME Project. RHAD is able to attract developers like “Rasterman” (although only for a short time) and Federico Mena-Quintero.

February 1998

The Cobalt Qube is announced and immediately becomes a favorite in the trade press due to its high performance, low price and cute form factor. Cobalt's Linux engineering is done by none other than David Miller, the source of much that is good in the Linux kernel.

The Linux user community wins *InfoWorld*'s technical support award; Red Hat 5.0 also won their Operating System award. But it was the tech support award that truly opened some eyes; everybody had been saying that Linux had no support. This was the beginning of the end of the “no support” argument.

Eric Raymond and friends come up with the term “open source”. They apply for trademark status and put up the opensource.org web site. Thus begins the formal effort to push Linux for corporate use.

March 1998

Consumer advocate Ralph Nader asks the large PC vendors (Dell, Gateway, Micron, etc.) to offer non-Microsoft systems, including systems with Linux installed.

April 1998

Linux is covered by the US National Public Radio news, marking one of its first appearances in the mainstream, nontechnical press.

O'Reilly holds the “first ever” Free Software Summit, featuring Larry Wall, Brian Behlendorf, Linus Torvalds, Guido van Rossum, Eric Allman, Phil Zimmermann, Eric Raymond and Paul Vixie.

May 1998

The Google search engine pops up. Not only is it one of the best search engines around, but it's based on Linux and features a Linux-specific search page.

Big databases start to arrive. Support for Linux is announced by Computer Associates for their Ingres system and by Ardent Software for their O2 object database.

June 1998

“Like a lot of products that are free, you get a loyal following even though it's small. I've never had a customer mention Linux to me.” —Bill Gates, *PC Week*, June 25, 1998

“...these operating systems will not find widespread use in mainstream commercial applications in the next three years, nor will there be broad third-party application support.” —The Gartner Group says there is little hope for free software.

A Datapro study comes out showing that Linux has the highest user satisfaction of any system; it also shows Linux to be the only system other than Microsoft Windows NT that is increasing its market share.

IBM announces that it will distribute and support the Apache web server after working a deal with the Apache team.

July 1998

The desktop wars rage as KDE and GNOME advocates hurl flames at each other. Linus gets in on the act, saying that KDE is okay with him. In this context, KDE 1.0 is released. The first stable release of the K Desktop Environment proves to be popular, despite the complaints from those who do not like the licensing of the Qt library.

Informix quietly releases software for Linux. Meanwhile, Oracle beats Informix to the punch PR-wise and makes a Linux-friendly announcement first, suggesting that they would soon be supporting Linux. Oracle promises to make a trial version available by the end of 1998, a deadline they beat by months. This, seemingly, was one of the acid tests for the potential of long-term success for Linux; a great deal of attention resulted from both Informix's and Oracle's announcements.

Informix announces support for Linux effectively moments after Oracle does so. Sybase later announces their support for Linux also.

Linus appears on the cover of *Forbes* magazine. A lengthy story presents Linux in a highly positive manner and brings the system to the attention of many who had never heard of it before. Linux begins to become a household word.

September 1998

LinuxToday.com is launched by Dave Whitinger and Dwight Johnson. The site, later acquired by Internet.com, arguably becomes the most well-read and visited Linux portal of all time.

Microsoft's Steve Ballmer admits that they are “worried” about free software and suggests that some of the Windows NT source code may be made available to developers. The same month Microsoft goes on to list Linux as a competitive threat in its annual SEC (US Securities and Exchange Commission) filing. Speculation abounds that their real purpose is to influence the upcoming antitrust trial.

October 1998

“For the moment, however, the company from Redmond, Washington, seems almost grateful for the rising profile of Linux, seeing it as an easy way of demonstrating that Windows is not a monopoly, ahead of its antitrust trial, scheduled to begin on October 15. That may be short-sighted. In the long run, Linux and other open-source programs could cause Mr. Gates much grief." — *The Economist*, October 3, 1998

Intel and Netscape (and two venture capital firms) announce minority investments in Red Hat Software. The money is to be used to build an "enterprise support division" within Red Hat. An unbelievable amount of press is generated by this event, which is seen as a big-business endorsement of Linux.

Corel announces that WordPerfect 8 for Linux will be downloadable for free for "personal use". They also announce a partnership with Red Hat to supply Linux for the Netwinder.

December 1998

A report from IDC says that Linux shipments rose by more than 200% in 1998, and its market share rose by more than 150%. Linux has a 17% market share and a growth rate unmatched by any other system on the market.

January 1999

"Microsoft Corp. will shout it out to the world when Windows 2000 finally ships. Linux creator Linus Torvalds announced the arrival of the next generation of Linux, version 2.2, with a simple note to the Linux-kernel mailing list." —Steven J. Vaughan-Nichols, Sm@rt Reseller

Samba 2.0 is released. It contains a reverse-engineered implementation of the Microsoft domain controller protocols, allowing Linux servers to provide complete services to Windows networks.

Hewlett-Packard and Compaq announce plans to offer Linux-based systems. Later, Dell also announces plans to begin selling Linux-installed systems. SGI contents itself with providing information on how to bring up Linux on its systems.

Loki Entertainment Software announces that it will port *Civilization: Call to Power* to Linux.

February 1999

Linux and BSD users unite for "Windows Refund Day". They visit Microsoft, hoping to return the unused Windows licenses that they were forced to acquire when they purchased a computer system bundled with the OS.

March 1999

“Like a Russian revolutionary erased from a photograph, he is being written out of history. Stallman is the originator of the Free Software movement and the GNU/Linux operating system. But you wouldn't know it from reading about LinuxWorld (Expo). Linus Torvalds got all the ink.” —Leander Kahney, *Wired* magazine, March 1999

The first LinuxWorld Conference and Expo is held in San Jose, California. As the first big commercial “tradeshow” event for Linux, it serves notice to the world that Linux has arrived; 12,000 people are said to have attended.

Linux Magazine debuts, bringing some additional competition to the Linux print business. Later, other magazines rise and fall including *Open*, *Journal of Linux Technology* (*JOLT*) and *Maximum Linux*.

VA Research buys the Linux.com domain for $1,000,000 and announces plans to turn it into a Linux portal. Microsoft's rumored bid for the domain is frustrated.

April 1999

“...please imagine what it is like to see an idealistic project stymied and made ineffective because people don't usually give it the credit for what it has done. If you're an idealist like me, that can ruin your whole decade.” —Richard Stallman on GNU/Linux

Al Gore's presidential campaign web site claims to be open source. That claim is gone, but the site still claims: “In the spirit of the Open Source movement, we have established the Gore 2000 Volunteer Source Code Project; www.algore2000.com is an 'open site'.“

HP announces 24/7 support services for the Caldera, Turbolinux, Red Hat and SuSE distributions. They also release OpenMail for Linux.

The Linux FreeS/WAN Project releases a free IPSec implementation, allowing Linux to function as a VPN gateway using what is now the industry standard.

“But the mere fact that there is now an official SEC document that includes the text of the GPL serves as fairly astonishing proof that the rules of the software business really are being rewritten.” —Andrew Leonard, *Salon*

May 1999

“Those two little words—open source—have become a magical incantation, like portal in 1998 or push in 1997. Just whisper them and all will be yours: media attention, consumer interest and, of course, venture capital.” —Andrew Leonard, *Wired*

August 1999

First Intel IA-64 “Merced” silicon. Although Intel had given simulators to several OS vendors, Linux is the only OS to run on the new architecture on its first day. *The Register* headline: “Merced silicon happens: Linux runs, NT doesn't”.

SGI announces the 1400L—a Linux-based server system. SGI also announces a partnership with Red Hat and begins contributing to kernel development in a big way.

Red Hat's initial public offering happens; a last-minute repricing helps to create difficulties for people participating in the community offering. The stock price immediately rises to $50; a value that seems high at the time.

“For the umpteenth time, someone paved paradise, put up a parking lot. For the thousands of Linux coders who've built the utopian open-source movement—offering free help to create a free operating system—the IPO of Red Hat Software was a sure sign of Wall Street cutting the ribbon on the new Linux mall.” —*The Industry Standard*

Motorola jumps into Linux announcements of embedded systems products, support and training services, and a partnership with Lineo.

Sun acquires StarDivision; it announces plans to release StarOffice under the Sun Community Source License and to make a web-enabled version of the office suite.

September 1999

“'Burlington Coat Factory Warehouse Corp. in Burlington, New Jersey is spending $1 million or so to buy 1,250 Linux-equipped PCs from Dell, but it won't pay Red Hat a dime for support', says Michael Prince, chief information officer. 'I suppose Red Hat's business model makes sense to somebody, but it makes no sense to us', he says.” —Daniel Lyons, *Forbes*, May 31, 1999. Then in September, Burlington ended up purchasing support from Red Hat.

The first big Linux stock rush happens. Shares in Applix more than double in volume, reaching nearly 27 million shares—three times the 9 million shares that are actually on the market.

SCO trashes Linux in a brochure distributed in Northern Europe: “Linux at this moment can be considered more a plaything for IT students rather than a serious operating system in which to place the functioning, security and future of a business. Because Linux is basically a free-for-all it means that no individual person/company is accountable should anything go wrong, plus there is no way to predict which way Linux will evolve.”

Stock in Red Hat hits $135/share. The price seems unbelievably high at the time.

October 1999

Sun Microsystems announces that it will release the source to Solaris under the Sun Community Source License. The actual release drew criticism: “In a move aimed at Linux, Sun said it will announce Wednesday that it is making the source code for its new Solaris 8 operating system 'open'. Webster's has lots of definitions for the word, including 'not sealed, fastened, or locked'. But when you dig into the details of Sun's announcement, you'll find that what it is offering doesn't come close to meeting the dictionary's definition, let alone that of the Open Source movement.” —Lawrence Aragon, Redherring.com, January 26, 2000

November 1999

“...if there's one thing about Linux users, they're do-ers, not whiners.” —Andy Patrizio,

Red Hat buys Cygnus for almost $700 million in stock. Rumors of other acquisitions by Red Hat begin to circulate and show no signs of stopping.

December 1999

VA Linux Systems goes public after two repricings (originally priced at $11-$13/ share). The final IPO price is $30/share; that price rises immediately to $300 before closing around $250. It sets the record for the biggest IPO rise in the history of the NASDAQ.

“Gee. Remember when the big question was 'How do we make money at this?”' —Eric Raymond

January 2000

VA Linux Systems announces SourceForge (although the site had actually been up and running since November 1999). SourceForge also makes the code for its operation available under the GPL. By the end of the year, SourceForge hosted over 12,000 projects and 92,000 registered developers.

Version 1.0 of Red Flag Linux is released in the People's Republic of China.

Transmeta breaks its long silence and tells the world what it has been up to—the Crusoe chip, of course.

The Linux Professional Institute announces the availability of its first Linux professional certification exam.

Linux wannabe press releases flow from companies trying to ride on the success of Linux stocks. Vitamins.com, for example, posts the following: “Vitamins.com has further distinguished itself in the competitive Internet health industry race by being one of the first to integrate the Linux Operating System, produced by Red Hat, the leading developer and provider of open-source software solutions.”

February 2000

The latest IDC report suggests that Linux now ranks as the “second-most-popular operating system for server computers”, with 25% of the server operating system sales in 1999. Windows NT is first with 38% and NetWare ranks third with 19%. IDC previously predicted that Linux would get up to the number two position—in 2002 or 2003. The revolution appears to be well ahead of schedule.

VA Linux Systems acquisition of Andover.net in a high-profile purchase that values Andover shares at 0.425 of VA's, or roughly $50/share. Andover.net is the owner of the popular web sites Slashdot.org and Freshmeat.org.

LinuxMall.com and Frank Kaspar and Associates also have made plans to merge. LinuxMall.com has been at the top of the retail side of Linux almost since the very beginning; Kaspar is one of the largest distribution channels.

Red Hat wins *InfoWorld*'s “Product of the Year” award for the fourth time in a row.

March 2000

“The law in open code means that no actor can gain ultimate control over open-source code. Even the kings can't get ultimate control over the code. For example, if Linus Torvalds, father of the Linux kernel, tried to steer GNU/Linux in a way that others in the community rejected, then others in the community could always have removed the offending part and gone in a different way. This threat constrains the kings; they can only lead where they know the people will follow.” —“Innovation, Regulation, and the Internet” by Lawrence Lessig for *The American Prospect*.

A new version of LILO is posted that is able to get past the 1024-cylinder boot limit that has plagued PC systems for years.

The latest Netcraft survey shows Apache running on just over 60% of the Web.

Caldera Systems goes public after a short delay, on March 21. The stock, which was offered at $14/share, began trading at $26 and closed at $29.44. It thus registered a 110% gain on its first day.

“Caldera knows of no company that has built a profitable business based in whole or in part on open-source software.” —Caldera SEC filing

Walnut Creek (the parent company for Slackware) and BSDi announce their merger. Yahoo! will be taking an equity investment in the new company.

Motorola Computer Group announces the release of its HA Linux distribution. This distribution is aimed at telecommunications applications that require very high amounts of uptime; it includes hot-swap capability and is available for the i386 and PowerPC architectures.

The Embedded Linux Consortium is announced. Its goal is “to amplify the depth, breadth and speed of Linux adoption in the enormous embedded computer market”. The initial leader will be Rick Lehrbaum, the man behind the LinuxDevices.com and DesktopLinux.com web sites, among other things.

Ericsson announces its “Screen Phone HS210” product—a Linux-based telephone with a touchscreen that can be used for e-mail, web browsing, etc. Ericsson and Opera Software also announce that Ericsson's (Linux-based) HS210 Screen Phone will incorporate the Opera web browser.

April 2000

Code is ruled to be speech. On April 4, 2000, the United States Court of Appeals for the Sixth Circuit published its decision regarding Peter Junger's challenge to the Export Administration Regulations that prevented him from posting information on the Internet that contained cryptographic example code. Most critical in the ruling: “Because computer source code is an expressive means for the exchange of information and ideas about computer programming, we hold that it is protected by the First Amendment.”

Andy Tanenbaum releases the the Minix operating system under the BSD license. Had Minix been open source from the beginning, Linux may never have happened.

May 2000

SuSE releases the first supported Linux distribution for the IBM S/390 mainframe.

“Approximately 140 distribution companies exist across the globe. We believe all but the top five will be bought, will go out of business or will be relegated to insignificance. Market-share leaders are currently defined around geographic boundaries. Red Hat has the largest global brand recognition and leading North American market share; SuSE leads in Europe, Turbolinux leads in Asia, and Conectiva leads in South America.” —Keith Bachman, an analyst for WR Hambrecht, predicting in *The Red Herring*

June 2000

Commercial considerations help prompt the relicensing of MySQL under the GPL. Now the two freely available databases that are widely used in the Linux and Free Software communities, PostgreSQL and MySQL, meet the Debian Free Software Guidelines and the Open Source Guidelines. In addition, Progress Software forms a new company, NuSphere, just for the purpose of supporting MySQL.

July 2000

“In a world of NDA-bound business agreements, Debian is an open book. In a world of mission statements, Debian has a social contract. At a time when commercial distributors are striving to see how much proprietary software they can pack into a box of Linux, Debian remains the bastion of software freedom —living proof that you can have a fully functional and usable operating system without needing any proprietary code.” —Evan Leibovitch, ZDNet

Sun announces that StarOffice is to be released under the GPL. The code is going to be reworked, integrated with Bonobo and GTK, and released as a set of reusable components. StarOffice will also be reworked to use a set of open XML-based file formats.

Oracle's Linux-based internet appliance system hits the shelves. The “New Internet Computer” (NIC) is the latest result of Larry Ellison's long personal crusade to make non-Microsoft systems available to the world. It's aimed at people who only want access to the Net; as such, it's essentially a $199 (without monitor) X terminal.

Reports first appear that SCO may be purchased by Caldera. Later in 2000 Caldera and SCO announce their intent for Caldera International to be formed from Caldera's existing operation and two of SCO's three divisions.

Ted Ts'o steps forward to become the new 2.4 status list maintainer. Alan Cox was doing the job until he said that it was time to “find someone else to maintain it”. Ted Ts'o responded to Linus' subsequent call for a new status list maintainer.

August 2000

HP, Intel, IBM and NEC announce the “Open Source Development Lab”, which makes large hardware available to Linux developers for benchmarking and testing.

September 2000

“I'm a bastard. I have absolutely no clue why people can ever think otherwise. Yet they do. People think I'm a nice guy, and the fact is that I'm a scheming, conniving bastard who doesn't care for any hurt feelings or lost hours of work if it just results in what I consider to be a better system.” —Linus Torvalds trying to change his image.

The RSA patent expires, allowing for secure web transactions without proprietary software.

Trolltech releases the Qt library under the GPL, putting a definitive end to a long-running and unpleasant license flame war.

The CueCat fiasco begins. Digital Convergence attempts to shut down programmers who have written Linux drivers for its CueCat bar code scanner. The company has given out large numbers of these scanners for free, expecting people to use them with its proprietary software and web site. The threats cause the drivers to become marginally harder to find for a short period, after which the company declares victory and moves on.

October 2000

Microsoft says that penguins can mutate in a European print ad that quickly becomes famous.

December 2000

“I was dumbfounded to discover that installing Linux was easy. Why? Well, the world has changed. No more do you have to understand everything about Linux before you install it, downloading the many chunks of code necessary to run a complete system and getting them all to work together. That was BSW—before shrink-wrap. With companies such as Red Hat and Corel putting all the software you need in a box, the pain is (nearly) gone.” —John Schwartz, *Washington Post*

IBM announces plans to invest $1 billion in Linux in 2001.

January 2001

The long-awaited 2.4.0 kernel was released on January 4.

The US National Security Agency (NSA) releases SELinux under the GPL. SELinux offers an additional layer of security checks in addition to the standard UNIX-like permissions system.

March 2001

The Linux 2.5 kernel summit is held in San Jose, California; it is, perhaps, the most complete gathering of Linux kernel hackers in history.

April 2001

IBM gets into trouble over its “Peace, Love and Linux” graffiti in several cities.

“Slackware has always made money (who else producing a commercial distribution can say that?), but with BSDi we ended up strapped to a sinking ship.” —Patrick Volkerding

May 2001

Sony's PlayStation Linux kit, shipped in Japan, sells out in eight minutes despite a doubling of the available stock.

June 2001

Sharp announces its upcoming Linux PDA based on Lineo's Embedix system.

VA Linux Systems exits the hardware business, choosing to focus on SourceForge instead. Later VA drops the word “Linux” from its name altogether, relaunching as VA Software Corporation.

“In a press release issued Wednesday afternoon, VA Linux CEO Larry M. Augustin called the shift in strategy a logical move. 'Our differentiating strength has always been our software expertise', Augustin said”. —*Wired*. You only thought VA was a hardware company.

July 2001

Free Dmitry! Dmitry Sklyarov is arrested in Las Vegas after Adobe complains about the Advanced eBook Processor. The following month he is charged with DMCA violations and conspiracy: the potential penalties add up to 25 years in prison. Dmitry's defense is based on constitutional challenges to the DMCA, on free speech and jurisdictional issues. Later in the year, charges are dropped, conditional on one year of good behavior and testimony in the ElcomSoft trial.

“Although Adobe withdrew its support for the criminal complaint against Dmitry Sklyarov, we respect the grand jury and federal government's decision to prosecute the company, ElcomSoft, and as a law-abiding corporate citizen, Adobe intends to cooperate fully with the government as required by law.” —Adobe's position

November 2001

Sharp Electronics Corporation begins a special Linux developer prerelease of the Zaurus PDA to attract free software developers to the hot new platform.

February 2002

Avaya, the former PBX and enterprise systems division of Lucent, announces Linux-based PBX systems.

“So there are some—and I'd list myself among them—who believe that the return to Earth is a good thing. There's nothing wrong with making a buck, but Linux doesn't benefit from being elevated beyond reality on a shaky foundation.” —Evan Leibovitch takes a look at the post-rush world of Linux.

What Others Have to Say about *Linux Journal*'s 100th Issue

Archive Index Issue Table of Contents

Advanced search

Supporting IPv6 on a Linux Server Node

Ibrahim Haddad

Issue #100, August 2002

This article provides a technical tutorial on setting up IPv6 on a Linux server and connecting it to the IPv6 Internet.

The current version of the IP protocol, IPv4, has proved to be robust, easily implemented, interoperable and has stood the test of scaling to the size of today's Internet, most of which uses IPv4—now nearly 20 years old. IPv4 has been remarkably resilient in spite of its age, but it is beginning to have problems. The initial design of IPv4 did not take into consideration several issues that are of great importance today, such as a large address space providing a solution for the address crunch problem, mobility, security, autoconfiguration and quality of service.

To address these concerns, the Internet Engineering Task Force (IETF) has developed a suite of protocols and standards known as the IP version 6 (IPv6), which incorporates many of the concepts and proposed methods for updating IPv4. Some of the IPv6 features include a new header format, a larger address space (128 bits), an efficient and hierarchical addressing and routing infrastructure, the availability of stateless and stateful address security, built-in security, better support of mobility and a new protocol for neighboring node interaction. As a result, IPv6 is not only going to fix a number of problems in IPv4, it also will add many improvements. IPv6 is expected to replace IPv4 gradually, with the two coexisting for a number of years during a transition period.

Linux IPv6 Implementations

There are two main IPv6 implementations for Linux: the implementation that comes as part of the Linux kernel and the USAGI (UniverSAl playGround for IPv6) implementation. The USAGI Project works to deliver a production-quality IPv6 protocol stack for Linux, tightly collaborating with the WIDE, KAME and TAHI Projects. It is run by volunteers from various organizations contributing to the Linux and the IPv6 communities via the delivery of the IPv6 protocol stack. Currently, there are many efforts in the different distributions teams, and USAGI is trying to unify them so that there is one IPv6 implementation for all Linux distributions.

For the purpose of this article, we use Linux kernel 2.4.5 from kernel.org. We first show how to build a kernel with IPv6 support, then how to upgrade the basic networking software to support IPv6 and finally, how to connect your IPv6-enabled server to the IPv6 Internet using the services from the www.freenet6.net Project.

Supporting IPv6 in the Linux Kernel

The first step is to download the Linux kernel from kernel.org and uncompress it:

```
tar -xzf linux-2.4.5.tar.gz
```

You will have a directory called linux. You need to move this directory into /usr/src and rename it linux-2.4.5 to reflect the kernel version. Next, you need to create a link to the 2.4.5 source directory:

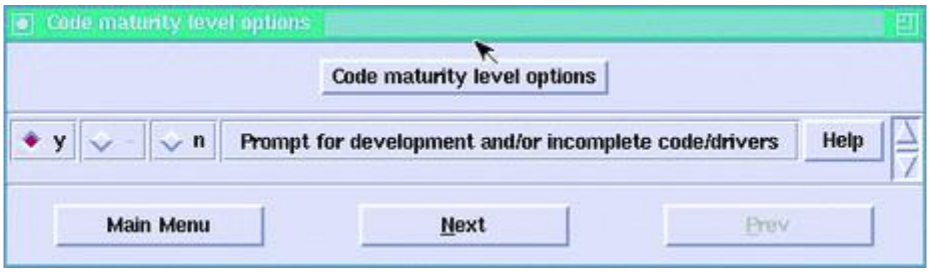
```
ln -s /usr/src/linux-2.4.5 /usr/src/linux
```

Having done that, you need to configure the new kernel to enable support for IPv6:

```
cd /usr/src/linux
make xconfig (or menuconfig)
```

We need to enable two options in the kernel configuration. First, go to Code Maturity Level and enable development/incomplete code/drivers:

```
"Prompt for development and/or incomplete
   code/drivers"    YES
```

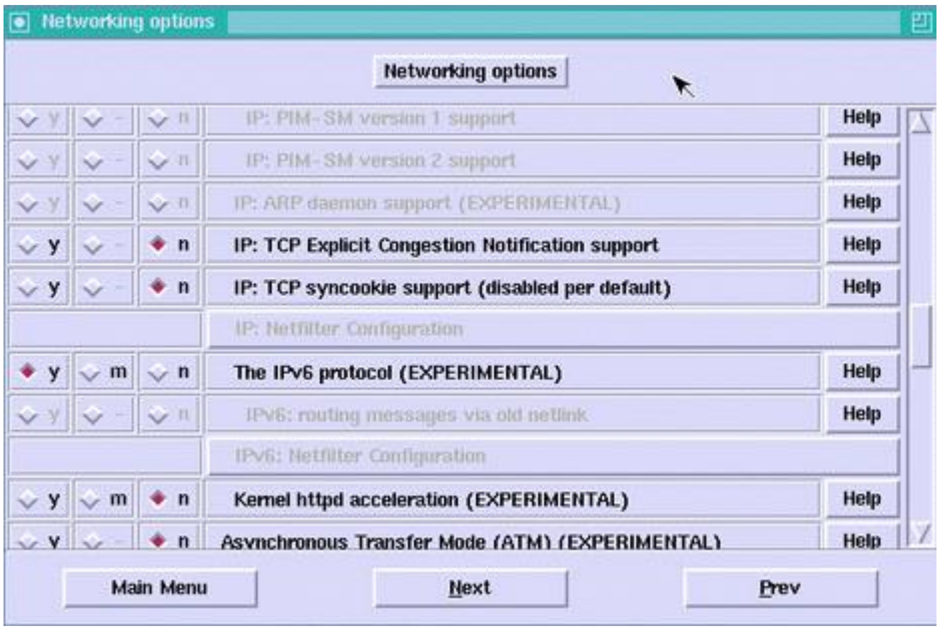


Prompt for development and/or incomplete code/drivers

Figure 1. Enabling Support for Experimental Features

Then go to the Networking Options. There you will enable the IPv6 protocol:

```
IPv6 Protocol (EXPERIMENTAL)    YES
```



```
y - n  IP: PIM-SM version 1 support                             Help
y - n  IP: PIM-SM version 2 support                             Help
y - n  IP: ARP daemon support (EXPERIMENTAL)                    Help
y - [✓ n]  IP: TCP Explicit Congestion Notification support     Help
y - [✓ n]  IP: TCP syncookie support (disabled per default)     Help
       IP: Netfilter Configuration
[✓ y] m n  The IPv6 protocol (EXPERIMENTAL)                     Help
y - n  IPv6: routing messages via old netlink                   Help
       IPv6: Netfilter Configuration
y m [✓ n]  Kernel httpd acceleration (EXPERIMENTAL)             Help
y - [✓ n]  Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)      Help
```

Figure 2. Linux Kernel IPv6 Configuration

This is all the configuration you need at the kernel level. Next, you should save this configuration and exit by clicking on the Save and Exit button (see Figure 3). This will create a .config file in /usr/src/linux, which is the kernel configuration file. Now you are ready to compile the kernel by following these steps:

```
make clean
make dep
make bzImage
```

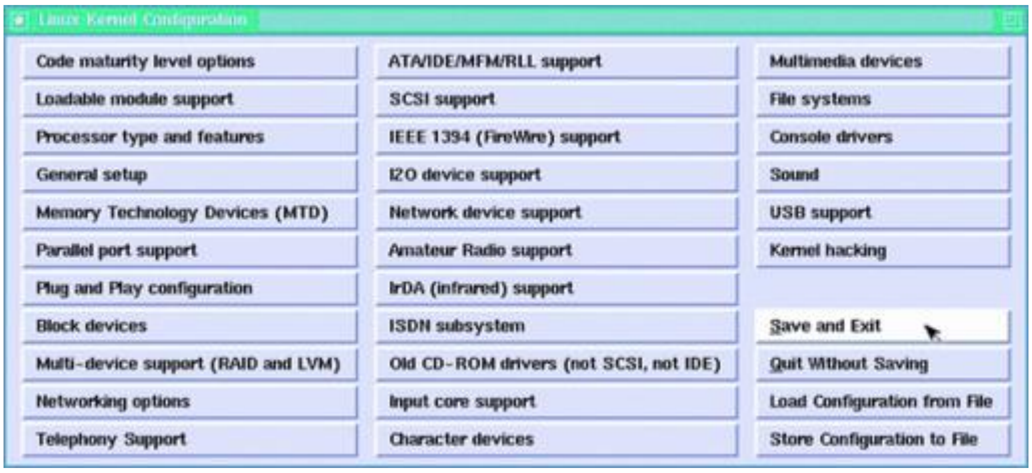


Figure 3. Saving the Configuration

The result will be a new kernel image created in /usr/src/linux/arch/i386/boot/. If you added other features as modules you need to compile and install the modules by applying:

```
make modules
make modules_install
```

At this point you need to copy the new IPv6-enabled boot image to /boot:

```
cp /usr/src/linux/arch/i386/boot/bzImage
    /boot/bzImage.ipv6
```

and update your System.map file:

```
cp /usr/src/linux/System.map \
/boot/System.map-2.4.5-ipv6
ln -fs /boot/System.map-2.4.5-ipv6 /boot/System.map
```

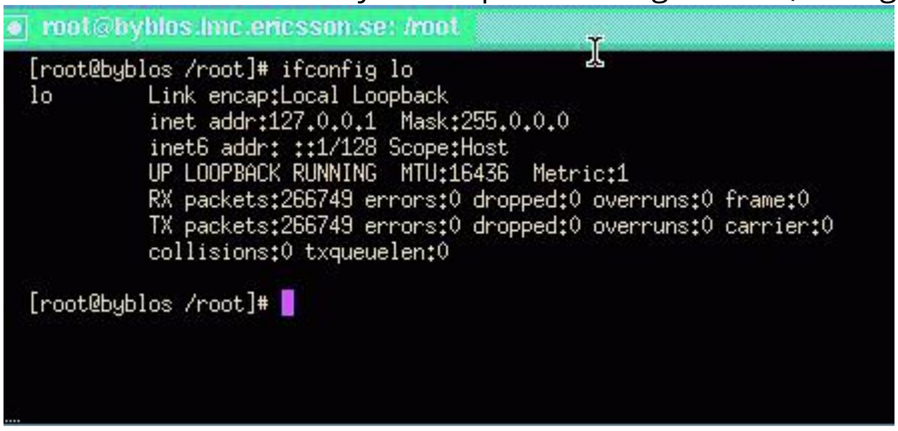
The only remaining step is to update /etc/lilo.conf file to add an entry for the new IPv6-enabled kernel. Edit the /etc/lilo.conf file and add a new entry as follows:

```
image=/boot/bzImage.ipv6
label=linux_ipv6
root=/dev/hda1 # change this to reflect your own
               # partition
read-only
```

Then update the LILO configuration by applying

```
/sbin/lilo
```

This will add an entry called linux_ipv6 that will be presented at LILO at boot time. You are now ready to reboot your server. When LILO comes up, choose to boot linux_ipv6. *Et voilà*! Your system will boot with IPv6 support in the kernel. Type **ifconfig** at the prompt to verify and see the outcome. It should show the IPv6 local address ::1 in your loopback configuration (see Figure 4).



```
[root@byblos /root]# ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:266749 errors:0 dropped:0 overruns:0 frame:0
          TX packets:266749 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

[root@byblos /root]#
```

Figure 4. ifconfig Loopback

IPv6 Binaries and Tools

Once the kernel supports IPv6, you need to install some tools to test your setup and to use IPv6 transport with other systems. First, you need the basic network utilities to support IPv6 to be able to configure your interface, in addition to some other IP utilities such as ping6, telnet6, etc., and to be able to do some basic IPv6 testing.

There are three packages that provide these utilities: net-tools, iputils and the NetKit utilities. The must-have packages are the first two; NetKit is optional. In addition, if you want to have the ability to debug potential IPv6 network problems that may arise, you need to have support for IPv6 in tcpdump and libpcap. In the following sections, we cover the installation of all the above-mentioned packages.

net-tools

net-tools is a collection of programs to control Linux's networking, which includes commands such as arp, hostname (domainname, dnsdomainname, nisdomainname), ifconfig, ipmaddr, iptunnel, netstat, rarp, route and plipconfig. This package is available for download from www.tazenda.demon.co.uk/phil/net-tools.

To install the package on your system first download the packages directly into /usr/src. Then enter these commands separately:

```
cd /usr/src
tar xIvf net-tools-1.60.tar.bz2
cd net-tools-1.60
./configure.sh config.in
```

The configuration will ask a few questions that you need to answer to be able to configure net-tools. It is suggested that you answer yes to the following questions:

```
INET6 (IPv6) protocol family (HAVE_AFINET6) [n]    y
SIT (IPv6-in-IPv4) support (HAVE_HWSIT) [n]        y
Build iptunnel and ipmaddr (HAVE_IP_TOOLS) [n]     y
```

Then, you need to compile and install net-tools:

```
make
make install
```

The binaries will be installed in /sbin and /bin, and you can start using them assuming that you have rebooted into an IPv6-enabled kernel.

iputils

This package includes the following tools: ping, ping6, traceroute6, rdisc, clockdiff, tftpd, tracepath, tracepath6 and arping. It may be that the installed IP utilities on your system do not support IPv6. The first step is to check if the currently installed version is IPv6-compliant by typing the following at the command shell:

```
rpm -q --qf "%{NAME}-%{VERSION}\n" iputils
```

If the outcome is iputils-20000121 or newer, then you do not need to perform the installation of the latest iputils package. Otherwise, you need to follow these steps to install the utilities on your system. First, download the package from ftp.inr.ac.ru/ip-routing. For our setup, we used iputils-ss001110.tar.gz. Next, untar the package in /usr/src:

```
tar -xzf iputils-ss001110.tar.gz
```

Finally, compile iputils with **make**.

The package does not provide a make install. Therefore, you are free to place the binaries in a directory of your choice. However, you need to make sure that the old versions of the tools you have do not conflict with the newer versions in your path. Save the new binaries in /usr/local/iputils/bin/. The package provides man pages for every tool. You also need to move the new man pages into a directory that man searches; to check which directories are in the man path, type **manpath**.

NetKit Utilities

These utilities are basic tools to work with and test your new IPv6 configuration. NetKit includes the following tools: ping, finger, telnet, rwho and their respective dæmons. These are very useful utilities because we can compile them with IPv6 support. The package is available for download from freshmeat.net/projects/netkit. The version we tested was nkit-0.5.1.tar.gz.

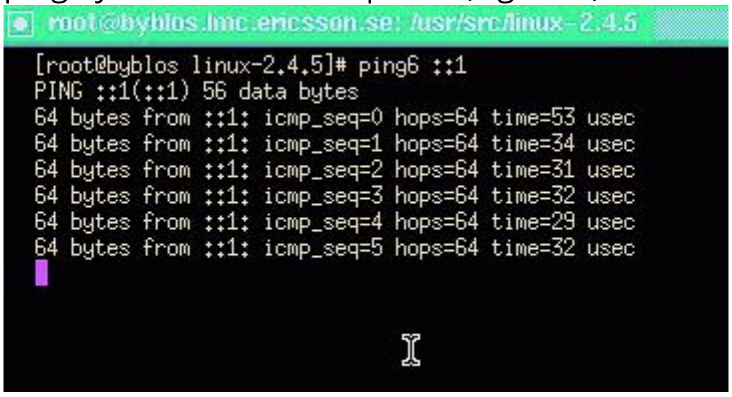
Here are the steps you need to follow to install these utilities on your Linux server. First, download the latest NetKit package from the referenced web site. Then move the downloaded file to /usr/src. Unpack the package with:

```
tar -xzf nkit-0.5.1.tar.gz
```

Next, run **./configure**. Compile with **make clean** and **make**, and copy the binaries to /usr/local/bin:

```
cp telnet/telnet /usr/local/bin/telnet6
cp telnetd/in.telnetd /usr/local/sbin/in.telnetd6
cp finger/finger /usr/local/bin/finger6
cp ping/ping /usr/local/bin/ping6
cp fingerd/in.fingerd /usr/local/sbin/in.fingerd6
```

At this moment, you should have basic functionalities, as you can, for instance, ping6 your local IPv6 loopback (Figure 5).



```
[root@byblos linux-2.4.5]# ping6 ::1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=0 hops=64 time=53 usec
64 bytes from ::1: icmp_seq=1 hops=64 time=34 usec
64 bytes from ::1: icmp_seq=2 hops=64 time=31 usec
64 bytes from ::1: icmp_seq=3 hops=64 time=32 usec
64 bytes from ::1: icmp_seq=4 hops=64 time=29 usec
64 bytes from ::1: icmp_seq=5 hops=64 time=32 usec
```

Figure 5. ping6 in Action

Please note that if you are using Red Hat 7.x, you need to apply a patch to the NetKit package. The patch is available from ftp.bieringer.de/pub/linux/IPv6/netkit.

Optional Utilities

There are several optional utilities that you can install on your system that extend their support for IPv6. For the purpose of this article we mention only three packages: libpcap, tcpdump and xinetd.

libpcap and tcpdump

If you need to understand what is happening at the packet level of your IPv6 network/connection, you need to have IPv6 with libpcap and tcpdump. **libpcap** is a system-independent interface for user-level packet capture that provides a portable framework for low-level network monitoring. On the other hand, tcpdump is a tool that provides network monitoring and data acquisition.

If you want these functionalities, you need to download the latest versions and install them on your system. The versions we tested were tcpdump 3.6.2 and libpcap 0.6.2. First, download the packages from www.tcpdump.org and move them to /usr/src. Then unpack them with:

```
tar -xzf libpcap-0.6.2.tar.gz
tar -xzf tcpdump-3.6.2.tar.gz
```

After unpacking, you will have two directories, one for each package. Next, you need to follow these steps for each package; however, you need to apply them first to libpcap and then to tcpdump. First, run the configuration script while enabling IPv6:

```
./configure --enable-ipv6
```

Then compile with **make clean** and **make**. Lastly, install the binaries with **make install**.

After following these steps, you need to adjust your path to include the new binaries that support IPv6. You also may want to edit /etc/profile and include /usr/local/sbin and /usr/local/bin within your PATH variable, and reload /etc/profile for the new changes to take effect:

```
source /etc/profile
```

xinetd with IPv6 Support

If you want to be able to telnet6 to your system, you need to compile xinetd with inet6 support. Normally, the installed inetd dæmon isn't ready to handle

IPv6 addresses. Therefore, you need to upgrade to xinetd. To download the latest version of xinetd go to synack.net/xinetd. Our setup was tested with xinetd-2.1.8.8p3.tar.gz.

Download xinetd-2.1.8.8p3.tar.gz (or latest) into /usr/src and unpack it with:

```
tar -xzf xinetd-2.1.8.8p3.tar.gz
```

Next, run the configuration script:

```
./configure --with-inet6 --prefix=/usr/local/bin
```

The **--prefix=/usr/local/bin** is used to specify that the resulting binaries should go under /usr/local/bin. Then compile and install:

```
make clean
make
make install
```

Next, you need to create a configuration file from your old inet.conf:

```
/usr/sbin/xconv.pl < /etc/inetd.conf > /etc/xinetd.conf
```

where /usr/sbin is the path to the xinetd executable.

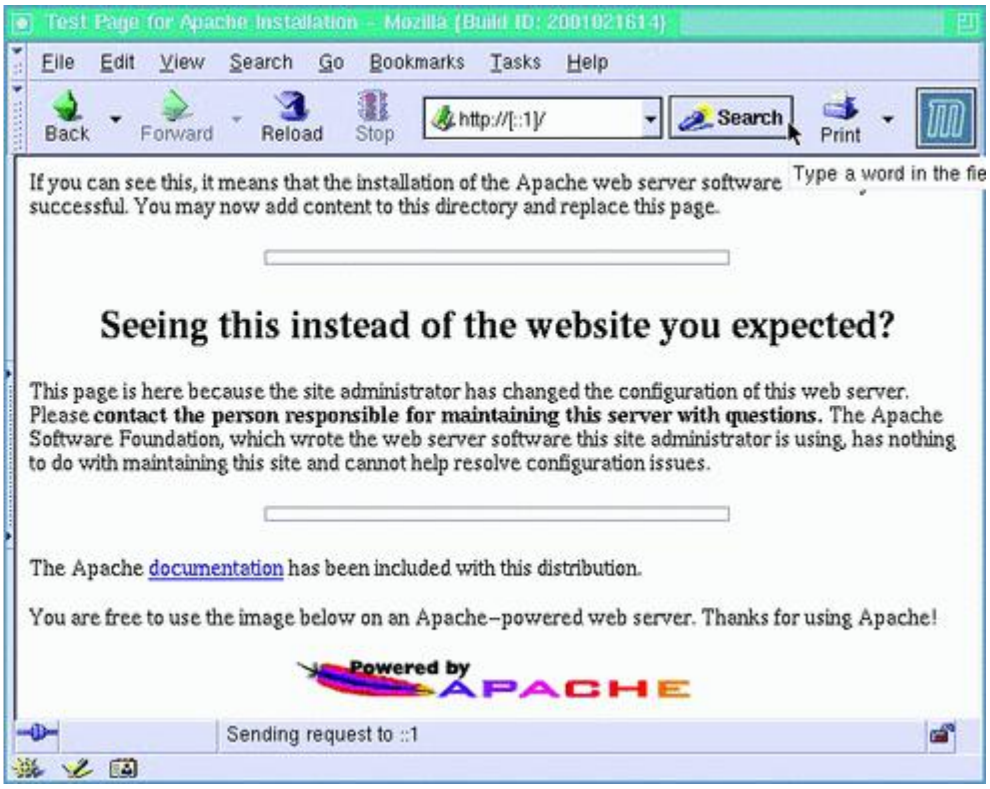
As a side note, you need to make sure that in the xconv.pl script, the first line contains the right path to the Perl binary to be able to execute.

Next, you need do some very minor changes in /etc/xinetd.conf to reflect the usage of the telnet6d and tftp6d, instead of the usual IPv4 Telnet and TFTP dæmons. Having done that, you will be set to Telnet and FTP to your system over IPv6.

IPv6 Applications

There is a wide range of applications that support IPv6. However, we are going to mention only one server application, the Apache web server. Apache is the most popular web server on the Internet (source: Netcraft.com). The latest beta release, Apache 2.0.16 beta, includes support for IPv6, which makes it a good application for testing your IPv6 setup. If you download the latest version of the Apache web server and install it on your system, you will be able to serve web pages over IPv6.

Figure 6 presents a screenshot of the Mozilla browser when trying to access "http://[::1], which is the IPv6 local loopback.


Test Page for Apache Installation - Mozilla {Build ID: 2001021614}
File Edit View Search Go Bookmarks Tasks Help
Back Forward Reload Stop
http://[::1]/
Search
Print
Type a word in the fie

If you can see this, it means that the installation of the Apache web server software successful. You may now add content to this directory and replace this page.

Seeing this instead of the website you expected?

This page is here because the site administrator has changed the configuration of this web server. Please **contact the person responsible for maintaining this server with questions.** The Apache Software Foundation, which wrote the web server software this site administrator is using, has nothing to do with maintaining this site and cannot help resolve configuration issues.

The Apache documentation has been included with this distribution.

You are free to use the image below on an Apache–powered web server. Thanks for using Apache!

Figure 6. A Request for ::1

For your convenience, you may want to update /etc/hosts file to include:

```
::1     ip6-localhost   ip6-localhost
```

Then, instead of using ::1, you can use ip6-localhost.

Don't forget to check the /etc/protocols. If the below-mentioned entries are not there, you need to append them for IPv6-protocol support:

```
ipv6   41 IPv6              # IPv6
ipv6-route 43 IPv6-Route    # Routing Header for IPv6
ipv6-frag  44 IPv6-Frag     # Fragment Header for IPv6
ipv6-crypt 50 IPv6-Crypt    # Encryption Header
                            # for IPv6
ipv6-auth  51 IPv6-Auth     # Authentication Header
                            # for IPv6
ipv6-icmp  58 IPv6-ICMP  icmpv6 icmp6M    # ICMP for
                                          # IPv6
ipv6-nonxt 59 IPv6-NoNxt    # No Next Header for IPv6
ipv6-opts  60 IPv6-Opts     # Destination Options
                            # for IPv6
```

Connecting to the IPv6 Internet

Back in 1996 when the first IETF specifications for IPv6 were done, there was an interest in having a test backbone for IPv6. During IETF-Montréal in 1996, the 6bone (IPv6 backbone) was born. It uses test (but still valid) addresses in the 3ffe::/16 range. At the beginning, most of the backbone was done using tunnels over the current IPv4 Internet. This makes a virtual IPv6 network over the IPv4 Internet. Nowadays, the 6bone is made of both native links and tunnel links.

The 6bone is there for testing, so there is no service-level agreement between the organizations, but this doesn't mean it is not reliable or valid. Any traffic from and to 3ffe::/16 is valid without any limitation.

In July 1999, the three regional registries, ARIN for Americas, RIPE for Europe and Africa and APNIC for Asia, started to give regular nontest addresses to providers, starting in the 2001::/16 range. All the sites that have addresses from that range form the production IPv6 Internet.

To connect to either the 6bone or the IPv6 Internet, you need (as in IPv4) a provider that offers the service. If you can't find one directly, or if your current one does not offer the service, then the easy solution is to make a tunnel to a provider or a site that is willing to offer you the transit service.

As in the early days of the Internet, a project aimed to help people start using IPv6 is offering a free and automated tunnel service that can connect any individual or organization to the IPv6 Internet. The project is called Freenet6.net and is run by Viagénie, a consulting firm, as a free, volunteer and run-on-a-best-effort basis. The service is very popular in the community because of the easy and fast access to the Internet.

Freenet6 is modeled from the tunnel broker (RFC 3053) where an IPv6-over-IPv4 tunnel is established between a node and the tunnel broker. Freenet6 is an enhanced version where the node is using a tunnel setup protocol (TSP) to negotiate the establishment of the tunnel with the server. The client node may be a host or a router. The TSP server Freenet6 provides not only tunnels but also a large address space to any user of the service. The address space provided is a /48, which gives (16 bits) 65,536 subnets, each may have up to 264 nodes (64 bits). This is much more than the entire current Internet! This address space is assigned to the user and will survive over a change in the IPv4 address of the client node. This enables any user or organization to have the freedom of billions of addresses for servers and services; this was not easy to do with NAT in IPv4.

An IPv6-over-IPv4 tunnel is made with both end points configuring the IPv4 and the IPv6 address of the other end point. When one of the end points changes its IPv4 address, then both end points of the tunnel need to change their configuration accordingly. This is especially cumbersome when the IPv4 node is doing dial-up or changing addresses often. TSP, as implemented in the Freenet6 service, can be configured to take care of this. Each time the tunnel client changes its IPv4 address, for example, at boot time with DHCP service, the TSP client sends updated and authenticated information to the server, so the tunnel remains active. Supported client nodes of the Freenet6 service are

Linux, FreeBSD, OpenBSD, NetBSD, Windows, Solaris and Cisco. Figure 7 illustrates the basic architecture of Freenet6.

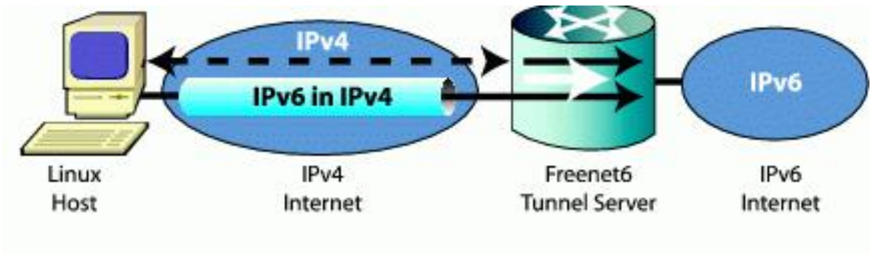


Figure 7. Freenet6 Architecture with One Host

To use the Freenet6 service after installing IPv6 on Linux, you have to take the following steps. First, go to www.freenet6.net and register a user name. Then download the TSP client for Linux. Follow the instructions for compiling and installing it. Next, configure the tspc.conf file provided. Add your user name and password. Then start the tspc client:

```
tspc -vf tspc.conf
```

You may want to put the tspc client command in your boot sequence so that it will automatically re-enable the tunnel at boot time, even if your IPv4 address changes.

Freenet6 can give you either one IPv6 address if you have a host, or it can give you a full /48 if you have a router. Freenet6 will configure Linux to fit the role.

In the router case, you will receive a /48, and the first subnet on your router will be configured for router advertisements. This means that hosts on that subnet will receive the prefix and autoconfigure themselves, as shown in Figure 8.

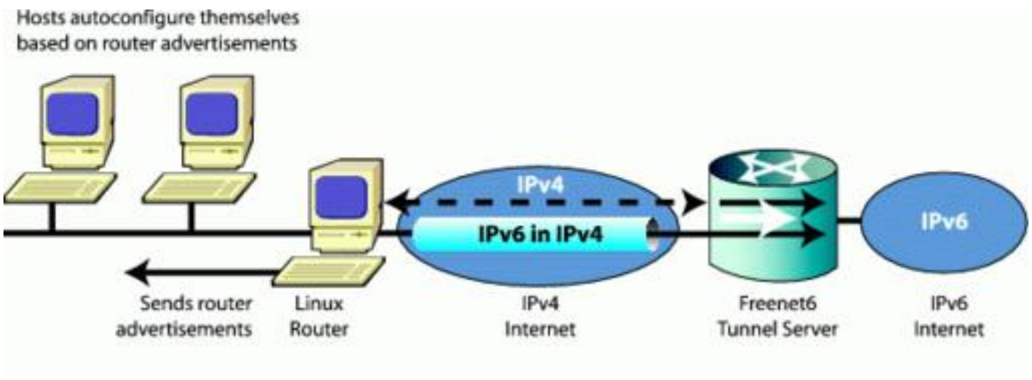

Hosts autoconfigure themselves based on router advertisements
IPv4
IPv6 in IPv4
IPv6
Sends router advertisements
Linux Router
IPv4 Internet
Freenet6 Tunnel Server
IPv6 Internet

Figure 8. Freenet6 Architecture with a Router and Multiple Hosts

There are many ways to connect to the IPv6 Internet. Freenet6, together with the TSP protocol, enables an easy IPv6-tunneled connection with a permanent address space so that if you change your IPv4 address, the IPv6 addresses and connection remains stable.

Conclusion

As part of our activities in the Open Architecture Research at Ericsson Research Canada, we are conducting several IPv6-related projects, such as supporting IPv6 on our telecom-grade server nodes, porting application servers to work with IPv6 and establishing research projects in different IPv6 areas.

One of the interesting activities we carried out was to experiment with the Linux IPv6 implementations currently available and present recommendations to decide which implementation to adopt for our Linux processors. The recommendations were based on IPv6-implementation characteristics, such as its development speed, its compliance to the standards and its performance vs. other implementations. The results were presented during the Linux Symposium in Ottawa, Canada, June 26-29 (www.linuxsymposium.org/2002/view_txt.php?text=abstract&talk=93).

For 2002, we continue to support IPv6 on our Linux clusters. Our current efforts are in the direction of supporting IPv6 on the SIP and SCTP implementations for Linux, as well as building an IPv6 benchmarking environment capable of testing the performance and scalability of platforms and applications running over IPv6.

In closing, it's important to know that one of the key advantages of IPv6 is addresses. Any individual can receive as many addresses as the current IPv4 address space. This empowers users with complete freedom to deploy servers and services without having to take care of NAT problems and limited address space. Welcome to the freedom of deploying services on the Internet!

Acknowledgements

We'd like to thank Ericsson Open Architecture Research for approving the publication of this article, Canarie, Inc. (www.canarie.ca) for funding part of the Freenet6 Project and David Gordon (David.Gordon@Ericsson.ca) for his help in the ECUR Lab setting up IPv6 nodes and editing and reviewing this article.

Resources

Ibrahim Haddad (Ibrahim.Haddad@Ericsson.com) currently is a researcher at the Ericsson Corporate Research Unit in Montréal, Canada. He is primarily involved in researching carrier-class server nodes for real-time, all-IP networks and represents Ericsson on the Technical Groups of the Open Source Development Lab. **Marc Blanchet** works at Viagénie, a consulting firm specialized in IP engineering, IPv6 and network security. He has been involved in IPv6 since 1995 and has written many IETF documents on IPv6. He also wrote Migrating to IPv6, published by Wiley.

Bare Metal Recovery, Revisited

Charles Curley

Issue #100, August 2002

Charles describes the additions he made to the scripts for his backup and recovery suite.

Imagine your disk drive has just become a very expensive hockey puck. Imagine you have had a fire, and your computer case now looks like something Salvador Dali would like to paint. Now what?

That's the way I started an article on this subject in the November 2000 issue of *Linux Journal*. The article described a process for backing up a computer and subsequently restoring it to the bare metal. The article described a suite of scripts that were part of both the backup process and the recovery process. Readers can find the article at www.linuxjournal.com/article/4175.

Since then I have added some scripts to the suite. Most of the new scripts are designed for network backups and take advantage of Secure Shell (SSH). (For more information on SSH, see Mick Bauer's “The 101 Uses of OpenSSH” in the January and February 2001 issues of *LJ*.) I've also made some changes to the scripts introduced in the original article. The suite of revised scripts is available at my home page (see Resources).

The Flaw

The biggest problem with my November 2000 article and the process it described is that the process required a lot of typing at the beginning of the recovery process. You have to enter partition boundaries and other data into fdisk manually, then check the results against your printout. (Printout!? for Murphy's sake!) Then you manually create the appropriate filesystems for each partition. Then you get to mount them, again manually.

This is a lot of typing. I don't know how many times I did test backups and restores on my test computer while I was writing the article. More than I ever want to do again, that's for sure. It's also error prone. After a while all those numbers start to blur together.

The obvious solution is a script or two. What we need is a script that will restore the partition information to a hard drive, then build the filesystems and mount them so that you can run the first stage restoration.

My first pass at this script is the script make.partitions, which is available in the tarball of scripts on my home page. It has two problems: first, it does not rebuild the partitions, so you still have to run fdisk manually; and second, it has to be created by hand for each computer. Add, delete, reformat or otherwise modify a partition, and you have to edit the script. That's not good enough. The script, which is GPLed, should look somewhat like Listing 1.

Listing 1. make.dev.hda

A Script-Writing Script

The second solution is a lot smarter. Why not automate the process? We use gcc to compile gcc. Heck, you can use gcc to compile Perl. Why not a script that creates the script that make.partitions should be? Why not a script-writing script?

make.fdisk parses the output from fdisk -l and mount -l and creates a new script for restoring a given hard drive.

Using Redirection

The first problem we face is one I mentioned in the original article: fdisk does not export partition information in a manner that allows it to be re-imported later on. While other versions of fdisk do allow exporting, tomsrtbt (the floppy-based distribution I recommend for bare metal restore) comes with fdisk, and I don't want to rebuild the tomsrtbt disk. We can handle this with something all well-behaved Linux programs have: I/O redirection. Given a program, foo and a file of commands for foo called bar, we can feed the commands to foo by redirecting foo's input from the keyboard to bar, like this:

```
foo < bar
```

So what we want to be able to do is:

```
fdisk /dev/x < dev.x
```

where *x* is the name of the hard drive to be rebuilt.

make.fdisk creates two files. One is an executable script, called make.dev.*x*, like Listing 1. The other, dev.*x*, contains the commands necessary for fdisk to build the partitions. You specify which hard drive you want to build scripts for (and so the filenames) by naming the associated device file as the argument to make.fdisk. For example, on a typical IDE system,

```
make.fdisk /dev/hda
```

spits out the make.dev.hda script and the input file for fdisk, dev.hda.

How It Works

As you look at the script make.fdisk shown in Listing 2 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/5484.tgz], keep in mind what happens at what time. Like C source code, some things happen later on, at runtime. Others happen at the time the program is compiled, like evaluation of defines and inclusion of header files.

On examining make.fdisk, the first thing we see is that it is a Perl script. Next is a brief description of what the script does. This is followed by a timestamp and two copyright statements. Then we see the usual announcement that the code is free software and distributed under the General Public License. Next is a detailed description of the problem with fdisk we've already seen—and the solution. It is good coding practice to document a program in this manner; it makes the program almost self-documenting.

Now we get to actual Perl code. The subroutine cut2fmt takes a series of column numbers and calculates a format string for later use with unpack. Right after the subroutine we use it to create a format string to unpack the output from fdisk.

After that is a series of definitions of the columns in fdisk's output. With these, we can index into the array created with unpack by name rather than by column number. This should make the script easier to read and more maintainable.

The directory where the rebuilt hard drive will be mounted is named $target so that the first stage restore can find it. Make sure this agrees with the definition of $target in your copy of the script restore.metadata.

Next, the code massages the device name to produce the filenames where we will send our output. Then we define the path to the directory where we will place the output files.

Disk Labels

Labels are tools that Linux uses to abstract partitions. The problem with using device filenames in fstab is that if you add or remove a hard drive you may affect which device file another partition shows up under. Labels travel with the partition, so that with mounting by label you always get the correct partition. They are a problem for us because tomsrtbt doesn't handle labels.

The next section of code executes mount with a command-line switch to make it show the labels. If there is a label in any given line, we save the label and the device filename in a hash. That way, later on when we make the filesystem in the partition, we can assign the label. Also, we need to mount the partition by a device filename so that we can restore to it. We make a hash mapping from device filename to mountpoint so that later on we can build the mountpoint directories and mount the partitions.

Next is a typical Perl command to spawn a process and put the results into a filehandle, in this case FDISK. It is complete with error checking. Then we open our output file, which will eventually be redirected as input to fdisk.

Now we begin a loop to parse each line of the output from the system call to fdisk. We are interested in any line that has the device in it. If we find one, we massage it a bit, unpack it into the array @_ and further massage the array members.

Disk Partitions

If a partition number is less than five, it is either a primary partition, meaning it can have a filesystem in it, or an extended partition, meaning it can have a number of logical partitions in it. In either case, we write the commands to build the partition to the output file. If it is a Linux swap partition, we have to tell fdisk to change its partition type.

If we see a primary partition that is either FAT (but, for now, not FAT32), Linux or Linux swap, we append the appropriate command to the $format to make the partition a FAT, ext2 filesystem or a swap partition. Later on, we'll use $format to create the output script.

A partition number of five or greater only can be a logical partition, that is, one contained within an extended partition. As far as we are concerned, these are either Linux ext3fs, Linux swap partitions, FAT or anything else. As above, appropriate fdisk commands are sent to the output file and appropriate commands to create filesystems are appended to $format.

We look to see if there is a label for each ext2 partition. If there is, we use a command that will recreate that label on the new partition, otherwise we use the same command without a label.

Bad-Block Checking

You will notice that there are two commands to make each filesystem, with one commented out. The one commented out makes the filesystem with no bad-block checking. If I were installing to a brand-new hard drive, I would consider using this. The other does bad-block checking. I prefer to check for bad blocks when reusing a hard drive. The bad-block check is a simple read-only test, which is reasonable most of the time. You can add a write test, which is much more thorough but takes longer, by adding -w to the command-line options for bad blocks. The write test is destructive, but since you will be building a new filesystem in the partition, you don't care.

At the end of our line-parsing loop, if any partition is marked “bootable” (typically a MS-DOS, Windows or Windows NT partition because LILO ignores the bootable flag), we send the commands to make it bootable to the command file.

The last thing we do for the command file is send a “v”, which will have fdisk verify the newly built partition table. Then we send a “w”, which will cause fdisk to write the partition table to the hard drive and then exit. We then close our two files.

Next, we open the file that will become our script and send an appropriate header to the script, similar to the header for this script. The first thing the script actually will do is use dd to write zeros over the first 1,024 bytes of the hard drive. This will clobber any existing master boot record (MBR) so that we don't have to worry about deleting partitions before creating the new ones.

The next step is to create the command that will partition the hard drive, using the command file we've already created. Then the code walks through the hash of mountpoints, creating a comment line, a command to create the directory and then a command to mount the device filename to the directory.

We have to mount starting at the root partition so that mountpoints are created in the correct partition. For example, suppose /usr/local is on its own partition; we have to mount /usr before we build /usr/local. To ensure that is done, we sort the keys of the hash and process the hash in that order.

The last thing we do is change the mode of the files we've just created. Since paranoids live longer, we disallow anyone but root from even reading the script, and make it executable.

Using the Script

The script make.fdisk should be run as a normal part of preparing for backing up for bare metal recovery. Run it before you run **save.metadata** so that the output files are saved to the ZIP drive. Better yet, have save.metadata call it, once for each hard drive.

When you are restoring, run **make.dev.*x*** for each hard drive you have. Again, this can be automated by including it in restore.metadata.

There are other things you can do with this script. Suppose you want to add a new partition. Use the bare metal backup process to save a hard drive, then edit the dev.x command file to change the partition definitions and restore using the edited file. I successfully added a 30MB Mess-DOS partition to my test computer with this technique.

Improvements

Some improvements that you can tackle if you like include having make.fdisk process several hard drives, all indicated on the command line; adding error checking for the argument(s) to make.fdisk, having it produce one script that builds all the hard drives, extending the FAT filesystem support (for one thing, right now the code ignores FAT32); and extending the code to support other filesystems.

Resources

Charles Curley (w3.trib.com/~ccurley) is a freelance software engineer, writer and occasional cowpoke in the wilds of Wyoming. Occasionally, while he's in his backyard working on an article, some deer wander through and he loses his train of thought.

Archive Index Issue Table of Contents

Advanced search

The Linux Router

Kaleem Anwar

Muhammad Amir

Ahmad Saeed

Muhammad Imran

Issue #100, August 2002

The performance of the Linux router makes it an attractive alternative when concerned with economizing.

Routers are amongst the most crucial components of the Internet, as each bit of information on the Internet passes through many routers. Most of the routers used on the Internet are made by Cisco. Although these have good performance, they come at a high price.

In situations where we need to economize, the Linux router is an attractive alternative. When used as a simple gateway for a LAN, it can be almost free. All that is needed is an old 486DX machine with more than one network interface. A monitor is not always necessary. If used for a sophisticated application you will need a Pentium PI 200MHz MMX, which is costs more but is still three or four times cheaper than a commercial router with comparable functionality.

If one has a small lab with several LANs and wishes to set up a reliable, as well as secure, connection to the Internet, the cost of a commercial router may not be justifiable. The most economical solution in this case is to use a low-cost processor running the LRP (Linux Router Project, www.linuxrouter.org) distribution, which is a networking-centric, micro-distribution of Linux.

LRP is so small that it can safely boot from a single 1.44MB floppy disk. It makes the building and maintenance of firewall, routers, switches, hubs, and so on, cheap and straightforward.

In this article we show how to set up a Linux router for two to four LANs and test its performance under different conditions. All of the work described here was done on Intel PIIIs running at 733MHz. For comparison we also used Pentium Is and IIs. Here we present the results of our investigation into the performance of the Linux router and compare it with a commercial router.

Setting Up a Linux Router

The most common function of the Linux router is a connection between two networks. Typically, this would be a LAN and the Internet. For our experiments, given the unavailability of a connection to the Internet fast enough to stress the router sufficiently, we used a server to simulate the Internet.

For performance measurements, we set up a simple router configuration as follows:

- Download a copy of the idiot image (lrp 2.9.8). See the Sidebar “Which Disk Image to Use” for details.
- Extract the image to a floppy disk (1.44MB/1.68MB super-formatted) and make it bootable. The best way to do so is to use WinImage (www.winimage.com).
- Get the kernel module for the Ethernet card you are using. We used RealTek Ethernet cards with the RTL8139 chipset, so the module we used was rtl8139.o. Add this to your kernel. Your Linux router is now ready for its configuration. See the Sidebar “Adding Kernel Modules for Ethernet Cards” for details.
- Boot from your LRP disk and open the network.conf file (located in /etc/network.conf). Now modify it so that it looks like Listing 1 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/5826.tgz]. Appropriate comments are there for modifications.
- Save the changes and back them up. Reboot.

The configuration of the Linux router is now complete. Now we'll describe its performance in different configurations. Because we are not using dynamic routing, we will define static routes in the following experiments according to the configuration of the experiment. Note: after you are done configuring the Linux router, write-protect the floppy disk you are using.

Performance of Linux Router

The test setup in our computer lab uses a 100Base-T Ethernet. The NICs and switching hubs are 100Base-T. All platforms are running Linux 2.2 kernels, and the Linux router is the default gateway for all of them. Performance is measured on different LRP boxes, such as PI 133MHz, PI 200MHz and PIII 733MHz.

Bandwidth Measurement

The first configuration uses one client and one server. We connected the server at the first NIC on the LRP box (eth0) and the client at the second NIC (eth1) through cross-UTP 100Mb cables. Then we set the ipchains rules on the Linux router for forwarding the traffic between client and server by issuing the following command:

```
ipchains -I forward -j ACCEPT -s 192.168.1.0/24
-d  192.168.0.0/24 -b
```

We measured the bandwidth of the Linux router when there was traffic flow between the server and the client. See Table 1 for the measurements for the different LRP boxes.

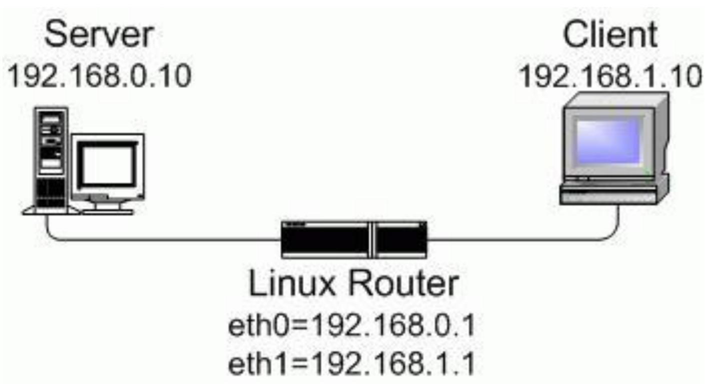


Figure 1. Setup Number One for Measurement of Bandwidth of LRP Box

Table 1. Bandwidth Measurement Results

In this case, the measurements for the Pentium I are misleading, as the bottleneck is the 90Mbps practical limit of 100Base-T Ethernet and not the capacity of the router.

Configuration two was done with one server and multiple clients. We connected a server on the first NIC of the LRP box (eth0) and three LANs through different hubs to the other three NICs respectively. The setup is depicted in Figure 2. The ipchains rules for this setup would look like:

```
ipchains -I forward -j ACCEPT -s 192.168.0.0/24
-d  192.168.1.0/24 -b
ipchains -I forward -j ACCEPT -s 192.168.0.0/24
-d  192.168.2.0/24 -b
ipchains -I forward -j ACCEPT -s 192.168.0.0/24
-d  192.168.3.0/24 -b
ipchains -I forward -j ACCEPT -s 192.168.1.0/24
-d  192.168.2.0/24 -b
ipchains -I forward -j ACCEPT -s 192.168.1.0/24
-d  192.168.3.0/24 -b
```

```
   ipchains -I forward -j ACCEPT -s 192.168.2.0/24
   -d  192.168.3.0/24 -b
```

You can write a script to run these rules eliminating the need to enter them at the command prompt every time you boot your LRP box. It should be placed in the root directory so that the user is able to run all the rules by just entering ***./filename***. We measured the bandwidth of the router when there was traffic between the server and more than one client (clients may be from the same or different LANs).

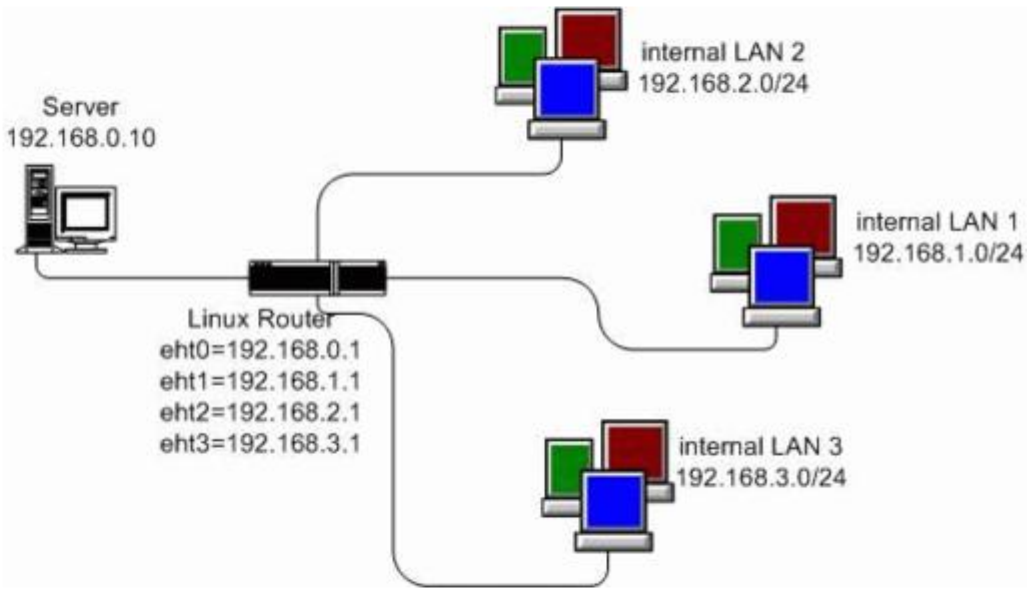


Figure 2. Setup Number Two for Measurement of Bandwidth of LRP Box

The graph in Figure 3 shows the performance of LRP while routing the traffic between the server and the clients. From this graph we conclude that a PI 133MHz-based Linux router is sustaining a bandwidth of about 51Mbps, and a PI 200MHz-based Linux router is sustaining a bandwidth of about 82Mbps. The measured bandwidth between two platforms that are on the same network segment (say both are at internal LAN1) was found to be equal to 90Mbps. In this case, the router is not involved in the communication. This is direct communication between two computers on 100Base-T Ethernet, start topology, so Ethernet has a practical limit of 90Mbps. The bandwidth of the PIII-based Linux router cannot be calculated due to the limitation of the physical medium of transmission.

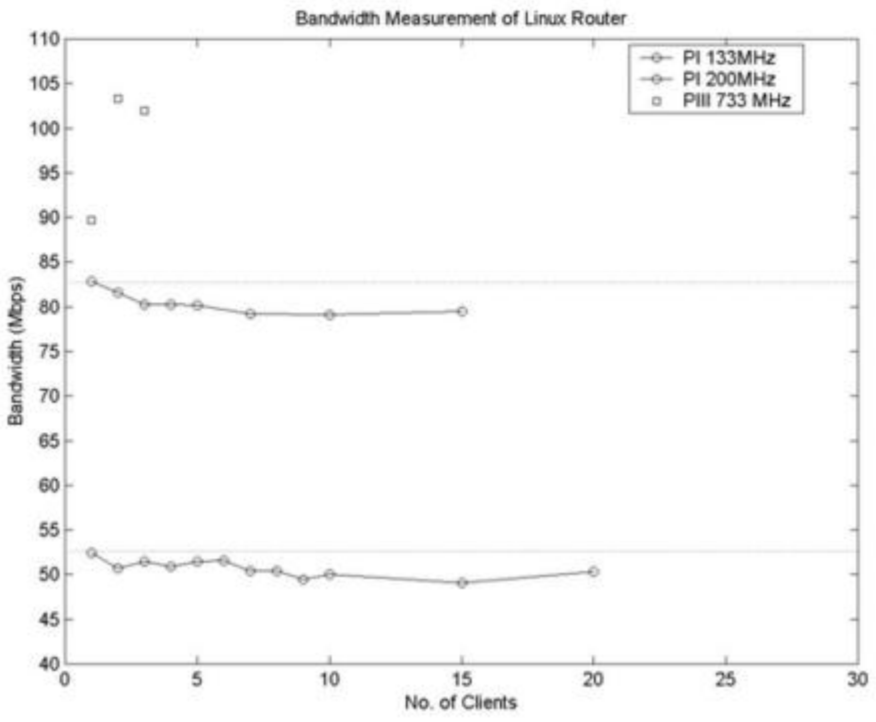


Figure 3. Bandwidth Measurement of Linux Router

For the third configuration we set up multiple servers and multiple clients (cross-pinging). In this test setup we used two servers connected on eth0 and eth2 of the LRP box.

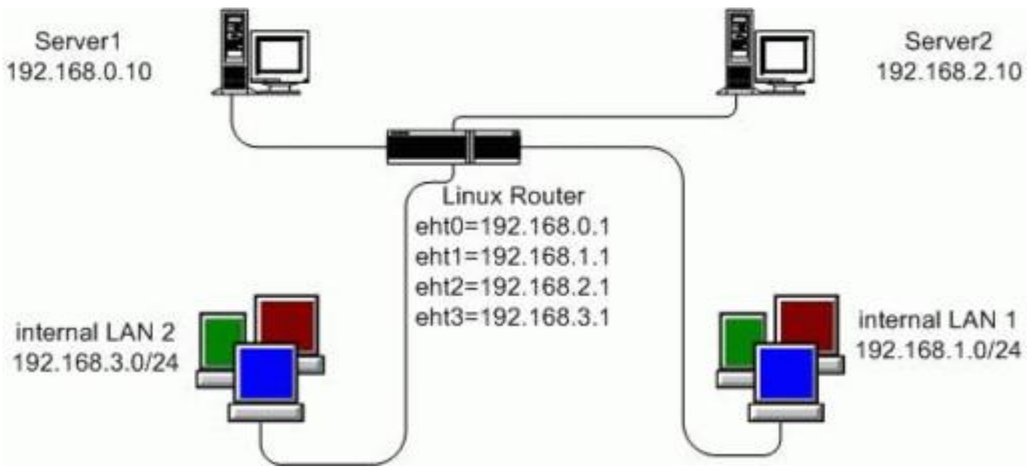


Figure 4. Setup for Cross-Pinging

A slight reduction (1-2% only) in the bandwidth of the Linux router was observed when there was cross-pinging of packets between server 1 and client 1 and server 2 and client 2, simultaneously.

Stability

The Linux router is very stable in its operation. We have run it for long periods, and it showed a very stable performance over the entire length of time. The graphs in Figures 5 and 6 show that the bandwidth of the Linux router is fairly constant with a great increase in the amount of data.

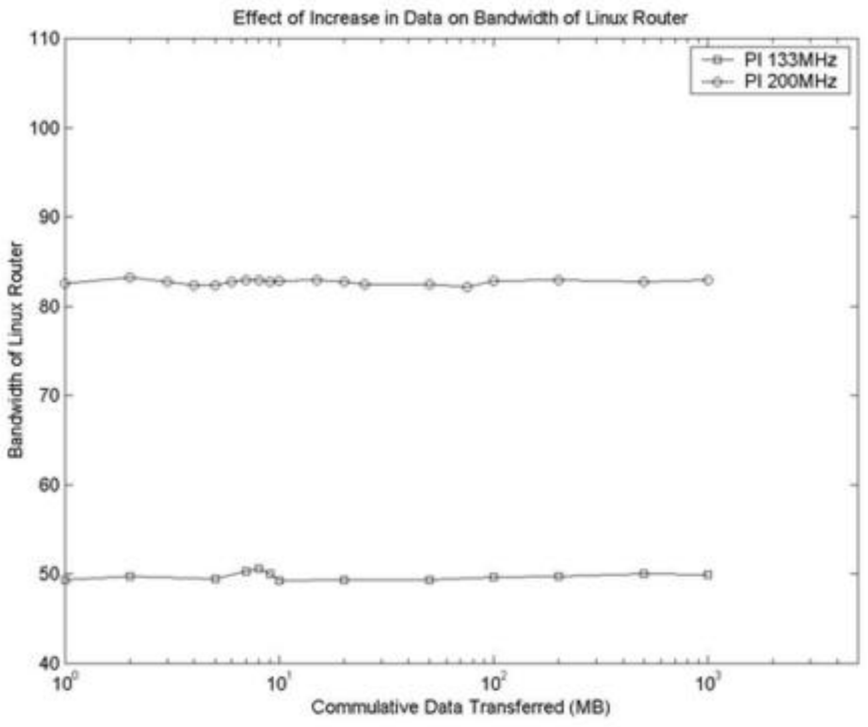


Figure 5. Effect of Increase in Data on Bandwidth of Linux Router

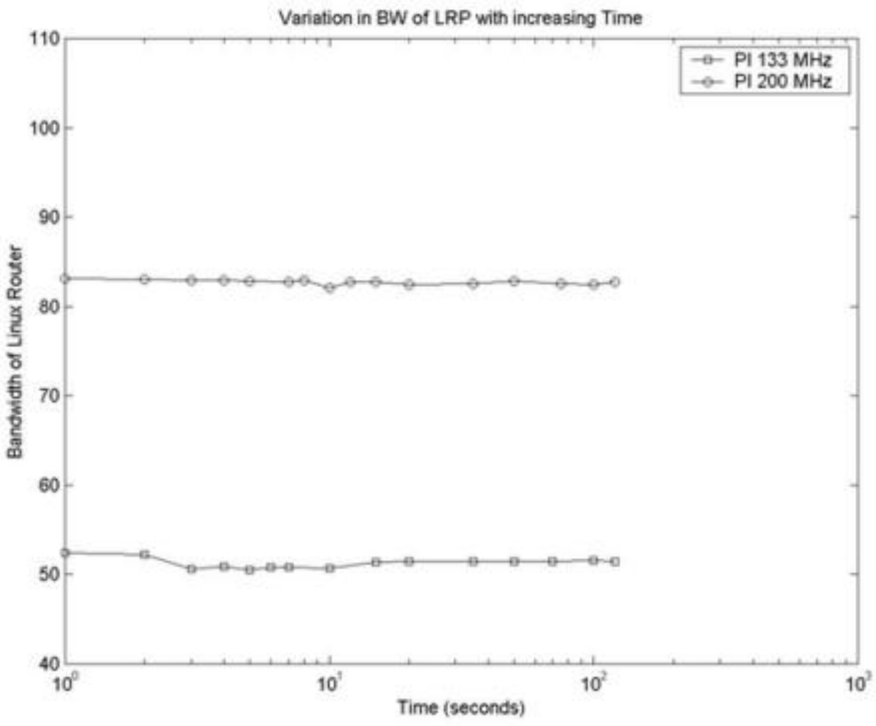


Figure 6. Variation in Bandwidth of LRP with Increasing Time

The write-protected medium for booting off the Linux router gives it increased security from crackers. Once booted, it runs exclusively off RAM. You may safely take your floppy out of the floppy drive and put it in a secure place until it's needed again. Also, a single floppy can be used to boot many identical Linux routers with a runtime change in configuration.

Easy to Handle

The Linux router is easy to handle and configure. It does not require any special care for its use other than that required for a normal PC. If there is a problem, configuring it only takes a few minutes. Moreover, it is basically software on a floppy disk; if your LRP box gets damaged because of power fluctuations (a common problem in the third world), you can instantly convert another available PC into your router by adding NICs from the corrupted LRP (if they are not corrupted) and boot it off the floppy disk. No configuration will be required for this router at all, except the runtime configuration. You can imagine what a great advantage this is—think of what would happen if your Cisco router were to be corrupted.

Comparison with a Commercial Router

The following is a comparison of the Linux router with the Cisco 2620 router available in our laboratory.

The cost of building a good Linux router (based on a Pentium I, 200MHz MMX) with 1FDD, 32MB of RAM is less than $100 US. (It may be nearly free if you use the minimum required hardware, i.e., a 486DX with 16MB RAM.) A monitor is not necessarily required. You can use a borrowed monitor temporarily at configuration time or configure via a remote serial connection (if you include support for that through the serial.lrp package). On the other hand, the cost of the Cisco 2620 with a 50MHz Motorola Processor, 16MB Flash RAM and 40MB DRAM is more than $3,500 US.

Although power consumption here is not of great concern, in most applications it is notable that the Linux router (running on PI 200MHz, MMX) consumes less than 30W of power, while Cisco 2600 series routers consume 75W.

You can add as many NICs in the Linux router as you wish (limited by the number of slots on the main board). In Cisco 2600 there is only one Fast Ethernet card available.

The modularity of the Linux router is matchless. Its packaging system allows easy removal and addition of features. You can add/remove packages, even at runtime, using the **lrpkg** command. You need to shut down the Linux router to add a module only if it requires some additional hardware. However, the kernel module for the hardware can be installed at runtime using **insmod**. The design of the Cisco router is not as modular.

For the Linux router there are a large variety of hardware and software products available in the open market as it has the complete structure of the ordinary Linux operating system. You can use the product of any manufacturer that has support for the Linux router. Cisco routers, on the other hand, are limited in this respect. Usually only Cisco products are used with Cisco routers.

Having Linux as the operating system on your router gives you the extra advantage that you can build your own packages according to your needs using shell scripting. You also can get a lot of help from the available literature for Linux. Cisco routers have their own specific operating system called Internet Operating System. The Cisco 2620 uses IOS release 12.1. Although it is developed on a regular basis, you can use only those features that are available in the specific IOS release used on your specific router.

Like Cisco routers, the Linux router also supports the multiprotocol feature. It has support for RIP, BGP, OSPF and many more that are added through packages.

Services such as Ethernet router, firewall, DNS and ISDN may be initialized on a Linux router. However, initializing services like DNS (which is highly CPU-bound) will degrade its performance. It is better to use a separate machine as a DNS server. The Cisco router has multiservice integration of voice, data and video. As with Cisco routers, IP masquerading, port translation, load balancing, transparent proxy and interface alias may all be implemented on a Linux router.

Cisco routers support IPX, Token Ring, VLAN, VPN, Apple Talk and DDR for advance routing. The Linux router also can support these features through proper packages. Although to do so, some expertise in Linux and some additional hardware are required, which will increase the cost of Linux router, but it still will be much less than that of a Cisco router.

Depending upon the model and series of the Cisco router, it has a limited number of WAN slots. In the 2620 there are two WIC (WAN Interface Cards) slots, one network module and one advance integrated mode slot. The two-port serial WAN card has a asynchronous speed of 115.2Kbps, and synchronous speed equals 2.048Mbps. Port 1 supports only synchronous mode. The Linux router also has support for WAN interface cards. Sangoma WICs (www.sangoma.com), which have a synchronous data rate of 8Mbps, are quite popular among LRP users. With these cards you can combine many LRP boxes. However, the disadvantage is that the cost of the LRP box increases—this card costs about $400 US.

Conclusion

The bandwidth of a 133MHz Pentium I-based Linux router is about 51Mbps and that of a 200MHz Pentium I-based Linux router is 82.5Mbps. The performance of the Linux router on a 733MHz PIII is so high (90Mbps) that it saturates the 100MHz Ethernet. We also studied the effect of RAM on routing. In this case it turned out that there is no effect on routing performance with an increase in RAM. However, by increasing RAM you can set up larger RAM drives that you may need if your routing table gets quite large.

We have explored the performance of Linux router, its stability, cost, highly modular design, low power consumption, and so on. More work on the Linux router is underway to improve its routing performance. For a small office or laboratory, where the pursuit of cost-savings is a major consideration, the Linux router is the ideal solution. A typical configuration for a small business would be as shown in Figure 7.

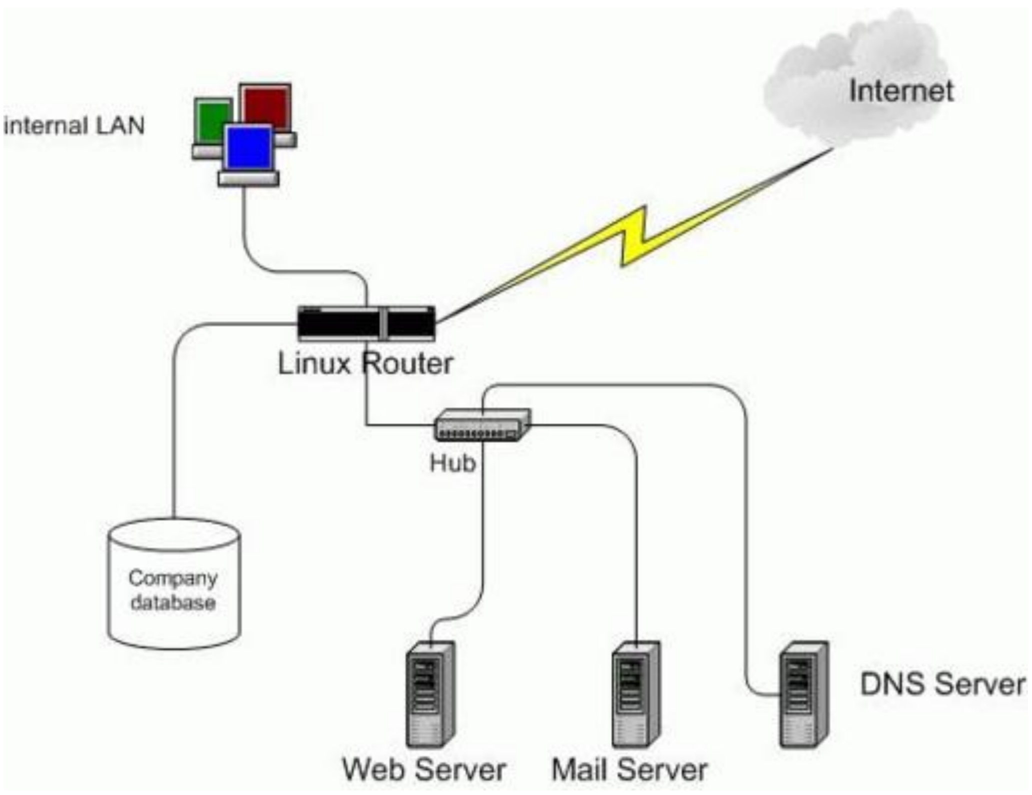


Figure 7. Typical Configuration for a Small Business

Acknowledgements

This work was performed in the Computer Communications Laboratory and the Digital Computers Laboratory at the Department of Electrical Engineering, University of Engineering & Technology, Lahore, Pakistan, under the supervision of Professor Shahid H. Bokhari.

Resources

Kaleem Anwar (kaleem_002@yahoo.com) is graduating in Electrical Engineering with a specialization in Computer Engineering from the Department of Electrical Engineering, University of Engineering and Technology Lahore, Pakistan. His fields of interest include Linux, Java, control systems, computer networks, algorithm design and digital signal processing.

Muhammad Amir (amirsher03@hotmail.com) is graduating in Electrical Engineering with a specialization in Computer Engineering from the Department of Electrical Engineering, University of Engineering and Technology Lahore, Pakistan. His fields of interest include Linux, Java, control systems, computer networks, algorithm design and digital signal processing.

Ahmad Saeed (electri17@yahoo.com) is graduating in Electrical Engineering with a specialization in Computer Engineering from the Department of Electrical Engineering, University of Engineering and Technology Lahore, Pakistan. His fields of interest include Linux, Java, control systems, computer networks, algorithm design and digital signal processing.

Muhammad Imran (imran_uet@hotmail.com) is graduating in Electrical Engineering with a specialization in Computer Engineering from the Department of Electrical Engineering, University of Engineering and Technology Lahore, Pakistan. His fields of interest include Linux, Java, control systems, computer networks, algorithm design and digital signal processing.

Archive Index Issue Table of Contents

Advanced search

The Beowulf Evolution

Glen Otero Ferri

Issue #100, August 2002

Second-generation Beowulf clusters offer single-process I/O space, thin slave nodes, GUI utilities and more for adaptability and manageability.

Imagine, for a moment, if you will, driving your car into a full-service gas station —a near anachronism—pulling up to the attendant and saying, “Fill'er up, check the oil and wipers, and...give me 20 more horsepower, would you?” The attendant, not phased by the request, looks at you and says, “Would you like four-wheel drive with that? I hear it might snow tonight.” You think for a moment and respond positively—four-wheel drive would be good to have.

If only automobiles, and Beowulf clusters, were so adaptable. Yet, the single most important distinguishing feature of Beowulf 2 technology is adaptability—the ability to add more computing power to meet changing needs. To understand and appreciate how Beowulf technology has become so adaptable, an understanding of Beowulf 1 is in order.

The Roots of Beowulf

As we all know by now, the original concept for Beowulf clusters was conceived by Donald Becker while he was at NASA Goddard in 1994. The premise was that commodity computing parts could be used, in parallel, to produce an order of magnitude leap in computing price/performance for a certain class of problems. The proof of concept was the first Beowulf cluster, Wiglaf, which was operational in late 1994. Wiglaf was a 16-processor system with 66MHz Intel 80486 processors that were later replaced with 100MHz DX4s, achieving a sustained performance of 74Mflops/s (74 million floating-point operations per second). Three years later, Becker and the CESDIS (Center of Excellence in Space Data and Information Services) team won the prestigious Gordon Bell award. The award was given for a cluster of Pentium Pros that were assembled for SC'96 (the 1996 SuperComputing Conference) that achieved 2.1Gflops/s (2.1

billion floating-point operations per second). The software developed at Goddard was in wide use by then at many national labs and universities.

First-Generation Beowulf

The first generation of Beowulf clusters had the following characteristics: commodity hardware, open-source operating systems such as Linux or FreeBSD and dedicated compute nodes residing on a private network. In addition, all of the nodes possessed a full operating system installation, and there was individual process space on each node.

These first-generation Beowulfs ran software to support a message-passing interface, either PVM (parallel virtual machine) or MPI (message-passing interface). Message-passing typically is how slave nodes in a high-performance computing (HPC) cluster environment exchange information.

Some common problems plagued the first-generation Beowulf clusters, largely because the system management tools to control the new clusters did not scale well because they were more platform- or operating-specific than the parallel programming software. After all, Beowulf is all about running high-performance parallel jobs, and far less attention went into writing robust, portable system administration code. The following types of problems hampered early Beowulfs:

- Early Beowulfs were difficult to install. There was either the labor-intensive, install-each-node-manually method, which was error-prone and subject to typos, or the more sophisticated install-all-the-nodes-over-the-network method using PXE/TFTP/NFS/DHCP—clearly getting all one's acronyms properly configured and running all at once is a feat in itself.
- Once installed, Beowulfs were hard to manage. If you think about a semi-large cluster with dozens or hundreds of nodes, what happens when the new Linux kernel comes out, like the 2.4 kernel optimized for SMP? To run a new kernel on a slave node, you have to install the kernel in the proper space and tell LILO (or your favorite boot loader) all about it, dozens or hundreds of times. To facilitate node updates the r commands, such as rsh and rcp, were employed. The r commands, however, require user account management accessibility on the slave nodes and open a plethora of security holes.
- It was hard to adapt the cluster: adding new computing power in the form of more slave nodes required fervent prayers to the Norse gods. To add a node, you had to install the operating system, update all the configuration files (a lot of twisty little files, all alike), update the user space on the nodes and, of course, all the HPC code that had configuration requirements of its own—you do want PBS to know about the new node, don't you?.

- It didn't look and feel like a computer; it felt like a lot of little independent nodes off doing their own thing, sometimes playing together nicely long enough to complete a parallel programming job.

In short, for all the progress made in harnessing the power of commodity hardware, there was still much work to be done in making Beowulf 1 an industrial-strength computing appliance. Over the last year or so, the Rocks and OSCAR clustering software distributions have developed into the epitome of Beowulf 1 implementations [see “The Beowulf State of Mind”, *LJ* May 2002, and “The OSCAR Revolution”, *LJ* June 2002]. But if Beowulf commodity computing was to become more sophisticated and simpler to use, it was going to require extreme Linux engineering. Enter Beowulf 2, the next generation of Beowulf.

Second-Generation Beowulf

The hallmark of second-generation Beowulf is that the most error-prone components have been eliminated, making the new design far simpler and more reliable than first-generation Beowulf. Scyld Computing Corporation, led by CTO Don Becker and some of the original NASA Beowulf staff, has succeeded in a breakthrough in Beowulf technology as significant as the original Beowulf itself was in 1994. The commodity aspects and message-passing software remain constant from Beowulf 1 to Beowulf 2. However, significant modifications have been made in node setup and process space distribution.

BProc

At the very heart of the second-generation Beowulf solution is BProc, short for Beowulf Distributed Process Space, which was developed by Erik Arjan Hendriks of Los Alamos National Lab. BProc consists of a set of kernel modifications and system calls that allows a process to be migrated from one node to another. The process migrates under the complete control of the application itself—the application explicitly decides when to move over to another node and initiates the process via an rfork system call. The process is migrated without its associated file handles, which makes the process lean and quick. Any required files are re-opened by the application itself on the destination node, giving complete control to the application process.

Of course, the ability to migrate a process from one node to another is meaningless without the ability to manage the remote process. BProc provides such a method by putting a “ghost process” in the master node's process table for each migrated process. These ghost processes require no memory on the master—they merely are placeholders that communicate signals and perform certain operations on behalf of the remote process. For example, through the ghost process on the master node, the remote process can receive signals, including SIGKILL and SIGSTOP, and fork child processes. Since the ghost processes appear in the process table of the master node, tools that display the status of processes work in the same familiar ways.

The elegant simplicity of BProc has far-reaching effects. The most obvious effect is the Beowulf cluster now appears to have a single-process space managed from the master node. This concept of a single, cluster-wide process space with centralized management is called single-system image or, sometimes, single-system illusion because the mechanism provides the illusion that the cluster is a single-compute resource. In addition, BProc does not require the r commands (rsh and rlogin) for process management because processes are managed directly from the master. Eliminating the r commands means there is no need for user account management on the slave nodes, thereby reducing a significant portion of the operating system on the slaves. In fact, to run BProc on a slave node, only a couple of dæmons are required to be present on the slave: bpslave and sendstats.

The Scyld Implementation

Scyld has completely leveraged BProc to provide an expandable cluster computing solution, eliminating everything from the slave nodes except what is absolutely required in order to run a BProc process. The result is an ultra-thin compute node that has only a small portion of Linux running—enough to run BProc. The power of BProc and the ultra-thin Scyld node, taken in conjunction, has great impact on the way the cluster is managed. There are two distinguishing features of the Scyld distribution and of Beowulf 2 clusters. First, the cluster can be expanded by simply adding new nodes. Because the nodes are ultra-thin, installation is a matter of booting the node with the Scyld kernel and making it a receptacle for BProc migrated processes. Second, version skew is eliminated. Version skew is what happens on clusters with fully installed slave nodes. Over time, because of nodes that are down during software updates, simple update failures or programmer doinking, the software on the nodes that is supposed to be in lockstep shifts out of phase, resulting in version skew. Since only the bare essentials are required on the nodes to run BProc, version skew is virtually eliminated.

Of course, having the ability to migrate processes to thin nodes is not a solution in itself. Scyld provides the rest of the solution as part of the special Scyld Beowulf distribution, which includes features such as:

- BeoMPI: a message-passing library that meets the MPI standard, is derived from the MPICH (MPI Chameleon) Project from Argonne National Lab and is modified specifically for optimization with BProc.
- BeoSetup: a GUI for creating BeoBoot floppy boot images for slave nodes.

- Beofdisk: a utility for partitioning slave node hard drives.
- BeoStatus: a GUI for monitoring the status of the cluster.

Let's take a look at how to use these tools while building a Scyld Beowulf cluster.

You can purchase the Scyld Beowulf Professional Edition (www.scyld.com) that comes with a bootable master node installation CD, documentation and one year of support. The Professional Edition is spectacular and supports many advanced cluster software tools such as the parallel virtual filesystem (PVFS). Alternatively, you can purchase a Scyld Basic Edition CD for $2.95 at Linux Central (www.linuxcentral.com). The Basic Edition is missing some of the features present in the Professional Edition and arrives without documentation or support. I've built clusters with both without any problems.

It's important that you construct your Beowulf similar to Figure 1, which illustrates the general Beowulf (1 and 2) layout. The master node has two network interfaces that straddle the public network and the private compute node LAN. Scyld Beowulf assumes you've configured the network so that eth0 on the master is the public network interface and eth1 is the interface to the private compute node network. To begin the installation, take your Scyld CD, put it in the master node's CD drive and power-cycle the machine.

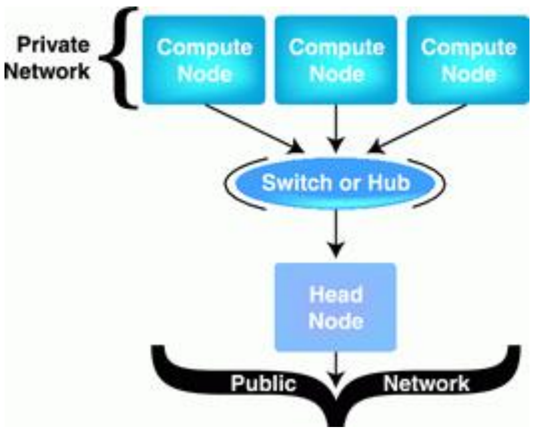


Figure 1. General Beowulf Layout

You'll discover that the Scyld Beowulf master node installation is almost identical to a Red Hat Linux installation. At the boot prompt, type **install** to trigger a master node installation. Allowing the boot prompt to time out will initiate a slave node installation by default.

Step through the simple installation procedure as you would for Red Hat Linux. For first-time cluster builders, we're going to recommend (and assume here) that you select a GNOME controller installation instead of a text-console-only installation. Choosing the GNOME installation will give you access to all the slick

GUI Beo* tools integrated into the GNOME desktop environment that make building the rest of the cluster a snap.

After the typical configuration of eth0, you'll come upon the key difference with the Scyld installation: the configuration of eth1 on the master and the IP addresses of the compute nodes. The installation will prompt you for an IP address (like 192.168.1.1) for eth1 and an IP address range (such as, 192.168.1.2-192.168.*x*) for your compute nodes. Simple enough, but make sure you select an IP range large enough to give each of your compute nodes its own address.

Continue through the remaining installation steps, such as X configuration. For simplicity's sake, select the graphical login option. Wrap up the master node installation by creating a boot disk, removing the CD (and the boot disk) and rebooting the master node.

Log in to the master as root and the Scyld-customized GNOME desktop is fired up for you, including the BeoSetup and BeoStatus GUIs and a compute node quick install guide.

Initially, all compute nodes require a BeoBoot image to boot, either on a floppy or the Scyld CD. Rather than move the Scyld CD from node to node, I prefer to create several slave node boot images on floppies, one for each slave node. The BeoBoot images are created with the BeoSetup tool by clicking the Node Floppy button in BeoSetup. Insert a blank formatted floppy into the master's floppy drive; click OK to create the BeoBoot boot image and write it to the floppy. Repeat this for as many floppies as you like. Insert the boot floppies into the slave node floppy drives and turn on the power.

What happens next is pretty cool but is hidden from the user (unless you hook up a monitor to a slave node). Each slave node boots the BeoBoot image, autodetects network hardware, installs network drivers and then sends out RARP requests. These RARP requests are answered by the Beoserv dæmon on the master, which in turn sends out an IP address, kernel and RAM disk to each slave node. This process, where the slave node bootstraps itself with a minimal kernel on a floppy disk, which is then replaced with a final, more sophisticated kernel from the master, is dubbed the Two Kernel Monte. The slave node then reboots itself with the final kernel and repeats the hardware detection and RARP steps. Then the slave node contacts the master to become integrated into BProc.

During the kernel monte business, slave node Ethernet MAC addresses will appear in the Unknown Addresses window in BeoSetup on the master. You can integrate them into your cluster by highlighting the addresses, dragging them into the central Configured Nodes column and clicking the Apply button. Once the master finishes integrating the slave nodes into BProc the nodes will be labeled “up”. Node status will appear in BeoStatus as well.

You can partition the slave node hard drives with the default configuration in /etc/beowulf/fdisk:

```
beofdisk -d
beofdisk -w
```

The -d option tells beofdisk to use the default configuration in /etc/beowulf/fdisk and the -w option writes the tables out to all the slave nodes. You then need to edit /etc/beowulf/fstab to map the swap and / filesystems to the new partitions. Simply comment out the $RAMDISK line in /etc/beowulf/fstab that was used to mount a / filesystem in the absence of a partitioned hard drive, and edit the next two lines to map the swap and / filesystems to /dev/hda2 and /dev/hda3 (/dev/hda1 is reserved as a bootable partition). If you would like to boot from the hard drive, you can write the Beoboot floppy image to the bootable partition like this:

```
beoboot-install -a /dev/hda1
```

You'll have to add a line in /etc/beowulf/fstab after doing this:

```
/dev/hda1     beoboot     ext2     defaults     0 0
```

Reboot all slave nodes for the partition table changes to take effect:

```
bpctl -S all -s reboot
```

It doesn't get much easier than that. Unlike Beowulf 1, building a Scyld Beowulf requires a full Linux installation on only the master node. Nothing is written out to permanent storage on the slave nodes during their installation, making them ultra-thin, easily maintained and quick to reboot.

To test your cluster you can run the high-performance Linpack benchmark included with the distribution from the command line: **linpack**.

For a little flashier demonstration, launch a visual Mandelbrot set with the included mpi-madel application. Starting mpi-mandel on five nodes from the command line would look like:

```
NP=5 mpi-mandel
```

Collectively, the single-process ID space, the ability to migrate quickly processes under control of the application, thin slave nodes and the GUI utilities for building and monitoring a Scyld cluster, provide a cluster solution that distinguishes itself from Beowulf 1 by its completeness, adaptability and manageability. So, the answer is yes, you really can add more horsepower to that cluster.

Acknowledgements

The authors would like to thank Donald Becker, Tom Quinn and Rick Niles of Scyld Computing Corporation and Erik Arjan Hendriks of Los Alamos National Lab for patiently answering all our questions related to second-generation Beowulf.

Resources

> **Glen Otero** has a PhD in Immunology and Microbiology and runs a consulting company called Linux Prophet in San Diego, California.

Richard Ferri is a senior programmer in IBM's Linux Technology Center, where he works on open-source Linux clustering projects such as LUI and OSCAR. He now lives in upstate New York with his wife, Pat, three teenaged sons and three dogs of suspect lineage.

How a Poor Contract Sunk an Open-Source Deal

Henry W. III

Issue #100, August 2002

Why the Progress and NuSphere vs. MySQL AB litigation is about sloppy deal making, not open-source integrity.

Many describe a new continuing lawsuit in federal court in Boston as “The first litigation testing the validity and enforceability of the General Public License” (GPL). So what?

Will this litigation really impact the future of Linux programmers? Does this dispute matter for companies betting their business models on the open-source trend? Will the judge get the chance to punish an arrogant American software vendor that broke the long-known rules of GNU and thereby defend the OSS cause, as some OSS advocates have suggested?

Sorry, probably not. Yes, the case is important. Yes, it is apparently the first GPL court test, by consensus. But it won't foretell the OSS future because it's a dispute about an extraordinarily poor contract in a context of chaotic, changing communications between the parties.

You can't project the prospects of a programming language from analysis of one short, poorly documented application coded in that language. And in this case, the underlying contract is an outlier that's so far from norms of modern prudent software management and licensing practices that by many orders of magnitude, it's off the map. It ultimately will prove more relevant for “Software Product Management 101” and “Beginner Software Contracts” training than for refining OSS strategies.

Snapshot of a Train Wreck

The story is told in the publicly available court pleadings. The contract underlying the litigants' dispute is a disclosed attachment to the answer filed by the Finnish authors of the well-known MySQL OSS database to the lawsuit initiated by the US software publisher/remarketer. (So the contract and the parties' various arguments, e-mails and affidavits are “open source” for tech managers, lawyers and trainers to study and use to improve work processes.)

This author obtained from court pleadings the original international agreement by which a publicly traded, long-established business software company based in Massachusetts obtained remarketing rights from a young, offshore, small developer in Finland. Ugly surprise: these two companies agreed to do a big-impact, large-dollar deal on a mere nine-paragraph contract. The agreement ran all of 1.25 pages.

Progress Software agreed to pay roughly $300,000 US to a dynamic foreign company in a new, unfamiliar (to Progress) industry segment, on the equivalent of the proverbial envelope. MySQL AB, the Finnish company, blessed the Massachusetts vendor's procurement of its key product by a short statement indicating some future contract would be utilized “later”, triggering “a total of up to $2.5 million”. The resulting fight shows precisely why experienced business people (including lawyers) frown at the optimistic idea of “let's just trust each other and figure out later the deal and the details.”

What's wrong with a little brevity and trust? Think of it this way: why do surgery before taking x-rays or reviewing a medical history? Why not dive head-first in to an unfamiliar river? You can both get hurt and hurt others by launching a major software initiative—OSS or proprietary—without first figuring out the basic rules. That's what happened here.

One purpose of most contracts is similar to the norms of much data processing: benchmarking, testing and standards. Here, fragmentary code got shipped. That is, an incomplete “agreement” was relied upon for too much action, too soon.

Deafening, Deadly Silence

What did this short and ultimately bitter contract omit? The majority of terms and conditions found in most software agreements, that's what. Conspicuous by their absence, among other points, were 1) When would the expected “later, superseding agreement” be completed? 2) Within what parameters for the business terms? 3) Exactly what degree of service would be required and provided for technical support? What did they mean by “enterprise level support” and “existing electronic support channels”? 4) Who would be the designated liaisons for intercompany coordination? 5) What does it mean to give your licensee “fair use” rights to your key trademark, as MySQL AB blessed here? What particular variations would be permitted and excluded? 6) What ongoing product enhancement services by the original author would be assured? 7) How would disputes be resolved or arbitrated, if necessary? 8) If there's a dispute due to one party's fault, will the nonbreaching party get its enforcement costs and damages reimbursed by the defaulting party? 9) Why omit all the often-derided generic or “boilerplate” provisions that are included in most contracts precisely because they help prevent disputes and enable enforcement?

Learning Lessons from Others' Wrecks: Code Your Contracts Like Your Software

Most modern, mature software businesses recognize the many issues that can and do arise in a software distribution deal. They design their deal (e.g., in a “terms sheet” or outline), then “code” (i.e., write a draft contract), then test and document their *agreements* (i.e., negotiate and refine the base contract and write and revise the necessary exhibits), just as they do their *applications*.

For example, many software projects identify “user requirements” in detail and in advance. This deal apparently lacked a joint “terms sheet” or “deal summary memo” as the anchor for the agreement.

Most applications get a look-over for quality control by programmer colleagues. Automated code-testing tools get deployed in some complex environments. This contract presumably was shipped out as the handiwork of one individual, or at least of a very small team.

Savvy software professionals include error-message features. This oblique agreement lacked the typical “notice of breach, then opportunity to cure the breach” provision.

Experienced coders include header files and other technical documentation in their work to assist later revisions and debugging. In your software transactions, include specified modes of communications between the author and publisher companies. Decide up front which particular individuals have the authorization to pass commercial instructions, objections and suggestions to some specified person(s) in the other organization.

Frightful Images: Ships Passing in the Night

The contract's brevity means the parties may raise legal issues that will muddy the waters or at least defer the outcome. Remember, the wheels of the justice system can grind very slowly, at least in the US.

OSS loyalists hoping for court affirmation of the GNU model may be frustrated: both sides of the suit have already raised legal arguments unrelated to the OSS issue. For example, MySQL AB has already obtained (on February 28) a partial injunction against Progress and its young OSS subsidiary NuSphere, but on trademark law grounds, not enforcement of the GPL. The federal judge found the GPL issue too uncertain to adjudicate in this litigation's early, summary phase.

Then there's the legal doctrine of “mutual mistake”. A contract sometimes can go unenforced when both parties inadvertently hold different, though reasonable, interpretations of the deal's predicate and terms. The classic case involves a similar cross-border mishap.

When Going to Rome, Study Ahead

The rashness of this saga is underscored by its multicountry context. Transnational transactions merit extra thinking and terms, just like multinational applications often require more modular screen messaging, two-byte code (for Asian character sets), accommodating different operating system iterations and other shrewd coding.

Doing deals with foreign companies requires extra consideration. For example, many offshore companies prefer (or insist on) the use of arbitration to resolve disputes, both as part of a strong cultural tradition and to avoid the rumored American tendency toward premature, extended and expensive litigation. (Here, the litigants filed 73 different court pleadings in the initial nine months of the case, with no end in sight.)

World travelers arrange translators, confirm supply lines and determine local communication protocols *before* setting out. In international contracts, many companies take similar extra steps. They pre-agree on minimum collaborative product planning, contractually commit to visit each other's headquarters and meet at major global tradeshows and include other contractual “glue code” to help refine the relationship. Common sense says to develop a map when venturing into unfamiliar territory. Here, the parties got lost and found themselves in court, with the resulting marketing disasters, big litigation bills and an uncertain product road map.

What to Think; What to Do

Some in the OSS community have attacked Progress and NuSphere, citing the accurate but fragmentary story that the MySQL code got modified and then marketed via a proprietary license, not the GPL or some other OSS license. True, NuSphere modified its model to use GPL, and in NuSphere's view thus fixed a mere short-term oversight. But that's not the full story. The pleadings suggest another perspective: criticize Progress instead for letting some product manager do a poorly documented contract, presumably without coordinating with counsel and other colleagues. Sentence this individual to attend a licensing workshop. Maybe commute the sentence due to time-to-market competitive pressures. And then bet good money that next time both companies will use traditional, coherent, complete software contracts, after learning from spending big bucks on litigators and losing time, managerial energy and market goodwill.

The Progress-NuSphere-MySQL fight ultimately may prove to be just another chapter in the long book of companies who practiced “ready, fire” without adequate “aim”.

Example of Poor Code

Henry W. (Hank) Jones, III is a 22-year software consultant, manager and lawyer who founded and leads the UC Berkeley Extension software licensing workshop and has worked with over 75 software companies. Operating as MemphisHank@aol.com, he regularly leads corporate training sessions and trade group panels on open source and other software and technology issues.

Archive Index Issue Table of Contents

Advanced search

Hey Embedded Developers! Buy This Magazine!

Don Marti

Issue #100, August 2002

We've added an embedded Linux development section with in-depth information on device-driver coding and more that even a desktop developer can use.

Those of you who are primarily interested in the development of embedded systems may ask why start reading *Linux Journal* now when you can just tell it's going to be full of non-embedded stuff? And why should regular Linux developers care about the embedded articles? The stuff you're interested in won't run on a Linux watch anyway, right?

The short answer is that Greg Kroah-Hartman's Driving Me Nuts column, on device drivers, will be starting this month. Greg maintains the Linux USB and PCI Hot Plug support and has written great articles for both *Linux Journal* and *Embedded Linux Journal*. Now we bring you Kernel Korner every month and Driving Me Nuts every other month, which averages out to a monthly helping and a half of kernel goodness you won't find anywhere else.

Now that you've sent in your subscription card, the long answer is that Linux, and free software in general, are inherently embeddable. By their nature, embedded projects thrive on software that is free to port, customize and sell. If you don't believe this, look at the spectacular success (ha!) of the much-hyped wannabe “Media OS”, BeOS, which sank without a splash while the real media devices, such as TiVo and Moxi, went Linux.

When I told Greg about the new Cypress USB chip, the SL811HS, and the driver for it that Cypress is releasing under GPL, his first reaction was that he was glad Cypress has “a clue”. It looks like a very useful part, too. It does both host and slave, so you could plug a PDA into your computer to sync files and plug USB peripherals, such as a mouse or keyboard, into the PDA later. Try one.

For those of you developing for desktop or server Linux, why make your users sit around and get carpal tunnel syndrome when they could be using a convenient embedded device where the real work (or fun) is? Or better yet, why not cut the user out of the boring stuff altogether and let the machine deal with the real world? Your software will effectively accomplish more in an “embedded” device than in the server room or office. Same goes for your web site. Try it on a Zaurus today.

But back to the clue part. Embedded Linux isn't just a shrink-wrap package that says “embedded” on it. It's a standard generic platform with a thorough collection of drivers that's growing every day (thanks Cypress), distributed under a license that lets you get some work done. Just like regular Linux, embedded systems are good software for clueful people.

1. An embedded system doesn't need system administration.
2. It comes with the application pre-installed. You don't have to install additional software to get something done.
3. It's responsive. Many embedded systems have formal real-time requirements, but even those that don't at least don't make you wait for silly things like fdisk.
4. It interfaces with the real world, not just the network and a carpal-tunnel-slaying keyboard and mouse.
5. It can't be reasoned with! It doesn't feel pity, remorse or fear!
6. And it absolutely will not stop, ever, until you are dead!

No, wait, the last two are “The Terminator”, not all embedded systems. But you get the idea. Embedded systems are what software wants to be when it grows up, and “regular Linux” is headed in the same direction.

> **Don Marti** is technical editor of *Linux Journal*.

The tty Layer

Greg Kroah-Hartman

Issue #100, August 2002

In the first of a series of articles on device-driver development, we'll start with how the kernel handles the system console and serial ports.

Welcome to a new column called Driving Me Nuts. Here we are going to explore the different Linux kernel driver subsystems and try to understand the wide range of different interfaces they provide and expect a driver to provide. If there are any specific subsystems that anyone would like explained, please drop me an e-mail.

There are a number of very good references on Linux kernel programming and Linux driver programming (see Resources). This column assumes you have at least skimmed these in the past or have them handy as a reference.

To start things off, let's look into the kernel's tty layer. This layer is used by all Linux users whenever they type at a command prompt or use a serial port connection.

Every tty driver needs to create a struct tty_driver that describes itself and registers that structure with the tty layer. The struct tty_driver is defined in the include/linux/tty_driver.h file. Listing 1 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/5896.tgz] shows what the structure looks like as of the 2.4.18 kernel version. This is a rather large and imposing structure, so let's try to break it into smaller pieces.

The “magic” field should always be set to TTY_DRIVER_MAGIC. It's used by the tty layer to verify that it is really dealing with a tty driver.

The driver_name and name fields are used to describe your driver, and driver_name should be set to something descriptive, as it will show up in the /proc/tty/drivers file. The name field is used to specify what the /dev or devfs name base is for your driver. As an example, the kernel serial driver sets the driver_name field to serial, the name field to ttyS if devfs is not enabled, and tts/%d if devfs is enabled. If devfs is enabled, it will use the name field when creating new device nodes for your driver. The %d portion of the name will be filled in with the minor number of the device when it is registered in the tty subsystem.

The name_base field is only necessary if your device does not start at minor number 0. For almost all drivers, this should be set to 0.

The major, minor_start and num fields are used to describe what major/minor numbers are assigned to your driver to the tty layer. The major field should be set to the major number assigned to your driver. If you are creating a new driver, read the file Documentation/devices.txt on getting a new major number for your driver. This file is also good reading for anyone who wants to see what major/minor number pair is used by what driver. The minor_start field is used to specify where the first minor number is for your device. If you have an entire major number assigned to your driver, then this should be set to 0. The num field describes how many different minor numbers you have assigned to your driver.

So if you have all of major 188 assigned to your driver, then your driver should set these fields to:

- major: 188,
- minor_start: 0,
- num: 255,

The type and subtype fields describe what kind of tty driver your driver is to the tty layer. The type field can be set to the following values:

- TTY_DRIVER_TYPE_SYSTEM: used internally by the tty subsystem to notify itself that it is dealing with an internal tty driver. If this value is used, then subtype should be set to SYSTEM_TYPE_TTY, SYSTEM_TYPE_CONSOLE, SYSTEM_TYPE_SYSCONS or SYSTEM_TYPE_SYSPTMX. This type should not be used by any normal tty driver.
- TTY_DRIVER_TYPE_CONSOLE: only used by the console driver. Do not use it for any other driver.
- TTY_DRIVER_TYPE_SERIAL: used by any serial type driver. If this value is used, then subtype should be set to SERIAL_TYPE_NORMAL or SERIAL_TYPE_CALLOUT, depending on which type your driver is. This is one of the most common settings for the type field.
- TTY_DRIVER_TYPE_PTY: used by the pseudo-terminal interface (pty). If this value is used, then subtype needs to be set to either PTY_TYPE_MASTER or PTY_TYPE_SLAVE.

The init_termios field is used to set up the initial termios (the line settings and speeds) for the device when it is first created.

The flags field is set to a mixture of the following bit values, depending on the needs of the driver:

- TTY_DRIVER_INSTALLED: if this bit is set, the driver cannot register itself with the tty layer, so do not use this value.
- TTY_DRIVER_RESET_TERMIOS: if this bit is set, the tty layer will reset the termios setting whenever the last process has closed the device. This is useful for the console and pty drivers.
- TTY_DRIVER_REAL_RAW: if this bit is set, it indicates that the driver guarantees to set notifications of parity or break characters up to the line driver if the line driver has not asked to be notified of them. This is usually set for all drivers, as it allows the line driver to be optimized a little better.
- TTY_DRIVER_NO_DEVFS: if this bit is set, then the call to tty_register_driver() will not create any devfs entries. This is useful for any driver that dynamically creates and destroys the minor devices, depending on whether the physical device is present. Examples of drivers that set this are the USB to serial drivers, the USB modem driver and the USB Bluetooth tty driver.

The refcount field is a pointer to an integer within the tty driver. It is used by the tty layer to handle proper reference counting of the driver and should not be touched by the tty driver.

The proc_entry field should not be set by the tty driver itself. If the tty driver implements the write_proc or read_proc functions, then this field will contain the driver's proc_entry field that will have been created for it.

The other field is only used by the pty driver and should not be used by any other tty driver.

Now we have some pointers to different tty structures. The table field is a pointer to an array of tty_struct pointers. The termios and termios_locked fields are pointers to an array of struct termios pointers. All of these arrays should have the same number of entries as you have set the minor field to above. They are used by the tty layer to handle the different minor devices properly and should not be touched by your tty driver.

The driver_state field is only used by the pty driver and should not be used by any other tty driver.

There is a large list of different function pointers in the tty_driver structure. These function pointers are used by the tty layer to call into the tty driver when it wants to do something. Not all of them have to be defined by a tty driver, but a few of them are required.

The open function is called by the tty layer when open(2) is called on the device node to which your tty driver is assigned. The tty layer calls this with a pointer to the tty_struct structure assigned to this device and a file pointer. This field must be set by a tty driver for it to work properly (otherwise, -ENODEV is returned to the user when open(2) is called).

The close function is called by the tty layer when release(2) is called on the file pointer that was previously created with a call to open(2). This means that the device should be closed.

The write function is called by the tty layer when data is to be sent to your tty device. The data may come from user space or kernel space (the field from_user will be set if the data comes from user space). This function should return the number of characters that are actually written to the device. This function must be set for a tty driver.

The put_char function is called by the tty layer when a single character is to be written to the device. If there is no room in the device for the character to be sent, the character may be ignored. If a tty driver does not define this function, then the write function will be called when the tty layer wants to send a single character.

The flush_chars function is called when the tty layer has sent a number of characters to the tty driver using the put_char function. The tty driver should tell the device to send all of the data remaining in it out of the serial line.

The write_room function is called when the tty layer wants to know how much room the tty driver has available in the write buffer. This number will change over time as characters empty out of the write buffers.

The chars_in_buffer function is called when the tty layer wants to know how many characters are still remaining in the tty driver's write buffer to be sent out.

The ioctl function is called by the tty layer when ioctl(2) is called on the device node. It allows the tty driver to implement device-specific ioctls. If the ioctl requested is not supported by the driver, it should return -ENOIOCTLCMD. This allows the tty layer to implement a generic version of the ioctl, if possible.

The set_termios function is called by the tty layer when the device's termios settings have been changed. The tty driver should then change the physical settings of the device, depending on the different fields of the termios structure. A tty driver should be able to handle the fact that the old variable might be set to NULL when this function is called.

The throttle and unthrottle functions are used to help control overruns of the tty layer's input buffers. The throttle function is called when the tty layer's input buffers are getting full. The tty driver should try to signal the device that no more characters are to be sent to it. The unthrottle function is called when the tty layer's input buffers have been emptied out, and it now can accept more data. The tty driver should then signal to the device that data can be received.

The stop and start functions are much like the throttle and unthrottle functions, but they signify that the tty driver should stop sending data to the device and then later resume sending data.

The hangup function is called when the tty driver should hang up the tty device.

The break_ctrl function is called when the tty driver is to turn on or off the BREAK status on the RS-232 port. If state is set to -1, then the BREAK status should be turned on. If state is set to 0, then BREAK should be turned off. If this function is implemented by the tty driver, then the tty layer will handle the TCSBRK, TCSBRKP, TIOCSBRK and TIOCCBRK ioctls. Otherwise these ioctls will be sent to the tty driver's ioctl function.

The flush_buffer function is called when the tty driver is to flush all of the data still in its write buffers. This means any data remaining in them will be lost and not sent to the device.

The set_ldisc function is called when the tty layer has changed the line discipline of the tty driver. This function is generally not used anymore and should not be set.

The wait_until_sent function is called when the tty layer wants all of the pending data in the tty driver's write buffers to be sent to the device. The function should not return until this is finished and is allowed to sleep in order to achieve this.

The send_xchar function is used to send a high-priority XON or XOFF character to the tty device.

The read_proc and write_proc functions are used if the driver wants to implement a /proc/tty/driver/<*name*> entry; <*name*> will be set to the name field described above. If either of these functions are set, then the entry will be created and any read(2) or write(2) calls will be passed to the appropriate function.

And finally, the next and prev fields are used by the tty layer to chain all of the different tty drivers together and should not be touched by the tty driver.

No Read?

One thing that might stand out in the above list of functions is the lack of a read function to be implemented by the tty driver. The tty layer contains a buffer that it uses to send data to user space when read(2) is called on a tty device node. This buffer needs to be filled up by the tty driver whenever it receives any data from the device. Because tty data does not show up whenever a user asks for it, or wants it, this model is necessary. That way the tty layer buffers any received data and the individual tty driver does not have to worry about blocking until data shows up on the tty line.

The Tiny tty Driver

Now that we have gone over all of the different fields, which ones are actually necessary to get a basic tty driver up and running? Listing 2 is an example of the most minimal tty driver possible. Once the steps shown there are completed, create the tiny_open, tiny_close, tiny_write and tiny_write_room functions and you are finished with a tiny tty driver.

Listing 2. Minimal TTY Driver

For an example implementation of a tiny tty driver, see Listing 3 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/5896.tgz]. This tty driver creates a timer to place data into the tty layer every two seconds in order to emulate real hardware. It also properly handles locking the device structures when it is run on an SMP machine.

Flow of Data

When the tty driver's open function is called, the driver is expected to save some data within the tty_struct variable that is passed to it. This is so the tty driver will know which device is being referenced when the later close, write and other functions are called. If this is not done, the driver can key off of the MINOR(tty->device) function, which returns the minor number for the device.

If you look at the tiny_open function, the tiny_serial structure is saved within the tty driver. This allows the tiny_write, tiny_write_room and tiny_close functions to retrieve the tiny_serial structure and manipulate it properly.

The open and close functions of a tty driver can be called multiple times for the same device as different user programs connect to the device. This could allow one process to read data and another to write data. To handle everything correctly, you should keep a count of how many times the port has been opened or closed. When the port is opened for the first time you can do the necessary hardware initialization and memory allocation. When the port is closed for the last time you can do the proper hardware shutdown and free any allocated memory. See the tiny_open() and tiny_close() functions for examples of how the number of times the port is opened can be accounted for.

When data is received from the hardware, it needs to be placed into the tty device's flip buffer. This can be done with the following bit of code:

```
for (i = 0; i < data_size; ++i) {
        if (tty->flip.count >= TTY_FLIPBUF_SIZE)
                tty_flip_buffer_push(tty);
        tty_insert_flip_char(tty, data[i], 0);
}
tty_flip_buffer_push(tty);
```

This example makes sure there are no buffer overflows in the tty flip buffer as the data is being added. For drivers that accept data at very high rates, the tty->low_latency flag should be set, which will cause the last call to tty_flip_buffer_push() to be executed immediately when called. In the example driver, the tty_timer function inserts one byte of data into the tty's flip buffer and then resubmits the timer to be called again. This emulates a slow stream of characters being received from a hardware device.

When data is to be sent to the hardware, the write function is called. It is important that the write call checks the from_user flag to prevent it from accidentally trying to copy a user-space buffer directly into kernel space. The write function is allowed to return a short write. This means that the device was not able to send all of the data requested. It is up to the user-space program that is calling the write(2) function to check the return value properly to determine if all of the data was really sent. It is much easier for this check to be done in user space than it is for a kernel driver to sleep until all of the requested data is able to be sent out.

The tty Interface over Time

The tty layer has been very stable from the 2.0, 2.2 and 2.4 kernel versions, with only a very minor amount of functionality added over time. So a tty driver written for 2.0 will successfully work on 2.4 with almost no changes. Throughout the 2.5 kernel series, the tty layer has been marked for a rewrite, so this article may describe things that are no longer true. When in doubt, read the include/tty_driver.h file of the kernel version for which you wish to develop. Also, take a look at any of the tty drivers in the driver/char kernel directory for examples of how to implement some of the functions that are not covered here.

Conclusion

We have covered the basics of the tty layer, explaining all of the different fields in the tty_driver structure for the 2.4 kernel tree and pointing out which ones are necessary for a driver to implement. The tiny_tty.c driver, see Listing 3 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/5896.tgz], is a good example of a very minimal tty driver that successfully works. Feel free to use this code as an example for your own tty drivers in the future.

Resources

Greg Kroah-Hartman is currently the Linux USB and PCI Hot Plug kernel maintainer. He works for IBM, doing various Linux kernel-related things and can be reached at greg@kroah.com.

Embedded System àla Carte

Peter Ryser

Michael Baxter

Issue #100, August 2002

Virtex-II Pro FPGA and MontaVista Linux provide a highly flexible and powerful solution for embedded system designers.

How many times have you faced the problem that after the specification phase is complete you simply cannot find a matching microprocessor with the required type and number of peripherals? How many times did you find out that for the closest matching microprocessor no Linux port is available? How hard was it to put together all the tools you need to design and debug your system? How many hours have you spent working around all of these issues? In any case, relax! The Virtex-II Pro FPGA family and the corresponding tools combined with MontaVista Linux OS and development environment enables you to pick your system à la carte and saves you a lot of time and sweat.

The Virtex-II Pro FPGA family offers up to four hard-core PowerPC 405 processors, 16 Rocket I/O 3.125Gbps serial transceivers, 3.8Mb of block RAM (BRAM) and four million system gates on a single programmable platform chip. This rich set of features opens the field for a wide range of applications and provides high flexibility for system designers. Single-chip systems with an arbitrary number and type of I/O devices are possible. For example, you can have five UARTs, a PCI bus and several Gigabit Ethernet ports all controlled by the on-chip processors.

An Xilinx partnership with IBM introduces support for the hard-core PPC405, a robust busing standard and a device fabrication process. Xilinx offers a wide variety of intellectual property (IP) cores for the Virtex-II Pro FPGA, which are predefined hardware blocks that directly snap into IBM's CoreConnect bus technology (Figure 1).

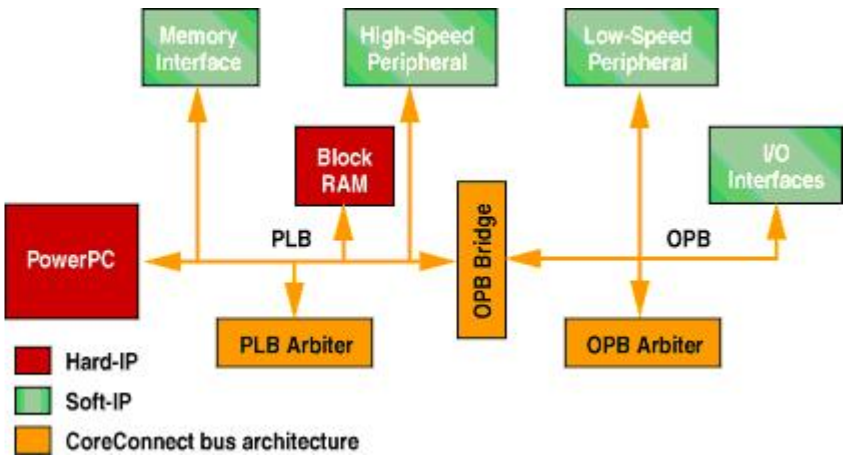


Figure 1. IP Cores Connected to CoreConnect

CoreConnect has several features for interconnecting the processor to the FPGA fabric in order to construct system designs. The key CoreConnect features are the application-oriented system buses, which include the processor local bus (PLB), the on-chip peripheral bus (OPB) and the device control register (DCR) bus. Selected IP cores include many forms of I/O and external memory controllers. For I/O, the cores include 16450- and 16550-compatible UARTs, IIC controller and SPI controller. The memory controllers include support for SRAM/FLASH, SDRAM, DDR SDRAM and ZBT devices. More sophisticated, system-level I/O is also available as IP cores: PCI cores, 10/100 Mb Ethernet, Gb Ethernet and the System ACE configuration interface. Each core is delivered with a corresponding driver where appropriate. The drivers are shipped as source code.

The powerful software design tools that help the user make use of the processors, the IP cores and, more generally the FPGA, will be described here in more detail. The interaction of this hardware and software is unique for the Virtex-II Pro FPGA because it enables the design of a completely custom embedded system solution on a single chip. The issues of hardware/software codesign include use of powerful application software tools for embedded Linux and some unique tools for embedded debugging. Before delving into codesign issues, let's look a little deeper inside the Virtex-II Pro FPGA.

The PowerPC 405 processors run at 300+ MHz and each have 16KB of data and 16KB of instruction cache. The PLB and OPB buses run at 100+ MHz. A powerful feature of the Virtex-II Pro FPGA is the implementation of the on-chip memory (OCM). The OCM is memory that has similar access characteristics as the processor caches but is managed by the user. The Virtex-II Pro FPGA implements the OCM as dual-ported BRAM, allowing for an extremely fast data path from peripheral devices to the processor or as a communication buffer between processors.

The Rocket I/O 3.125Gbps serial transceivers support many different communication standards, listed in Table 1. Important features like buffers, 8B/10B encoding and decoding, and CRC calculation and checking are implemented on-chip and do not take space away from the FPGA fabric.

Table 1. Virtex-II Pro Platform FPGAs support these protocols and baud rates.

The BRAMs can be used as memory accessible by the processors and can be connected to CoreConnect or as OCM. While OCM offers better performance and a direct way to the processor, hooking up the BRAM to the PLB makes it available for DMA transfers where the processor is not involved. It is up to the system designer to decide how the BRAM is used best.

The FPGA fabric can be filled with IP cores provided from Xilinx or with user-specified designs. While some of the IP cores interact with each other, others work completely in parallel.

Because FPGAs can be reconfigured dynamically you can even switch the number and type of devices while the system is running. Such a flexible platform needs a powerful operating system and tools for hardware and software engineers to develop and debug their embedded applications.

Depending on their application, users may choose from a wide variety of COTS (commercially available off the shelf) real-time operating systems, proprietary RTOS or embedded Linux environments. In the Linux arena, Xilinx chose MontaVista Linux as a suitable embedded OS and development environment for the Virtex-II Pro FPGA family because of its extreme versatility. Linux, by default, supports a wide range of devices. Xilinx and MontaVista Software chose to use a layered device-driver approach. It is split into a low-level, OS-independent layer that directly sets up on the hardware and an OS specific adaptation layer that sits in the middle between the OS and the low-level drivers. Xilinx provides the low-level drivers written for optimal performance and best use of the functionality of the IP cores. MontaVista Software implements the adaptation layer for Linux and uses its expertise for a seamless integration of the drivers. Both driver layers are pushed into the open-source repository and are released under the GPL (General Public License). All device drivers can be compiled into the kernel or are available as loadable modules. Being able to load and unload driver modules supports hardware that can be reconfigured while the system is running. Replacing hardware in a running system is a well-known process for external devices like USB devices and PC cards, but replacing hardware on the chip and dynamically loading the corresponding Linux driver is something completely new.

The Linux port is first targeted toward the Virtex-II Pro ML300 Platform from Xilinx (shown in Figure 2), and it supports most of the IP cores and hardware on this board. Since the ML300 is an elaborate board with a lot of functionality, it will be easy for users and customers to adapt the port to their own specific hardware. MontaVista Software offers professional support for such projects.

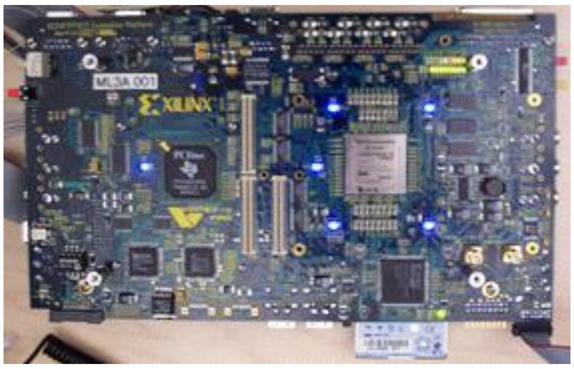


Figure 2. Virtex-II Pro ML300 Platform—Bottom View

Still, hardware and software engineers need powerful tools to develop, boot and debug their systems. With System Generator for Processors, GDB/XMD, System ACE and ChipScope Pro, Xilinx has a complete tool suite for all aspects of hardware, software and systems engineering.

System Generator for Processors (SGP) helps you put together the design for the Virtex-II Pro FPGA both on the hardware and on the software sides. In user-friendly dialog boxes you can specify all the parameters for the system, like base addresses for the peripherals, interrupts to be used and amount of memory present. As a result, SGP emits the hardware design files that are ready for FPGA implementation or simulation and a parameter file that is used when the Linux kernel is built and the corresponding driver modules are created. Listing 1 shows an excerpt of the parameter file. In this case, the user assigned parameters to configure the interrupt controller in the system.

<u>Listing 1. Excerpt of the Parameter File to Configure the Device Drivers</u>

SGP gives system architects the flexibility to investigate different options and variations for their new embedded systems. Setting the parameters to different values tailors the hardware and software to the specific requirements. Only functionality that will be accessed and used is included in the system; other functionality is stripped out of the design. Additionally, devices are preconfigured with the default parameters. As a result the hardware and software use less space, offer better performance and the initialization process is much simpler (in some cases it is not even required). SGP and its companion tools use an open interface that makes it simple for users to add their own hardware functionality and software drivers. XMD is a debugging server using the on-chip debug (OCD) protocol to communicate with GDB on the host system. It controls the target system through the JTAG port of a processor in the Virtex-II Pro FPGA. At the same time, XMD serves multiple GDBs. Thus, it is possible to debug more than one processor at once. More specifically, all four processors within one Virtex-II Pro FPGA can be debugged simultaneously. On Linux, GDB runs on the command line or with one of the different front ends. It is compiled with support for the Insight GUI but can also be used with DDD and Emacs.

In GDB the PPC405 architecture was added and the “target ocd” command was extended to support all the features of the Virtex-II Pro FPGA. As a result all the registers of the PPC405, the caches, the TLB (translation look aside buffer) entries and the contents of the OCM cannot only be inspected and changed but also can be mapped into the memory space of the processor.

Debugging an embedded system traditionally has been a difficult process. You have to look at hardware and software at the same time. An external, non-intrusive software debugger like GDB combined with XMD is a big help. Additionally, the PPC405 supports hardware breakpoints and allows freezing the processor on exceptions. But especially when processors and peripherals are integrated on the same chip, it is difficult to see what transactions are executed and how (in which order) memory is accessed. All the important signals are buried within the chip, and there is normally no way to get access to these signals. The Virtex-II Pro FPGA does not have such a limitation because it integrates peripherals as soft hardware. All signals are visible and can be accessed with the appropriate tool.

ChipScope Pro is an integrated hardware logic analyzer. It consists of a logic analyzer running on the debugging host system and a set of trigger and data units compiled or inserted into the hardware design. The integrated logic analyzer (ILA) units can be hooked up to any number of signals inside the FPGA and can trigger on user-defined conditions or processor bus transactions. Multiple ILA units can be active at the same time. Sometimes it is useful to hook up multiple ILAs to the same signals. In one case, we hooked up two different ILA units to the PLB address and data lines to resolve a problem with corrupted memory. One ILA was hooked up to the PLB signals connected to the PPC405. We knew that the processor would do a memory access at the time when the corruption would occur. The other ILA was hooked up to the PLB signals connected to the DDR memory controller. By comparing the address and data lines reported by the two ILA units, we were able to isolate the problem and fix it. Having access to the hardware and being able to watch bus transactions is very useful, especially in Linux where the MMU is used. From a software perspective, the same physical block of RAM can be mapped into many different virtual address spaces. On the hardware level, all addresses are physical.

The combination of ChipScope Pro, GDB and XMD gives the developer extremely high visibility into the system. The software tools share a common cable and communicate through the JTAG port of the FPGA. The friendly cooperation between the tools reduces the number of cables and makes the setup of the debugging environment much easier.

The boot process is an imminently important phase when an embedded system is powered up. In a few steps the components on the board, the processor, the memory system and the communication infrastructure are brought up. On the Virtex-II Pro FPGA, the boot process happens in two steps. On one hand, the FPGA is configured, and on the other hand, the processor is started. The FPGA needs to be configured with its functionality in one of many different ways. We will show one recommended method later in this article. Often a specialized primary boot loader is used to start the processor, bring up the system, load the Linux kernel into memory and transfer control to the entry point of the kernel. The Virtex-II Pro FPGA supports this traditional boot method where the primary boot loader resides in external ROM or in internal BRAM. The latter case removes the need for external ROM in that the primary boot loader is included in the bit stream that configures the FPGA. Immediately after the FPGA is configured the processor is released from reset, starts reading instructions from the internal BRAM and executes the primary boot loader.

A new and, especially with MontaVista Linux, powerful solution to boot the system uses System ACE. System ACE is a companion chip external to the Virtex-II Pro FPGA and allows booting a system without having any ROM. It has two main functions. For one, it boots the system by configuring the FPGA, the processor and any device on the processor bus through the JTAG chain from a CompactFlash card or a Microdrive. And two, it uses the same storage device as a filesystem accessible by Linux.

The Microdrive contains a FAT12 or FAT16 and a Linux partition. The Linux kernel is configured with support for the System ACE device, compiled, converted into a System ACE-specific file format, concatenated with the configuration bit stream for the FPGA and stored on the FAT filesystem. On power-up, System ACE reads the configuration file from the FAT filesystem, configures the FPGA and boots the kernel. During the boot process, the Linux kernel mounts the Linux partition on the Microdrive as a root filesystem. The non-obvious advantages of booting with System ACE are that no memory at all is required at the reset vector of the processor, different boot configurations can be stored on the FAT partition and the boot configuration can be changed by normal file operations.

System ACE works on the JTAG chain like an external debugger through the debug port of the processor. Code, data and, if required, a RAM disk, for the

Linux kernel are loaded through the JTAG chain and the processor bus into system memory. Configuration of any devices in the system accessible by the processor can be done in the same way before the kernel is loaded. And at the end, the program counter of the PPC405 is set to the start address of the Linux kernel and directly executed from this location.

A switch points System ACE to one of eight active configurations. The configuration to be loaded also can be set by software. A running Linux system selects the new configuration, resets the system and boots into this new configuration that may consist of a different set of peripherals.

The FAT filesystem allows Linux to update the System ACE file while the system is running—a very powerful solution to upgrade hardware and software in-system and on the fly.

The Virtex-II Pro Developer's Kit adds another dimension to a successful system design experience. The kit allows you to simulate your embedded system before you build real hardware and introduces another abstraction level during the debugging phase. Each component of the system can be simulated on its own. Once the whole system is put together, hardware and software can be run in the simulation to verify the functionality of the embedded system. Problems observed in real hardware can be taken back into simulation and tracked down. GDB/XMD can be configured to connect to HDL simulators and enables an engineer to follow the program execution step by step and watch the bus transactions and hardware state changes as they occur. The completeness of the combination of the Virtex-II Pro FPGA with MontaVista Linux makes it an ideal platform for many different applications. The Rocket I/O serial multigigabit transceivers make it interesting for telecommunications, for example, in base stations where complex calculations have to be combined with high bandwidth and enormous computing power. The same transceivers can also be used as a backplane interconnect between multiple devices. The available peripherals combined with up to four processors also make it an ideal platform for data and graphics terminals or even as a workstation.

The integration of processors into the FPGA fabric offers some opportunities for interesting system architectures and future development. On the architectural side, in a simple system, multiple processors can be interconnected by a shared PLB. A more complex system uses a switched approach to prevent congestion on the bus and gain better performance. Due to the nature of FPGAs, system designers might start with the simple approach and later change their strategy. In any case, Linux will have to support the architecture. Since semaphores and mutexes are easily implemented by dual-ported BRAM, resource management and access to shared memory are straightforward. Hardware/software coprocessing will improve system performance a great deal. While hardware is fast and can execute in parallel, software is much more flexible. Linux will call certain system functions that in reality are implemented in hardware. It will be a challenge for the system designer to find the functions for which it makes sense to off-load into hardware, but it pays off in a faster and more dynamic system. In an even more complicated system Linux will use dynamic coprocessing. It partially reconfigures the FPGA with the desired hardware functions optimized for the currently running applications. While one application calculates extensive FFT transformations, another application searches for patterns in a data stream. Whenever the scheduler transfers control to one of the two applications, it also replaces the corresponding IP. Based on statistical data, the scheduler decides whether the application will be hardware accelerated or whether a corresponding software function is used. The Virtex-II Pro FPGA and MontaVista Linux combined with the corresponding system generation, debugging and configuration tools is a powerful and flexible solution. It enables you to implement the design in your specification and not the one given by hardware and software limitations, increases the integration factor without losing observabilitiy, reduces time-to-market because of available IP cores and related software drivers, and finally, opens a new dimension to your creativity with respect to hardware/software codesign.

Michael Baxter has been working in computer technology since he was nine, imprinted by a 1969 viewing of 2001: A Space Odyssey. He is an experienced computer architect, system, board and FPGA logic designer. Michael holds ten US patents in computer architecture and logic, plus five patents as a co-inventor. His interests also include hiking, amateur radio and programming in Lisp.

Peter Ryser works as a systems design engineer for Xilinx, Inc. He is responsible for various embedded software-related projects for Virtex-II Pro and can be reached at peter.ryser@xilinx.com.

Archive Index Issue Table of Contents

Advanced search

GNU Bayonne Is for Telephony

David Sugar

Issue #100, August 2002

The Bayonne Project makes running telephony software on any vendor's card possible.

Three years ago I came to realize that we had a serious need in free software. Although free software had expanded to fill almost every other void in the enterprise infrastructure, we had not addressed the needs of telecommunications. Telecommunications are not only a part of the infrastructure of every business, but they are also an often overlooked part of the desktop user's experience. At the same time, the hardware required to create telephony services for the public telephone network has become more widely available under commodity PC platforms and operating systems, including GNU/Linux.

In choosing to address telecommunications with free software, I and a few others decided to create a framework describing what all these services might be, ranging from the needs of desktop users and application programmers to the needs of the largest commercial carriers. This project later became known as GNUCOMM when it was officially folded into a GNU project working group.

One area we chose to define was the idea of a telephony application server. Such a server should make it both possible and easy to create and deploy new telephony application services. These would be applications specifically written to interact with real people that call the server over regular telephone lines and interact with the application with both a voice and a telephone keypad.

Applications of this nature typically include things like voice-mail systems or prepaid (debit card) calling platforms. All of these systems are complex and sometimes programmable systems and specialized computer telephony hardware are needed to provide an interface between the PC platform and the public telephone network. This can be hardware that talks to individual analog telephone lines or even hardware that provides multiport voice control over

ISDN and T1 digital voice circuits, which larger enterprises can get directly from a local carrier's central office.

With full consideration that such systems in the past were generally very expensive, always proprietary and often hard to program, I chose to solve all of these problems at once by writing a server under the best supported free software platform available at the time: GNU/Linux.

When we started the project, few companies provided telephony hardware under GNU/Linux, so we used what was available. Even now, each telephony card is different from every other one and tends to include its own API. Since neither the hardware nor the APIs are in any manner standardized, most people that produce telephony applications do so for only a single vendor's card family, and they do so using exclusively the vendor's supplied API. This practice also means that any vendor in the computer telephony business has to provide a very broad family of hardware because one could not substitute easily other products to fit gaps in a product offering. All these things have made it difficult for new telephony card vendors to come into existence and easy for the limited vendors that do exist to maintain their markets without much change.

This is not to say that no efforts were made to standardize APIs. After all, there is the ECTF (European Community Telework/Telematics Forum). Being an industry consortium of proprietary vendors, they would have to come up with, through committees, a complicated set of standards and proposals for how proprietary vendors could develop and maintain computer telephony solutions. Furthermore, they would need to do so in ways that expand the need for specialized knowledge, increasing the stranglehold of their existing members on the computer telephony marketplace.

Another popular organization is the ITU (International Telecommunications Union), best known for the fact that appointment often is handled by national governments. In the US, for example, this is done as a political appointment by the state department, rather than from among the best and brightest minds.

Our goal was not only to produce a telephony server as free software, but we wanted also to make telephony application services as readily and easily approachable as creating and administering a web site. We also wanted to abstract the telephony driver and APIs to the point that they were both irrelevant and invisible in the development of application services. Doing so would mean anyone could substitute hardware as they wished, rather than being locked into the offering of a single vendor.

First Came ACS

Since we wanted to abstract everything within the server at a low level, the first thing we needed was a portable class foundation written in C++. I wanted to use C++ for several reasons. First, it seemed natural to use class encapsulation for the driver interfaces because of their abstract nature. Second, I found I could write bug-free C++ code faster than I could write C code. In fact, this would become my first large-scale C++ project.

Why we chose not to use an existing framework is also simple to explain. We knew we needed threading, socket support and a few other elements. No single existing framework did all these things except a few that were larger and more complex than we needed. For example, we wanted a small footprint for a telephony server. The most adaptable framework at the time was ACE (Adaptive Communication Environment), which typically added several MBs of core image for the runtime library. Since we were looking at running on machines with as little as 8-12MBs of memory, this seemed an unacceptable overhead.

GNU Common C++ (originally APE) was created to provide an easy-to-comprehend and portable class abstraction for threads, sockets, semaphores, exceptions and so on. APE has since grown and is now used as a foundation for a number of projects in addition to being a part of GNU.

As to creating services themselves, we realized we needed a new way to create telephony applications—one that would make the process approachable for the average system administrator. For simplicity we choose to use a common scripting language, which later became known as GNU ccScript. By writing scripts and recording audio samples to create telephony application services, virtually anyone could participate without needing specialized knowledge or deep understanding of fantastically complex APIs like those promoted by the ECTF. Because the underlying telephony hardware is both invisible and abstracted away from the application scripting language, the cycle of dependence on using a single card family is also broken.

But what form should this new scripting language take? Many extension languages assume a separate execution instance (thread or process) for each interpreter instance, making them unsuitable for our project. Many extension languages assume expression parsing with nondeterministic runtime. An expression could invoke recursive functions or entire subprograms, for example. Again, we did not want to have a separate execution instance for each interpreter instance, and we did not want to have each instance respond to the leading edge of an event callback from the telephony driver as it steps through a state machine, so none of the existing common solutions like Tcl, Perl, Guile, etc., would immediately work for us. Instead, we created an entirely new nonblocking and deterministic scripting engine for our first server.

Our scripting language is unique in several ways. First of all, it is step executed and nonblocking. Statements can either execute and return immediately or schedule their completion for a later time with the executive. This allows a single thread to invoke and manage multiple interpreter instances. While a telephony server can potentially support interactions with hundreds of simultaneous telephone callers on high-density carrier scale hardware, we do not require hundreds of native “thread” instances running in the server, and we have a very modest CPU load.

Another way our scripting is unique is in support for memory-loaded scripts. To avoid delay or blocking while loading scripts, all scripts are loaded and parsed into a virtual machine (VM) structure in memory. When we wish to change scripts, a brand new VM instance is created to contain these scripts. Calls currently in progress continue under the old VM, and new callers are offered the new VM. When the last old call terminates, then the entire old VM is disposed of. This allows for 100% uptime, even while services are modified.

Finally, since we were building a C++ scripting system, we allowed direct class extensions of the script interpreter as a means to add new script functionality. This allows one to create a derived dialect specific to a given application or, if needed, specific to a given telephony driver, simply by deriving it from the core language through standard C++ class extension.

While the server scripting language can support the creation of complete telephony applications, it was not designed to be a general-purpose programming language or to integrate with external libraries the way traditional languages do. Nonblocking requires that any module extensions created for the server be highly customized. Instead, we wanted a general-purpose way to create script extensions that could interact with databases or other system resources. To that end we chose a model essentially similar to how a web server did this when our ACS (Adjunct Communication Server) Project was created.

The TGI model for our server is similar to how CGI works for a web server. In TGI, a separate process is started, then is passed information on the phone caller through environment variables. Environment variables rather than command-line arguments are used to prevent snooping of transactions that might include things such as credit-card information that could be visible with a simple ps command.

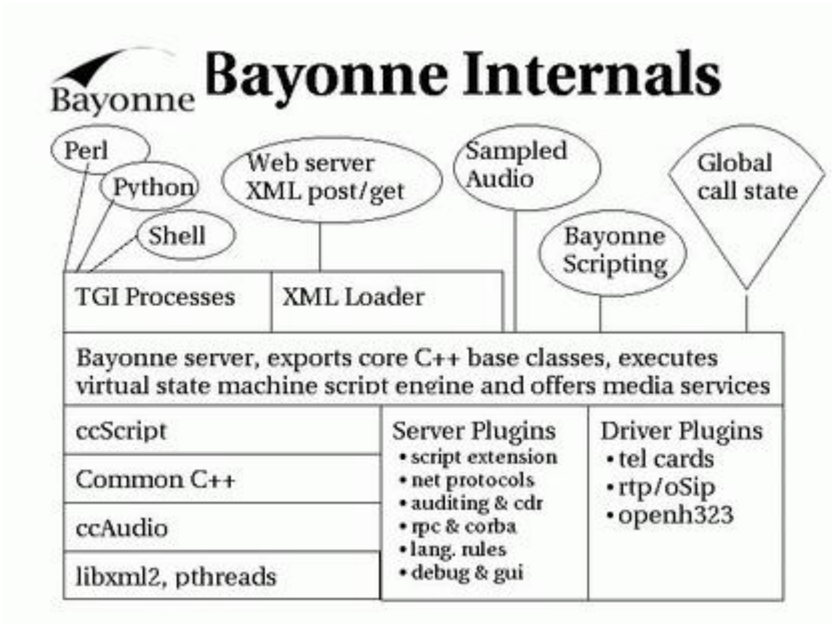
The TGI process is tethered to the server through **stdout** and any output the TGI application generates is used to invoke server commands. These commands can do things like set return values, such as the result of a database lookup, or they can do things like invoke new sessions to perform outbound dialing. Rather than creating a gateway for each concurrent call session, a pool of available processes are maintained for TGI gateways so it can be treated as a restricted resource. It is assumed that gateway execution time represents a small percentage of the total call time, so maintaining a small process pool always available for quick TGI startup is efficient. This helps to prevent stampeding if, say, all the callers hit a TGI at the same moment.

With these basic tools, it was possible to create interactive voice response applications. As soon as it was functional, our first telephony server was used commercially by Open Source Telecom and other companies. This wide adoption was a result in part of how simple it is to create new application services and to integrate telephony applications under this server with other aspects of a commercial enterprise. As noted, the only requirements are some skill in constructing a server-side script, the ability to play and record audio and some knowledge of common tools like Perl.

A typical application for our server might look like the one shown in Listing 1 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/6077.tgz], the playrec script. This script demonstrates the different concepts in the current scripting language, including symbol scope and event trapping, which, used under named script references, form a chain of logic for processing an interactive telephony application. In Listing 2 [available at ftp.linuxjournal.com/pub/lj/listings/issue100/6077.tgz], we have an example of the server's use of Perl with the TGI.pm module and the tgigetdbval.pl Perl script.

How ACS Became GNU Bayonne

As noted earlier, we achieved all these goals some two years ago with the first of these telephony application servers, which as previously stated, which was called Adjunct Communication Server or ACS, for short. Unfortunately, ACS suffered from a name problem, and I received many letters from different people pointing out that ACS was used by several other projects, including Al's Circuit Simulator. Clearly this was a problem.



Bayonne Internals

At the same time, the ACS architecture was showing its limitations. First, it was based on the idea of building a server directly bound to the telephony card, much the way XFree86 3 bound the X server to a given family of video hardware. This meant separate servers had to be compiled for each card family, and a lot of code was being duplicated needlessly.

I chose to rewrite the entire core server from scratch, and this was completed over a period of a few weeks. The first thing I did was create a concept for supporting plugins, using a somewhat different idea from how most people had done plugins in the past.

Typically, a plugin would be a small object file dynamically loaded with a known symbol name or structure that could be found easily once the loaded file was examined. Hence, one can use **dlopen** to open the plugin and **dlsym** to find a known symbol, thereby calling functions within the plugin.

I came up with a different method: I made the new server export its own dynamic symbols. The server then had a bunch of base classes with constructors that would initialize a registration structure. The plugins were written instead as C++ derived classes, where the base class was defined in the server and had static objects for these derived classes. When the plugin was loaded with dlopen, the constructors of these static objects would be invoked automatically, and the base class references to the server image resolved automatically. The base class held inside the server image would be invoked from the constructor, and it would register the plugin object. Hence, a single dlopen would both load the plugin and perform all initialization as a single operation.

Furthermore, things that were part of ACS got spun off as separate packages. This is when GNU ccScript and GNU ccAudio became separate class libraries, as these represented the already useful scripting engine and audio processing functionality found in ACS. In particular, we were looking at using the scripting language in other servers that would be part of GNUCOMM.

GNU ccAudio has proven to be a useful general-purpose audio processing library. It can be used to pregenerate single- and dual-frequency tones that can be played later from memory, and it can assemble audio from multiple input files into packed, fixed-sized frames with silence filling at the end, as is commonly needed for feeding DSPs (digital signal processing). This feature makes it a bit unique because other audio processing libraries typically do not concern themselves with these issues. Ideally, I would like to extend GNU ccAudio into a full, general-purpose audio processing framework that can also be used to provide host-based DSP-like processing.

So we had a new server, only it lacked a name. Since we wanted something distinct and unlikely to be used by someone else, we decided not to use yet another acronym. Instead, since the server was essentially a bridge between the computer and telephony world, it seemed natural to choose a bridge for a metaphor. But what bridge?

There of course is the Brooklyn Bridge. But overused and having bad connotations, it seemed a bad choice. Similarly, Golden Gate is extremely overused and, in any case, associated with IBM's Java initiative. Tacoma Narrows was a possibility, but considering it was famous for self-destructing, we thought we would leave that one alone, perhaps for a proprietary vendor in Washington.

There is a bridge not far from where we are in New Jersey: the Bayonne Bridge. Virtually nobody has heard of it, and in any case, the name is little-used.

Today and Tomorrow

Summer 2002 marks the introduction of the 1.0 release of GNU Bayonne. At present, GNU Bayonne is part of not only the GNU Project, but it has been packaged and distributed as a standard part of several GNU/Linux distributions, including GNU/Debian and Mandrake. In that we wish to make telephony application services universally available to free software developers, this is a positive development.

GNU Bayonne is widely used already in every part of the world. Users range from commercial carriers in Russia to state and federal government agencies in the US, and they include many enterprises that are looking for either specialized telephony-enabled web services or a platform for enterprise applications such as voice messaging.

GNU Bayonne does not exist alone but is part of a larger metaproject: GNUCOMM. The goal of GNUCOMM is to provide telephony services for both current and next-generation telephone networks using freely licensed software. These services can be defined as 1) services that interact with desktop users, such as address books that can dial phones and soft phone applications; 2) services for telephone switching, such as the IPSwitch GNU Softswitch Project and GNU oSIP proxy server; 3) services for gateways between current and next generation telephone networks, such as troll, and proxies between firewalled telephone networks such as Ogre; 4) real-time database transaction systems, such as preViking Infotel and BayonneDB; and 5) voice application services, such as those delivered through GNU Bayonne.

Even before GNU Bayonne 1.0 had been finalized, work started in late 2001 on a successor to GNU Bayonne. This successor attempts to simplify many of the architectural choices that were made early on in the project, with the hope of making it easier to adapt and integrate GNU Bayonne in new ways. The choice of design and much of the initial planning occurred during a two-day period late in 2001, while I was in London meeting with the people who developed the preViking telephony server. Some of these changes involved bringing the preViking Project directly into GNU Bayonne development.

One of the biggest challenges in the current GNU Bayonne server is creating telephony card plugins. These often involve the implementation of a complete state machine for each and every driver, and often the code is duplicated. GNU Bayonne 2 solves this by pushing the state machine into the core server and making it fully abstract through C++ class extension. This allows drivers to be simplified, but it also enables us to build multiple servers from a single code base.

Another key difference in GNU Bayonne 2 is much more direct support for carrier-grade Linux solutions. In particular, unlike its predecessor, this new server can take ports in and out of service on a live server, allowing for cards to be hot-plugged or hot-swapped. On a carrier-grade platform, the kernel will provide notification of changeover events, and application services can listen for and respond to these events. GNU Bayonne 2 is designed to support this concept of notification for management of resources it is controlling.

Finally, GNU Bayonne 2 is designed from the ground up to take advantage of XML in various ways. It uses a custom XML dialect as a configuration language. It also acts as a web service with both the ability to request XML content that describes the running state of GNU Bayonne services and the ability to support

XMLRPC. This fits into our vision for making telephony servers integrate with web services, representing part of how we envision the project going forward.

David Sugar has been involved in developing free software over the last 20 years and is the principal author of a number of packages in the GNU Project, including GNU Bayonne. David Sugar is a founder of Open Source Telecom and chairs the DotGNU steering committee.

Kernel Locking Techniques

Robert Love

Issue #100, August 2002

Robert explains the various locking primitives in the Linux kernel, why you need them and how kernel developers can use them to write safe code.

Proper locking can be tough—real tough. Improper locking can result in random crashes and other oddities. Poorly designed locking can result in code that is hard to read, performs poorly and makes your fellow kernel developers cringe. In this article, I explain why kernel code requires locking, provide general rules for proper kernel locking semantics and then outline the various locking primitives in the Linux kernel.

Why Do We Need Locking in the Kernel?

The fundamental issue surrounding locking is the need to provide synchronization in certain code paths in the kernel. These code paths, called critical sections, require some combination of concurrency or re-entrancy protection and proper ordering with respect to other events. The typical result without proper locking is called a race condition. Realize how even a simple i++ is dangerous if i is shared! Consider the case where one processor reads i, then another, then they both increment it, then they both write i back to memory. If i were originally 2, it should now be 4, but in fact it would be 3!

This is not to say that the only locking issues arise from SMP (symmetric multiprocessing). Interrupt handlers create locking issues, as does the new preemptible kernel, and any code can block (go to sleep). Of these, only SMP is considered true concurrency, i.e., only with SMP can two things actually occur at the exact same time. The other situations—interrupt handlers, preempt-kernel and blocking methods—provide pseudo concurrency as code is not actually executed concurrently, but separate code can mangle one another's data.

These critical regions require locking. The Linux kernel provides a family of locking primitives that developers can use to write safe and efficient code.

SMP Locks in a Uniprocessor Kernel

Whether or not you have an SMP machine, people who use your code may. Further, code that does not handle locking issues properly is typically not accepted into the Linux kernel. Finally, with a preemptible kernel even UP (uniprocessor) systems require proper locking. Thus, do not forget: you must implement locking.

Thankfully, Linus made the excellent design decision of keeping SMP and UP kernels distinct. This allows certain locks not to exist at all in a UP kernel. Different combinations of CONFIG_SMP and CONFIG_PREEMPT compile in varying lock support. It does not matter, however, to the developer: lock everything appropriately and all situations will be covered.

Atomic Operators

We cover atomic operators initially for two reasons. First, they are the simplest of the approaches to kernel synchronization and thus the easiest to understand and use. Second, the complex locking primitives are built off them. In this sense, they are the building blocks of the kernel's locks. Atomic operators are operations, like add and subtract, which perform in one uninterruptible operation. Consider the previous example of i++. If we could read i, increment it and write it back to memory in one uninterruptible operation, the race condition discussed above would not be an issue. Atomic operators provide these uninterruptible operations. Two types exist: methods that operate on integers and methods that operate on bits. The integer operations work like this:

```
atomic_t v;
atomic_set(&v, 5);  /* v = 5 (atomically) */
atomic_add(3, &v);  /* v = v + 3 (atomically) */
atomic_dec(&v);             /* v = v - 1 (atomically) */
printf("This will print 7: %d\n", atomic_read(&v));
```

They are simple. There are, however, little caveats to keep in mind when using atomics. First, you obviously cannot pass an atomic_t to anything but one of the atomic operators. Likewise, you cannot pass anything to an atomic operator except an atomic_t. Finally, because of the limitations of some architectures, do not expect atomic_t to have more than 24 usable bits. See the “Function Reference” Sidebar for a list of all atomic integer operations.

Function Reference

The next group of atomic methods is those that operate on individual bits. They are simpler than the integer methods because they work on the standard C data types. For example, consider void set_bit(int nr, void *addr). This function will atomically set to 1 the “nr-th” bit of the data pointed to by addr. The atomic bit operators are also listed in the “Function Reference” Sidebar.

Spinlocks

For anything more complicated than trivial examples like those above, a more complete locking solution is needed. The most common locking primitive in the kernel is the spinlock, defined in include/asm/spinlock.h and include/linux/spinlock.h. The spinlock is a very simple single-holder lock. If a process attempts to acquire a spinlock and it is unavailable, the process will keep trying (spinning) until it can acquire the lock. This simplicity creates a small and fast lock. The basic use of the spinlock is:

```
spinlock_t mr_lock = SPIN_LOCK_UNLOCKED;
unsigned long flags;
spin_lock_irqsave(&mr_lock, flags);
/* critical section ... */
spin_unlock_irqrestore(&mr_lock, flags);
```

The use of spin_lock_irqsave() will disable interrupts locally and provide the spinlock on SMP. This covers both interrupt and SMP concurrency issues. With a call to spin_unlock_irqrestore(), interrupts are restored to the state when the lock was acquired. With a UP kernel, the above code compiles to the same as:

```
unsigned long flags;
save_flags(flags);
cli();
/* critical section ... */
restore_flags(flags);
```

which will provide the needed interrupt concurrency protection without unneeded SMP protection. Another variant of the spinlock is spin_lock_irq(). This variant disables and re-enables interrupts unconditionally, in the same manner as cli() and sti(). For example:

```
spinlock_t mr_lock = SPIN_LOCK_UNLOCKED;
spin_lock_irq(&mr_lock);
/* critical section ... */
spin_unlock_irq(&mr_lock);
```

This code is only safe when you know that interrupts were not already disabled before the acquisition of the lock. As the kernel grows in size and kernel code paths become increasingly hard to predict, it is suggested you not use this version unless you really know what you are doing.

All of the above spinlocks assume the data you are protecting is accessed in both interrupt handlers and normal kernel code. If you know your data is unique to user-context kernel code (e.g., a system call), you can use the basic spin_lock() and spin_unlock() methods that acquire and release the specified lock without any interaction with interrupts.

A final variation of the spinlock is spin_lock_bh() that implements the standard spinlock as well as disables softirqs. This is needed when you have code outside a softirq that is also used inside a softirq. The corresponding unlock function is naturally spin_unlock_bh().

Note that spinlocks in Linux are not recursive as they may be in other operating systems. Most consider this a sane design decision as recursive spinlocks encourage poor code. This does imply, however, that you must be careful not to re-acquire a spinlock you already hold, or you will deadlock.

Spinlocks should be used to lock data in situations where the lock is not held for a long time—recall that a waiting process will spin, doing nothing, waiting for the lock. (See the “Rules” Sidebar for guidelines on what is considered a long time.) Thankfully, spinlocks can be used anywhere. You cannot, however, do anything that will sleep while holding a spinlock. For example, never call any function that touches user memory, kmalloc() with the GFP_KERNEL flag, any semaphore functions or any of the schedule functions while holding a spinlock. You have been warned.

If you need a lock that is safe to hold for longer periods of time, safe to sleep with or capable of allowing concurrency to do more than one process at a time, Linux provides the semaphore.

Semaphores

Semaphores in Linux are sleeping locks. Because they cause a task to sleep on contention, instead of spin, they are used in situations where the lock-held time may be long. Conversely, since they have the overhead of putting a task to sleep and subsequently waking it up, they should not be used where the lock-held time is short. Since they sleep, however, they can be used to synchronize user contexts whereas spinlocks cannot. In other words, it is safe to block while holding a semaphore.

In Linux, semaphores are represented by a structure, struct semaphore, which is defined in include/asm/semaphore.h. The structure contains a pointer to a wait queue and a usage count. The wait queue is a list of processes blocking on the semaphore. The usage count is the number of concurrently allowed holders. If it is negative, the semaphore is unavailable and the absolute value of the usage count is the number of processes blocked on the wait queue. The usage count is initialized at runtime via sema_init(), typically to 1 (in which case the semaphore is called a mutex).

Semaphores are manipulated via two methods: down (historically P) and up (historically V). The former attempts to acquire the semaphore and blocks if it fails. The later releases the semaphore, waking up any tasks blocked along the way.

Semaphore use is simple in Linux. To attempt to acquire a semaphore, call the down_interruptible() function. This function decrements the usage count of the semaphore. If the new value is less than zero, the calling process is added to the wait queue and blocked. If the new value is zero or greater, the process obtains the semaphore and the call returns 0. If a signal is received while blocking, the call returns -EINTR and the semaphore is not acquired.

The up() function, used to release a semaphore, increments the usage count. If the new value is greater than or equal to zero, one or more tasks on the wait queue will be woken up:

```
struct semaphore mr_sem;
sema_init(&mr_sem, 1);      /* usage count is 1 */
if (down_interruptible(&mr_sem))
  /* semaphore not acquired; received a signal ... */
/* critical region (semaphore acquired) ... */
up(&mr_sem);
```

The Linux kernel also provides the down() function, which differs in that it puts the calling task into an uninterruptible sleep. A signal received by a process blocked in uninterruptible sleep is ignored. Typically, developers want to use down_interruptible(). Finally, Linux provides the down_trylock() function, which attempts to acquire the given semaphore. If the call fails, down_trylock() will return nonzero instead of blocking.

Reader/Writer Locks

In addition to the standard spinlock and semaphore implementations, the Linux kernel provides reader/writer variants that divide lock usage into two groups: reading and writing. Since it is typically safe for multiple threads to read data concurrently, so long as nothing modifies the data, reader/writer locks allow multiple concurrent readers but only a single writer (with no concurrent readers). If your data access naturally divides into clear reading and writing patterns, especially with a greater amount of reading than writing, the reader/writer locks are often preferred.

The reader/writer spinlock is called an rwlock and is used similarly to the standard spinlock, with the exception of separate reader/writer locking:

```
rwlock_t mr_rwlock = RW_LOCK_UNLOCKED;
read_lock(&mr_rwlock);
/* critical section (read only) ... */
read_unlock(&mr_rwlock);
write_lock(&mr_rwlock);
```

```
/* critical section (read and write) ... */
write_unlock(&mr_rwlock);
```

Likewise, the reader/writer semaphore is called an rw_semaphore and use is identical to the standard semaphore, plus the explicit reader/writer locking:

```
struct rw_semaphore mr_rwsem;
init_rwsem(&mr_rwsem);
down_read(&mr_rwsem);
/* critical region (read only) ... */
up_read(&mr_rwsem);
down_write(&mr_rwsem);
/* critical region (read and write) ... */
up_write(&mr_rwsem);
```

Use of reader/writer locks, where appropriate, is an appreciable optimization. Note, however, that unlike other implementations reader locks cannot be automatically upgraded to the writer variant. Therefore, attempting to acquire exclusive access while holding reader access will deadlock. Typically, if you know you will need to write eventually, obtain the writer variant of the lock from the beginning. Otherwise, you will need to release the reader lock and re-acquire the lock as a writer. If the distinction between code that writes and reads is muddled such as this, it may be indicative that reader/writer locks are not the best choice.

Big-Reader Locks

Big-reader locks (brlocks), defined in include/linux/brlock.h, are a specialized form of reader/writer locks. Big-reader locks, designed by Red Hat's Ingo Molnar, provide a spinning lock that is very fast to acquire for reading but incredibly slow to acquire for writing. Therefore, they are ideal in situations where there are many readers and few writers.

While the behavior of brlocks is different from that of rwlocks, their usage is identical with the lone exception that brlocks are predefined in brlock_indices (see brlock.h):

```
br_read_lock(BR_MR_LOCK);
/* critical region (read only) ... */
br_read_unlock(BR_MR_LOCK);
```

Use of brlocks is currently confined to a few special cases. Due to the large penalty for exclusive write access, it should probably stay that way.

The Big Kernel Lock

Linux contains a global kernel lock, kernel_flag, that was originally introduced in kernel 2.0 as the only SMP lock. During 2.2 and 2.4, much work went into removing the global lock from the kernel and replacing it with finer-grained localized locks. Today, the global lock's use is minimal. It still exists, however, and developers need to be aware of it.

The global kernel lock is called the big kernel lock or BKL. It is a spinning lock that is recursive; therefore two consecutive requests for it will not deadlock the process (as they would for a spinlock). Further, a process can sleep and even enter the scheduler while holding the BKL. When a process holding the BKL enters the scheduler, the lock is dropped so other processes can obtain it. These attributes of the BKL helped ease the introduction of SMP during the 2.0 kernel series. Today, however, they should provide plenty of reason *not* to use the lock.

Use of the big kernel lock is simple. Call lock_kernel() to acquire the lock and unlock_kernel() to release it. The routine kernel_locked() will return nonzero if the lock is held, zero if not. For example:

```
lock_kernel();
/* critical region ... */
unlock_kernel();
```

Preemption Control

Starting with the 2.5 development kernel (and 2.4 with an available patch), the Linux kernel is fully preemptible. This feature allows processes to be preempted by higher-priority processes, even if the current process is running in the kernel. A preemptible kernel creates many of the synchronization issues of SMP. Thankfully, kernel preemption is synchronized by SMP locks, so most issues are solved automatically by writing SMP-safe code. A few new locking issues, however, are introduced. For example, a lock may not protect per-CPU data because it is implicitly locked (it is safe because it is unique to each CPU) but is needed with kernel preemption.

For these situations, preempt_disable() and the corresponding preempt_enable() have been introduced. These methods are nestable such that for each *n* preempt_disable() calls, preemption will not be re-enabled until the *n*th preempt_enable() call. See the “Function Reference” Sidebar for a complete list of preemption-related controls.

Conclusion

Both SMP reliability and scalability in the Linux kernel are improving rapidly. Since SMP was introduced in the 2.0 kernel, each successive kernel revision has improved on the previous by implementing new locking primitives and providing smarter locking semantics by revising locking rules and eliminating global locks in areas of high contention. This trend continues in the 2.5 kernel. The future will certainly hold better performance.

Kernel developers should do their part by writing code that implements smart, sane, proper locking with an eye to both scalability and reliability.

Rules

Robert Love (rml@tech9.net) is a Computer Science and Mathematics student at the University of Florida and a kernel engineer at MontaVista Software. Robert is the maintainer of the preemptible kernel and is involved in various other kernel development projects. He loves Jack Handy books and Less than Jake.

Archive Index Issue Table of Contents

Advanced search

At the Forge: Why Linux?

Reuven M. Lerner

Issue #100, August 2002

Reuven steps back this month and offers an overview of Linux's progression in business.

Break out the champagne! This month, *Linux Journal* is celebrating its 100th issue, and I've decided to take a break from my exploration of open-source web/database technologies to join the party.

There are plenty of good reasons for Linux users (and advocates of open-source software in general) to be happy. Despite the downturn in the high-tech economy, open-source software development continues at an extremely rapid pace. When *Linux Journal* was first published, few people had ever heard of the free operating system created by a Finnish student. Nowadays, many people have heard of Linux, even if they don't understand what it is or what it can do for them.

Indeed, while many of my clients know that I push for open-source solutions, they are always curious to know why I favor them and, more importantly, why choosing such solutions is in their interest as well. So at the risk of preaching to the converted, this month's column reviews some of the reasons why Linux is such an excellent platform for building server-side applications. I hope some of the ideas I put forth here will help you evangelize free software solutions with your own colleagues and clients in the years to come.

Cost and Stability

Hackers are interested in technologies and tools that teach new skills and perspectives. But in the real world, people are interested in getting their jobs done as quickly and cheaply as possible. Software is a means to an end, rather than an end in and of itself.

For this reason, I've found that the best way to sell people on open-source software is to say that it does more and costs less. Either one of these factors isn't enough by itself; it's easy to find expensive, high-quality software and useless to install cheap, poor-quality software. As consumers, my clients are always eager to get more for less, and free software appeals to them in this way.

When I pitch solutions to my clients, I begin by explaining that I'm offering them something they might have thought impossible: inexpensive software that does what they want, without crashing. When I explain to Windows users that I have yet to see a Linux system crash in over six years of running dozens of systems, they are shocked and incredulous. When I tell them that this software is freely available on the Internet, they find it even harder to believe.

My clients often wonder who is supporting the software and what happens if things go wrong. They are relieved to hear that not only can I offer them the support they need but that they can look for support elsewhere if they don't approve of my work. This, of course, contrasts sharply with the attitude and restrictions that many consulting firms impose on software installations. The open-source approach is thus friendlier to consumers than the traditional software model, reducing costs and encouraging competition.

Of course, not all free software is of high quality, and not all consultants really know what they're doing. The community development process can produce excellent results, but that doesn't mean everything released on the Internet is guaranteed to be safe and stable. Indeed, it's clear that many programs, including some popular ones, were uploaded without undergoing any testing. Programs like these give the entire Open Source community a bad name and often do more harm than good. Several times per year, clients call me in to fix a program they have downloaded that worked fine at first, but eventually proved itself to be insecure, unstable or full of bugs.

Fixing Bugs

Even if you find that your server depends on a bug-ridden, insecure, open-source application, all is not lost. That's because the nature of free software ensures that you can modify it to suit your needs or fix it when problems arise. In this way, shared-source licenses, which allow users to view the source code but not to modify or fix it, miss the point. Buying a house or a car entitles you to fix it on your own; why should software be any different?

True, the shared-source license does mean that more people will look over the code, so security and stability problems will be identified and fixed more quickly. But being able to read the source code isn't nearly as important as being able to improve it. Moreover, folding these improvements back into the community version means that everyone else will benefit from your adjustments and be able to make further improvements. Thus, contributing to the community process is in the interest of everyone who uses open-source software; it's not simply a nice thing to do.

Because I tend to use mature tools such as Linux, Apache, Perl and Python, it's relatively rare for me to find bugs in the software I download. But several times per year, I will discover a problem or limitation in the software I use. Having access to the source code guarantees that I can get up and running as quickly as possible, and it also means that others will not have to suffer through the bugs that I've fixed.

It's ironic that I can still use this argument today, given that a similar problem with printer drivers was what drove Richard Stallman to found the Free Software Foundation, whose GNU Project has been crucial to the success of Linux and free software. It's also amazing to discover how quickly we get used to having the source available and to being able to inspect or modify every part of our computer systems.

Along the same lines, Linux systems tend to come with “batteries included”, to borrow a phrase from the Python world. I recently began work on a project that will be deployed on Solaris, and I soon remembered how much richer and better-stocked a typical Linux distribution is when compared with a standard Solaris installation. True, I can spend half a day downloading and installing gcc, Perl, Python and the rest. But after years in which gcc was available on every machine I ran, it felt like I had been thrown back into the Dark Ages of UNIX.

No Secrets

Engineers are notoriously bad at keeping secrets, as Scott Adams has occasionally pointed out in his *Dilbert* comics. Indeed, one of the things that attracts me to open-source software is the fact that there are no secrets. Clients hire me because they want to save themselves time or because they lack expertise in a particular area, not because they are forced to do so. For this reason, I tend to think of myself as analogous to a lawyer or accountant, both of whom offer advice and documents based on freely available information.

This “no secrets” philosophy tends to work well with my clients, including those who aren't at all interested in knowing how the software works. They know they can ask me questions, and I'll give them the best answer I can, without having to hide behind marketing hype, mandatory updates or double-talk. My technical clients, of course, enjoy knowing they can dive into the code or read the documentation; the only thing stopping them from knowing what I know is time and experience.

My nonprofit clients, who are in many ways the perfect audience for open-source software, are often excited by the possibility of using such tools. In particular, I have found that educational institutions like the idea of sharing information and community involvement, in software as in other spheres of life. Telling them they can both save money and participate in a community of like-minded people is a powerful combination. Moreover, nonprofits typically have little incentive to keep changes within the company, meaning that they can more easily participate in the Open Source community.

High-Quality Toolkits

When I first began to write this column for *Linux Journal*, most server-side web applications were handwritten CGI programs. A huge number of web sites still use such programs. But as the Web has become more sophisticated, people have demanded toolkits that make it easier to develop high-quality, scalable web applications in a short amount of time. It shouldn't come as any surprise that many proprietary software companies have come to fill this void. It is shocking, however, to find out how much money they want for their software—sold on the condition that their consultants are hired to customize it, followed by a mandatory service contract.

Luckily, the open-source world has responded. A number of open-source toolkits can be used for creating sophisticated server-side applications. Zope, as we have seen in recent months, is a fantastic (if complicated) application server, making it possible to create web applications that connect to databases and other information sources. Next month, we'll begin to look at OpenACS, designed to make on-line community systems easy to build and modify. Furthermore, such environments as mod_perl, Mason and the numerous Java- and XML-related tools sponsored by the Apache Software Foundation increasingly mean that locating the right tool can be as difficult as installing and using it.

But as wonderful as these toolkits are, we must remember that not everyone will be won over. My harshest lesson on this front came last year when a potential client decided against hiring me to create a simple content management system for producing a product catalog for the Web. I was told that my bid came in at $800,000 less than my closest competitor. However, because I was using open-source software and the competition was a well-known name in the world of content management, I lost out. (That client has had a round of layoffs and quarterly losses since then, and their web site still appears to be managed by hand, so at least I won a moral victory of sorts.)

We should also remember that not every player in the open-source sphere can be trusted to follow through on their promises to the community. Many open-source advocates were surprised and disappointed when Lutris pulled the plug on its open-source Enhydra Enterprise Java application server last year, turning it into a proprietary product. Luckily, there are alternatives; not only has the GPL-licensed JBoss application server dramatically grown in popularity over the last year, but Sun recently made it clear that nonprofit, open-source J2EE implementations will be able to receive official certification in the coming months. This should help to reduce further the stigma that some businesses associate with open-source software.

But even if you suffer setbacks, don't be fooled: as IBM, HP and even Sun now acknowledge, Linux and open-source software are powerful, stable and should be taken seriously. “World domination” hasn't yet arrived, but brand-name recognition, financial realities and admiration from academics and commercial entities alike are helping us move ahead.

Happy Birthday!

I'll conclude this column with a salute to the hardworking staff that makes this magazine possible. I read *Linux Journal* for nearly a year before starting to write this column, and I continue to read it when it comes in the mail every month. The articles consistently reflect the diversity, sophistication and interests of some of the most creative software developers on the planet.

The fact that *Linux Journal* has now reached 100 issues should prove, beyond a shadow of a doubt, that Linux and open-source software are here to stay. I hope to write a similar article when the 200th anniversary issue comes around.

email: reuven@lerner.co.il

Reuven M. Lerner is a consultant specializing in web/database applications and open-source software. His book, Core Perl, was published in January 2002 by Prentice Hall. Reuven lives in Modi'in, Israel, with his wife and daughter.

Cooking with Linux: Strike up the Band and Celebrate!

Marcel Gagné

Issue #100, August 2002

Marcel introduces two media players, XMMS and KDE's Media Player, and shows how to dress them up with skins.

It is good to see you again François, *mon ami*. I trust that you took good care of the restaurant while I was away on my whirlwind tour of Europe, *non*? Well, thank you for asking, *mon ami*. It was indeed, a fine journey. *Oui*, the food was great as well, but no restaurant is blessed with such a waiter as yourself.

I have seen cities of lights and cities of music—London, Salzburg, Rome, Florence and many others. Then, of course, there was the ultimate city of lights, *Paris*. It seems somehow fitting that it is in this atmosphere of celebration that I had to prepare the menu for *Linux Journal*'s special 100th anniversary issue. *Mais oui*, François, what better way to celebrate than with music and light? Quickly before our guests arrive, head down to the cellar and bring up the 1989 Pessac-Léognan.

But our guests are already here. François. *Vite*! Please, *mes amis*, sit and make yourselves comfortable. Today's menu of fine Linux cuisine is a bit of an occasion. *Linux Journal* is 100 issues old—a fine vintage indeed. Have a look at the first item on the menu, a little something called XMMS. Basically a standard Linux media player, XMMS is much more than a music player. Properly used, it is a spectacular light show as well. It supports Ogg Vorbis, MP3 and WAV formats. With the right extensions you also can use it to play RealAudio.

Every major Linux distribution comes with XMMS, so you don't have to go far to find it. If it isn't already part of the installation, then have a look on your distribution CD-ROM. If all else fails, go to the source at www.xmms.org for the latest and greatest.

Building XMMS is easy and follows familiar steps. The current version is based on GTK 1.2.2 or better, and you may need the OpenGL or Mesa libraries if you want to use the OpenGL plugin (more on that later). Starting with the source, here is what we do:

```
tar -xzvf xmms-1.2.7.tar.gz
cd xmms-1.2.7
./configure
make
su -c "make install"
```

To start the program, type **xmms &** and press the Enter key. If this is the first time you start XMMS, you'll see something that looks like the amplifier on your home stereo system (Figure 1).

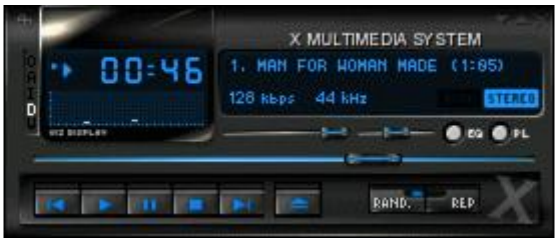


Figure 1. Your Basic XMMS

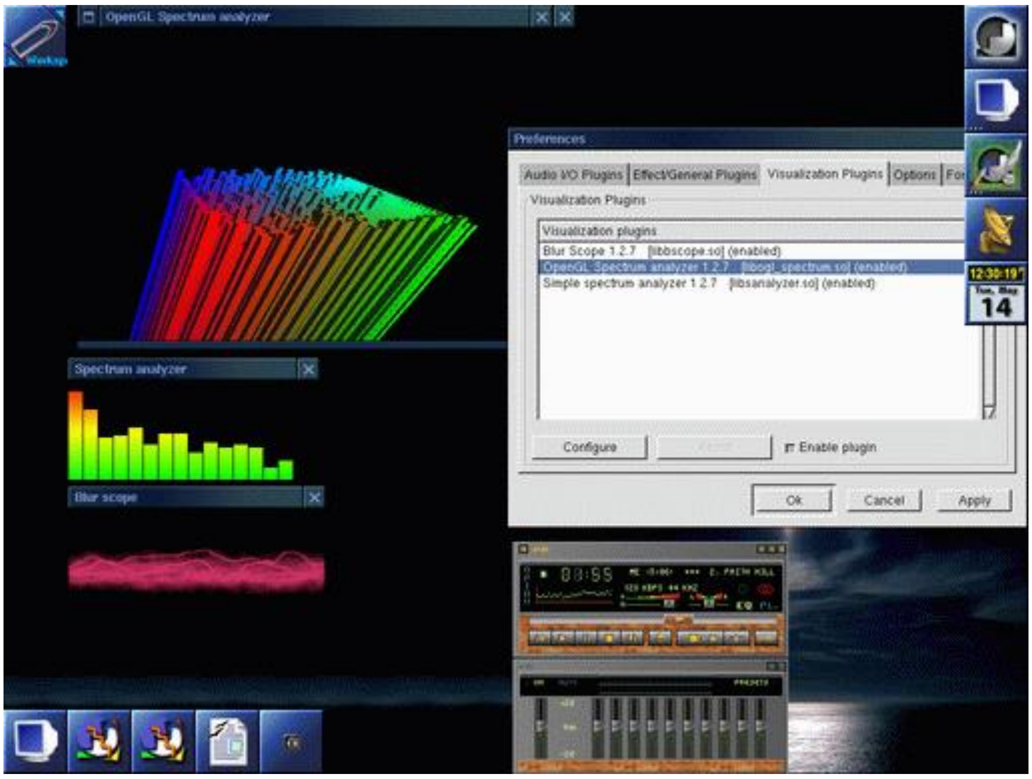
That is only the first module. Look at the buttons on the right of the amplifier. You'll see one labeled EQ (the equalizer) and PL (the playlist). Clicking these buttons will bring up two additional modules for your stereo system. Because each of the three modules can be moved about individually on the screen, you may find yourself re-adjusting their position more often than you care to. The easy way to solve this is by right clicking on the amp module, choosing Options from the menu and then choosing Easy Move. There's a Ctrl-E keyboard shortcut as well.

XMMS has extensive plugin support for input, output and visualization. To get at these, use the Preferences menu (the shortcut is Ctrl-P). A new window will pop up offering you tabs for various runtime options, fonts and so on. This is also where you find the control for the various audio I/O, special effects and visualization plugins. If you find you are having any problems with sound when you first start up XMMS, this is the place to look. Look under the audio section and check the output plugin. Having built my XMMS from scratch, I had to set the output plugin to OSS Driver.

I could spend a great deal of time talking about the various options, but instead, I invite you to check out the various options on your own. In terms of visualization plugins, this is where the light show begins. You'll notice that there are things like Spectrum Analyzer and Blur Scope. Earlier on, I mentioned the OpenGL Spectrum Analyzer, another cool plugin that provides colorful 3-D visuals to accompany your music—you can even launch that one full screen; sit back and enjoy the show.

One of my favorite features of XMMS (and there are quite a few) is its *skinability*. Using skins, I can change XMMS' look from its default black metal face to something more classic, like cherry wood or a refined brushed aluminum. Using the Ctrl-S shortcut brings up the Skin Browser, which you can also select through the right-click menu. Of course, if you just finished installing XMMS, you probably won't see anything there. You need to get yourself some skins. For that, head back to the XMMS web site and click Skins on the menu. I guarantee you won't be getting bored anytime soon. There are literally tons of skins available, including the aforementioned cherry-wood finish.

So, how do you install these skins? All of the skins on the web site are in tar.gz format. Find one that appeals to you, download it and save it to your $HOME/.xmms/Skins directory. You don't need to extract the file—just save it to the directory. If I turn out to be wrong, and you do get bored with the list on www.xmms.org, you'll be happy to know that XMMS supports Winamp skins as well. A visit to www.winamp.com should keep you more than busy. Just click Skins off the menu and have fun. In Figure 2, you'll see my classic cherry-wood XMMS system with various light-show plugins running.



```
Visualization plugins
Blur Scope 1.2.7   [libbscope.so] (enabled)
OpenGL Spectrum analyzer 1.2.7   [libogl_spectrum.so] (enabled)
Simple spectrum analyzer 1.2.7   [libsanalyzer.so] (enabled)
```

Figure 2. The XMMS Visualization Light Show

I'm going to leave XMMS behind on this topic of skins because the next application does skins in a great way as well. If you are running KDE, you have a great little program called Media Player. You can access the program either by looking under the K menu, choosing Multimedia, then clicking on the KDE Media Player, or you can type **noatun &** at the command line. The problem is that when you fire it up for the first time, it tends to look a little boring, as in Figure 3.

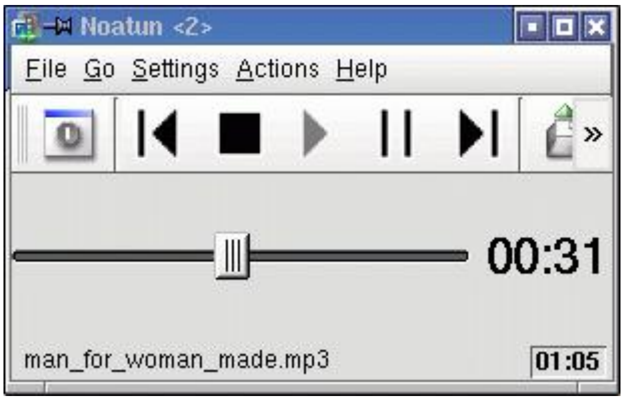


Figure 3. The KDE Media Player (Default Skin)

Don't let that disappoint you. That is the default skin, named “Excellent”. Click on Settings followed by Configure Noatun. From the pop-up menu that appears, choose Plugins, which will then give you a tabbed menu. Under Interfaces, you'll see four options for player styles. The skinnable styles are K-Jofol and Kaiman. In both cases, you can find additional skins on the KDE-Look web site at www.kde-look.org. Actually, *mes amis*, this is a great site to visit for numerous ways to dress up your KDE environment, from Noatun skins to desktop themes to replacement icon sets.

Start by clicking off the “Excellent” interface and clicking on the K-Jofol interface. You'll see the menubar at the left change. It now shows a K-Jofol Skins option. The same would happen with the Kaiman interface. If you click on this menu option, you'll see a drop-down list on the right with a preview of the various installed skins. For Noatun, skins live under the $HOME/.kde/share/apps/ noatun/skins directory.

To add new skins, switch to that directory, extract the skins you downloaded from www.kde-look.org, and you have just added a new skin. There is a catch, however. Unlike XMMS, the skins must be in PNG format and must be extracted into the skins directory. You can use skins you find on other sites, but you need to convert the JPG images to PNG. Luckily, this is extremely easy with the **convert** command (part of the ImageMagick package):

```
convert image.bmp image.jpg
```

That information is for KDE 2.x only. With KDE 3.x, you'll find that things are a little different. In the $HOME/.kde/share/apps/noatun/skins directory, you will find two other directories called winamp and kaiman. If they don't already exist, create them. Then, in the winamp directory, extract any of the XMMS skins that you find appealing. For instance, let's go back to my cherry-wood example:

```
   cd $HOME/.kde/share/apps/noatun/skins/winamp
   tar -xzvf /path_to/cherrywood.tar.gz
```

Now, when you click Configure Noatun, you should see this new skin under the Winskin list on the preferences menu.

When I told you about XMMS, I mentioned the variety of cool plugins you could use for visualization. KDE's Media Player, Noatun, has these as well. Among my favorites are Tyler and Blur Scope. Try some of these out for yourself, but be warned. The one called Madness is *truly* madness. Don't try it unless you are looking for a glimpse of temporary insanity. You have been warned! For a sample of various plugins in action, have a look at Figure 4.

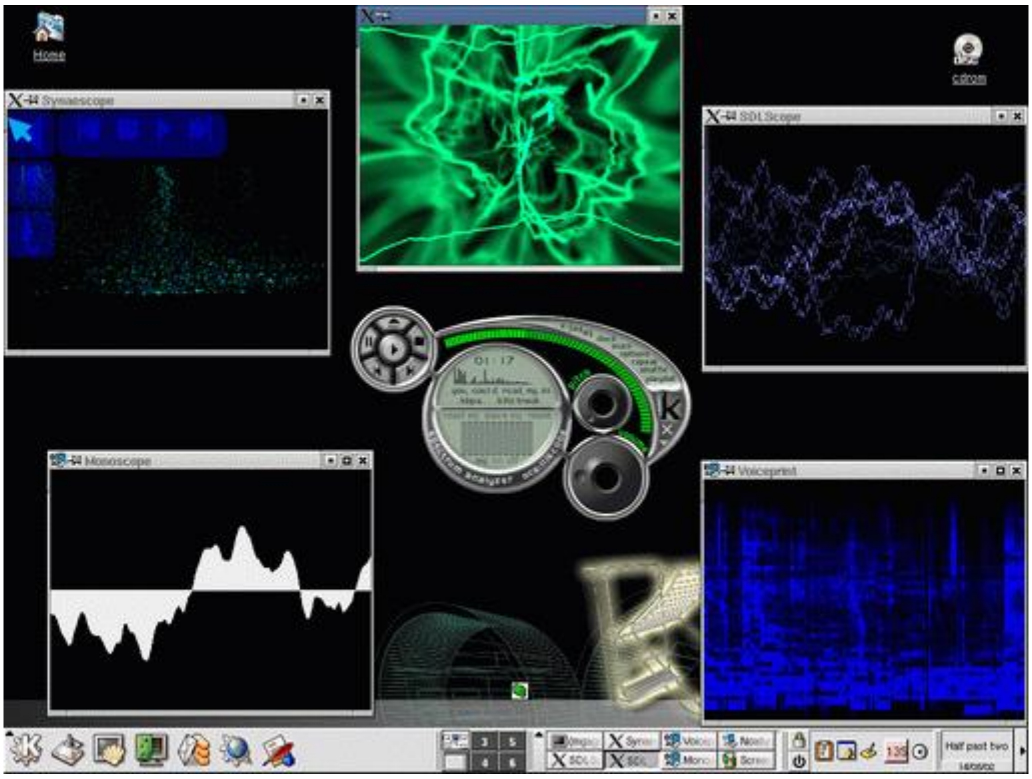


Figure 4. The KDE Noatun Light Show

Turn up that volume control, *mes amis*, or should I say “Crank it!”? François (who has managed to arrange a cameo in this very issue, see page 25) will refill your glasses. Tonight, the cellar is open and you may have anything you desire! Let the music play and the lights dance. It is time to celebrate!

Until next month. *A votre santé*! *Bon appétit*!

Resources

Marcel Gagné lives in Mississauga, Ontario. He is the author of Linux System Administration: A User's Guide (ISBN 0-201-71934-7), published by Addison-Wesley (and is currently at work on his next book). He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his web site at www.marcelgagne.com.

Archive Index Issue Table of Contents

Advanced search

Paranoid Penguin: Using iptables for Local Security

Mick Bauer

Issue #100, August 2002

Mick explains how to use the owner match extension for iptables to prevent local users from violating your network's security protocols.

Most of us think of iptables strictly as a firewall tool for keeping remote attackers at bay. But did you know it also can be used to keep local users in line? The experimental match extension owner adds new iptables options that can be used to help prevent local users from sending packets through other local users' network processes.

For example, suppose one of root's cron jobs uses Stunnel to send files to a remote rsync process. While that tunnel is open, any local user also may use it to access the remote rsync server. **iptables** can help you prevent such sponging; this month's column shows how.

The Problem

Tunneling utilities comprise one of the most important new categories of security tools at our disposal. They allow us to wrap insecure services, such as Telnet, IMAP and POP3, in encrypted virtual “tunnels”, transparently and effectively. I've written at length in these pages about the Secure Shell and its powerful port-forwarding capabilities; Stunnel and SSLWrap are similar free tools that can be used for this purpose under Linux.

But what happens when you set up such a tunnel on a multi-user system? What's to stop unauthorized local users from sending their own traffic through the tunnel? Until recently, practically nothing. Since most tunneling utilities work by creating a new local listener (e.g., localhost:992) for the near side of the tunnel, and since normally any local user can connect to a local listening-port, it's usually up to the server application at the other end of the tunnel to authenticate users.

For example, suppose I use Stunnel to create a secure sockets layer (SSL) tunnel from my local system “crueller” to the remote system “strudel”, over which I'm going to run Telnet. (Never mind that this sort of transaction is simpler with SSH; maybe I don't want SSH installed locally for some reason.) On the remote host, which is already running the Telnet dæmon (via inetd) on TCP port 23, I run Stunnel in dæmon mode with this command:

```
stunnel -d 992 -r localhost:23 -p \
/etc/stunnel/strudel.pem
```

On the local host, I'll run Stunnel in client mode, also listening on the local port TCP 992 but forwarding connections to TCP port 992 on strudel:

```
stunnel -c -d 992 -r strudel:992
```

If you've never used Stunnel before and these two commands mean nothing to you, don't worry. The important thing to understand is that to use this example tunnel to Telnet securely from crueller to strudel, I'll use this command on crueller:

```
telnet localhost 992
```

At this point, I'll be prompted for a user name and password by strudel, and unlike with normal Telnet, my login credentials will be encrypted by Stunnel rather than transmitted over the network as clear text. (The remote Stunnel process will decrypt the packets and hand them to strudel's local Telnet process. This happens for the *entire* Telnet transaction, not just the authentication part; Stunnel acts as a middleman for both parties during the entire transaction, a middleman who neither knows nor cares what he's tunneling as long as it's TCP.)

So far so good; I've got encryption, which I didn't have without Stunnel, and I've got a modest amount of authentication by virtue of Telnet itself. The problem is that *any* user on crueller can Telnet to the local Stunnel listener on TCP 992 and try to log in to strudel. Maybe I'm worried about someone guessing my strudel password and maybe I'm not; but how to stop them from sending *any* packets down the tunnel to begin with? With iptables and its new owner match extension, that's how.

Stunnel Who?

The Tool

iptables' owner match extension adds four match criteria to the iptables command:

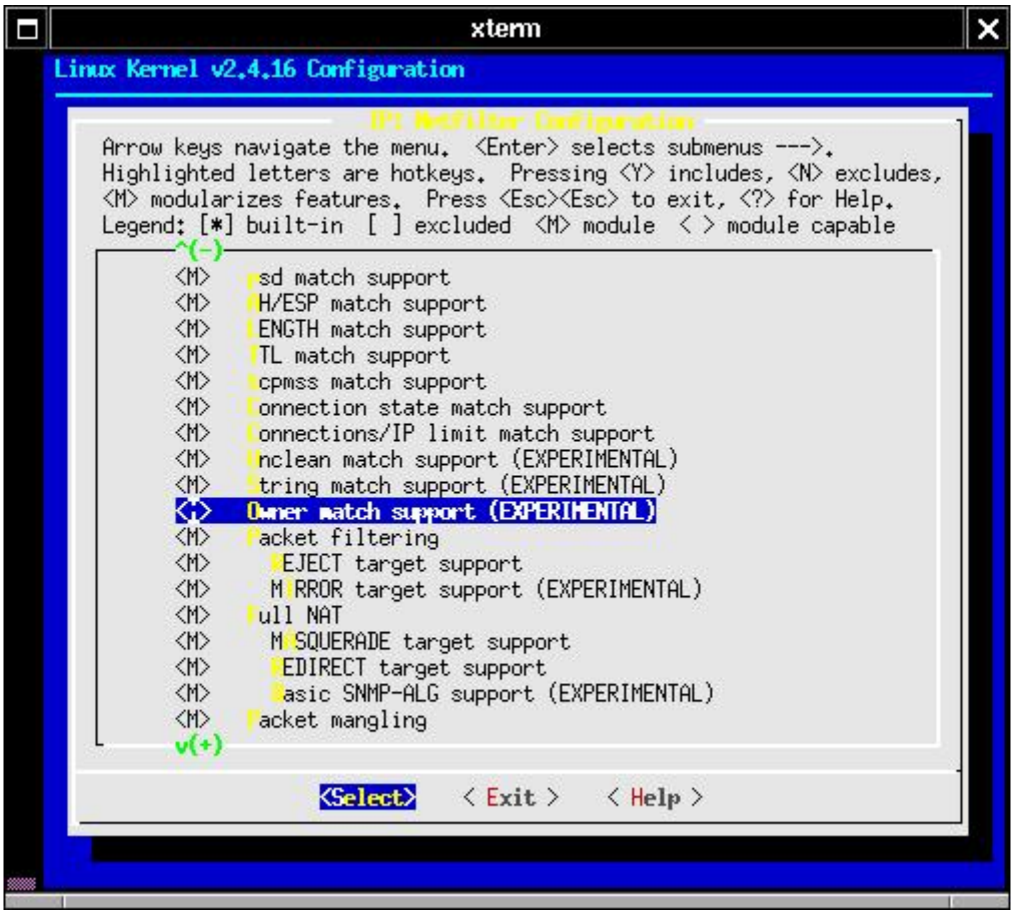
- —uid-owner *UID*: matches packets generated by a process whose user ID is *UID*.
- —gid-owner *GID*: matches packets generated by a process whose group ID is *GID*.
- —pid-owner *PID*: matches packets generated by a process whose process ID is *PID*.
- —sid-owner *SID*: matches packets generated by a process whose session ID is *SID*.

Of these four, the first two are the most useful for our purposes here.

The owner match extension isn't necessarily included in your distribution's stock kernel; it's considered an experimental feature (by the Linux kernel team, not necessarily by the iptables team), so you may need to compile it yourself. Its source code, however, *is* part of the standard 2.4 kernel codebase, so this is done easily with any recent version of your distribution's (2.4.x) kernel source package.

When recompiling your kernel, you'll need to set several things explicitly. First, under Code maturity level options, select “Prompt for development and/or incomplete code/drivers”.

Next, in addition to the other network protocols and features you customarily select in Networking options, make sure to select “Network Packet Filtering”. This will enable the subgroup IP: Netfilter Configuration, shown in Figure 1. You may compile these options either into the kernel (by selecting them with an asterisk) or as modules (with an M), but most people compile them as modules because all are seldom used at one time.



```
xterm
Linux Kernel v2.4.16 Configuration
                          IP: Netfilter Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
        ^(-)
        <M>    sd match support
        <M>    H/ESP match support
        <M>    ENGTH match support
        <M>    TL match support
        <M>    cpmss match support
        <M>    onnection state match support
        <M>    onnections/IP limit match support
        <M>    nclean match support (EXPERIMENTAL)
        <M>    tring match support (EXPERIMENTAL)
        < >    Owner match support (EXPERIMENTAL)
        <M>    acket filtering
        <M>      EJECT target support
        <M>     M RROR target support (EXPERIMENTAL)
        <M>    ull NAT
        <M>     M SQUERADE target support
        <M>      EDIRECT target support
        <M>      asic SNMP-ALG support (EXPERIMENTAL)
        <M>    acket mangling
        v(+)

                    <Select>    < Exit >    < Help >
```

Figure 1. Compiling a Kernel with owner Match Extension Support

Naturally you can select as many of the Netfilter modules as you like. They don't take up much disk space, and if compiled as modules they needn't be loaded unless necessary. The one we're most concerned with right now, though, is owner Match Support.

The rest of the procedure for compiling and installing the Linux kernel and its modules is well documented elsewhere (notably in the kernel source's own README file). Once you've compiled, installed and rebooted with your kernel, you can use your shiny new owner module, which will be named ipt_owner.

To load this module, use the modprobe command:

```
modprobe ipt_owner
```

In practice you'll probably want to load your iptables rules from a startup script in /etc/inet.d. If so, make sure you add the above modprobe line to the beginning of this script (i.e., above any iptables commands that use owner matches).

Note: neither Bastille-Linux's automated firewall configuration functionality nor SuSE Linux's SuSEfirewall scripts support owner matching without major hacking. This should hardly be surprising; they and other simple packet-filter rule generators are intended primarily for low-impact internet protection, not for the advanced control of local user access. For the latter, you need to write your own iptables rules.

Let's return to our example Stunnel client, crueller. Suppose crueller's kernel has been compiled with the ipt_owner module. You've loaded this module with modprobe and, for the time being, iptables isn't configured, i.e., nothing's being filtered yet.

Suppose further that you wish to restrict use of the Telnet-over-Stunnel socket we considered at the beginning of the article to root only. (You may recall we set up a Stunnel listener on crueller at TCP port 992, which encrypts and forwards packets to the same TCP port on strudel.)

If crueller isn't a firewall, we may be able to get away with an accept-by-default policy for the OUTPUT chain. On firewalls, all chains should have a drop-by-default or reject-by-default policy, but single-homed (single-network-interface) bastion hosts may sometimes have a more permissive stance on outbound traffic. If this is the case on crueller, then we need only one filtering rule to achieve the desired restriction:

```
  iptables -A OUTPUT -p tcp --dport 992 -d localhost \
  -m owner ! --uid-owner root -j REJECT
```

Let's dissect that command line one field at a time:

- -A OUTPUT: tells iptables we want to add a rule at the end of the chain OUTPUT. Since owner matches apply only to packets originating locally, and since outbound traffic is handled in the OUTPUT chain, this is the only chain in which you can use owner matches.
- -p tcp: tells iptables to match only TCP packets and to load iptables' TCP options.
- —dport 992: this TCP-specific option tells iptables to match only TCP packets destined for port 992.
- -d localhost: tells iptables to match packets destined for the localhost (i.e., the loopback interface 127.0.0.1).
- -m owner: tells iptables to load the owner match extension.
- ! --uid-owner root: tells iptables to match only packets not created by processes owned by root.
- -j REJECT: tells iptables to reject packets that meet all match expressions in this line.

In summary, this rule tells the kernel (via iptables) to drop packets sent to the local TCP port 992 unless they're sent by one of root's processes.

Suppose now that crueller has the more cautious default OUTPUT policy of DROP rather than ACCEPT. A drop-by-default policy is preferable on most iptables installations; the Principle of Least Privilege is one of the most important concepts in information security (i.e., “that which is not explicitly permitted must be denied”).

Now, however, we'll need a longer OUTPUT chain. Starting again with an empty chain, first we'll need to tell iptables to pass packets belonging to sessions it has already accepted:

```
iptables -I OUTPUT 1 -m state --state \
ESTABLISHED,NEW -j ACCEPT
```

The -state match extension provides iptables with crucial state-tracking abilities, allowing iptables to evaluate packets in relation to actual sessions and data streams. Aside from the desirability of this intelligence for its own sake, it also drastically reduces the number of rules you need to specify in order to accommodate a single transaction. Without state tracking, you'd need two rules rather than one to allow, for example, an outbound Telnet transaction; one each in the OUTPUT and INPUT chains. This is why the above rule should nearly always be used at the top of any chain whose default policy is DROP.

Next, we need to allow Stunnel itself to connect to strudel:

```
stunnel -A OUTPUT -p tcp -dport 992 -d strudel \
-j ACCEPT
```

This command appends a new rule to the bottom of the OUTPUT chain that permits outbound connections to TCP port 992 on strudel.

Finally, we enter a command similar to the one in the accept-by-default example, but this one is for the target ACCEPT rather than REJECT and for the absence of the negating exclamation point before the --uid-owner option:

```
iptables -A OUTPUT -p tcp --dport 992 -d localhost \
-m owner --uid-owner root -j ACCEPT
```

Remote Users and Tunnel Ports

Let's look at one more example. **rsync** is a powerful file-transfer utility that can perform differential file transfers. It can compare a remote file with a local copy and download only those parts that differ. **rsync** can be used in conjunction with SSH or, you guessed it, with Stunnel.

Suppose you've got a cron job on crueller that uses rsync to compare the file stuff.txt on strudel with a local copy and downloads any differences. Suppose further that stuff.txt contains some sensitive stuff, so you use Stunnel to encrypt these transfers. But only the local administrators, all of whom belong to the group “wheel”, need to control the script or use the tunnel.

On strudel, rsync is running in dæmon mode, having been configured to share a module (virtual volume) named attic. Assuming /etc/rsyncd.conf is properly configured (the specifics of which are beyond this article's scope), the command to run rsync in dæmon mode is simply:

```
rsync --daemon
```

In addition, strudel also has a Stunnel listener on TCP port 273 that decrypts and forwards traffic to the rsync process (which is itself listening on TCP port 873). The command to run Stunnel this way on strudel would be:

```
stunnel -d 273 -r localhost:873 -p /etc/stunnel/
    strudel.pem
```

On crueller, a corresponding client-mode Stunnel listener would be invoked like this:

```
stunnel -c -d 273 -r strudel:273
```

Okay, we now have a tunnel set up whereby packets sent to TCP port 273 on crueller will be encrypted and sent to TCP port 273 on strudel, where they'll be decrypted and forwarded to strudel's local rsync process on TCP 873.

In the absence of iptables rules, if the ordinary user plebian on crueller tries to use the tunnel, he or she will succeed:

```
rsync --port=273 -v localhost::attic/stuff.txt .
stuff.txt
wrote 508 bytes  read 575 bytes  2166.00 bytes/sec
total size is 48188  speedup is 44.49
```

Unless, that is, we add an iptables rule on crueller that restricts local use of the rsync tunnel to members of the group wheel:

```
iptables -A OUTPUT -p tcp -d localhost --dport 272 \
-m owner ! --gid-owner wheel -j REJECT
```

Now, plebian's attempt to pilfer the new stuff.txt file will fail:

```
rsync --port=273 -v localhost::attic/stuff.txt .
rsync: failed to connect to localhost:
    Connection refused
rsync error: error in socket IO (code 10)
    at clientserver.c(97)
```

But if wheel group member admin7 tries to connect, this will succeed:

```
rsync --port=272 -v localhost::chumly/stuff.txt .
stuff.txt
wrote 508 bytes  read 575 bytes  2166.00 bytes/sec
total size is 48188  speedup is 44.49
```

Hopefully, you noticed that this presumes a default allow policy. If OUTPUT instead uses a default drop policy, we'd need a rule in the OUTPUT chain allowing an outbound connection to TCP 273 on strudel. The OUTPUT chain also would need to begin with an allow established/related sessions rule. Since both these rules would resemble strongly those in the previous example, I won't bother showing them here.

Miscellaneous Notes on owner Matching and Stunnel

As you can see, the uses of --uid-owner and --gid-owner are pretty straightforward. One thing I haven't mentioned yet is that both options accept names, as I've shown in the examples, or numeric IDs.

Another issue I've dodged is TCP Wrappers-style access controls. On any system that uses TCP Wrappers (or whose stunnel binary was compiled with support for libwrapper), you must add appropriate entries to /etc/hosts.allow for Stunnel to work properly, whether you run Stunnel in client mode or dæmon mode on that host. This is a good thing; rather than being one more thing capable of preventing Stunnel from working, you should think of it as another layer of your security onion.

Finally, I'm leaving it to you to tinker with --pid-owner and --sid-owner. I will give you a hint, though. Many dæmons write their parent PID in a predictable place on startup, that is, /var/run/sshd.pid. By reading such a PID file into a variable in your iptables startup script, you can match packets originating from a specific process. Good luck!

Resources

Mick Bauer (mick@visi.com) is a network security consultant for Upstream Solutions, Inc., based in Minneapolis, Minnesota. He is the author of the upcoming O'Reilly book Building Secure Servers With Linux, composer of the “Network Engineering Polka” and a proud parent (of children).

Internet Abuse

David A. Bandel

Issue #100, August 2002

Help stop spam, create tests for students and watch your packets.

I had an interesting phone conversation about a month ago. It went something like this (abbreviated for space): “Hey David, I have problem with a client's mail server [Caldera eServer 2.3]. It's been running great since I installed it two years ago, but a power failure caused it to shut down. Now, the system comes up, but the Ethernet interfaces won't come up.” Me: “Well, let's bring them up manually and see why they don't come up at bootup. Any problems with fsck?” Friend: “None. I've tried running ifconfig, but it won't configure the interfaces.” Me: “Run **lsmod**, if the drivers aren't installed. Let's do that first, then humor me and run **ifconfig** exactly as I give it to you.” Friend: “Modules are installed; ifconfig segfaults.” Me: “Segfaults? (With alarm bells going off in my head.) Let's replace the ifconfig RPM in case it was damaged when the system crashed (as I'm thinking, ifconfig is one of the least likely apps to go south). You'll have to use **--force** to replace the ifconfig package.” Friend: “It appears ifconfig can't be replaced, even with --force.” Me: “Please run **lsattr ifconfig** and tell me what you see.” Friend: “I see an 'i' to the left of the name.” Me (with *Star Trek*'s “whoop, Intruder Alert” playing in my mind): “Humor me again. Run **locate ifconfig**.” Friend: “/sbin/ifconfig; /dev/sdg/.azgub/backup/ifconfig, /usr/man/man8/ifconfig.8.gz.” Me: “Well, you just found a rootkit hidden in /dev. Your client has been broken into. By the way, have any of the security patches been applied to that server since installation?” (No answer, but I assume not.)

It's a month since I sent a quote to fix his security problems and install a firewall (among other services). The intruder is still in this system, the Ethernet cards are in promiscuous mode, and the client seems oblivious to the dangers (says he's changed his passwords, so he's taken precautions—right). Who's inside? Why? Has this system with a fairly large pipe been used to break into other systems and/or act as a zombie to perform DDOS attacks? Some folks should not be allowed to remain connected to the Internet. Is he alone? Not hardly. My servers are pounded daily, my bandwidth being eaten by virii, automated attacks, etc. And, I'm paying for unnecessary bandwidth because of it. There should be a law.

SmtpRC sourceforge.net/projects/smtprc

Need to find out if you (or someone on your network) are contributing to the spam problem? This tool is all you need to help stop the spam problems on your network. Requires: libpthread, glibc.

GTK-Agenda brufal.kleenux.org/proyectos.shtml

This is an excellent start on a nice GTK agenda. It holds names, phone numbers and e-mail addresses in a PostgreSQL database. Currently available only in Spanish, changing the labels, etc., to English, German, whatever, should be fairly simple—although Internationalization would be the way to go. You can send e-mails from within this application, but you'll need an SMTP dæmon running. Should be an easily extended application for your database needs. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, libpq, libssl, libcrypto, libcrypt, libresolv, libnsl, glibc.

x86info sourceforge.net/projects/x86info

If you need to find out more than you can get from /proc/cpuinfo, this may be what you need. It reads directly from the CPU registers, so it can provide more information than what most of us will need or even understand. Requires: glibc.

ILIAS www.ilias.uni-koeln.de/ios/index-e.html

If you're looking for a way to give courses and tests over the Internet, ILIAS is another tool that will allow you to do this. Data on students, test scores, etc., are all maintained in a MySQL database. Requires: Apache w/PHP and MySQL, MySQL server, GD, zlib, freetype, libjpeg, ImageMagick, zip, unzip.

pktstat www.itee.uq.edu.au/~leonard/personal/software/#pktstat

There are a lot of utilities out there for watching packets on your network, but this one is slightly different. It looks at the percentage of bandwidth use for packets. This little jewel can tell you very quickly that one of your abusers running Kazaa is gobbling 99.7% of the available bandwidth. Requires: libm, glibc.

DNSMan www.xsta.cc/dnsman

This web application is probably one of the easiest ways to maintain your BIND zone files, even easier than Webmin's BIND module. Requirements are small, but it does mean running a web server on your DNS platform. I'll be watching this one as it develops, as the author has a number of interesting items on his to-do list. Requires: web server (Apache) capable of running CGI scripts, Perl, BIND 8 or 9.

ntop www.ntop.org

This month's pick from three years ago wavered between two great programs: ntop and stickerbook, a great program for children (mine love it, but ntop won). This is now a *much* improved version of ntop. It has developed from a simple ncurses utility to an advanced web client using HTTPS or HTTP for connections, with graphing in gdgraph (optional). **ntop** now bears little resemblance to its former self and is easier to use and read. If you need a top-like utility for your network, you need this. Requires: libmysqlclient, libcrypt, libm, libssl, libpthread, libresolv, libnsl, libdl, libgdbm, libz, glibc. Until next month.

David A. Bandel (david@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is coauthor of Que Special Edition: Using Caldera OpenLinux.

Where No Penguin Has Gone Before

Rick Lehrbaum

Issue #100, August 2002

Tux arrives at the North Pole, making the spread of embedded Linux truly global. And, more cool toys for your entertainment needs.

Is embedded Linux a success? Judging from the steady stream of new devices coming to fruition, it certainly is. In this month's column, we learn of an incursion by Tux into the arctic region of the globe—perhaps a search for Santa? Then we take a brief look at a handful of other cool gadgets that run on embedded Linux.

Tux Searches for Santa

Thanks to a project of the US National Oceanic and Atmospheric Administration (NOAA), a webcam has been installed for the first time at the North Pole—one that runs on embedded Linux, no less. The device was installed on April 28, 2002 and is logging four images a day—take a look at NOAA's web site: www.arctic.noaa.gov/gallery_np.html.

Figure 1. North Pole NetCam

The webcam's images will track the North Pole snow cover, weather conditions and the status of NOAA's Pacific Marine Environmental Laboratory's North Pole instrumentation, according to James Overland, head of NOAA's North Pole Project.

Although the webcam is capable of transmitting camera video at the rate of one image per second, NOAA keeps the device powered off most of the time in order to conserve its solar-charged battery power. Four times a day the webcam wakes up, places a phone call via the Iridium low Earth orbit satellite system and transfers the latest images at 2,400 baud to NOAA's servers using PPP.

The webcam being used is the NetCam from StarDot Technologies, a 20-person company located in Buena Park, California. StarDot created their first webcam design in 1996, developed a custom digital camera control chip in 1997 and went on to sell over 65,000 webcams based on that design (manufactured by a Taiwanese partner and imported to the US by StarDot).

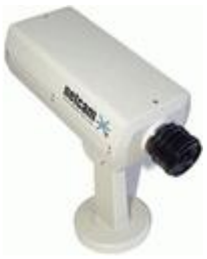


Figure 2. StarDot Technologies NetCam

The NetCam's embedded computer, which runs μClinux (www.uclinux.org), is based on a 54MHz Motorola ColdFire microprocessor equipped with 8MB of RAM memory and 2MB of nonvolatile Flash memory. I/O ports include a pair of RS-232 serial ports, a 10Mbps Ethernet port, an I2C serial bus and four bits of digital I/O. The device implements a web server function based on Boa (www.boa.org), which allows it to be accessed remotely from a web browser anywhere in the world, provided it's connected to the Internet via either Ethernet, modem or wireless. (See www.stardot-tech.com.)

More Cool Gadgets with Embedded Linux Inside

snom 100 VoIP Phone: The snom VoIP phone supports a wide range of open telephony standards including SIP, H.323/H.450 and Asterisk, along with data protocols like HTTP, TAPI and LDAP. The company says the snom 100's IP-based voice quality is indistinguishable from that of “normal” ISDN phones and, in addition, the 128 × 64 pixel graphical LCD display and web-browser interface make calling, remote management and configuration even easier. In addition to its IP functionality, the device also provides typical telephone features like call hold, call waiting, call forward/transfer, call divert, caller line identification, and so on. Inside the snom 100, there's a 50MHz Motorola MPC855T PowerQUICC Integrated Communications processor equipped with 16MB of RAM. The system's embedded OS, based on a Linux 2.4.18 kernel, was developed in-house, and snom created their own drivers and libraries for the use of the 128 × 64 pixel graphical LCD and even wrote their own embedded HTTP server. (See www.snomag.de/snom100_en.htm.)

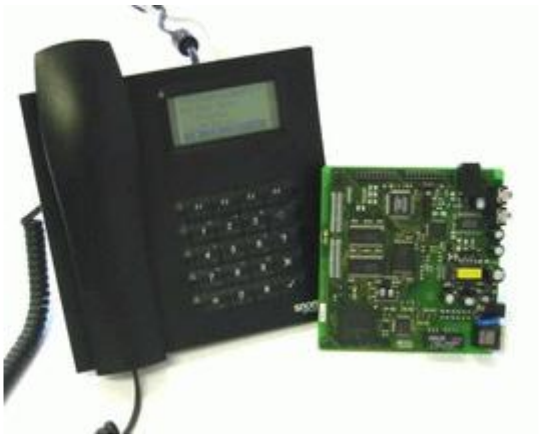


Figure 3. snom Phone

hippo Internet Phone: The hippo Internet Phone looks a lot like an ordinary desktop telephone, but it can make calls over either Ethernet LAN or normal telephone lines (via PPP, through the user's local ISP). Instead of being carried as analog signals over normal phone lines, the calls are carried digitally over the Internet. At the other end, the calls can be received by either another Internet Phone (PC-based or dedicated Internet Phones) or by a standard PSTN (Public Switched Telephone Network) telephone anywhere in the world, including cell phones. Obviously, substantial reductions in phone bills are possible using this approach because no long-distance phone service is necessary. The hippo Internet Phone has a 4 × 20 character LCD screen, a 12-button telephone keypad, six function keys and a telephone handset with hook/switch, ringer and normal-sounding dial and call progress tones that simulate “normal” telephone operation. Inside hippo's Internet Phone is an embedded computer based on a 48MHz Motorola MPC850/823 PowerPC system-on-chip processor that is equipped with 16MB of DRAM and also runs an embedded Linux OS derived from MontaVista's Hard Hat Linux. (See www.hippoinc.com.)

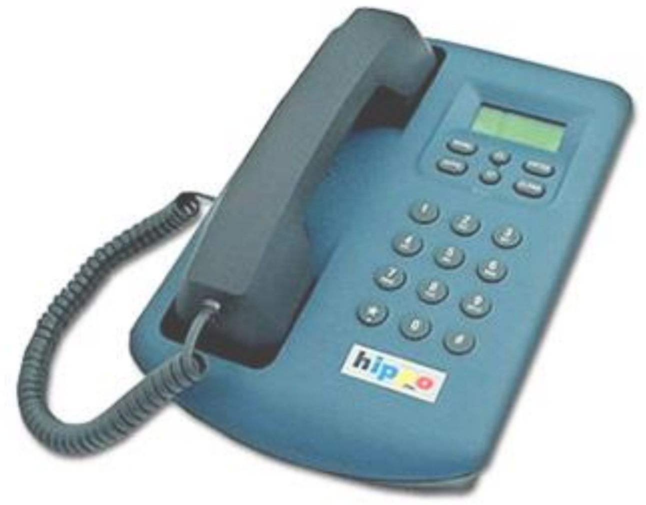


Figure 4. hippo Internet Phone

SONICblue Rio Central: This high-fidelity home stereo component that stores up to 650 CDs' worth of music on its built-in 40GB hard drive is said to be “as simple to use as a CD player, but as smart as a PC”--a Linux-based PC, that is. The system is intended to function as a standalone audio system, making it easy to store your music (“you load your CDs, the device does the rest”) and create an essentially unlimited number of customized playlists. Thereafter, a large display, intuitive interface and advanced search features are meant to make it easy to find the music you want instantly. Plus, the system optionally offers suggestions based on your tracked listening habits. Inside, there's a 206MHz Intel StrongARM processor along with 16MB of system RAM, running an embedded Linux OS derived by the system's developers from Debian/ARM sources. In addition to its standalone operation, the Rio Central can also be used as the basis of a broader Rio experience, serving one or more companion Rio Receiver thin clients (which also run embedded Linux) via HomePNA (phone line networking) or Ethernet. It also can download files to Rio portable MP3 players via USB. The device is broadband-ready (via an external USB-to-Ethernet adapter), includes a built-in 56Kbit/sec modem (for users without broadband access) and provides a 10Mbps HomePNA connection. (See www.sonicblue.com.)

Figure 5. SONICblue Rio Central

Cyclades Device Server: The Cyclades TS100 is a powerful yet highly compact device server used to connect various serial devices to a TCP/IP network. Typical applications include industrial automation and control, out-of-band network management, retail automation and connecting various types of serially interfaced devices to networks. The device provides interface ports for both 10/100Mb Ethernet and RS-232/RS-485 serial lines, allowing it to integrate legacy instruments and systems having serial interfaces to broadband networks using TCP/IP. Unlike its competitors, which are typically based on proprietary software models, the TS100's built-in embedded Linux OS and other open-source software make it easy to customize operation of the TS100. Although not much larger than a deck of playing cards (2.8" × 3.4" × 1.2"), the TS100 contains not just one but two microprocessors. This is possible thanks to the use of Motorola's MPC855T “PowerQUICC Integrated Communications Processor”. The MPC855T is a dual-core system-on-chip processor that includes a PowerPC core processor (running at 50MHz) plus a separate RISC engine specifically designed to off-load communication tasks. Memory resources consist of 16MB of SDRAM, plus a 4MB Flash disk from which the firmware is uncompressed and loaded into a RAM disk at boot time. The device's embedded Linux OS is based on a 2.2.14 kernel, along with a variety of open-source utilities including the GoAhead web server (for web-based setup and management of the device), Portslave, OpenSSH 3.1, crontab, BusyBox, net-tools, rsyncm and others. Cylades started from MontaVista's Hard Hat Linux 1.2 and added their own in-house customization. (See www.cyclades.com.)

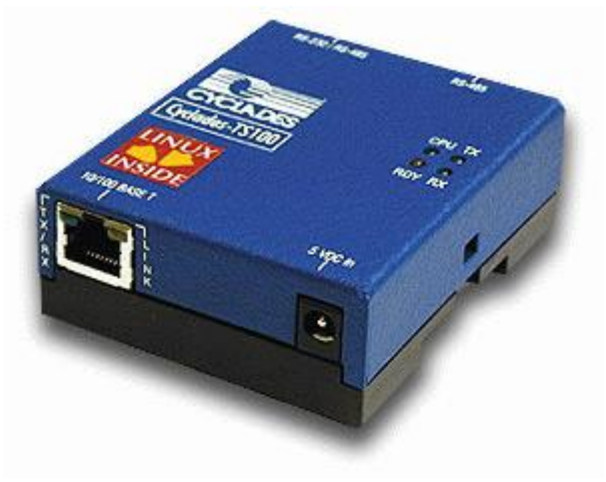


Figure 6. Cyclades TS100

Linksys Wireless Presentation Gateway: Using this embedded Linux-powered device, wireless mobile PC users can project presentations and other data using VGA-equipped devices such as multimedia projectors, monitors and LCD panels without having to physically wire each PC to the projector. The WPG11 lets users take turns controlling the display instantly. Users are each assigned unique key codes for access and control of the device. Because cabling and setup time is eliminated, WiFi-enabled users can take turns controlling the presentation display simply by typing in the pre-assigned key codes. In real time, participants can offer instant visual input that follows the verbal discussion. Perhaps it even solves the inevitable incompatibilities between the laptops and the projectors? (See www.linksys.com.)

Figure 7. Linksys Wireless Presentation Gateway

And More All the Time

Keep abreast of all the latest cool gadgets that have Linux embedded inside by visiting LinuxDevices.com's on-line “Embedded Linux Cool Devices Quick Reference Guide”: www.linuxdevices.com/articles/AT4936596231.html.

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com and DesktopLinux.com web sites. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in creating and launching the Embedded Linux Consortium.

Archive Index Issue Table of Contents

Advanced search

Linux for Suits: Scoring 100

Doc Searls

Issue #100, August 2002

Before the dot-com era, there was no such thing as a “Linux company”. Afterward it was hard to find a major company in computing that didn't run Linux or sell goods that ran on it.

The biggest word in today's *Wall Street Journal* is LINUX. Actually, LINUX is tied with UNIX, WINDOWS, CHECKBOOKS, ROCKS and DB2. All are in the headline of a full-page ad by IBM on page B3—the first page that accepts advertising—in the Marketplace section of the newspaper.

This afternoon (I'm writing this in late May) Hewlett-Packard is announcing a deal with Reuters to move that company's Market Data Systems to HP ProLiant servers running Linux. The deal spans three to five years and could exceed $200 million.

HP says it won that piece of business over IBM and Sun, both of which also are growing Linux providers. IBM has all but embarrassed itself with its public declarations of love for Linux, getting busted last year for scrawling Linux graffiti on public property. Sun, meanwhile, has been struggling to find a way to be pro-Linux without hurting its high-end Solaris business.

So it would understate the matter to say Linux is a hit with big systems OEMs, and the trend hardly stops there. A year ago Linux was big in only one obvious category: web servers. Now it's spreading out. HP says its latest sale “demonstrates the growing presence of Linux in the financial services market”, a market that used to be synonymous with highly proprietary software and hardware. HP pointedly adds, “This also places HP in a strategic position as the financial services market moves from Sun Solaris to Linux.”

Proprietary UNIX systems aren't the only ones threatened. Yes, Windows is still a monopoly, but for how long? Several years ago that would have been a ridiculous question, but now it's not. Microsoft seems to have declared war on

Linux and open-source software—a strategy that is backfiring terribly. Today's papers bring news that Microsoft reportedly has been trying to convince the US Department of Defense that open-source software poses both a threat to security and to the company's intellectual property. Meanwhile the DOD had a report prepared by MITRE Corp. that identified 249 uses of open-source systems and tools, including the Defense Intelligence Agency's web portal and network security software for the US Army and Air Force. MITRE itself maintains a library of open-source products, including mobile mesh networks and CVW (Collaborative Virtual Workspace). MITRE even maintains some of its open-source software on SourceForge.

Even where Linux isn't the operating system of choice, it has opened the door for other free and open forms of UNIX, such as BSD, which was adopted into Darwin, the open-source foundation of Apple's OS X. These events have increased vastly the population of the world's Linux-friendly PCs and servers. At a recent conference, I asked Steve Jobs if it was true that the company's new rackmounted X servers were expected to augment, rather than replace, existing UNIX systems. He said “yes”. After I asked him what other kinds of UNIX crops prevail in potential customers' server farms, he started his list with Linux.

His answer is not surprising. Here's a rundown of Google results for a variety of operating systems and related topics:

- Linux: 47,100,000
- Windows: 43,800,000
- UNIX: 12,200,000
- Open Source: 2,740,000
- Free Software: 1,760,000

This is a far cry from 1993, when Phil Hughes included me in an e-mail list that explored opportunities for a free-software magazine. After batting ideas around for a while, Phil suddenly announced that his little company, SSC, was going to start a magazine for Linux, the brainchild of a 21-year-old guy from Finland.

I thought Phil was nuts. But Phil is very instinctive about stuff other people don't see. A few years later, during a visit to my house, Phil showed me KDE running office applications that looked remarkably like what one saw on Windows. Yet everything he showed me was free and open. It blew my mind so much that I took him up on his offer to join the masthead. This was in early 1999, when the venture capital was still flowing like Niagara and Linux had an effect on investors that was something like Viagra. By the end of that year, three of the biggest IPOs in the history of American business were for companies strongly identified with Linux: Red Hat, Cobalt and Andover/VA

Linux. VA's IPO in December flew to over $300 per share on day one before settling at over $200. For a few months there was no shortage of Linux billionaires.

I recently purged my pile of business cards and took a picture before tossing the stack into recycling. They told an interesting story.

- Some companies were out of business. That was the case with Linux Laptops, Rebel.com, Loki, Eazel and OpenSales, which changed its name to Xelerate (was it one l or two?) before going away.
- Other companies, like Kerbango and Cobalt, were bought up by larger companies—in these cases 3Com and Sun. 3Com killed off Kerbango and its cool little radio, while Sun has reportedly done pretty well using Cobalt to hold up the low end of the company's server business. The Cobalt Qube remains a benchmark Linux server appliance.
- Others are shifting their focus away from Linux .VA Linux, which was once so identified with the L word that they paid seven figures for the Linux.com domain name, paid over $1 billion in stock for Andover and traded on Wall Street as LNUX, has changed their name to VA Software.
- Corel, after trying to create a kind of consumer/productivity software business around Linux, jettisoned Linux along with a lot of other ballast in an effort to stay afloat.
- Linuxcare is still alive, although without its founders, who are off making news with Sputnik, a new company in the mobile wireless network market.
- Other companies, like Boxx (not Boxx Technologies, currently operating in Austin, Texas), which made (or at least prototyped) a fancy Linux/Windows hybrid laptop, simply vanished.
- Two related market experiments (which I was very enthused about), CoSource and SourcExchange, quietly folded.
- There are success stories too. Linux-based TiVo is synonymous with the small but closely watched digital video recorder business. Borland, no longer burdened by the forgettable name Inprise (which was on several of my tossed business cards), is reportedly doing very well with Kylix, its cross-platform Linux development environment.
- SuSE shrank its operations in the US but remains the leading Linux distribution in Europe and is strong worldwide as well. Caldera is holding on pretty well. So is Turbolinux.

The big winner is a company with no cards in my pile, Red Hat, which still proudly flies the penguin flag and remains by far the leading Linux distribution. (We modestly point out that the name of the editor atop our first issue's masthead was none other than Red Hat founder Bob Young. Coincidence?)

It's easy to put down all the dot-com enthusiasm and to damn Linux with the failure of the whole dot-com, um, “model”. The phenomenon had its upside as well. It helped make Linux a household word and funded a variety of projects that thrive today. One example is SourceForge, which hosts over 40,000 projects and 430,000 registered users.

Here at *Linux Journal*, we've been through the hard times along with the rest of the surviving Linux companies. What's kept us going is the steady march of Linux toward what Linus Torvalds half-jokingly called world domination.

After 100 issues, *Linux Journal* has become the leading Linux how-to magazine for countless technologists—software and hardware companies, government organizations, medical and scientific institutions and third-world economies that need maximum productivity with minimal cost. We're in great shape, and so is the world Linux now dominates: operating systems.

Linux has finally done what UNIX devotees have wanted for decades: it has driven the OS to ubiquity. Today the only UNIX systems with any future are free, open and ready for improvement by anybody who wants to jump in and help. That's one heck of a success story.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal*.

Fair Use

Lawrence Rosen

Issue #100, August 2002

Trying to define fair use is stickier than you might think.

Occasionally I hear the “fair use” defense made to excuse someone's breach of someone else's copyright. Unfortunately, the word *fair* has colloquial meanings that are different from the legal meaning of the phrase *fair use*. Copyright law says you can't copy certain software even though it may be fair to have a backup copy at home. You can't create derivative works from certain software even though it may be fair to build whatever programs you want. You can't reverse-engineer certain software even though it may be fair to be able to fix defects, ensure security and interwork with your other software.

The law doesn't say that any licensing practice you find distasteful or that you morally oppose can be ignored if to do so would be fair. The constitutional foundation for the fair use doctrine was described by one court this way:

> The fundamental justification for the [fair use] privilege lies in the constitutional purpose in granting copyright protection in the first instance, to wit, “To Promote the Progress of Science and the Useful Arts.” [Const., Art. I, Sec. 8, Cl. 8] ... To serve that purpose, “Courts in passing upon particular claims of infringement must occasionally subordinate the copyright holder's interest in a maximum financial return to the greater public interest in the development of art, science and industry.” *Rosemont Enters. v. Random House, Inc.*, 366 F.2d 303 (2d Cir. 1966), *cert. denied*, 385 U.S. 1009 (1976).

Under this formulation, fair use is a privilege and not only a defense. Your right to make certain uses of copyright materials is guaranteed by the Constitution in the same sentence that allows an author to obtain a copyright on his or her works. The monopoly that copyright law confers is a limited one, limited not only as to duration but also limited as to the author's exclusive rights.

Congress codified the fair use doctrine in the Copyright Act, 17 U.S.C. § 107, to make it clear what forms of use are to be considered fair. The statute lists “purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research”.

Whenever you see the phrase “purposes such as” in a statute, be prepared for a debate. Congress obviously couldn't list all purposes, so what other purposes will pass muster? For example, copying your office word processing program onto your home computer because you're only using it in one location at a time is not one of the purposes listed in the statute, but can you argue that it is like one of those purposes? Is reverse-engineering to detect security flaws in software a form of criticism or research? Can a teacher make thousands of copies of a software program available for a course taught over the Internet? And as if that isn't enough vagueness for a statute, Congress then proceeded to list four factors to consider in determining whether a use is fair:

> 1. **The purpose and character of the use**, including whether such use is of a commercial nature or is for nonprofit educational purposes. News reporting, scholarly research and teaching are examples of favored fair uses of copyright material. Commercial use is not favored.
>
> 2. **The nature of the copyrighted work**. The law generally recognizes a greater need to disseminate factual works than works of fiction or fantasy. To the extent one must permit expressive language to be copied in order to assure dissemination of the underlying facts, copying may be more justified.
>
> 3. **The amount and substantiality of the portion used in relation to the copyrighted work as a whole**. Fair use is less appropriate if entire works, or the most valuable parts of them, are copied.
>
> 4. **The effect of the use upon the potential market for or value of the copyrighted work**. If copying a work will prevent the owner of the copyright from profiting from it, even if such profit is only potential, the copying is less justified. This factor may tip the balance in favor of fair use where there are no other known copies of the work in existence, where the copyright owner is unidentifiable, or where there is no ready market by which copies can be sold.

None of these factors is determinative standing alone. In evaluating them, a court must undertake a “sensitive balancing of interests”. *Financial Information, Inc. v. Moody's Inv. Serv.*, 751 F.2d 501 (2d Cir. 1984). So too, when you seek to infringe someone's copyright, you should perform your own “balancing of interests” analysis. Consider whether your use is similar enough to one of the purposes listed in the statute, and go through each of the factors, asking whether that factor balances in your favor. If you can't convince yourself that you pass the fair use test, don't infringe.

If you use open-source software such as Linux, fair use is generally not an issue. An open-source license safeguards the rights of anyone, anywhere, for any purpose whatsoever, to use, copy, modify and distribute (sell or give away) the software and to have the source code that makes those things possible. All uses are licensed by the copyright owner, so you don't need to defend your use with the fair use doctrine. This is yet another reason why free and open-source software is better than proprietary software. With proprietary software, be careful to have a valid fair use argument if you do anything not permitted by the license. With free and open-source software, enjoy your broad and comprehensive fair use rights.

Legal advice must be provided in the course of an attorney-client relationship specifically with reference to all the facts of a particular situation and the law of your jurisdiction. Even though an attorney wrote this article, the information in this article must not be relied upon as a substitute for obtaining specific legal advice from a licensed attorney.

email: lrosen@rosenlaw.com

Lawrence Rosen is an attorney in private practice, with offices in Los Altos and Ukiah, California (www.rosenlaw.com). He is also executive director and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (www.opensource.org).

ASA 2URS3 Rackmount 2U Server

Logan G. Harbaugh

Issue #100, August 2002

The 2URS3 offers high performance and top-of-the-line engineering at a reasonable price.

Many people come to Linux looking for an inexpensive server platform, but web sites, databases and other Linux-based application servers have ways of growing in size, complexity and importance that affect the company's bottom line. When this happens, it becomes desirable to upgrade the server hardware from that old 486 desktop system to a real server platform.

Servers come in a wide range of qualities and prices. At the low end are inexpensive systems with desktop cases and motherboards, and at the other are name-brand, custom-configured, redundant-everything rackmount boxes with five- or six-figure price tags. ASA Computers offers a multitude of servers in 1U (1.75"), 2U (3.5") and 4U rackmount configurations, as well as pedestal (desk-side) cases with up to four processors. The 2URS3, in a configuration intended to be a database server, offers high performance and top-of-the-line engineering at a reasonable price.

The 2U rackmount server I tested included an Intel SCB2 motherboard with on-board SCSI, dual 1.13GHz Pentium III processors, 2GB RAM, an ICP Vortex 64-bit/66MHz Ultra 160 RAID controller, one IBM 9GB Ultra 160 boot drive and four Seagate Cheetah 10,000RPM 18GB Ultra 160 drives in a hot-swappable RAID-5 configuration. It also included integrated dual 10/100 Ethernet interfaces, an integrated ATI Rage 8MB video controller, an Intel PRO/1000 Gigabit Ethernet NIC and a Hewlett-Packard 12/24GB DDS-3 DAT drive.

The keyboard and mouse (not included) can share a single PS/2 port via a Y-cable, or two USB ports also are available. There is no parallel port, and the one serial port has an RJ-45 connector rather than the standard DB-9. (There's only so much real estate available on a 2U case.) The standard warranty is one year for parts and three years for labor. On-site service is not currently available.

The motherboard supports up to six 64-bit/66MHz PCI cards, three full-height and three low-profile, with each set of three on a separate PCI bus. Two slots were used for the Intel Gigabit Ethernet NIC and the ICP RAID controller. The motherboard supports up to 6GB of PC133 interleaved RAM and one or two 1.0, 1.13, 1.26 or 1.4GHz Pentium III processors. The system uses a nicely engineered Intel SR2200 case that is capable of redundant 350 watt-power supplies, though only one was installed. All sheet metal edges are folded over to prevent sliced fingers; construction is solid and the case supports up to six hot-swap SCA drives. A floppy/CD combo drive (or a seventh hot-swap hard drive) can be added, and there is room for a tape drive as well (or a floppy or CD, if the seventh hard drive option is in use). A rackmount rail kit is also included.

The only quibble I have with the case is that the locking button for the slide-off top can be easily bent too far, which then keeps the top from locking closed. This won't be much of an issue when the case is mounted in a rack.

The system is a real screamer, fully capable of running with the fastest Intel-based systems around. Throughput on the Gigabit Ethernet adapter was over 900Mbps, and sustained file transfers reached speeds of over 30MB/sec using the RAID volume. This speed could be increased by adding spindles to the RAID. In a effort to benchmark the overall system performance, I used an Antara FlameThrower to generate requests against the web server, downloading graphics and sound files as well as text pages. The overall performance numbers were about 1.6 times as high as for a 1.4GHz single-processor Compaq server, indicating that SMP is working well and overall system performance is comparable to the best systems around.

The price quoted by ASA Computers for the system as configured is $5,542. Using Dell's web site, a comparable PowerEdge 2550 2U server came in at $7,911. The ASA Computers system is a well-engineered system with high-quality, name-brand components and a huge bang for the buck. So what don't you get for the price? Two things—name recognition and hand-holding.

ASA Computers is a relatively small company (compared to Compaq, Dell or HP) that has been around since 1989, a fact that has both pluses and minuses. ASA is much more likely to respond to special requests from customers and give those customers quick results. On the other hand, it can't offer the kind of service organization and sales support that the big companies can. For example, the system I received had no documentation other than the vendor documentation for the motherboard, chassis and RAID controller. A simple hardware build sheet didn't include any information on what software was installed, what packages were installed, how the RAID was configured or even what the root password was for the system.

For administrators who intend to install their own standard server software, or who are comfortable with rooting around in the depths of the system, this lack of documentation might not be a problem. When I called, the support technician at ASA was able to give me the information I needed immediately, with no time spent on hold. ASA does offer customized hardware/software configurations that could be specified to include documentation and software at additional cost. However, the sales support that you may expect from one of the big companies, which could help you select the OS, database software, appropriate storage devices, configuration details for the devices (what RAID works best for a given type of database?), configure the OS and database software, install it at your site and provide ongoing support, is not available. This is the kind of capability you pay the extra money for when you buy servers from IBM, HP, Compaq and other, larger companies.

If you can provide your own support, however, ASA offers a wide variety of equipment. Everything from 1U web servers to four-way Xeon servers with ten drive bays are available, all with best-of-breed components and all at great prices.

Product Information/The Good/The Bad

Logan G. Harbaugh (lharba@awwwsome.com) is a freelance writer specializing in networking. He has worked as an information technology manger and manager of systems integration and has been a networking consultant for more than 15 years. He has also written two books on networking.

Archive Index Issue Table of Contents

Advanced search

ImageStream's Rebel Router

Paul M. Holzmann

Issue #100, August 2002

Nothing on the market today can touch the Rebel Router from a cost/savings perspective.

The box arrives and the glee of a child at Christmas washes over me—a new toy to play with. Only this is no ordinary toy. This is a DS3 (T3) router that costs under $4,200 US. Three years ago I would have scoffed at the possibility of such a thing. I was used to the world of Nortel and Cisco, a world in which a router this powerful cost as much as a new car and, in some cases, a small house. It was a world in which RAM and card upgrades became necessary due to the growing number of routes on the Internet, and the aforementioned companies would charge in excess of $15,000 US for an upgrade that really didn't allow your router to do anything it wasn't doing the year before. And it was aworld in which good technical support meant signing a $10,000/year contract. Boy, the world has changed since then.

Anyhow, back to the box. Opening it, I find the new and improved Rebel Router from ImageStream. For those of you who don't know, ImageStream makes Linux-based routers that are capable of wire-speed throughput yet are incredibly flexible and inexpensive. I have been using theses routers for the past two years, both in my network and in customer networks, with great success. The new Rebel Router comes in a black rackmount case about the size of a Cisco 2500 (height: 1U, depth: 10.75"). The front is adorned with the ImageStream logo and a blue LED that really makes it stand out on a network rack. The back panel includes dual 10/100 Ethernet ports, an auxiliary console port, a couple of fans for cooling and two PCI card slots. In order for me to test this router for *Linux Journal*, I requested a configuration that I could use to replace my current network router. Because of those needs, my configuration includes a single DS3 card and a Quad T1 card.

Upon opening the case, I was struck by the neat layout. The case is segmented into three parts. The left side of the case contains the single power supply. The central section contains what is essentially a PC-based motherboard sans PS/2 and parallel ports. The right side contains the area for the serial cards. ImageStream sells cards for this router in the following capacities: single or quad 10/100 Ethernet; single, double, quad or octal port T1; and single DS3. ImageStream does sell Dual DS3 cards, but they don't recommend them for this router. Instead they recommend a platform with multiple CPUs for a multiple DS3 configuration. On a further note, under the recommended configurations they guarantee wire speed.

Now to the real point of this review: what can the Rebel Router do? One great feature of this router is its ability to bond multiple interfaces. For instance, you can bond two T1s to get a 3Mb channel as opposed to a fractional DS3. This can come in handy if you want to avoid the high local loop charges associated with most DS3s. As the marketing literature says, you can provide connectivity for up to 16 T1/E1s or one DS3/E3, but aside from the level of throughput, the most impressive part of this router is the routing capabilities. The Rebel Router uses GateD, originally developed by Merrit. These are the same people that developed the Radius standard, as well as many other technological advances used on the Internet today. GateD supports static, RIP, OSPF and, most importantly (for ISPs and large companies), BGP-4 routing.

For those unaware, BGP-4 is the standard used to route traffic across the Internet dynamically, and it is used in almost all configurations that are truly redundant (i.e., connections to more than one upstream provider). As the Internet has exploded in growth over the past few years, so have the number of routes in the routing tables. This has caused many ISPs to have to upgrade their RAM, which can be prohibitively expensive, as I mentioned above. ImageStream uses a relatively off-the-shelf, unbuffered RAM that is inexpensive when compared to the proprietary RAM sold by other router manufacturers. Combine this with the fact that these routers are slightly modified PCs using Intel processors, that you have your choice of the 2.2.14 or 2.4 Linux kernel, and you get an incredibly powerful router without the hefty price tag of something that is totally proprietary.

Some of the other wonderful features of this router are the result of its Linux-based nature. For example, If you are like me and wouldn't use Telnet on your servers to save your soul, then guess what? You don't have to on your routers either. All ImageStream routers allow you to turn off Telnet in favor of SSH. If you don't feel like learning a new firewall language or messing around with access lists, then you can use the ipchains that come with the current 3.2 distribution. The 2.4 kernel, and therefore iptables, are also available. In addition, if you truly hate to use menus to get things done, you can always drop to the shell and do anything that you would do in a normal Linux system.

Because all of the files for configuring your interfaces, routing, firewalling, etc., are in ASCII format, you can open them in vi or Pico and make your changes.

Other great features of the Rebel include the ability to see the traffic flowing through the router in real time. ImageStream routers come with a program called Stats that works just like the top command (also available from the command line), except that instead of seeing system resources you can see your interface-usage statistics. This feature helps a great deal when you're troubleshooting. Another great feature is the QOS system that uses Diffserv to allow you to limit bandwidth and shape parts of your network down to the single-IP level. This can become essential in a limited bandwidth environment if you need to give priority to certain types of traffic.

One of the new services that this router provides is IPSec using FreeS/WAN. This is something I had never played with prior to this router test, so I ended up getting on the phone with the support staff at ImageStream because I did not want to chance doing something wrong and taking my network down. I spoke with Josh, who was very knowledgeable and friendly and helped me set up a test VPN. We first set up a subnet-to-subnet VPN. This setup took about 15 minutes, as I was a newbie and wanted to understand everything about it. Once that was done we were able to set up additional VPN tunnel variations rapidly, including subnet-to-PC and PC-to-PC.

Over the span of the week that I tested the router, it performed flawlessly. Because I have used ImageStream routers before, I felt totally confident that it would perform without a hitch as the core of my network, and it didn't let me down. During that time period I added an additional T1 to the network, changed the BGP-4 AS number and added another block of addresses in the normal course of business. In addition, I tested the IPSec service and did DS3 throughput tests, all without a hitch.

Finally, the most important feature of these routers is the support. I have used ImageStream routers for almost two years now and have found that the biggest savings in the purchase of these routers is the support. ImageStream's support personnel are knowledgeable, friendly and go out of their way to ensure that you are happy with the product. The best part about dealing with ImageStream is that there are no support contracts. They provide 24/7 tech support and free software updates for the life of the product, and they warranty the hardware for a year. That being said, there is nothing on the market today that can touch the Rebel Router from a cost/savings perspective.

Information/The Good/The Bad

Paul Holzmann currently lives in Grand Rapids, Michigan where he is the technology manager for TAK Consultants, Inc., a local accounting, consulting and internet firm. In addition, he and his business partner, Thomas Korth, have recently started a new business called Worldwide Dialup (www.worldwidedialup.net) that provides global dialup internet access.

OmniCluster Technologies' SlotServer

Linda Hypes

Issue #100, August 2002

Note: this review is specific to the SlotServer 1000. OmniCluster's SlotServer 3000 has additional features and memory.

Wouldn't it be nice to have a complete, multiplatform network at your fingertips—one that fits right on your desk? That may not be the first item on everyone's wish list, but as a company that creates solutions for integrating different operating systems, it certainly ranked high on ours.

The biggest limiting factor to accomplishing this was obviously space, but with the help of OmniCluster Technologies, our dream became a reality. We first met OmniCluster at LinuxWorld 2001 in San Francisco, where they were demonstrating what looked like some type of network expansion board, but turned out to be a complete Linux blade server on a PCI card.

The SlotServer is a half-length industry standard PCI card that contains a fully compatible x86-based server blade with two network adapters. The standard bracket provides an integrated 100Mbps network interface. A second “Modular Network” is also provided as a peer network connection between the host system and each card. This network adapter is essentially two Gigabit Ethernet adapters set back to back in the same integrated circuit. One of these adapters is connected to the PCI bus of the SlotServer processor and the other connects to the PCI bottom-edge connector and hence to the “host” system. Multiple peer modular networks can then form a high-speed local private network with the host machine. This high-speed connection allows each SlotServer to operate diskless Windows or Linux from a “virtual partition” of the host machine's filesystem. This permits a RAID filesystem to be shared by many SlotServers and enables a provided utility, called Virtual Disk Manager, to create, replicate, remove and associate operating system images for all the SlotServers in a system. The SlotServer kit even includes one sample OS image.

The SlotServer can be loaded with Windows 2000, NT, XP, Linux and FreeBSD operating system environments and can run any application supported by those operating systems. It fits into any standard PCI slot, allowing you to have complex server clusters and multiple operating systems within a single PC base. The minimum of two network ports per SlotServer permits processing information in-line to or from the host.

We had previously looked at other blade servers from various vendors, but they were considerably more expensive and required an additional proprietary chassis, bumping the cost up even more. So, we decided to give the SlotServer a try.

We obtained our first SlotServer for use in our support organization. Our products run on Windows and provide file-sharing capabilities with any operating system that uses NFS (network filesystem), including UNIX or Linux, so we get support calls involving a wide range of operating systems. With the different variations of UNIX and Linux available, we would require a large number of servers in-house if we did support using traditional methods. The ability to use SlotServers to run these operating systems has allowed us to support them by using only a couple of mini-tower-based host systems. We have saved countless hours in building and rebuilding systems to different operating systems or specific versions of an operating system, enabling us to test and resolve issues more quickly for our customers.

The SlotServer provides the same benefits of multiple servers and operating systems within one system for our developers. Our products run on and are developed on the Windows operating system. However, since the products are used to share files between Windows and other operating systems, a minimum of two machines are needed for even a cursory test. Using the installed SlotServer, the developers have a complete test environment consisting of two different operating systems within a single machine. This allows them to perform functionality tests efficiently without needing external resources. It's also very helpful when testing precertified, unstable code. If a blue screen occurs on the SlotServer, their host machine will not be affected.

The SlotServer has also proven to be very beneficial for us at tradeshows. They're lightweight and easy to ship—we used to spend a small fortune just shipping the number of various machines we needed to demonstrate our products. With the ability to configure different operating systems on each SlotServer within one mini-tower system, we are able to demonstrate all of our products at a tradeshow using one mini-tower and two laptops.

We can also easily change our server configuration, for example, from Red Hat to SuSE or any of the many other operating systems supported on the

SlotServer. If we are demonstrating our DiskAccess NFS client or X-Win32 X server with Red Hat NFS server loaded on the SlotServer, and someone wants to see our products working with a SuSE NFS server, we can quickly comply with their request. The SlotServer allows you to boot up in different operating system images by way of OmniCluster's host-resident Virtual Disk Manager (VDM). We simply assign the SuSE boot image to the SlotServer and reboot it via VDM without affecting the host system or any of the other SlotServers installed within the host system. One operating system and its associated applications can be exchanged in a minute or so. With this ease of configuration, we can demonstrate our products within another company's server by simply installing the SlotServer in an available PCI slot and powering up their system.

While our use of the SlotServers has advantages from which others could benefit, the different applications that can share a mutual benefit with these devices is tremendous. Some of the more mainstream uses for the SlotServers include front-end servers for highly available web clusters, generic application mirroring for automatic failover and even a dedicated firewall solution (Check Point) that runs inside mission-critical servers. The use of industry-standard PCI as the SlotServers host interface means that blade systems made with SlotServer are not proprietary. We can use any available PCI system as the host and any of thousands of PCI peripheral cards can be mixed in our configurations as necessary.

Note: this review is specific to the SlotServer 1000. OmniCluster's SlotServer 3000 has additional features and memory. For additional information on both of these products, please visit www.omnicluster.com.

Product Information/The Good/The Bad

Specs

Linda Hypes is sales and marketing director at Shaffer Solutions for the AccessNFS Product Suite.

Benchmark's ValuSmart Tape 80

Cosimo Leipold

Issue #100, August 2002

A new generation half-height DLT drive.

Do you have mission-critical machines that need to be backed up and need a reasonably priced solution? The VS Tape 80 DLT backup drive may be the answer. As part of a new generation of half-height DLT drives, this device is geared toward users who require tape drives that will fit into an existing rackmount system. In other words, if you have little rackmount space to spare, this drive may very well suit your needs.

Benchmark (www.4benchmark.com) was kind enough to provide me with an external VS Tape 80 with an Adaptec 29160 card. The test machine consisted of a dual PIII with 800MHz processors, 7200RPM IDE disks and 512MB RAM. We used Red Hat 7.2 and BRU-Pro 2.0 as our backup solution.

Prior to receiving the drive, I was unsure whether corners had been cut in order to achieve a half-height form. This was, after all, the world's first half-height DLT. Fortunately, that doesn't appear to be the case. The drive itself feels like a quality product and also performs like one. It comes with three simple LEDs on the front: Drive Error, Ready and Clean Media. It also has an unload button that has a nice soft feel and, unlike many DLT drives, doesn't require you to flip a locking mechanism to eject or insert. My only concern was the sound the drive made when inserting or ejecting media. It was a loud, grainy sound that didn't sound like any other drives I'd heard before. That said, it was probably nothing serious, as BRU-Pro 2.0 performs checksums on the buffer level, and I never once found an error. If any of you are prone to taking naps during backups (as I am), it should be mentioned that the drive is incredibly quiet during operation, and the slight humming sound greatly facilitates sleep.

From a performance standpoint, this puppy won't scream like an LTO drive (which can reach 15-30MB/s for the HP Ultrium 230 models—see the review in the December 2001 issue of *LJ*, on-line at www.linuxjournal.com/article/5412).

However, it also costs less than a third of what the HP Ultrium LTO drives cost. According to Benchmark, the drive performs at 3MB/s native and 6MB/s compressed. BRU-Pro allows you to tweak the size of the block size it writes. I was unable to get any significant changes in speed by tweaking this value, though the 64K buffer seemed to offer the best performance. In the end, I was able to achieve writing speeds between 3.3 and 4.5MB/s, with an average speed of around 4MB/s. Restores were typically a little slower, averaging about 3.8MB/s. I called TOLIS Group, makers of BRU-Pro, because I knew they used these drives for some of their internal testing and training. They told me they've managed to push the drive to 5MB/s in their tests. I was unable to reach this speed but came close.

For the individual looking to back up more than 80GB, Benchmark has a product called the 640 Blade, which uses the same drive with eight tape cartridges that rotate in a carousel into the drive for a total capacity of 640GB (compressed). It's the first 2U DLT autoloader designed to fit in rack-optimized servers. At $4,000 US, the entire system costs less than a single HP Ultrium 230 upgrade kit for their SureStore line. For those looking for both speed and space, Benchmark makes the argument that at such low cost, several of their 640 Blades still cost less than a typical autoloader and offer more reliable backups.

The primary backup system we use here is an HP SureStore 2/20 with two HP Ultrium 230 drives. The HP SureStore 2/20 costs about $21,000 US and can push up to 15MB/s native (or 1.08TB/hour) on each drive. For about the same cost one could purchase five 640 Blades. From a purely mathematical view, it would appear that the Blades would be slower. Each Blade can push about 10.8GB/hour (native). With five you would top out at around 54GB/hour.

However, I think the Blades, depending on your configuration, might be a faster solution. How? The catch is that the Ultrium 230s are so fast that we've had trouble pushing data to them as fast as they can write. In our environment we have a handful of fast machines that can push data at about 8-11MB/s to the Ultriums and about 20 clients on a slower network that can only push data at about 3-4MB/s. At any given point in time we can back up two clients, one to each drive (we do not use multiplexing because of the serious performance hit it causes on restores). The bottleneck in our case isn't our network but rather the slower, older client machines. If instead of the Ultrium drives we had five 640 Blades, we could write five clients at any given point in time, making our effective backup rate faster than the Ultriums'.

It also should be noted that with the 640 Blades you would have multiple drives and multiple backup solutions, so if one did unexpectedly get destroyed in a freak accident (such as spilling coffee on it during one of your naps), you wouldn't be left high and dry. Certainly from that perspective, the 640 Blade offers peace of mind and added reliability. I would recommend the Blade 640s for small to medium-sized environments, clustered environments or physically disperse environments in which having several backup servers makes sense. They are certainly worth considering when cost is a major deciding factor. Benchmark will also be coming out with a VS Tape 160 drive with double the capacity and an impressive 8MB/s native speed by the time you read this.

If you need a top-of-the-line system with all the speed and power you can muster, provided you can push data fast enough, LTO is probably the way to go, and our system from HP has performed well (knock on wood). If what you need is something small, well-priced and reasonably powerful, the VS Tape 80 is an excellent choice. I've grown quite enamored with the drive since I plugged it in (and also with BRU-Pro 2.0—see my review at www.linuxjournal.com/article/6068), and I wish I could afford one for my own personal use. Donations are welcome.

Product Information/The Good/The Bad

Cosimo Leipold is an analyst for DiamondCluster International in Chicago. He spends his free time skiing, scuba diving and trying to take as much vacation as possible. He welcomes comments, thoughts and ideas at cosimo@hypnotic.net.

Archive Index Issue Table of Contents

Advanced search

Advanced search

Letters

Letter Editor

Issue #100, August 2002

Readers Sound Off

Eyes Opened—Looking for Lurkers

I'd like to thank Charles Curley and (obviously) *Linux Journal* for writing the “Emacs: the Free Software IDE” article in the June 2002 issue. I've been using XEmacs for four years for my programming projects, and Charles' article opened my eyes to a bunch of options I wasn't even aware of. I guess I'll be spending the next couple of weeks (or months) going through the Emacs documentation to see what else is lurking in the 20MB of source code. Thank you *LJ* for publishing such an informative (and well-written) article.

—Robert James Kaes

Humor in Advertising

It's great that you can afford to spend an entire page on humor, even though it's not the April issue. And it's even that dry, sarcastic humor that you usually find only in British publications. I'm referring, of course, to the Compaq ad parody on page 51 of your June 2002 issue, where they brag about being such an early adopter of Linux when “their” employee, a Mr. Jon “maddog” Hall, ported Linux to “their” Alpha system. And how “they” funded and otherwise supported Linus as early as 1994.

As a matter of fact, wasn't Mr. Hall employed by DEC in 1994, when he ported Linux and helped to get funding for Linus to support the Alpha architecture? In 1994, wasn't Compaq, as well as Dell and all the other major OEMs, claiming that no one wanted a Linux system?

I realize that Compaq now owns all the assets of DEC and that marketing people have no shame, but if this ad is believable, perhaps I should be able to buy a Picasso and then claim that I painted it.

—Bill Peterson

Jon “maddog” Hall replies: *Linux Journal* was kind enough to allow me to answer your Letter to the Editor.

No one felt worse about the demise of Digital Equipment Corporation and its purchase by Compaq than I did. I had used DEC products since 1969, and it was on a PDP-8 through the use of a DEC training manual that I taught myself assembly language programming. When I was the Department Head of Data Processing at Hartford State Technical College, DEC repaired my PDP-11/70 for free because it had taken them so long to find the problem, and the school had no budget that year for the “time and materials” it would have taken to fix it. Later I learned UNIX and was system administrator to six VAX 11/780 machines at Bell Laboratories.

Through the years I got to know a lot about DEC culture and its commitment to its customers. As part of this commitment I was involved with DECUS, and from the DECUS members (Kurt Reisler, in particular) learned about Linux. Digital Equipment Corporation was more than a company, it was a family.

Yes, I was working for Digital at the time that the Alpha Project was started. Yes, it was several years before Compaq bought Digital. Yes, I felt a pang as Compaq took over Digital's role as the “first system vendor to join Linux International” (even before VA).

On the other hand, to deny Compaq the right to that claim denies the work that the people who still are employed by Compaq (or should I say HP?) did at that time. I know that Maurice Marks, who basically funded the Alpha Linux work in the first days is still at the “new HP”. Jay Estabrook (Hi Jay!) who did a lot of the low-level porting work is also still there, helping to continue support of the Alpha with Linux. The Alpha high-performance group, who started some of the first commercially available Beowulf clusters is still working away. The Digital compiler group, who ported their compiler technology to Alpha Linux is still (to my knowledge) working away. There were many, many people besides me who contributed their time and expertise, paid and unpaid, employee and customer, to the Alpha Linux Project. They did it because of the love of Linux and the love of Digital. They did it both with the support of their upper management and sometimes in spite of it.

To deny them their role in the support of Linux just because their company now has a new name is not fair either.

And if Compaq does not claim the right to say that they supported Linux first, who does? Digital Equipment Corporation is no more, and all the wishing to bring it back will not make a bit of difference. Soon I will expect to see Hewlett-Packard take up the mantle. And now, to set the record straight, a lot of other companies had people supporting Linux early on. Sun supported David Miller to do Linux porting work to Sparc. Compaq had people in Houston who helped to write device drivers and do porting. I know there were people at IBM and HP who were also doing Linux work. But for one reason or another the cultures of their companies did not allow them to have the visibility that Digital's culture allowed me.

In 1998, when a lot of companies held up their hands to say “they supported Linux” a lot of people were able to come “out of the closet”, but that does not mean those people were not active before. I had just come out a little sooner, and with a little more fanfare. Then again, I have never been a shrinking violet.

The world of computer company buyouts and mergers creates little tricks in time and space, and we should learn to live with them. A friend of mine, David Mosberger, who did a lot of work on Alpha Linux, recently wanted to return an Alpha system lent to him by Digital in 1995. But he did not know who to return it to, since Digital no longer existed. I told him to just hold on to it, and that things will be made right again. You see, for the last several years David has worked for Hewlett-Packard. His system is back home again.

The bottom line of this is that if I were writing a history book, I would mention the contributions of Wang, Prime and a host of other defunct computer companies like Digital Equipment Corporation. But I live in the here and now, so the current company that supported Linux in 1994 is Hewlett (née Compaq, née DEC) Packard.

Trivial Correction

Regarding the June 2002 issue, page 8, trivia question 1—please lay this myth to rest. Yes, Grace Hopper did find a moth in a relay. That log page is now in the Smithsonian Institute, I believe. But “bug” in the current sense has been around since Thomas Edison's days. See www.byte.com/art/9404/sec15/art1.htm and/or www.catb.org/~esr/jargon/html/entry/bug.html.

—Felix Finch

Suggestions Welcome?

Editor's note: The following Letter to the Editor came to us written by hand on a sheet of yellow legal pad paper.

From the Desks of Adrian and Mike: We understand that yours is a magazine that would like to appeal to Linux enthusiasts. Therefore, we have some suggestions regarding your fluffy layout:

1. Use fixed-sized, monospaced font throughout the entire publication. All pictures must be represented as ASCII art or PostScript files, and there must be no color. You've got black; you've got white. What more do you need?
2. The cover should not contain anything but the title and the date.
3. The binding should be staples and not glue, which is bad for the environment and embarrassingly corporate.
4. All type must be printed with high-impact printers to give each page a unique and profound texture.
5. You should change the name of your magazine from *Linux Journal* to *Linux Kernel* and, under no circumstances, write about anything that is not a part of or that cannot be directly compiled into the kernel.
6. There is no need to use English for all the articles. C and Bourne shell scripts are languages much more recognized by the global Linux community.
7. We could go on for pages but feel strongly that if you just covered the first 6, just a half a dozen honest, down-to-earth and from-the-heart suggestions, your circulation would multiply by a factor of ten (we know we would by ten issues each month just cause they were that cool); your revenues would skyrocket, never mind the inherent irony about the relationship between your capitalist intentions and the brilliant and revolutionary open-source model that Linux embraces, and, not to mention, we'd buy you beer.

—Adrian and Mike

Social Destruction

On a whim I was reading Harriet Beecher Stowe's *Uncle Tom's Cabin* today. I was struck by how, at the time, abolitionists were considered socially destructive extremists—and the South's reflexive reaction was censorship. The essence of freedom is freedom of thought and creativity, bounded by essential (not submissive!) respect for others. I believe that, perhaps in 50 years, perhaps 2,000 years, but with the same certainty that humanity will survive, intellectual property will come to be regarded with the same revulsion and embarrassment as human beings as property.

—Stephen Schaefer

Protocol Problems

The Linux for Suits column, “The Protocol Problem”, by Doc Searls in the July 2002 issue of *Linux Journal*, was of interest to me because it is an issue that I think we are seeing several major issues start to develop around. There seems to be an overwhelming amount of “If all you have is a hammer, everything looks like a nail” mentality going around, at least insofar as trying to put nearly the entire world on top of IP—this is only making, what Doc Searls referred to as “the gating factor”, worse.

As technology develops, the lower layers of the technology become more and more abstracted and hidden from the upper layers. Right now, it is doubtful if many people care or concern themselves with whether their IP packets ride over fiber or copper, the particular framing or link-level protocols being used, etc. Because we can connect fiber and copper and wireless networks together, we are not limited by the individual mediums.

Eventually, this is what is going to have to happen with IP and any other protocols occupying this space—the “end-to-end (network) protocol” mentality must give way to the “end-to-end application” mentality. One day, eventually, the IP network will be replaced by something else. It's going to happen—but the general mentality seems to be that an IP network *must* be IP end to end. Building for that, designing for that, that attitude will slow down the deployment of any future protocol technology...like IPv6.

One day, just as most people do not care whether their data are going end-to-end over a particular physical media (as long as the desired quality of service parameters are met), they will not care about the network protocol. When data networks are not an “IP-only” club where only IP through-and-through networks can play, but a true inter-net (as opposed to the current IP Internet, which is more of a giant shared-address space Intranet) where the intra-net protocols may all be different, then the barriers will be gone. Organizations will be freer to develop and deploy new protocols and technologies internally without getting ostracized by their peers for being different.

Erratum

In my interview in the June 2002 issue, my affiliation was mistakenly listed as “Director of PythonLabs” rather than the correct “Director of PythonLabs at Zope Corporation”.

—Guido van Rossum

Archive Index Issue Table of Contents

Advanced search

UpFRONT

Various

Issue #100, August 2002

HP to Hardware Vendors: Peddle That NDA Somewhere Else

HP to Hardware Vendors: Peddle That NDA Somewhere Else

Bruce Perens, Linux czar of Hewlett-Packard, has what he calls a “very good” chance of getting a preference for Linux-compatible hardware made into corporate policy at the new largest Linux vendor, the merged HP/Compaq.

The policy would mandate a company-wide “at design-in time, preference for devices that have publicly documented interfaces”.

For HP engineers, it means that if you have a choice between two hardware products, select the one for which an open-source driver exists or for which the vendor publishes programming information. If a part comes with a nondisclosure agreement, don't use it unless there's no alternative, or the alternative would be prohibitively expensive.

How much less will NDA-shrouded hardware be worth to HP? Perens doesn't put a number on it. Or, to look at it the other way, how much of a price premium will HP be willing to pay to use a publicly documented device? The advice to designers is “use your head”, he says.

“I was concerned about graphics display chips”, Perens said. ATI's low-end graphics hardware is fairly clean, but at the high end, “They're all crazy”, he said. The policy has already resulted in HP dropping NDA-covered modem chips for openly documented ones.

Although the documented hardware policy was approved at a “corporate policy level” within HP before the merger and was ready to become the standard within the company, Perens has to get management of the new, merged company to approve it all over again. “The merger has held off a lot of things, and I have to get [Compaq managers] to be cognizant of the reason we need it”, he said.

Just because HP laptops and PDAs will be Linux-friendly doesn't mean the company will formally support Linux on them. There are no plans to offer a Linux laptop or to commercialize Jim Gettys' work on running Linux and X on the Compaq-now-HP iPAQ.

“Most of what I'm doing with Linux is servers”, Perens said. The merger's overall effect on Linux at HP and Compaq? “Unless something awful happens it should make it better.”

—Don Marti

Hot and Cool Linux Dot-Com Actually Makes Money

HOTorNOT.com offers a simple and democratic answer to a common but embarrassing question: how good do I look? Voters play a kind of whack-a-mole, only they whack human beings instead of moles and use a mouse instead of a mallet, rating each candidate on scale of 1 to 10. As soon as one gets rated, another pops up next to a thumbnail of the last one, with the current rating.

The site was conceived in October 2000 by James Young and Jim Hong, a couple of 27-year-old Berkeley-trained hackers, roommates and drinking buddies. They were going to put it up on XMethods, their web site for publicly available web services.

For what instantly became obvious reasons, they made HOTorNOT.com an independent site. In just over a week, the site was getting almost two million page views per day. By May of this year, HOTorNOT had counted over 2.1 billion votes and had over one million user accounts.

But here's what's really hot: it runs on Linux. “In fact”, James Hong recently told us, “we couldn't have done it without Linux.” By “it”, he means make money. According to Hong, HOTorNOT pulls in more than enough income to pay for itself and its staff, which still consists of the original two guys.

That's because they've adapted quickly. “When the advertising business began to crash, we added a paid 'meeting' service to the system.” The result was countless dating success stories, including more than a few marriages. But the most important success story is HOTorNOT's own.

“HOTorNOT is a viable business built entirely on Linux”, Hong says. More specifically, “Linux (Red Hat), Apache, MySQL and PHP on 35 1U rackmounts, mostly from Rackable Systems.” (See picture.)

—Doc Searls

LJ Index—August 2002

1. Number of countries considering a bill or motion to mandate or promote free-software use by the government: 9
2. Number of complete sets of tapes required for one year of tape backups: 29
3. Number of sound cards and chipsets supported by ALSA (Advanced Linux Sound Architecture): 94
4. Number of cards for which docs are available but no ALSA driver yet exists: 20
5. Number of sound cards for which manufacturers are refusing to provide documentation to the ALSA Project: 4
6. Length in characters of a Perl regular expression that matches any valid URL: 7,579
7. Names in CREDITS for Linux 1.0: 80
8. Size of Linux 1.0 compressed: 1.2MB (15.4KB per contributor)
9. Names in CREDITS for Linux 2.4.18: 411
10. Size of Linux 2.4.18 compressed: 28.8MB (71.5KB per contributor)
11. Size in billions of dollars of the e-mail marketing industry: 1
12. Predicted average number of spam e-mails per inbox per year by 2006: 1,500
13. Number of spams accumulated in an idle inbox on Earthlink over the year leading up to August 2001: 1,200
14. Number of spams accumulated in the same inbox between April 18 and May 13, 2002: 1,124

15. Number of spams, received by one *Linux Journal* editor's inbox on May 1, 2002: 197

Sources

1: www.lugcos.org.ar/serv/mirrors/proposicion/doc/referencias/#ref.#1

2: com/~rick/linux-info/tape-backup

3-5: www.alsa-project.org/soundcards.php3

6: www.foad.org/~abigail/Perl/url2.html

7-10: kernel.org

11: *San Jose Mercury News*

12: Jupiter Media Metrix, www.jmm.com

13-15: *LJ* senior editor

Danish Navy Develops on Linux, Deploys on LynxOS

The Royal Danish Navy is going to sea with applications developed on Linux but running on the Linux-compatible, real-time LynxOS from LynuxWorks (lynuxworks.com). “The Linux interfaces are becoming the real-world definition of open systems”, said Dr. Inder Singh, LynuxWorks' CEO.

“LynxOS is very unique in being a hard real-time operating system that is ABI-compatible with Linux”, he added. The Royal Danish Navy doesn't even need to recompile to move applications from Linux-based development systems onshore to the LynxOS platform on ship.

LynxOS has been on the market for more than a decade and was designed into Space Station Freedom before the project was re-organized as the International Space Station. It is also used in Airbus jets. On the ground, LynxOS is also used on HP LaserJet printers and Xerox copiers.

“Requirements for certification are pretty strenuous”, Singh said. Extensive documentation is required, and no Linux-based system has yet been certified for real-time space or military use. “There have been many attempts, but no one has been able to get one certified”, Singh said.

Observers of Linux's inexorable progress add, “Yet”.

—Don Marti

Stop the Presses: Commons, 2; Hollywood, 0

The entertainment industry's war with technology goes back a long way. In 1984, MPAA Chairman Jack Valenti said, “The VCR is to the American film producer and the American public what the Boston Strangler is to a woman home alone.” But we can trace the current system of fears and balances back to 1908, when music publishers claimed player piano rolls violated music copyrights. That case lost in the Supreme Court, but the industry prevailed on Congress to establish the “mechanical license”, which established the right to reproduce published music in return for a regulated royalty. In 1931 some music composers claimed that a hotel's radio station violated copyrights by playing their music. The composers lost that one. In the 1960s, book publishers failed in their suit against photocopiers, but the Supreme Court ruling on the case allowed “fair use” of published works. In the late 1960s and early 1970s, broadcasters took on Community Antenna Television or CATV. (The movie industry was also involved, forcing theater employees to wear buttons that said “Fight Pay TV”.) Supreme Court rulings on two cases opened the way for the cable TV systems we have today. So it's clear we're still in this fight for the very long haul.

But at least we can pause to observe two current victories for technology over those who would control it: the launch of Creative Commons and the Librarian of Congress' rejection of CARP's (Copyright Arbitration Royalty Panel) recommendations for imposing stiff requirements and performance use fees on internet radio stations.

Creative Commons (www.creativecommons.org) was launched at the O'Reilly Emerging Technologies Conference in Santa Clara, California, on May 16, ending months of quiet development. It is led by Lawrence Lessig, Stanford Law professor and author of two highly influential works (*Code and Other Laws of Cyberspace* and *The Future of Ideas*), both of which argue against the entertainment industry's constant campaign to bend copyright law in their favor and to replace the Internet's commons with a highly regulated system for the supply-controlled distribution of “content”.

The project lives at Stanford University, where it is highly involved with the Stanford Law School Center for Internet and Society. It also receives what it calls “generous support” from the Center for the Public Domain (formerly the Red Hat Center), which is headquartered at Duke University and chaired by Red Hat founder Bob Young.

Rather than simply lobbying against the entertainment industry, Creative Commons offers concrete solutions to the first sources of creative goods: the artists themselves. In his speech to the conference Lessig said,

> This content that the law says is mine, I should be able to make available on my own terms. We need machine-readable expressions of the author's intentions about the nature of the content. The world should not be divided between those who believe in control and those who believe in access. Those who want both, on an individual level, should be able to compromise.

While Creative Commons was in development, internet radio, which includes thousands of stations (many of them making resourceful use of Linux and other open-source software), was under severe threat by the CARP's recommendations, which were issued in February 2002 and widely expected to knock most stations off the air. If adopted, the CARP recommendations would have become regulatory fact on May 22, 2002. But on May 21, the Librarian of Congress issued an “order rejecting the panel's determination”, which would become final one month later. Stations and advocacy groups like SaveInternetRadio.org breathed a public sigh of relief.

But the issue is far from resolved. CARP is a creature spawned by the Digital Millennium Copyright Act, which is still in force. More importantly, the whole idea of the Commons is still not well understood—especially on Capitol Hill, where the influence of the entertainment industry remains enormous.

In his speech Larry Lessig said, “We will never succeed in advocating the importance of this space until ordinary people get it. And they won't until technologists begin to express to politicians how important these values that they built into technology are to freedom and creativity.”

—Doc Searls

They Said It

Today, if you've got end-to-end encrypted mail going in and out of your company, it's probably somebody dealing drugs or sending or receiving pornography inside your company.

—Greg Olson, chairman and cofounder, Sendmail.com

The true cause of the enormous ills that now dismay so many Americans—the universal sleaze and “dumbing down”, the flood tide of corporate propaganda, the terminal insanity of United States politics—has risen not from any grand decline in the national character...but from the inevitable toxic influence of those few corporations that have monopolized our culture.

—Mark Crispin Miller

As bad as the cell phone carriers' quality of service is, the last thing they want is to lose control over their quality of service. That's why they don't understand the appeal of 802.11. You can't do anything viral in their environment.

—Dave Sifry

If you operate a web site and wish to link to this site, you may link only to the home page of the site and not to any other page or subdomain of us.

—*Dallas Morning News*, Terms of Service (on its registration page)

A man is judged by his Values; his Values are marked by that which he will not compromise.

—G. E. Nordell

The social object of skilled investment should be to defeat the dark forces of time and ignorance, which envelop our future.

—John Maynard Keynes

“Did you really think we want those laws observed?” said Dr. Ferris. “We *want* them to be broken. You'd better get it straight that it's not a bunch of boy scouts you're up against....We're after power and we mean it....There's no way to rule innocent men. The only power any government has is the power to crack down on criminals. Well, when there aren't enough criminals one *makes* them. One declares so many things to be a crime that it becomes impossible for men to live without breaking laws. Who wants a nation of law-abiding citizens? What's there in that for anyone? But just pass the kind of laws that can neither be observed nor enforced or objectively interpreted—and you create a nation of law-breakers—and then you cash in on guilt. Now that's the system, Mr. Reardon, that's the game, and once you understand it, you'll be much easier to deal with.”

—Ayn Rand, *Atlas Shrugged*

Take what you want and pay for it, says God.

—Spanish proverb

Decisions are made by those who show up.

—Aaron Sorkin

The only obligation which I have a right to assume is to do at any time what I think right.

—Henry David Thoreau

The Internet is obviously a critical part of any e-business. But the Internet is only a common set of protocols for the transport of information.

—Sybase advertisement

And reading is only the common set of protocols for the translation of oral words into written marks. And Sybase's products are only the semi-intentional arrangement of bits.

—David Weinberger

Welcome to the 100th Issue of *Linux Journal*

Richard Vernon

Issue #100, August 2002

While I'm sure that this is only the first hundred-mark of many, you might want to keep this issue in good shape for putting in your children's time capsule.

I still tend to think of myself as a newcomer to the magazine, having been on the masthead for only one-fifth of the now 100 issues. But those 20 issues have been an enriching experience. Each issue has presented its own challenges and dilemmas.

Some issues even hosted their own minor polemics. Nothing sparks comment like nudity, and our staff had a lot of fun reading and choosing which of the letters regarding the nude cover of the Python supplement to print. Then there was the “American” debate, begun in the November 2000 issue, which also spanned more than a couple of issues. Who could forget the QSOL advertisement, also from the November 2000 issue? The “improper” implication spawned some memorable press, but I'm sure it would never have happened if we hadn't made the first step and put the naked man on the cover, obviously giving QSOL the impression that we were “that kind of magazine”. QSOL's following ad was as frightening as the previous ad was suggestive, perhaps in a effort to take revenge on on those not yet prepared for their style of advertising.

These events made the letters section perhaps the most-read part of the magazine, but the most fun part of working for *Linux Journal* has been the interaction with both the Linux community and the rest of the hardworking *Linux Journal* staff.

As our magazine is one that relies on community-submitted content, I've been fortunate to be able to get to know many very talented hackers. And I'm not just talking about the famous members of the Linux community. Just as some of the best actors in the world are only seen on community stages, some of the smartest coders are garage hobbyists or work for small companies. The general high level of goodwill that emits from our authors, regular contributing editors and is evident throughout the community, consistently impresses me and makes my job a much happier one. Our reliance on you—our readers and contributors—makes this 100th issue celebration as much yours as it is ours. Sincere thanks for all of your help and support.

Richard Vernon is editor in chief of *Linux Journal*.

Wish you were here.

Advanced search

Best of Technical Support

Various

Issue #100, August 2002

Questions and Answers

Where's My FireWire CD-ROM?

I recently bought a Sony R505 laptop and when I tried to install SuSE Linux, I discovered that after the boot, Linux could not see the CD-ROM. It turned out that the docking station where the CD-ROM lives is accessed from the computer via FireWire somehow.

—Steven Smith, sjs@chaos-tools.com

Your confusion may be due to differing procedures for the various distributions available. For SuSE, there is a basic HOWTO guide in their knowledge base, which you can access by visiting sdb.suse.de/en/sdb/html/tbraza_dosinst.html. This should get you headed in the right direction.

—Chad Robinson, crobinson@rfgonline.com

I Have No Qt and I Must Run KDE 3.0

Are there RPM packages for the latest versions of Qt and KDE?

—Dan, dhdeang@SoftHome.net

To locate RPMs, I find rpmfind.net to be a helpful portal. In this case, there is a page specifically for Qt and KDE: rpmfind.net/linux/RPM/Development_KDE_and_QT.html. On that page I note that Qt 3.x is still listed as a development package, but it is available.

—Chad Robinson, crobinson@rfgonline.com

Raiders of the Lost DDS-3 Tape

I'm trying to read a DDS-3 tape made on an HP9000 machine running HP-UX 11. I'm trying to read it using my Linux server that runs Red Hat 7.2 (2.4.9.31SMP kernel). The Linux server tape drive is a Sony SCSI device, /dev/st0. The tape contains Oracle export files of a large Oracle database. The HP machine is no longer available, and I'm trying to rebuild the DB on Linux. When I try to read the tape with either **tar -tvf /dev/st0** or **tar -xvf /dev/st0** I always get:

```
Input/Output error at the beginning of the tape, error is not
recoverable, exiting now
```

Why?

—Adrian Manship, adrian.manship@skynet.be

Check two things related to the tape drive. First, DDS drives support the concept of hardware compression, which may have been used by HP-UX and may not be enabled by Linux. If there is a mismatch in this setting, that can cause problems reading the tape.

Second, although tar itself is a relatively standard format, is it possible that the tape was not created using tar? Depending on the version of Oracle used to create the tape, it may have internal support for reading/writing tape devices as backup media. Check a small sample of the file to be sure it is actually in tar format by running **dd if=/dev/st0 of=/tmp/tape bs=512 count=16**, replacing the bs value with a block size that works well for your drive and the count value with a count that yields a sufficient number of those blocks to take a quick look. This example will read 8K from the tape.

Perhaps the tape itself is bad. If it succeeds, you should be able to use a hex editor or text editor that can handle high-ASCII characters without choking to verify the format of the contents of the tape. In case you aren't familiar with the general format, each file entry should start with its filename, followed by a header that contains miscellaneous file information such as size and permissions, followed by the file itself.

—Chad Robinson, crobinson@rfgonline.com

The easiest thing to do, would be to buy/borrow/steal a similar machine to do the restore.

—Christopher Wingert, cwingert@qualcomm.com

I used to work with HP-UX DDS tapes, and most of the time they were used with the cpio utility. Keep in mind that cpio is a complicated utility with many options, so check the manual pages (**man cpio**) and play around with some of the options. Be careful not to write your tape accidently; I suggest that you physically enable the write-protect feature on the cartridge. Look at www.lns.cornell.edu/public/COMP/info/cpio/cpio_2.html for a simple cpio tutorial.

—Felipe E. Barousse Boué, fbarousse@piensa.com

No Screensaver for root

When I sign on using root, the screensaver does not work. When I sign on with another ID, it does. How can I make it work when signed on as root?

—James Logan, jlogon@mail.ewu.edu

xscreensaver will not work as root on purpose. It's a security feature; you're not supposed to run X as root (log in as a user and use **su -** where needed). This is explained in the FAQ, along with a solution to how to run X programs as root if you need to (www.jwz.org/xscreensaver/faq.html#root-lock).

—Marc Merlin, marc@merlins.org

New Install Won't Boot

I attempted to install Red Hat 6.2 on my spare machine. The install ran fine and completed. I removed the disks and rebooted, and I keep getting a nonsystem disk error telling me to replace the disk.

—Tim Dreas, timdreas_@hotmail.com

It seems like either LILO or GRUB, the boot loaders, were never installed correctly. If you made a rescue disk during the installation (of course you did, right?), use it to boot your computer. When you get the root Linux shell prompt (#), type **lilo -v**, which should attempt to write the boot loader into your hard disk.

—Felipe E. Barousse Boué, fbarousse@piensa.com

How Do I Remove a User

How do I remove a user from a group? (We deleted the user and now are receiving error messages that there is a permanent fatal error when someone e-mails this group.) How do we correct this?

—Barbara Viola, bviola@viotechsolutions.com

I assume you mean mail alias and not a UNIX group. Check out /etc/aliases and remove the user. Then run **newaliases**.

—Christopher Wingert, cwingert@qualcomm.com

To remove a user from a group, not just from an e-mail alias, use the **gpasswd** command as root:

```
gpasswd -d name_of_deleted_user
```

Take a look at **man gpasswd**. It gives an explanation and options for other group administrating functions.

—Paul Christensen, pchristensen@penguincomputing.com

I Have No Static IP Address and I Must SMTP

Is it possible to have an internet mail server with a dial-up connection to an ISP? I know that the IP address of ppp0 may change whenever I establish a new connection with the ISP. The MX records need to point to the mail server; the IP address of the server also must be specified at dewdesigns but will not be valid if or when I get a new connection. Do I need to run DNS locally or can I use the ISP DNS? I recently read an article by Marcel Gagné regarding small-office mail servers, but I feel that I am missing some pieces of the puzzle.

—Daryl E. Murray, daryl@Planet4us.net

There are ways to have an SMTP server on a dial-up dynamic IP and set up DNS so that it gets updated every time you change your IP address, but trust me, you do not want to go there. Short of running UUCP, which is the correct way to route mail in your case (UUCP is quite old, not well-known by most system administrators and probably not supported by your ISP), you should use Fetchmail to download your mail. If you need to download mail for many accounts, you can have your ISP spool all your mail in one mailbox, download that with Fetchmail, and split it up again, looking at the Envelope-To: field or whatever field in which your ISP stored the original Envelope-To.

—Marc Merlin, marc@merlins.org

You only can use Fetchmail to download all of your site's mail if your ISP consistently applies an Envelope-To: header to your mail. See the warning at www.catb.org/~esr/fetchmail/fetchmail-man.html#25.

If you have a dial-up with a static IP address, and your ISP is willing to queue incoming mail for you, you can do an **SMTP ETRN** when the connection comes up. Sendmail includes a utility to do this.

—Don Marti, info@linuxjournal.com

New Products

Heather Mead

Issue #100, August 2002

The latest technology.

Eclipse Database Center

Linux NetworX announces their Eclipse Database Cluster, based on the Oracle9i Real Application Cluster architecture. The ready-to-run system offers scalability, reliability and high availability for data warehousing, on-line transaction processing and data management applications. The cluster is built with multiple Xeon processors, Dot Hill storage components and high-speed Gigabit Fibre Channel interconnects. The Eclipse is integrated with ICE cluster management tools to provide remote power control, temperature sensing and real-time monitoring.

Contact Linux NetworX, 8689 South 700 West, Sandy, Utah 84070, 801-562-1010, www.linuxnetworx.com.

Tetra Platform

The Tetra hardware platform from Equator Technologies provides a compact and modular reference design that allows rapid development of products such as IP smart cameras, digital video recorders and IP-based internet streaming video appliances. The modular design includes the Tetra CPU board, an open peripheral interface for add-on modules and various add-on personality modules. All common CPU board core features, such as the BSP chip family, memory subsystems and Ethernet interface, are on the CPU board for a final size of 2.75" × 4". It comes with 64MB SDRAM and 4MB Flash memory; analog audio out and SPDIF audio in/out, 10/100Base-T Ethernet; and support for Linux or VxWorks.

Contact Equator Technologies, Inc., 300 White Oaks Road, Campbell, California 95008, 408-369-5200, sales@equator.com, www.equator.com.

Cobalt RaQ 550

Sun's Cobalt RaQ 550 is a 1U rackmount server appliance preconfigured with a host of software and hardware for the deployment of e-mail, web hosting and other internet applications. Designed for small to mid-sized companies looking to handle their internet and web applications in-house, the RaQ 550 has a 1.26GHz processor, up to 2GB memory and two 80GB drives with support for RAID 0 and 1. Preconfigured software includes Apache web server, Apache Tomcat, Sendmail, Bind DNS server, JSP, InterBase 6 SQL, MySQL, PostgreSQL, PHP, Perl, Python, SSH v2, 128-bit SSL, SNMP agent and Legato NetWorker and Knox Arkeia backup clients.

Contact Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, 800-555-9786 (toll-free), www.sun.com/cobalt.

BladeRack RS-1100V

RackSaver offers its latest entry in the blade server market, the BladeRack (patent-pending). The BladeRack houses up to 66 RS-1100RV RackBlades, each containing up to two 2.4GHz Xeons for a total of 132 processors in a 7' portable cabinet. AMD-based processors also may be used. The BladeRack can be configured with 66GBs of DDR ECC PC-2100 Corsair memory, based on Supermicro's P4DPR-6GM+ motherboard. In addition, the BladeRack is easily scalable and accessible, with hot-swap capabilities for the fans, networking equipment and servers themselves. Using nonproprietary, interchangeable off-the-shelf components, the BladeRack is targeted at 3-D rendering, research centers, universities and other data-intensive uses.

Contact RackSaver, Inc., 9449 Carroll Park Drive, San Diego, California 92121, 858-874-3800, inquiries@racksaver.com, www.racksaver.com.

Xcelerix Embedded IMDB

Xcelerix announced the formal launch of the company and the general availability of Xcelerix In-Memory Database (IMDB), formerly known as ERDB. Xcelerix IMDB's architecture prejoins database tables based on known operational queries and uses binary tree indexing, helping to eliminate bottlenecks and enabling up to one million transactions per second (SQL benchmark results). A small footprint and a 20:1 compression of memory resources ratio allows the Xcelerix IMDB to be embedded in response-sensitive, real-time applications. It is available in both 32- and 64-bit versions for a variety of platforms. Xcelerix IMDB supports integration with disk-based database systems, in addition to SQL, TCP/IP, IEEE 802.3 and POSIX.

Contact Xcelerix Corporation, 9000 Keystone Crossing, Suite 900, Indianapolis, Indiana 46240, 800-473-9012 (toll-free US), www.xcelerix.com.

Linux USB 2.0 Driver

Cypress Semiconductor released a Linux USB host driver that supports their SL811HS embedded host/peripheral controller, enabling developers to add USB host functionality to a variety of embedded applications, such as cell phones, PDAs and network appliances. Because the SL811HS controller is a memory-mapped device, the driver lets the Linux host stack provide USB support without realizing a non-OHCI/UHCI type host controller is present. The SL811HS host controller has a dual-role port that can function as either a USB host or a peripheral supporting both full- and low-speed USB devices. Source code for the driver is available at www.cypress.com/press/linux.

Contact Cypress Semiconductor Corporation, 3901 North First Street, San Jose, California 95134, 408-943-2600, www.cypress.com.

Fitrix

Fitrix is a source-code business software package that provides accounting, order processing and custom software tools for small to medium-sized businesses. In addition to accounting and distribution modules, Fitrix includes the application source code (written in Four J's Business Development Language (BDL), a 4GL derivative that is optimized for business database applications), 4GL programming tools, a RAD toolset and an SQL database. Custom functions and modules can be written in C and Java and directly embedded into the 4GL code. Fitrix also uses a thin client, so processing is done on the host system, keeping traffic to a minimum and allowing multiple clients to use one set of software simultaneously.

Contact Fourth Generation Software Solutions, 2814 Spring Road, Suite 300, Atlanta, Georgia 30339, 770-432-7623, info@fitrix.com, www.fitrix.com.

Archive Index Issue Table of Contents

Advanced search