Advanced search

Linux Journal Issue #118/February 2004

]

Will your favorite OS be your new favorite BIOS too?

LAMP Development at Public Sector Web Sites

Tom Adelstein

Issue #118, February 2004

A new breed of IT firm is helping federal, state and local governments create a “public infostructure” of interoperable, effective Web-based applications.

Linux runs right under the radar, not exactly in stealth mode, at many high-profile government agencies, including the Departments of Defense, State and Labor; the General Services Administration; the Census Bureau; and USAID. At the heart of these applications lies the LAMP framework, which stands for Linux, Apache, MySQL and PHP, Perl or Python. Tom Walker, former Lt. Commander of Navy Special Operations and adamant proponent of open-source technologies, said, “The next computing revolution, like that of the Internet, will emerge from unexpected and perhaps even modest sources...the next great leap in computing and software technology will come from small firms who have the creativity and strength to stand up...by offering advanced solutions using Open Source technologies.” Firms like Walker's group and a few others have led the charge on Capitol Hill and elsewhere with open-source software (OSS).

Driving Linux

Lack of an integrated response within the US intelligence community prior to the attacks of September 11, 2001 has heightened government awareness of the disconnected databases in use throughout the country. As the Congressional Report on Intelligence Actions and the September 11 attacks revealed, “The intelligence community continues to be fragmented.” In response to this situation, Peter Gallagher and Martin Hudson of devIS state, “The need for governments to share information and solutions is fueling a new framework for eGovernment problem solving.” Gallagher explains how Linux has made its way into the public sector:

> Early on we were able to use OSS only because the applications were outsourced. We could not talk about Linux, for instance. Now our clients really are

> considering how OSS might improve their internal environments—they want to talk to us about it. This is a big change. It always is a pleasure to see the look on a customer's face when they come to us asking about how to take advantage of OSS and then we remind them that together we have been for years!
>
> We have shown that it works rather than just talking about it. I expect that in a few years there won't be interest in OSS for eGov per se, it will be just another option—we think the winning option.

Open-Source Software Institute

John Weathersby sits in his office at the University of Southern Mississippi pulling together geographically diverse groups of people with a mission. He runs the Open-Source Software Institute (OSSI), a nonprofit organization comprised of corporate, government and academic representatives. OSSI exists to serve as an advocate and as a collective resource and venue for the promotion, development and implementation of open-source software solutions between corporate, government and academic entities. Additionally, OSSI serves as a forum for working committees to address open-source issues in regards to government/industry standards, academic research, economic/market and legislative policy.

OSSI members John Weathersby (front) and Richard Kuebler (middle) pause with Mark Goodman (left) and Matthew Schick (right) from the University of Southern Mississippi as they tour campus. The Administration building stands in the background.

Shortly after inception of the Web site Government Forge (governmentforge.org), Weathersby reached out and began supporting Project Leopard, a framework for developing LAMP applications in government. Within two weeks, he pulled together open-source developers to facilitate building a common infrastructure. He next found an immediate application need within the judiciary branch.

Other recent wins include a Cooperative Research and Development Agreement (CRADA) with the Naval Oceanographic Office delivered in August 2003. The Institute also serves as the coordination body securing government certification for OpenSSL under FIPS 140-2 approved cryptography. In addition, Weathersby has coordinated an effort in which the North Mississippi Education Consortium (NMEC) will lead a pilot program designed to provide free and open-source software to Mississippi's public school system. The program, called Freedom to Learn, is part of a PhD-level study exploring alternative technologies and methods of reducing costs while increasing efficiency and student productivity within public school systems.

devIS

Peter Gallagher and Martin Hudson direct traffic from the second floor of the historic Underwood Building in Arlington, Virginia. Gallagher, a former Peace Corps volunteer in Senegal, West Africa, saw the need for more appropriate technology solutions in developing countries. Hudson had a deep interest in making computers more useful and was pushing desktop applications. They met as consultants, worked together at two different firms and then decided to start their own company, Development Info Structure (devIS).

Using LAMP became Gallager and Hudson's stock and trade. As Peter explains:

> The opportunity to develop a public info structure with less redundancy, lower costs, greater flexibility and better service is the eGovernment challenge in this new world. devIS anticipates further expansion in the coming years based on accelerating interest in open standards, eGovernment and efficient use of open-source software solutions. devIS has capabilities in software development and outsourcing that are unique for a small business, and projections indicate rising demand and associated revenues. State and local governments are showing increased interest in open systems. devIS has begun actively soliciting partnerships with these groups. Recent changes to federal procurement rules now allow state and local governments to purchase services from federal GSA contracts, providing standard access to any governmental agency at various levels.

> OSS was essential to devIS as a small business competing against mega-corporations for federal work. Some of the high-end proprietary tools are so expensive to get started with—you pay for partner licensing and all types of required training, including marketing...just to try the product.
>
> The next big thing, now that we all know we need to share data using XML standards and Web services, will be to share components. I know it will be hard, but it has to happen—eGovernment is accelerating and the logic of public infostructure is too compelling—the necessary standards and architectural boundaries are becoming understood. Shared OSS components will move eGovernment ahead quickly.

Martin Hudson adds:

> The Government's adherence to published standards, at multiple levels, is making the market more competitive, making it possible for small companies like devIS to compete on larger, mission-critical, applications. When we formed devIS the higher order systems looked more like fiefdoms for large integrators—small business could not get in the door.
>
> Our ability to implement inter-networking applications fully—we host data servers for state, USAID, GSA and labor—makes us different from most of the small businesses in our sector. And we are able to do that largely because of our roots in open source.

Gallager and Hudson's recent wins include the US Department of Labor's Workforce Connections program. They elaborate:

> This just-in-time dynamic content publishing environment powers over 50 federal Web sites, including DisabilityInfo.gov, the official portal for US government information on people with disabilities. The Workforce Connections application environment also publishes structured learning content, including question and answer interactions, all using the same object-oriented engine.
>
> The tool exceeds federal specifications for Section 508, which is the federal implementation of the W3C Accessibility Guidelines. IT contractors now are legally liable to meet these requirements just as a construction projects must provide handicapped access. The system also meets another standard important to the federal government called SCORM, shareable content object reference model.
>
> SCORM is an XML standard that makes it possible to share and re-use learning objects independent of proprietary authoring/presentation systems.

> Workforce Connections allows for distributed maintenance and instantaneous publishing by government content experts through a secure administration interface. The software was created in Python using the Zope content application server and runs on GNU/Linux Debian with the Apache Web server. Many of the sites are private. devIS currently is working with the DoL to release the product under an open-source license.

devIS also is doing work for the US Agency for International Development's TraiNet Project. Gallagher and Hudson describe TraiNet as:

> ...a secure, Internet-enabled visa application pre-processor [that allows] worldwide staff to comply with new security rules for training foreign nationals in the US. A Web-based work-flow interface, connected to a federated system architecture that relies on XML messaging to compensate for inconsistencies in connectivity among developing countries, provides a robust environment.
>
> The system is in use at over 300 locations around the world to monitor training programs worth hundreds of millions USD for thousands of students. The visa processor has a secure machine-to-machine link with Department of Homeland Security systems to facilitate centralized production of the special student visas used for government-funded programs. OSS technologies used include GNU/Linux Debian, Apache, PostgreSQL and XML Blaster. Server-side applications are written in Python. devIS built, hosts and manages this application, including help desk and other operational support.

gOSapps LLC

As a career Naval Officer, Tom Walker gained extensive experience in programming and Web enabling computer systems that support international military and special forces operations. After leaving the service, he used his leadership abilities to build a client list for gOSapps LLC of more than 400 customers. To date, his organization has installed more than 500 LAMP applications.

Walker's enthusiasm for open-source software is evident. He recently spoke at a Department of Defense staff briefing on the security, reliability and performance of open-source software. He told his audience:

> The DOD decision [to use open-source software] will result in widespread changes in software development and acquisition throughout the federal government and with government contractors. The challenge for many departments is that right now the new policy has

> raised more questions than answers in a fast growing segment of the technology industry.

One of gOSapps LLC's projects is the Open Source Initiate Review (OSIR). OSIR provides formal expert analysis of existing system architectures and applications to help government agencies reach a high level of preparedness for open-source transition.

Recent wins for Walker include the US Navy's Technical Support Group Summary. Walker writes:

> We provided the architecture, design and deployment of secure Web and CD-based training programs. The Technical Support Group (TSG) required a reliable method to deliver training on secure advanced communications system to remote locations. This training required delivery by a variety of transfer methods.
>
> Additionally, due to ever-changing technological changes, the data had to be updated easily, minimizing the costs for program changes. We deployed special strategies and created a multimedia training system that was both informative and entertaining. The data is transferable on CD or over secure communication links.
>
> Additionally, by allowing last minute compilation of data from a secure database, the training is always up to date. This data compilation is performed automatically, minimizing the training costs and time requirements for the technical support staff. An internal object-oriented framework utilized results in lower development time and costs, far fewer pre-QA defects and a richer feature set. The MySQL relational database allowed us to use complex data-driven applications, coupled with reliability and speed. We had a direct role in all aspects of the development cycle, coupled with close communication and feedback from the client, resulting in an advanced yet intuitive interface to meet client needs.

Walker's team also developed an on-line LAMP application to manage the database of storm water facilities and their ratings for James County, Virginia. Walker explains:

> We developed the front-end interface for searching and viewing facility information. We also developed complete functional design specifications, carried through to development and deployment of multi-tiered relational database-driven Web applications.
>
> The system is designed to meet the county's specific need to post and broadcast notifications of watershed

> quality results. The entire application is maintained through an intranet system, all managed by a unified administrative application.

National Center for State Courts

James E. McMillan, employed at the National Center for State Courts has fashioned a Web site that says, “If you have ever wanted to try out court E-filing, now you can. Just click on the E-File a Document link above or the button below and fill in the forms.” The Web site also states, “You will be sent a password (you must have a working e-mail address) and attach your document. Or, you can fill out our demonstration complaint form.” James directs the Court Technology Laborartory (CTL, ctl.ncsc.dni.us/about_jim_mcmillan.htm).

McMillan has made his LAMP Project available to all the courts in the world. The inCounter Web site provides the downloadable source code to the inCounter Electronic Filing Manager Project. Look for a link that says “inCounter Open Source E-filing System” on James' Web page mentioned above. The link will take you to the current location of his OSS project.

Why are we doing this? To help the courts and legal system adopt electronic communication. Specifically, the inCounter Electronic Filing Manager Project is an effort to build the core functionality of an electronic filing inbox that has the following initial goals:

- Demonstrate electronic court document filing.
- Demonstrate a simple-to-use system (limited initial scope).
- Create an expandable and customizable system through use of open-source code.
- Demonstrate support for CourtXML/OASIS LegalXML filing standards.
- Demonstrate support of the W3C SOAP XML communications standard (to connect commercial and advanced systems).
- Demonstrate the use of free Linux, Apache, Perl and MySQL software in a court application.

McMillan joined the National Center for State Courts in October 1990. Since then, more than 1,000 visits from courts in 50 states and more than 70 foreign nations have been held in the CTL. Over 10,000 people have viewed remote CTL presentations. In November 2000, the TIES-CTL Project received the State Justice Institute's Howell Heflin Outstanding Project award.

With credentials comes credibility and McMillan has plenty. He previously held positions with the US Department of Justice and the Los Angeles Superior Court. He was a keynote speaker at the Fifth National Court Technology Conference and a lecturer at the National Judicial College, University of

Southern California Judicial Administration Program, Smithsonian Associates, and many other national and international court, law and technology interest groups.

Conclusion

Few of us have ever heard of the open-source organizations mentioned above. They have success because they deliver government solutions. In so many ways, they have paved the way for the rest of us. I hope you enjoyed finding out about them as much as I did in locating them.

Tom Adelstein works as a Linux consultant in Dallas, Texas. His current interest lies in the field of eGovernment applications. Tom is involved in the launch of Government Forge, a Web site devoted to state and local governments interested in Linux and open source. He also is a member of the Open-Source Software Institute.

Archive Index Issue Table of Contents

Advanced search

The REDACLE Work-Flow Management System

Giovanni Organtini

Luciano M. Barone

Issue #118, February 2004

A MySQL-based system handles the data management, quality control and bookkeeping for building a new scientific instrument with 500,000 parts. Here's how you can adapt it for your manufacturing process.

Subnuclear particles, tiny objects indeed, need to be revealed and measured by huge detectors. This field is known as high-energy Physics (HEP), and experimental HEP is a cutting-edge science. It uses and promotes the most recent technologies, it invents new tools and it encourages knowledge exchange. For all of these reasons, HEP has long been the realm of open-source software.

The bad news is HEP has become increasingly complicated; what was built in a craftsman-like style yesterday is now an industrial process that requires dedicated management software, usually expensive. We are living this experience in our experiment: a large international collaboration engaged in the construction of a 12,500-ton detector, called CMS (Compact Muon Solenoid), scheduled to take data at the CERN, Geneva, Large Hadron Collider in 2007. Our group in Rome, endowed by the Italian Institute for Nuclear Physics (INFN) and located in the Physics Department of the University La Sapienza, is working on the construction of the electromagnetic calorimeter. The calorimeter is made from about 500,000 parts, including scintillating crystals and photo-detectors. This process requires data management, quality control and bookkeeping, all of which relies on work-flow management.

Figure 1. Today's high-energy Physics is an industrial process.

Work-Flow Management

A work-flow management system (WFMS) is "software that enables the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (www.e-workflow.org). Using a WFMS allows a coordinator to establish the flow of operations needed to realize a product. Operators are guided through the construction sequence, and unforeseen deviations from the sequence are avoided. Each operation generates data, such as measurements, comments and tags, that are recorded in a database.

Originally, a WFMS based on proprietary components was used in our production for about two years. It proved to be clumsy, slow, resource-demanding, hard to resume after hang-ups and troublesome to integrate with other tools. When the flow of incoming calorimeter parts became higher and the assembly rate could not catch up, we made the decision to develop our own solution, based on open-source components. Our requirements were to avoid the previous inefficiencies, to interface transparently with input and output data and to have a flexible solution. We chose to implement a system based on the LAMP (Linux, Apache, MySQL and Perl/PHP/Python) platform. Each component of LAMP has an important role: Linux and Apache provide the basic infrastructure for services and programming; MySQL is the back end of our WFMS; and Perl/PHP manage the interaction operator database.

REDACLE: the Database Design

Our WFMS is called REDACLE (Relational ECAL Database at Construction LEvel). In more detail, our requirements for the database design were:

1. High flexibility: the database structure should not change when adding new products or activities.
2. Ability to store quality control (QC) data: quality assurance is an important part of our work and collected data must be available to everyone for statistical analysis.
3. Variety of access: the database should be able to be queried through different methods, including shells, programs, scripts and the Web.

Requirement 3 automatically was satisfied by MySQL, and this fact, together with its simplicity and completeness, was the main reason we adopted LAMP. In order to satisfy the first two requirements, we developed a set of tables following a pattern, which is a common and standard way to solve a given problem, as in OO programming. We used the pattern called homomorphism, which is a simple representation of a many-to-one relationship. In practice, each part of the specific process with which we are dealing is represented in the database as records in two tables, an object table and an object definition table. Each object definition has an ID, actually a MySQL primary key number. Many objects share the same object definition, and the relationship between them is provided by a foreign key in the object table containing the corresponding definition ID.

An example might explain this design better. As stated in the introduction, our calorimeter is composed of many parts of different types. Each kind of part, such as a crystal, has many instances. The whole calorimeter has about 75,000 crystals. Parts and part definitions are kept in two separate database tables, as shown in Tables 1 and 2. Different instances of a part share the same part definition by the proper part ID in the partDefinition_id column. In these two tables, the part 33105000006306 is a type 1L barrel crystal, as shown by its partDefinition_id 195 found in the partDefinition table.

Table 1. The Part Table in REDACLE

ID	partDefinition_id
33105000006306	195
33105000006307	196
33105000006308	197

ID	partDefinition_id
33105000006309	198
33105000006310	196

Table 2. The partDefinition Table in REDACLE

ID	Name	Subname	Type
195	crystal	Barrel	1L
196	capsule	Barrel	T4
197	Alveola	Barrel	3
198	subunit	Barrel	5

The real benefit of this approach is flexibility. If, for any reason, new parts enter the game, the REDACLE database structure will not be modified. It is enough to add a new record to the definition table and relate it to new parts. But the REDACLE database is even more flexible; if we were building cars rather than calorimeters, the database structure would be exactly the same.

Activities are represented using the same approach: an Activity table holds instances of records described in the ActivityDescription table. Inserting a new activity within the work flow is a matter of supplying its description to the definition table and relating it to its occurrences in the Activity table. Again, with this design it is possible to describe a completely different business seamlessly. For mail delivery, for instance, the definition records could contain the description of the operations to be done on reception, shunting and delivery, while the Activity table could contain records with information about when a given operation was done on which parcel.

The work flow is defined by collecting several activity definitions and defining the order in which they should be executed. The interface software then checks that the activity being executed at a given time follows, in the work-flow definition, the last completed activity performed on a part. Activities can be skipped or repeated according to the interface software.

For quality control data we adopted the same homomorphic pattern by adding a further level of abstraction. We defined characteristics as data collected during a given activity performed on a part. The Characteristics table, however, does not store actual values, because they can be of a different nature—strings, numbers or even more complex types. The Characteristics table simply is a collection of keys: one of them links the characteristics to its definition in the charDefinition table. Actual characteristics are kept in separate tables according to their type.

Our process has three data types: single floating-point numbers, triplets of numbers and strings. The length of a crystal, for example, is a single number and is stored in the Value table. Some measurements are taken at different points along the crystal axis and in different conditions. The optical transmission, for one, is measured every 2cm at different wavelengths. It constitutes a triplet, the first number representing the position, the second the wavelength and the third the transmission. Each triplet is stored as a record in the multiValue table. The same is true for strings: operators perform a visual inspection of each crystal before manipulation, and they may provide comments to illustrate possible defects. In Tables 3 through 7, we show the above-mentioned tables for characteristics representation. The part 33101000018045 has been measured for length and transmission (TTO). Length is 229.7815mm in table value. The char_id field is 134821 pointing in the Characteristics table to charDefinition_id=6 corresponding to crystal length. The TTO is a set of triplets in the multiValue table. The visual inspection of that crystal resulted in the comment nonhomogeneous in the charValue table.

Table 3. charDefinition

ID	Description	Name	Unit	activityDef_id
2	result of visual inspection	VIS_I_OPER		2
6	crystal length	DL	mm	3
26	transversal transmission	TTO	mm#nm#%	4

Table 4. Characteristics

ID	charDefinition_id	part_id	activity_id
106035	2	33101000018045	10660
134821	6	33101000018045	16093
135252	26	33101000018045	16182

Table 5. Value

ID	x	char_id
37614	229.7815	134821

Table 6. multiValue

ID	x	y	z	char_id
748867	15	700	76.1	135252
748907	35	700	75.7	135252
748947	55	700	75.9	135252
748987	75	700	76.1	135252
749027	95	700	76	135252
749067	115	700	75.5	135252
749107	135	700	76	135252
749147	155	700	75.7	135252
749187	175	700	76.3	135252
749227	195	700	76	135252
749267	215	700	74.6	135252

Table 7. charValue

ID	value	char_id
2872	nonhomogeneous	106035

Again, this method makes REDACLE qualified for different types of businesses; in a dairy it could be used to record the bacterial load for each batch of milk, besides the producer (a character string), as characteristics. In addition, a completely new data type, such as pictures or sounds, could be added to the database without disturbing the schema simply by defining a new table. Adding pictures, for instance, implies the creation of a table with three fields: primary key, picture data as a BLOB and the relation to the Characteristics table, which is expressed by an integer ID. The MySQL code to create such a table is:

```
CREATE TABLE picture (
        id INT NOT NULL AUTO_INCREMENT,
        data BLOB,
        char_id INT,
        INDEX (char_id),
        PRIMARY KEY (id)
);
```

REDACLE Interfaces

In our application, humans interact with the database in a multitude of ways—with the MySQL client, C++ and Java programs, Perl scripts and PHP scripts through Web pages. The use of a Web browser to render a graphical user interface (GUI) provides considerable advantages. The GUI is portable and does not require installation of specific components, no time is wasted on graphics, and the Web browser environment is well known by now to both operators or customers.

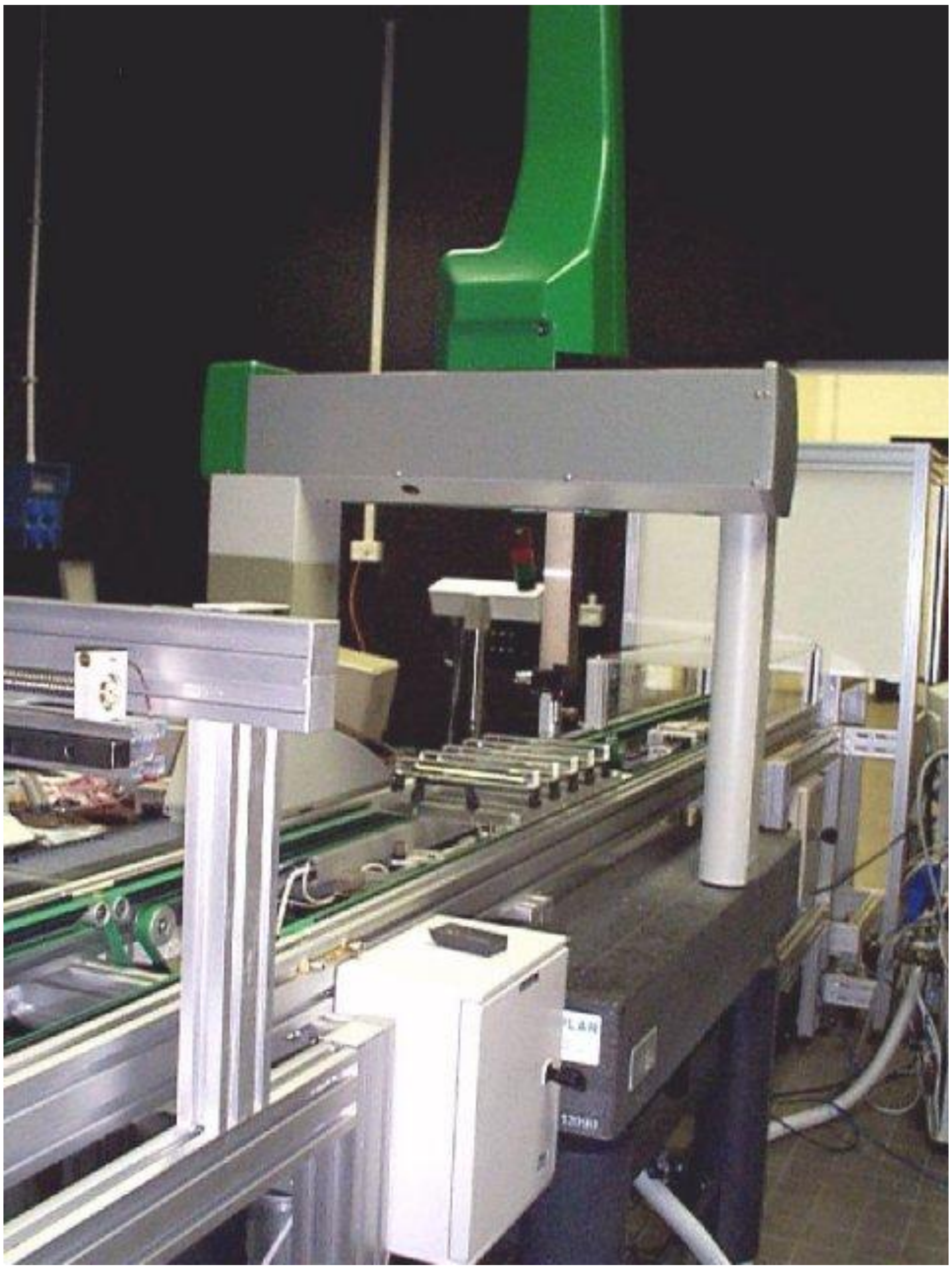


Figure 2. A set of five crystals being measured to determine their dimensions. Instruments need to report their measurements to the work-flow management system.

Another significant feature of REDACLE is its interface to other machines. During the calorimeter construction process, automatic machines take measurements of crystals and other parts without any human support (Figure 2). These machines, then, must be able to interact with REDACLE to learn the right sequence of operations to perform, to inform it about the start and the end time of the operations and to provide data to be stored as characteristics.

Our goal was to create a system that would allow almost any device to interact with REDACLE. We avoided imposing a given programming language or providing libraries for all the possible devices, because it does not scale with the market. Furthermore, some devices can be embedded systems with proprietary software.

We developed a dæmon called the Instrument Agent (IA) to act as an interface between REDACLE and instruments. The IA is a process that connects to an Internet port and is able to read ASCII characters and write them to that port. Instruments are required only to be able to connect to the network and send strings over the connection.

The sequence of operations is as follows:

- After connecting, an instrument declares the part on which it is operating.
- The IA queries the REDACLE database and searches for the last completed activity for that part in the work flow. The command the instrument should execute is stored in the database as a description field in the activityDefinition table.
- The IA sends the instrument the proper command.
- Upon recognition of the command, the instrument executes it, and the IA inserts a new activity in the REDACLE database after acknowledgement.
- At the end of the job, the IA updates the activity just inserted, marks it as FINISHED and gets the data from the instrument as XML-formatted strings.

The result of the activity may contain both multiValue and charValue fields. Single values are formatted as follows:

```
<RE><FI>field name<VA>field value</VA></FI>...</RE>
```

<RE> stands for result, <FI> is field and <VA> is value. From the field name, the instrument agent obtains the characteristics definition ID and fills in the appropriate table according to the field value format (value for numbers and charValue for strings). The multiValue table is populated if the result is of the form:

```
<RE><NT>ntuple name
```

```
<FI>field name<VA>field value</VA></FI>
...
</NT></RE>
```

<NT> here stands for n-tuple, a collection of n elements.

Instrument software developers need not have knowledge of the details of the REDACLE database; they simply have to be instructed on the string formats to be used. No libraries to link to the program are prescribed, nor files to be included. The programming language is not imposed. The only requirement is to be able to provide a network connection to the IA.

Besides the GUI and instrument interfaces, we developed a set of ancillary command-line scripts for administrators and coordinators. In addition, we created a small library to run C++ programs and Perl scripts over the database without needing to formulate SQL queries.

Experience and Perspectives

REDACLE was released in our laboratory four months after the first discussions of the project were held. The whole system contains about 10,000 lines of code in Perl, C++, PHP and Java. The resources needed to run the software are small compared to the ones requested by the former system, hosted on a dual 800MHz Pentium III server. That system saturated the CPU at about 100% and occupied almost all of the 512MB of RAM. We also needed to upgrade all the client PCs, doubling their memories to support Java GUIs. So we planned a server upgrade to a dual 1GHz Pentium III with 1GB of RAM to improve the performance of the previous system. When using REDACLE, we discovered, amazingly enough, that CPU load was negligible and the average memory usage was 140–200MB.

It became clear that we had a need for tools to import from or export to the previous database, which still was used in other labs. These tools were built in Perl quickly, to read or write XML files, and we were able to import all the old data into REDACLE in one day.

Currently, we have about 13,000 parts in the database. The stored characteristics are 97,000, each of which may be composed of several values, for a total database size of 50MB. Out of the 15 tables the multiValue table, containing more than 1,000,000 records, is the largest at 41MB.

But the most spectacular result was obtained by comparing the time spent by operators in the calorimeter assembling. Before the introduction of REDACLE, 25% of the operators' time was wasted in the interaction with the work-flow manager. Using REDACLE, the interaction between operators and the database takes a negligible amount of time, improving the overall detector assembling efficiency.

Moreover, operators soon familiarized themselves with REDACLE's flexibility and started requesting new tools and interfaces. What before required weeks to develop or might have been almost impossible to build, now can be implemented in a short time with REDACLE and LAMP—between a few hours and two or three days.

The extraordinary flexibility of the REDACLE database design makes it practical for many different business processes and industrial applications, clearly illustrating how open-source software can be superior to proprietary offerings. In the future, we plan to support even more complex work-flow models as well as a library of frequently used queries and functions to be employed in the development of other REDACLE-based projects.

Figure 3. Success: a finished module being loaded into a container for transportation to CERN.

Resources

The Cern Site: www.cern.ch

The CMS Experiment in Rome: www.roma1.infn.it/exp/cms

MySQL Reference Site: www.mysql.com

PHP Reference Site: www.php.net

The Workflow Portal: www.e-workflow.org

Giovanni Organtini (G.Organtini@roma1.infn.it) teaches Computing and Programming for Physicists at the University of Rome. He has been a Linux user both for research and teaching for nine years, and he has been married to Federica for ten. He is involved in the design and realization of particle detectors and advanced computing for high-energy Physics.

Luciano M. Barone (Luciano.Barone@roma1.infn.it) is Associate Professor of Physics at Rome University La Sapienza. Born to physics as a high-energy experimentalist, he quickly turned to computing applications in the same field. He likes to tackle large problems and tame them to the finest detail.

Archive Index Issue Table of Contents

Advanced search

Magnatune, an Open Music Experiment

John Buckman

Issue #118, February 2004

It's a record label, but it's not evil. How to build a business without ripping off artists and annoying customers, and why you might need three different kinds of Web server software.

Magnatune is an Internet music record label. It was born out of personal experiences from my wife releasing her CD on a British record label and some observations I'd gathered about the music industry. Magnatune is different from traditional labels in the following ways:

- We split the sale price of all purchases 50/50 with our artists.
- We sell only downloads and never use digital rights management (DRM). Purchasers may download albums as perfect-quality WAV or FLAC files, as high-quality variable bit-rate Ogg Vorbis files or MP3s or as 128k MP3s. Buyers can choose how much they want to pay, from $5 to $18 US.
- You can listen to all our music as streaming 128k MP3s (entire albums, not samples) as well as on Shoutcast MP3 stations. Two clicks on Magnatune queues a never-ending selection of our music in the genre of your choice. Our assumption is you eventually will hear something you like and want to buy it.
- All our free music is licensed using the Creative Commons Attribution-NonCommercial-ShareAlike license. This allows noncommercial use of the music at no cost, as well as derivative works, as long as the same Creative Commons license applies. If someone uses our music for commercial purposes, they have to license it for a modest fee. All our licensing is done on-line with a standard rate calculator, and we don't discriminate based on the kind of use. For example, we can't and won't block your use of our music if we don't agree with your views or your musical style.
- We're successful and profitable. Our top artists are making about $6,000 US a year in royalties, while the average musician makes about $1,500 US

> per year. We work directly with musicians or musician-owned labels, never with labels who funnel the money to themselves.

Magnatune was born out of personal experiences from my wife releasing her CD on a British record label and some observations I'd gathered about the music industry.

When my wife was signed to a British record label, we were really excited. In the end, she sold 1,000 CDs, lost all rights to her music for ten years (even though the CD has been out of print for many years) and earned a total of $45 in royalties.

The record label that signed her wasn't evil. They were one of the good guys, and gave her a 70/30 split of the profits, of which there were none. The label got battered at every turn: distributors refused to carry the label's CDs unless it spent thousands on useless print ads, record stores demanded graft in order to stock the albums and, in general, all forces colluded to destroy this small, progressive label.

Industry Observations

Radio is boring. Everyone I know listens to interesting music, yet good music rarely is played on the air. Most musical genres barely are visible in record stores, and they are totally absent from the airwaves. These days, radio plays mostly Country, Pop and Rock, with a little bit of dull, safe Classical thrown in.

CDs cost too much, and artists receive only 20 cents to a dollar for each CD sold, if they're lucky. And, most CDs quickly go out of print. I buy more CDs from eBay than from Amazon.

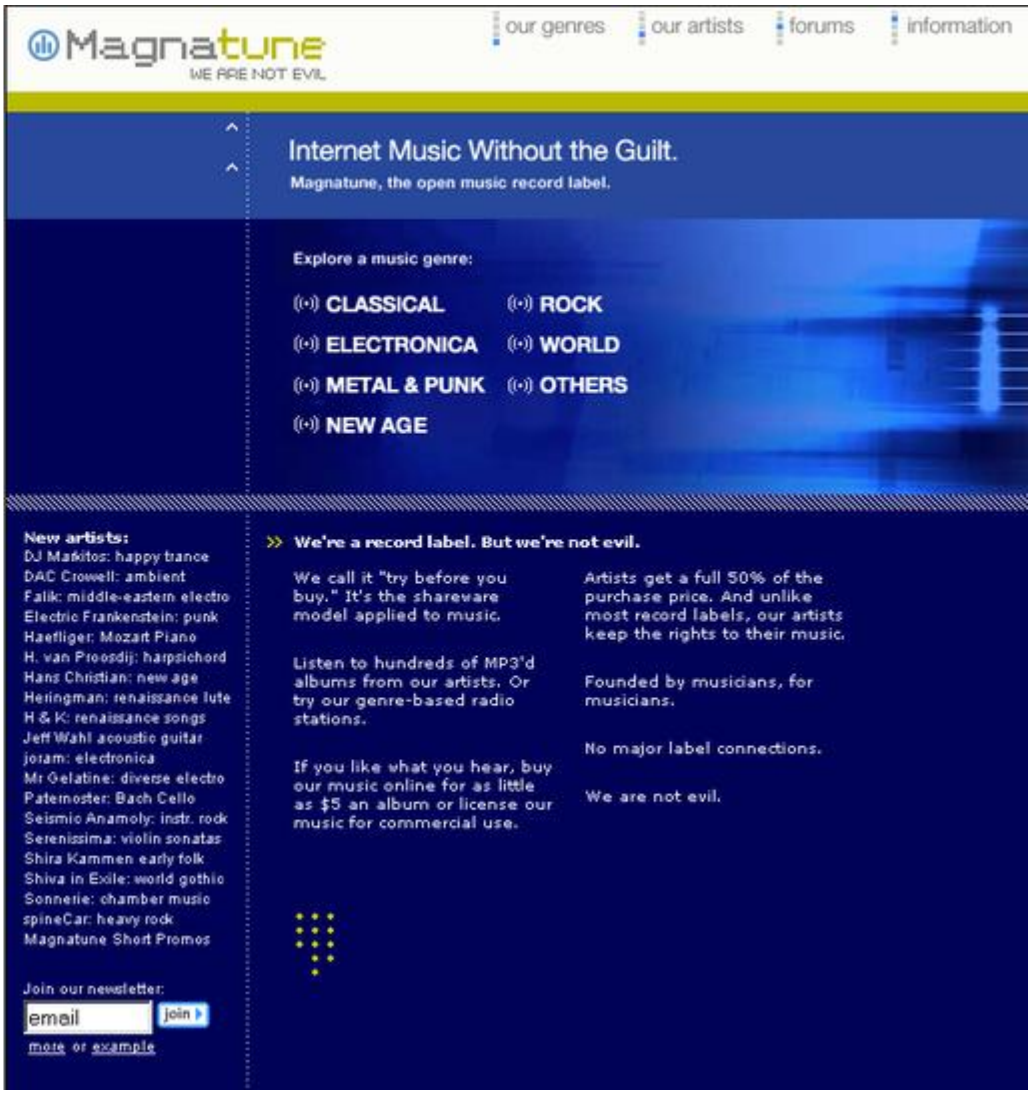
On-line sales, such as Amazon.com, often cost artists 50% of their already pathetic royalty, due to a common record contract provision. International sales and markdowns often net the artist no royalties.

Record labels lock their artists in to legal agreements that hold them for a decade or more. If the agreement is not working out, labels don't print the band's recordings but nonetheless keep artists locked in to the contract, forcing them to produce new albums each year. Even hugely successful artists often end up owing their record labels money because the advances they're initially paid are structured as a loan to the artist. Furthermore, the label does its accounting in such a way that profits rarely are shown on the record company's books, hence no money exists to pay down the advance.

Using the Internet to listen to music usually is tedious. There are too many ads, too many clicks and the sound quality is not great. Simply put, it's too much work for not enough reward. A well-run Internet radio station, such as Shoutcast or Spinner, solves that, but the entrenched record industry wants to kill them too, through onerous licensing terms and annoying DRM schemes.

Given all these factors, I thought, “why not build a record label that has a clue?” Create a label that helps artists get exposure, make at least as much money as they would make at traditional labels and get fans and concerts. Magnatune is my project. The goal is to find a way to run a record label in the Internet reality: file trading, Internet Radio, musicians' rights, the whole nine yards. These aren't things that can be ignored.

The Home Page



Internet Music Without the Guilt.

Magnatune, the open music record label.

New artists:
DJ Markitos: happy trance
DAC Crowell: ambient
Falik: middle-eastern electro
Electric Frankenstein: punk
Haefliger: Mozart Piano
H. van Proosdij: harpsichord
Hans Christian: new age
Heringman: renaissance lute
H & K: renaissance songs
Jeff Wahl acoustic guitar
joram: electronica
Mr Gelatine: diverse electro
Paternoster: Bach Cello
Seismic Anamoly: instr. rock
Serenissima: violin sonatas
Shira Kammen early folk
Shiva in Exile: world gothic
Sonnerie: chamber music
spineCar: heavy rock
Magnatune Short Promos

» We're a record label. But we're not evil.

We call it "try before you buy." It's the shareware model applied to music.

Listen to hundreds of MP3'd albums from our artists. Or try our genre-based radio stations.

If you like what you hear, buy our music online for as little as $5 an album or license our music for commercial use.

Artists get a full 50% of the purchase price. And unlike most record labels, our artists keep the rights to their music.

Founded by musicians, for musicians.

No major label connections.

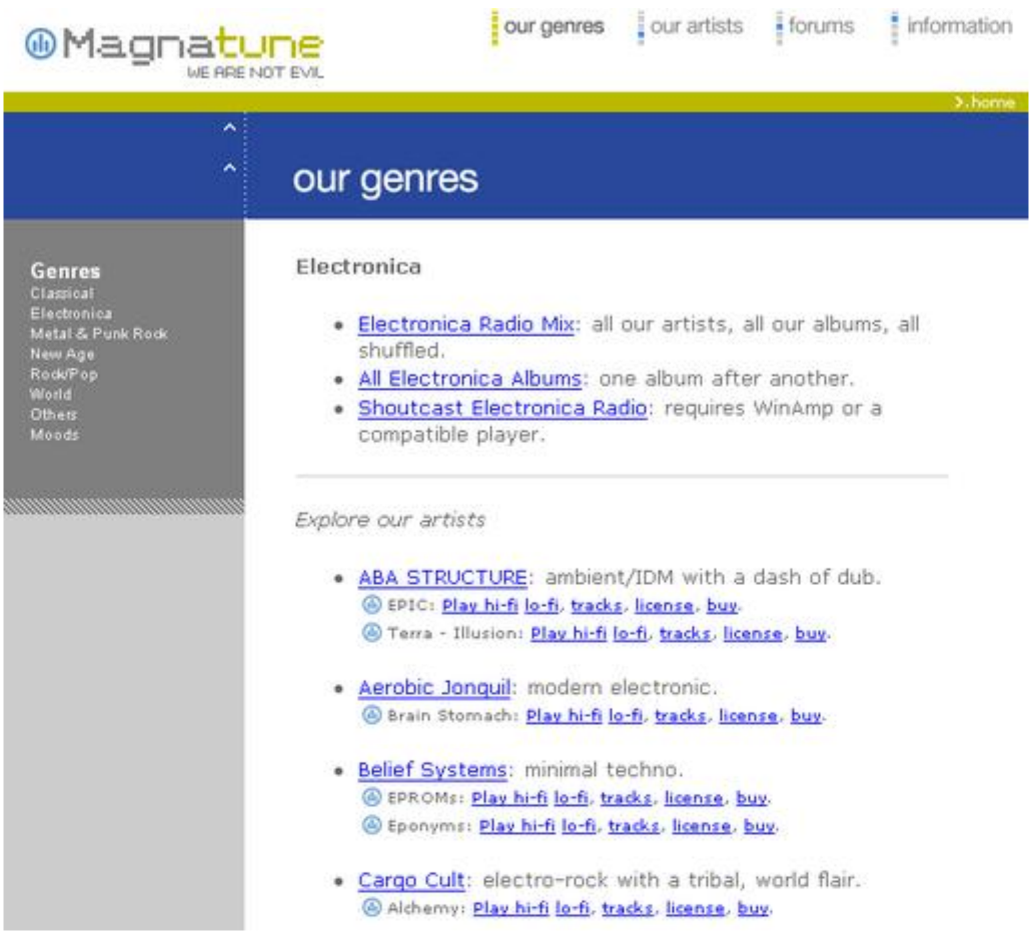
We are not evil.

Figure 1. The Magnatune home page (magnatune.com) answers visitors' top eight questions.

I believe that the home page of any successful company, project or organization needs to address eight issues immediately. As shown in Figure 1, Magnatune's home pages answers all eight:

1. Where am I? — a graphic logo on the top left or top right does the trick. It's even better if you have a catchy line. For Magnatune, it's “We are not evil”.
2. Why should I care? — a one-line description of what you do and, if possible, why someone should be interested. For Magnatune, it's “Internet music without the guilt” followed by “Magnatune, the Open Music Record Label”.
3. What do you want me to do? — for first-time visitors, it should be clear what the next step is. For Magnatune, I want people to listen to the music immediately, so it says “Explore a music genre: Classical, Electronica, Metal & Punk, New Age, Rock, World, Others”.
4. Why is this cool? — there are way too many sites on the Internet, and people have a limited amount of time. You've got the visitor's attention for a few seconds, so you need to explain quickly why this is something he or she wants to support. If you're doing e-commerce, expect that your visitors are jaded. If you answered the second question well, you've got another 30 seconds of their attention. Magnatune starts with: “We're a record label. But we're not evil. We call it 'try before you buy.' It's the shareware model applied to music.” The concepts of record label, not evil and shareware are an odd combination, so now they're interested.
5. What's new? — give people an incentive to come back to your site by making it easy to see what's changed. There's a lot of new stuff at Magnatune (new press coverage, for example), but most people care only about our new artists and albums, so that's what's on the home page.
6. Newsletter signup — every Web site should have a newsletter. If you put the signup on the home page, you can expect 2% to 5% of Web site visitors to sign up.
7. I want to know more — an “about” section also is crucial. The founders should explain why they created the site, project or company.
8. I want to steer — despite all these hints on what to do next, visitors often want to decide for themselves where to go. On Magnatune, 15% of people coming to the home page click on the Artists tab. Make the major site navigation options clear.

The Genre Page



Electronica

- Electronica Radio Mix: all our artists, all our albums, all shuffled.
- All Electronica Albums: one album after another.
- Shoutcast Electronica Radio: requires WinAmp or a compatible player.

Explore our artists

- ABA STRUCTURE: ambient/IDM with a dash of dub.
  EPIC: Play hi-fi lo-fi, tracks, license, buy.
  Terra - Illusion: Play hi-fi lo-fi, tracks, license, buy.
- Aerobic Jonquil: modern electronic.
  Brain Stomach: Play hi-fi lo-fi, tracks, license, buy.
- Belief Systems: minimal techno.
  EPROMs: Play hi-fi lo-fi, tracks, license, buy.
  Eponyms: Play hi-fi lo-fi, tracks, license, buy.
- Cargo Cult: electro-rock with a tribal, world flair.
  Alchemy: Play hi-fi lo-fi, tracks, license, buy.

Figure 2. “Play” links on the Electronica page (magnatune.com/genres/electronica) offer immediate gratification.

My main gripe with most music sites is they take too much of my time, when what I want to do is play music and get back to my work. My second gripe is they usually don't give you enough music and not at a high enough quality to make a decision. Most of the people I know buy music by hearing it first, either on the radio, at a concert, at a friend's house, at a restaurant and so on. Because only a small number of visitors to a music site already know the bands, isn't it reasonable to let visitors listen and make up their minds? At Magnatune, you can listen all you want, with minimal effort, to high-quality 128k MP3s.

Once a user clicks on the Magnatune home page to select a genre, such as Electronica (see Figure 2), a Web page is displayed that shows four main choices: 1) listen to every album in Electronica, one album after another, 2) listen to a mix of our Electronica artists, 3) listen to the entire album of any of our Electronica artists and 4) click on an artist to learn more. The first three choices offer users immediate gratification by allowing them to hear music immediately.

People would prefer to find music from their friends or on their own than be force-fed by the major music outlets—radio and MTV. In the 1980s, most software couldn't be evaluated before it was purchased, yet today no one buys any software without first trying it out. Eventually, all music will be shareware: the competitive advantage simply is too great.

Software Used

Magnatune runs on open-source software. Five 2.4GHz 1U rackmount boxes run Linux. Disk storage is provided by hardware RAID on four rackmount drive arrays, each with seven drives for a total of 28 drives. Each array is configured for RAID5 with one hot spare.

Apache 2 running PHP and OpenSSL serves all the HTML pages. When Magnatune was Slashdotted, I found that Apache could not keep up with the load for images. All HTTP image requests now are off-loaded to AOLserver, which had the lowest latency to serve images at high loads.

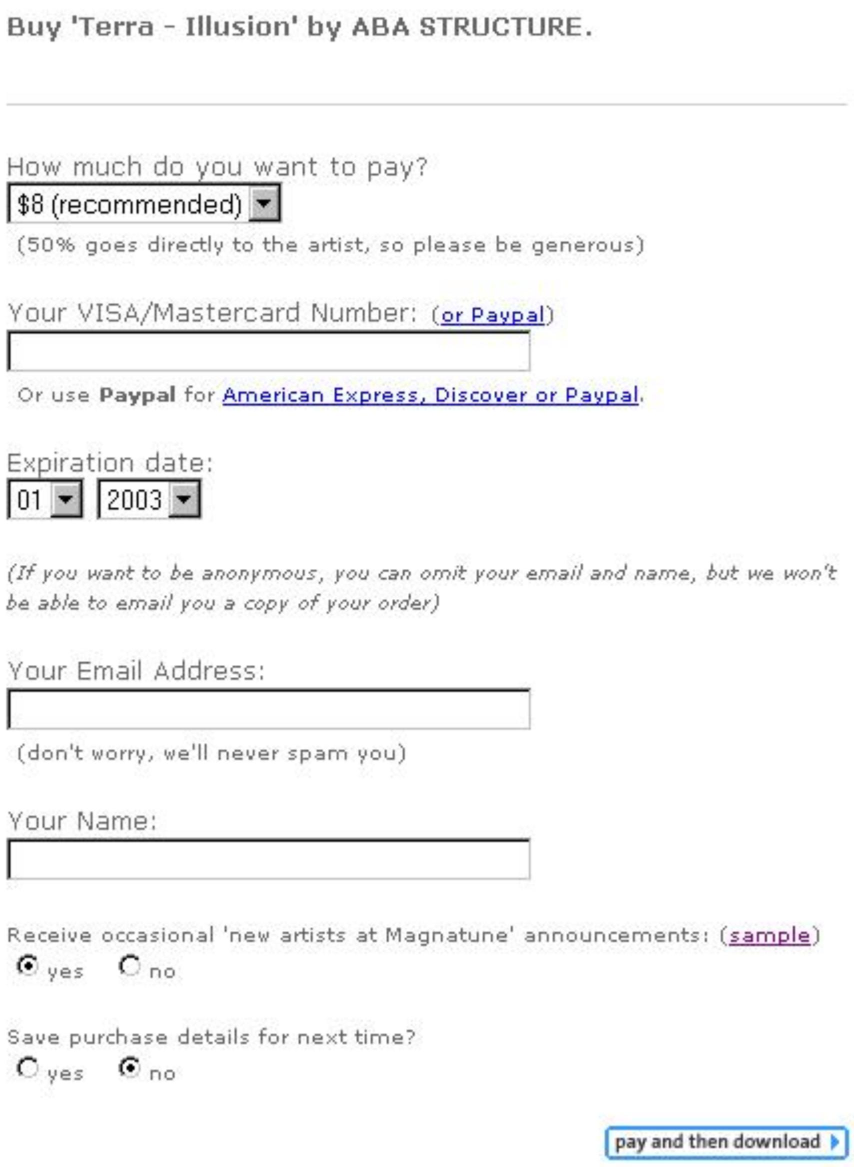
Mathopd, a single-threaded asynchronous HTTP server, is used to serve all MP3 files, as it is extremely scalable with large files. We customized Mathopd to return the same `Expires: HTTP` response header on images that Yahoo uses. Mathopd has more latency than AOLserver, which is why we don't use it for serving small images.

All Web pages in Magnatune run PHP. Purchases are logged to a MySQL database. A Perl script creates the track listings and m3u playlists. Tcl scripts handle all other tasks, such as making ZIP, Ogg Vorbis and FLAC files and making per-album password configuration files for Apache. Apache HTTP passwords are used to protect all purchasable downloads. We use rsync to distribute files among the servers.

E-Commerce

Several unusual features can be found in Magnatune's music-download purchasing process. Buyers determine how much they want to pay, we don't use a shopping cart, we support PayPal and anonymous Visa purchases are allowed.

Clicking the Buy button brings up a page with the question “How much do you want to pay?” and offers choices from $5 to $18, with $8 as the recommended price (see Figure 3). A disclaimer states, “50% goes directly to the artist, so please be generous”. The average purchase price in September 2003 was $9.82, which shows that when given a choice, buyers prefer to pay more than we ask.


Buy 'Terra - Illusion' by ABA STRUCTURE.

How much do you want to pay?
$8 (recommended)
(50% goes directly to the artist, so please be generous)

Your VISA/Mastercard Number: (or Paypal)

Or use **Paypal** for American Express, Discover or Paypal.

Expiration date:
01 2003

(If you want to be anonymous, you can omit your email and name, but we won't be able to email you a copy of your order)

Your Email Address:

(don't worry, we'll never spam you)

Your Name:

Receive occasional 'new artists at Magnatune' announcements: (sample)
[✓ yes] no

Save purchase details for next time?
yes [✓ no]

pay and then download

Figure 3. Because 50% goes to the artists, buyers choose to pay more than the minimum price.

I decided not to use a shopping cart at Magnatune because of the widely discussed problem of shopping-cart abandonment. When people are excited about an album and click the Buy button, that's the time to capture the sale. With a shopping cart in place, a user often goes looking for more things to add and that initial buying compulsion is gone if they don't find something.

On any given day, we find that 30%–50% of purchasers pay with PayPal. I believe many people prefer PayPal because with a credit card the merchant still has your credit-card number after the transaction completes. Your personal data is out of your control. With PayPal, the transaction ends with your payment, and no future risk is possible.

PayPal offers two ways for merchants to be notified of payments. IPN is a system where PayPal issues an HTTP callback to your Web server with a specific

ID. Your Web server software then takes the ID and calls PayPal over HTTP to fetch transaction details. This is a highly secure option, but it has two problems: it is significant work to code, and it doesn't allow an immediate download after purchase, because your Web server needs to wait for the notification. Many shopping-cart programs support IPN and deliver download instructions by e-mail. We do not use an IPN/e-mail-based system, because we found that ISP-based antispam programs blocked many of our download-instruction e-mails, which led to very unhappy buyers who were not aware of the blocked e-mail.

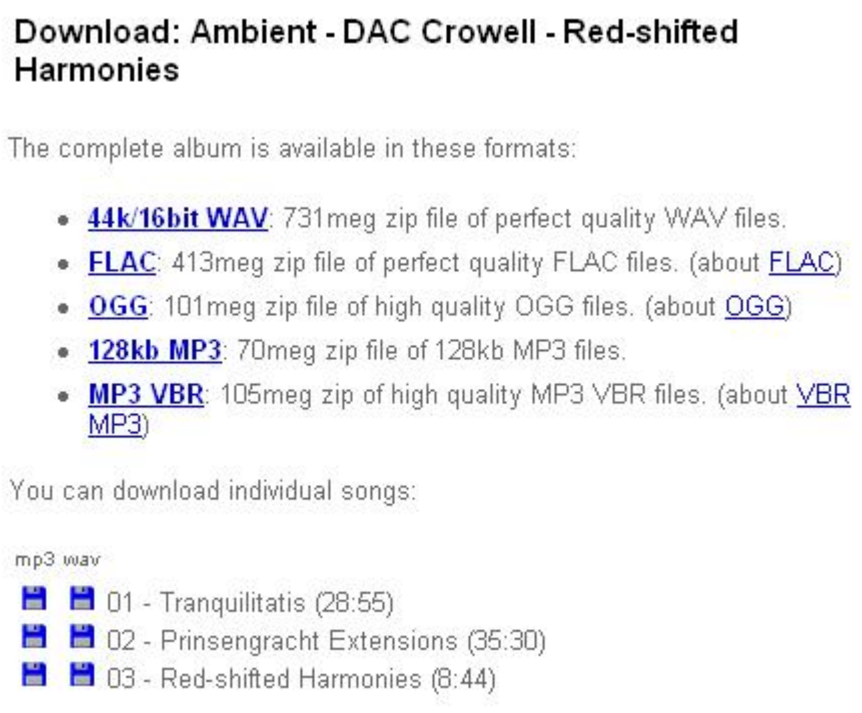
We use the second, simpler PayPal system, in which PayPal sends the user back to our site with HTTP POST data after a transaction occurs. This is simpler to program and fairly reliable. Not all PayPal payments are instantaneous, however. The payment does not arrive in the vendor's account for several days and may not come at all, so the merchant has to decide whether to allow the immediate download. We decided to trust our users and give them the download immediately.

Our credit-card processor would like us to ask for a name, phone number and postal address (as well as three-digit AVS) on credit-card transactions, but we're under no obligation to do so. Our credit-card company charges us a 1% fee for not having this data at checkout, but on a $10 purchase this is only 10 cents. This fee pales in comparison to the 25-cent Visa fee and the 25-cent Internet-gateway fee, so we feel it's well worth it.

On the Magnatune Buy page, we require only a customer's credit-card number and expiration date—that's all we need to complete the transaction. We do ask for a name and e-mail address, stating: “If you want to be anonymous, you can omit your e-mail and name, but we won't be able to e-mail you a copy of your order.” Asking for a minimal amount of data at purchase time makes it quick for people to fill out the purchase form. Additionally, not requiring an e-mail address means the buyer doesn't need to worry about e-commerce spam resulting from the purchase.

After the Payment

Immediately after making your purchase, you are sent to a Thank You page that contains a URL for downloading the music. A simple four-character HTTP name and password is given. The instructions also are e-mailed if an e-mail address was provided with the purchase. The download page provides the album in a variety of formats (see Figure 4). You can download a perfect-quality WAV or FLAC file and have an exact replica of the original CD. Ogg Vorbis, MP3-128 and MP3-VBR files also are provided. You can download any and all variations you like for as long as you like (passwords don't expire). The entire album is available as a single ZIP file, for ease of downloading. The ZIP file provides a modest amount of compression, about 10% on WAVs. More importantly, it means you can click Download on one file, let it run for an hour (on a DSL or cable line) and have the entire album without any more fuss.



Download: Ambient - DAC Crowell - Red-shifted Harmonies

The complete album is available in these formats:

- 44k/16bit WAV: 731meg zip file of perfect quality WAV files.
- FLAC: 413meg zip file of perfect quality FLAC files. (about FLAC)
- OGG: 101meg zip file of high quality OGG files. (about OGG)
- 128kb MP3: 70meg zip file of 128kb MP3 files.
- MP3 VBR: 105meg zip of high quality MP3 VBR files. (about VBR MP3)

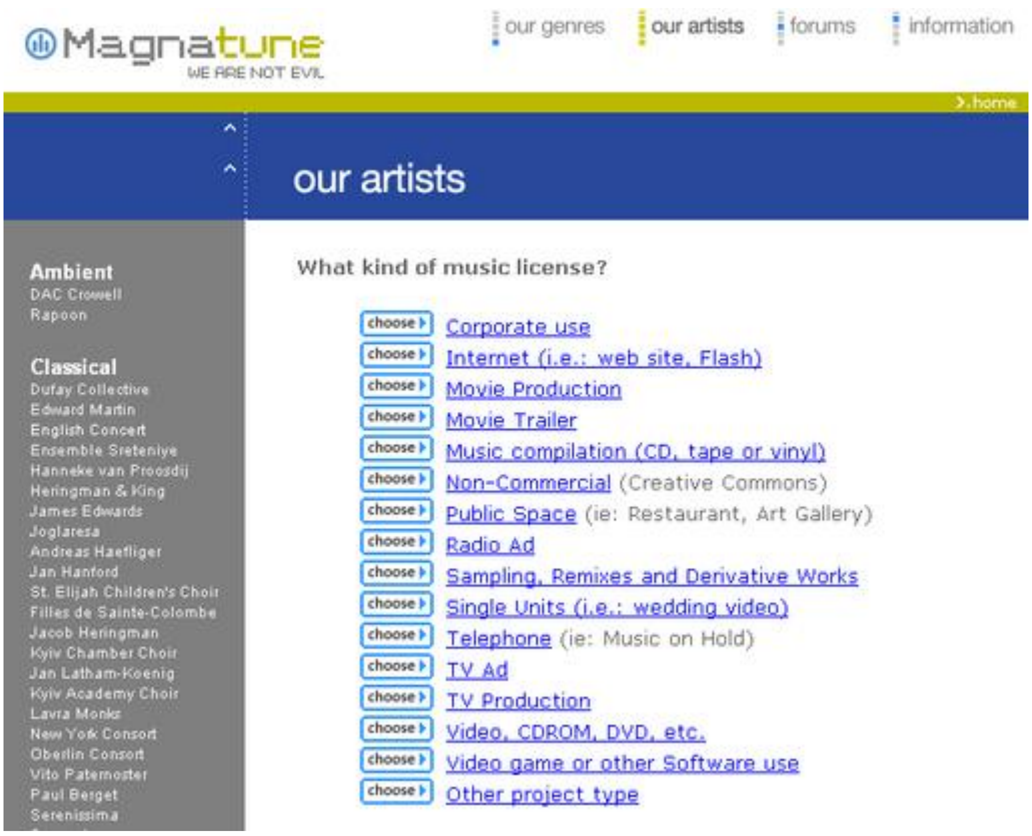
You can download individual songs:

mp3	wav	
💾	💾	01 - Tranquilitatis (28:55)
💾	💾	02 - Prinsengracht Extensions (35:30)
💾	💾	03 - Red-shifted Harmonies (8:44)

Figure 4. It's download time. Pick a format, any format.

Licensing

Clicking the License button next to any album brings up a page (see Figure 5) with the question “what kind of music license?” Sixteen different scenarios are displayed, such as Movie Use, Radio Advertising, Web Site, CD Compilations and even On-Hold Music for telephone systems. The connected page outlines the relevant variables for that scenario, and a price quote is given. The user enters the project details and then is sent to a Check Out page with a music license agreement, which is valid as soon as the fee is paid.



Ambient
DAC Crowell
Rapoon

Classical
Dufay Collective
Edward Martin
English Concert
Ensemble Sreteniye
Hanneke van Proosdij
Heringman & King
James Edwards
Joglaresa
Andreas Haefliger
Jan Hanford
St. Elijah Children's Choir
Filles de Sainte-Colombe
Jacob Heringman
Kyiv Chamber Choir
Jan Latham-Koenig
Kyiv Academy Choir
Lavra Monks
New York Consort
Oberlin Consort
Vito Paternoster
Paul Berget
Serenissima

What kind of music license?

- choose ▸ Corporate use
- choose ▸ Internet (i.e.: web site, Flash)
- choose ▸ Movie Production
- choose ▸ Movie Trailer
- choose ▸ Music compilation (CD, tape or vinyl)
- choose ▸ Non-Commercial (Creative Commons)
- choose ▸ Public Space (ie: Restaurant, Art Gallery)
- choose ▸ Radio Ad
- choose ▸ Sampling, Remixes and Derivative Works
- choose ▸ Single Units (i.e.: wedding video)
- choose ▸ Telephone (ie: Music on Hold)
- choose ▸ TV Ad
- choose ▸ TV Production
- choose ▸ Video, CDROM, DVD, etc.
- choose ▸ Video game or other Software use
- choose ▸ Other project type

Figure 5. Making a movie? Need legal music for your music-on-hold system? Try the licensing page for each album.

What's Next?

Since its launch in May 2003, Magnatune has been discussed on everything from Slashdot, Fark and BoingBoing to *Wired* and NPR. During these press events, we've experienced huge bursts of Web traffic. I run five servers at two different locations, each with 100Mb feeds, and send load-balanced pages with dynamic PHP URLs to each of them. I've found this to be sufficient for the current demands. However, as Magnatune has been growing by 30% in traffic and revenue each month, I'm seriously looking at peer-to-peer as a load balancing and scalability solution.

As I'm writing this article, Magnatune has about 60 musicians and 130 albums and is growing by about 15 musicians and 30 albums per month. My top musicians should see about $6,000 US of royalties in a year. On average, musicians should see about $1,500 US each year. If the 30% growth holds up, I'll be able to pay my musicians even more. More than anything, that's what excites me: that I'm able to have a material impact on these musicians. They are excited by the success, people are hearing their music, and the artists have the financial wherewithal to continue recording great albums.

John Buckman is the founder of Magnatune, an Internet-based record company. He is the Webmaster of Piazzolla (www.piazzolla.org), a Tango music site, runs the Internet Lute Society (www.lutesociety.com) and co-runs www.JSBach.org, the main Web site for the classical composer J. S. Bach.

Archive Index Issue Table of Contents

Advanced search

DIY-IT: How Linux and Open Source Are Bringing Do-It-Yourself to Information Technology

Doc Searls

Issue #118, February 2004

Follow the conventional IT media and you'll miss the new level of self-reliance and participation in Linux at companies large and small. Executives from Ernie Ball, Morgan Stanley and Ticketmaster explain the shift to “do-it-yourself”.

Without a doubt, Linux and open source are changing IT (Information Technology) at companies of every size. But how? When you read the IT magazines, go to the IT conferences and listen to the IT analysts, you get the same message you got ten or twenty years ago: vendors are in charge.

Of the 62 stories in Computerworld.com's current Week In Review (mid-November 2003), 51 stories, or 82%, are either about vendors—“Red Hat goes Live with Fedora”—or refer to vendors in their headlines—“IBM's Palmisano says US must innovate to keep jobs.” In reality, though, vendors' and customers' IT worlds are steeped in a variety of development communities. Both vendors and customers develop goods for themselves, as well as for sale and for use by the rest of the world.

Over the past year, I've been on assignment by *Linux Journal* to study what's really happening in the IT marketplace and the deeper roles played by Linux and open-source development in that marketplace. My first report was “How Linux Makes Companies Smarter” /article/6585, in the July 2003 *Linux Journal*. This second report focuses on changes in IT itself. What I've found is an increasing reliance on personal and development community initiative and the freedom and trust making that possible. In sum, what I'm seeing is a do-it-yourself movement in companies everywhere, a growth in self-reliance I'm calling Do-It-Yourself IT (DIY-IT).

Phil Moore, the Executive Director of Enterprise Application Infrastructure for Morgan Stanley and Company, explains:

> Open source has lowered the threshold at which do-it-yourself is possible. You can't do everything with building blocks from vendors. They pretend they're selling you a prefab building and they're not. They're selling you pipes and fittings and stuff to put it together.
>
> In reality, to build an enterprise, you have to have a set of experts in your IT shop who can put it all together. Certainly, historically you need a lot of expertise to get anything done, because this stuff really isn't easy to put together. But if you've been led by the vendors to believe that everything dovetails together nicely, like you see in the .Net ads, or in any major marketing campaign that promises nirvana, you've got a problem.
>
> You always need a certain amount of do-it-yourselfness. Consultants don't walk in, deliver an enterprise and walk out, saying “call me in six months for an upgrade.” It's organic. An enterprise is changing constantly. Even the walls in your house right now are on their way to needing another paint job.

Although DIY-IT involves a reduction in dependency on vendors, it doesn't mean vendors are bad or that they don't play extremely important roles in the marketplace. It does mean that the marketplace no longer belongs to them. It means a new balance of power exists between supply and demand as does a new division of responsibilities between vendor, customer and development communities. It seems the software business is growing up.

In this report, we look at several key factors involved in the DIY-IT movement: what leadership really means, the role of the Net, the rewards of courage, the cost-savings imperative, valuing talent, where we stand, untold stories and perspective.

Follow Which Leader?

Looking to leaders for leadership is natural. But what about the leadership of developments that have no direct leader—developments where the leadership comes largely from within, from shared conviction and the practices that express it?

That's what we have with Linux, with free software and with open source. Linux is a development project, not a company. It is not contained by a corporate structure. Like a tectonic plate, it is held together by cohesion more than by organizational forces. Free software and open source are value systems and development methodologies. To treat them strictly as populations or as classes of goods is to miss the nature and scope of what they're about.

The Net not only supports much of what we now take for granted in the technical world, it puts everybody and everything in a position to get more connected, more informed, more intelligent. Public bits outsmart secret ones, even if secret ones still have economic and other forms of value.

Craig McLane, VP of Technology at Ticketmaster, puts it this way:

> The best thing about the Internet, to me, is it mitigates tremendously the friction imposed by time and distance. You can take people who have individual passions and great talents, unite them and eliminate many of the obstacles to communications.
>
> The Open Source community truly is global, and that matters an awful lot to us. We're an international company and we like access to very smart people worldwide as we grow—people who still are accessible and relatively close-knit.
>
> The notion that you can take part in this community, do great things, be supported and foster this whole breeding ground of innovation is absolutely incredible.

The Networked IT World

Trying to imagine a civilized corporate or governmental organization today that is not sustained by the Net, and therefore also by open-source software, is almost impossible. Organizations everywhere also are coming to recognize that they depend on open-source values as well as open-source talent and code. In just the last year it became clear to governments around the world that their computing infrastructure needed to be built on stuff that is open, that has no secrets. They want to be able to inspect it to make sure it's sound, reliable and open to improvement.

Take Sterling Ball, the leader and namesake of Ernie Ball, the guitar string maker. You can't press SCAN on the car radio without hearing an Ernie Ball string. What makes Sterling Ball an open-source revolutionary isn't his technical chops, it's his independence—his guts. This became evident after Ernie Ball was raided in 2000 by federal marshals as part of an unannounced software audit by the Business Software Alliance (BSA), which found unauthorized “pirated” software on some of Ernie Ball's computers. The BSA still brags about the raid on its Web site. Here's how Sterling Ball described it when I interviewed him at LinuxWorld in August 2003:

> A disgruntled ex-employee saw a nail-your-butt opportunity, so he called the BSA. I was sued under federal seal. There was no warning. We were raided at 10 am on a Friday. We were shut down and ordered not to touch our computers. There were armed

> marshals. Our employees were sitting there going "What's the matter? Is our company criminal? Are we crooks?" Then they sent out press releases....It's coincidental that they always send these out after a business is closed.
>
> We're the number one employer in terms of manufacturing in San Luis Obispo. We're a big fish in a little pond. The headline reads, "Ernie Ball Raided for Piracy", and the story says, "Company officials unavailable for comment". Well, no [surprise]. I was at home. And I never say "No comment". So, when it came time to tell my story, I said, "They came for bear and got squirrel."

Ernie Ball went to court and paid a fine, but that didn't end the matter:

> The worst thing was when Microsoft printed a four-color reproduction of that newspaper article on an executive's desk, sent it to every registered Microsoft user and said "Don't get caught like Ernie Ball—a fine company that found out just how hard it is to stay compliant. Call us. We'll give you a free audit and sell you software at 20% off." Keep in mind that we already had downloaded the BSA self-auditing software and it didn't work. This was fear-based marketing, with government help.

Sterling Ball didn't get mad or get even—he got out:

> Everybody thought I was crazy. The IT people thought they were going to get fired. I said "no", because I've never seen any greater programming in the world than "You can't do business unless you've got an office suite on your desk." Hey, I'm talking here at LinuxWorld because I changed my word processor. The solution everybody [at our company] uses is a cocktail of open-source stuff. Nobody showed us how to do it. We had to figure it out ourselves.

Today Ernie Ball's servers run Red Hat Linux. Its desktops are GNOME on thin Sun clients, with applications that run off a Linux server. The company time clock and security software run on Linux. The company e-mail is Ximian's Evolution, and their office suite is OpenOffice.org.

The lesson here isn't about technology. The lesson is about independence, integrity and the courage to break free of mental programming. It's a lesson about the souls of individuals, of organizations and of a marketplace that still thinks vendors are in charge, even though the success of the Net and the Open Source movement prove they are not.

Breaking free is akin to awakening, and it doesn't happen only for companies the size of Ernie Ball. Take Ticketmaster, for example. Here's Craig McLane again, speaking at LinuxWorld:

> We support 8,000 clients in ten countries. In 2002 we sold 95 million tickets through channels that represented over four billion US dollars. This puts us in the top 25 of all retail Web properties—actually number two, between Dell and Amazon. So we're doing a lot of business, but on behalf of other people who have entrusted us with their business.
>
> We have 3,500 outlets, 19 call centers and the Ticketmaster.com Web site, which does about 50% of our business. We also provide box-office solutions. If you've ever purchased a ticket at a box office, that's also a Ticketmaster system, with the same inventory bucket.
>
> Our product and technology organization is the cornerstone of Ticketmaster as a company. We've got 250 people devoted to product and technology in an organization that has about 2,000 full-time employees.
>
> We provide solutions and systems, but we also support those 8,000 clients. In many cases, because of the nature of the business—highly customized, highly variable traffic and all kinds of strange configurations that actually are much different from any other retail businesses—there are no commercial solutions available for what we need to do.
>
> In fact, we are one of the first application service providers: extending the service to thousands of clients, actually writing the code, hosting the system, providing the customer service and charging a fee per unit sold. We have to build high-volume systems for very specific and peculiar businesses. Open source allows us to do [this] as well or better—at least in our experience—at half the price of commercial solutions.
>
> That's why Ticketmaster converted the Ticketmaster.com site primarily to open-source technologies over the course of the last 18 months.

McLane showed a small spreadsheet at this point in the talk (Table 1) that outlined the cost of the company's computing needs in terms of open-source and proprietary options. He continues:

Table 1. Ticketmaster Computing Options

Ticketmaster.com	Open Source	Proprietary
400 PC-based systems	$1,000,000	$1,000,000
Operating system	$0	$600,000
Web server software	$0	$120,000
Database software	$0	$240,000
Total	$1,000,000	$1,960,000

> It's really all about licensing costs. We buy the the same class of machine, same configuration, from the same vendor. But we're using all open-source technologies in these areas, and we pay nothing for licensing. So you can see that we save 50 cents on every dollar we invest and get the same or better performance. And we see better support from the community than we typically get from commercial vendors.
>
> The Web site costs, in hardware and capital development, a couple million bucks. So it's a small fractional part. It doesn't materially change the business from Wall Street's perspective or from the CFO's perspective. But it matters to us because we can use the money that's made available to employ more smart people. And that's really the key.

Valuing Talent

To Craig McLane, open-source human resources are collective as well as individual. To explain, he quotes T. S. Elliot: “No poet, no artist of any art, has his complete meaning alone”. He adds:

> By providing really robust tools to our people and by pulling in more people who are interested in solving problems, who can work autonomously, but who like to have the benefit of a larger community while they're working....
>
> [People like] Stas Beckman are doing core work for us...mod_perl 2.0. [He's] doing a lot of the work that will benefit the community, which benefits us. He's also working on database connection pooling....So we get to take the lead on doing some things that have broad general benefits for everybody. Geoffrey Young is going to start working with us. He's going to be working on things that can be done autonomously, that benefit the community but also provide immediate benefit for our business when deployed in very specific ways.

> Our teams have ownership over their tools. They also don't have an excuse. They can't say that a vendor doesn't have an answer or isn't getting back on the phone. Because everybody knows that there's a community out there and you have access to the source. Everything is in front of you. So there's an accountability that's reinforced when you have source code and a community that knows so much and is so willing to respond.
>
> You're also more motivated. When people pick their tools, the work invariably makes sense to them. They're also working side by side with the people creating the tools that we're using day by day. You can't get that anywhere else.

Once again, a company gets smart and saves money by aligning itself with its own smart employees and the development communities to which they belong.

What's different between now and 10 or 20 years ago? McLane says it's “the amplifying effects of the Internet on the power of the individual”, adding, “only a fool would ignore that”.

Where We Stand

We read about IT brass going gaga for Linux almost every day. Ken Harris, CIO of Gap Corp., recently said he's in favor of “Anything touching Linux”. Emea Harris of Lehman Brothers said, “We're very aggressive around migrating to Linux.”

When I spoke about DIY-IT at the O'Reilly Open Source Convention in July 2003, the majority of those attending the talk (about 100 in the room) was rank-and-file IT guys, mostly from large companies. Phil Moore was one of them. One audience member said some companies feel that their own open-source developments give them a competitive edge, and they don't want to talk about it for that reason. “They don't want their competitors to know how they do it faster and cheaper”, she said.

Untold Stories

What other kinds of stories are we not hearing, then? Here are a few, in no particular order:

- Debian: “I'm seeing far more Debian than any report gives it credit for”, says one technologist working for a large vendor that has partnerships with Red Hat and SuSE. “Red Hat and SuSE may sell more, so they show up on surveys that follow sales. But in terms of actual implementation, Debian is pretty big.”

- Education: at different periods during the past 30 years, companies like Digital, Apple, IBM and Microsoft have had successful programs for getting students hooked on their goods. Now those students are swimming in a sea of free software and old or cheap PCs on which to run it. Web services consultant and author Doug Kaye says "High schools and colleges are now all about open source....It's LAMP everywhere."
- The power of gravy: a number of IT people have told me that vendor relationships are valued highly, period, and always will be. "You get freebies. Tickets to games. Free dinners. Trips to conferences. A lot of people love that gravy train."
- Small consultant opportunities: one IT guy at a large company told me:

> The do-it-yourself movement inside IT is lowering the barriers to entry for small contractors too. Thirty years ago, big companies went to big vendors for big solutions. That's not the case anymore. You've got small vendors and consultants with 15, 20 or 40 people going in and delivering hugely successful solutions to Fortune 50 companies.

Phil Moore adds:

> When you contract for an open-source solution, all the middle layers of the contracting nightmare you get when you go through big vendors are gone. If you contract for a simple change from a big vendor, you can be in for spending a lot of money. The cheapest vendor contract I've ever been involved with is a half-million dollars. At the very least you've got two gigantic legal departments involved to begin with, just to negotiate the contract. Yet I've gone and got similar order-of-magnitude technology changes on open-source products for below five figures. This is a gold mine. This frees up a huge part of my budget. And time-wise the process is highly streamlined.

Perspective

Without vendors, we wouldn't have magazines or tradeshows, to name two of my favorite things. It's important to the market's ecology for vendors to push their goods and tell their stories. The problem we've had—and still have—is a long lag between what's happening in the marketplace and how we cover the subject. And I believe that lag derives from the young ages of the industries involved. The computer industry is about 50 years old. The software industry is half that age. The Internet—which changed everything—began supporting business only about nine years ago. What we need are more stories from the demand side of the marketplace and more courage by those in positions to tell them. We also need publications that welcome those stories, with authors and editors and analysts to help tell them.

My friend Christopher Lydon (blogs.law.harvard.edu/lydon), a former reporter for the *New York Times* and host of NPR's “Connections”, believes what's happening in our industry—this DIY-IT movement—is profoundly Emersonian. He points to this encouraging prose from the author's seminal essay, “Self-Reliance” (www.emersoncentral.com/selfreliance.htm):

> To believe your own thought, to believe that what is true for you in your private heart is true for all men, that is genius. Speak your latent conviction, and it shall be the universal sense; for the inmost in due time becomes the outmost....There is a time in every man's education when he arrives at the conviction that envy is ignorance; that imitation is suicide; that he must take himself for better, for worse, as his portion; that though the wide universe is full of good, no kernel of nourishing corn can come to him but through his toil bestowed on that plot of ground which is given to him to till. The power which resides in him is new in nature, and none but he knows what that is which he can do, nor does he know until he has tried.

We'll be doing our part here at *Linux Journal*. And, as always, we rely on your help as well.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal*. His monthly column is Linux for Suits, and his biweekly newsletter is SuitWatch.

Improving Perl Application Performance

Bruce W. Lowther

Issue #118, February 2004

The four basic performance-tuning steps to improve an existing application's performance.

A fellow developer and I have been working on a data collection application primarily written in Perl. The application retrieves measurement files from a directory, parses the files, performs some statistical calculations and writes the results to a database. We needed to improve the application's performance so that it would handle a considerable load while being used in production.

This paper introduces four performance-tuning steps: identification, benchmarking, refactoring and verification. These steps are applied to an existing application to improve its performance. A function is identified as being a possible performance problem, and a baseline benchmark of that function is established. Several optimizations are applied iteratively to the function, and the performance improvements are compared against the baseline.

Identifying Performance Problems

The first task at hand in improving the performance of an application is to determine what parts of the application are not performing as well as they should. In this case I used two techniques to identify potential performance problems, code review and profiling.

A performance code review is the process of reading through the code looking for suspicious operations. The advantage of code review is the reviewer can observe the flow of data through the application. Understanding the flow of data through the application helps identify any control loops that can be eliminated. It also helps identify sections of code that should be further scrutinized with application profiling. I do not advise combining a performance code review with other types of code review, such as a code review for standards compliance.

Application profiling is the process of monitoring the execution of an application to determine where the most time is spent and how frequently operations are performed. In this case, I used a Perl package called Benchmark::Timer. This package provides functions that I use to mark the beginning and end of interesting sections of code. Each of these marked sections of code are identified by a label. When the program is run and a marked section is entered, the time taken within that marked section is recorded.

Adding profiling sections to an application is an intrusive technique; it changes the behavior of the code. In other words, it is possible for the profiling code to overshadow or obscure a performance problem. In the early stages of performance tuning, this may not be a problem because the magnitude of the performance problem will be significantly larger than the performance impact of the profiling code. However, as performance issues are eliminated, it is more likely that a subsequent performance issue will be harder to distinguish. Like many things, performance improvement is an iterative process.

In our case, profiling some sections of the code indicated that a considerable amount of time was being spent calculating statistics of data collected off the machine. I reviewed the code related to these statistics calculations and noticed that a function to calculate standard deviation, std_dev, was used frequently. The std_dev calculation caught my eye for two reasons. First, because calculating the standard deviation requires calculating the mean and the mean of the sum of squares for the entire measurement set, the naï¿½e calculation for std_dev uses two loops when it could be done with one loop. Secondly, I noticed that the entire data array was being passed into the std_dev function on the stack rather than being passed as a reference. I thought these two items together might indicate a performance issue worth examining.

Benchmarking

After identifying a function that could be improved, I proceeded to the next step, benchmarking the function. Benchmarking is the process of establishing a baseline measurement for comparison. Creating a benchmark is the only way to know whether a modification actually has improved the performance of something. All the benchmarks presented here are time-based. Fortunately, a Perl package called Benchmark was developed specifically for generating time-based benchmarks.

I copied the std_dev function (Listing 1) out of the application and into a test script. By moving the function to a test script, I could benchmark it without affecting the data collection application. In order to get a representative benchmark, I needed to duplicate the load that existed in the data collection application. After examining the data processed by the data collection application, I determined that a shuffled set of all the numbers between 0 and 999,999 would be adequate.

Listing 1. The Baseline Implementation of std_dev

```
sub mean {
  my $result;
  foreach (@_) { $result += $_ }
  return $result / @_;
}


sub std_dev {
  my $mean = mean(@_);
  my @elem_squared;
  foreach (@_) {
    push (@elem_squared, ($_ **2));
  }
  return sqrt( mean(@elem_squared) - ($mean ** 2));
}
```

In order to yield a reliable benchmark, the std_dev function must be repeated several times. The more times the function is run, the more reliable or consistent the benchmark will be. The number of times to repeat the benchmark can be set specifically with the Perl Benchmark package. For example, run this benchmark 10,000 times. Alternatively, the package accepts a time duration, in which case the benchmark is repeated as many times as possible within the allotted time. All benchmarks shown in this article use an iteration parameter of 10 seconds. Calculating the standard deviation of 1,000,000 data elements for at least 10 seconds produced the result:

```
12 wallclock secs (10.57 usr + 0.02 sys
    = 10.59 CPU) @ 0.28/s (n = 3)
```

This information indicates that the benchmark measurement took 12 seconds to run. The benchmark tool was able to execute the function 0.28 times per second or, taking the inverse, 3.5 seconds per iteration. The benchmark utility was able to execute the function only three times (n = 3) in the allotted 10 CPU seconds. Throughout this paper, results are measured using seconds per iteration (s/iter). The lower the number, the better the performance. For example, an instantaneous function call would take 0 s/iter, and a really bad function call would take 60 s/iter. Now that I have a baseline measurement of the std_dev performance, I can measure the effects of refactoring the function.

Although three samples are enough to identify issues with the std_dev calculation, a more in-depth performance analysis should have more samples.

Refactoring and Verification

After establishing the benchmark shown in Listing 1, I refined the std_dev algorithm in two iterations. The first refinement, called std_dev_ref, was to change the parameter passing from “pass by value” to “pass by reference” in both the std_dev function and the mean function that is called by std_dev. The resulting functions are shown in Listing 2. Theoretically, this will increase the performance of both functions by avoiding copying the entire contents of the data array onto the stack before the call to std_dev and the subsequent call to mean.

Listing 2. Replacing Call by Value with Call by Reference

```
sub mean_ref {
  my $result;
  my $ar = shift;
  foreach (@$ar) { $result += $_ }
  return $result / scalar(@$ar);
}

sub std_dev_ref {
  my $ar = shift;
  my $mean = mean_ref($ar);
  my @elem_squared;
  foreach (@$ar) {
    push (@elem_squared, ($_ **2));
  }
  return sqrt( mean_ref(\@elem_squared) -
               ($mean ** 2));
}
```

The second refinement, called std_dev_ref_sum, was to remove the mean function altogether. The mean and the mean of the sum of squares are combined into one loop through the entire data set. This refinement, shown in Listing 3, removes at least two iterations over the data. Table 1 contains a summary of the benchmark times.

Listing 3. After Elimination of the Mean Function

```
sub std_dev_ref_sum {
  my $ar = shift;
  my $elements = scalar @$ar;
  my $sum = 0;
  my $sumsq = 0;

  foreach (@$ar) {
    $sum += $_;
    $sumsq += ($_ **2);
  }
  return sqrt( $sumsq/$elements -
               (($sum/$elements) ** 2));
}
```

Table 1. Baseline and Two Refinements

	s/iter
std_dev	3.53
std_dev_ref	2.93

	s/iter
std_dev_ref_sum	1.37

As hoped, an incremental improvement between each of the refinements is shown in Table 1. Between the std_dev and std_dev_ref functions there is a 20% improvement, and between std_dev and std_dev_ref_sum functions there is a 158% improvement. This seems to confirm my expectation that pass by reference is faster than pass by value in Perl. Also, as expected, removing two loops through the data improved the performance of the std_dev_ref_sum function. After both of these refinements, the function can calculate the standard deviation of 1,000,000 items in 1.37 seconds. Although this is considerably better than the original, I still think there is room for improvement.

Hasn't Someone Already Done This?

A number of open-source Perl packages are available. Hopefully, I could find a standard deviation calculation that was faster than my best attempt so far. I found and downloaded a statistics package from CPAN called Statistics::Descriptive. I created a function called std_dev_pm that used the Statistics::Descriptive package. The code for this function is shown in Listing 4.

Listing 4. The std_dev_pm Function

```
sub std_dev_pm {
  my $stat = new Statistics::Descriptive::Sparse();
  $stat->add_data(@_);
  return $stat->standard_deviation();
}
```

Using this function, however, produced a result of 6.80 s/iter; 48% worse than the baseline std_dev function. This is not altogether unexpected considering that the Statistics::Descriptive package uses an object interface. Each calculation includes the overhead of constructing and destructing a Statistics::Descriptive::Sparse object. This is not to say that Statistics::Descriptive is a bad package. It contains a considerable number of statistical calculations written in Perl and is easy to use for calculations that don't have to be fast. However, for our specific case, speed is more important.

An Out-of-Language Experience

All languages have good and bad qualities. Perl, for example, is a good general-purpose language but is not the best for number-crunching calculations. With this in mind, I decided to rewrite the standard deviation function in C to see if it improved performance.

In the case of the data collection application, it would be counter-productive to rewrite the entire project in C. Quite a few specific Perl utilities make it the best language for most of the application. An alternative to rewriting the application is to rewrite only the functions that specifically need performance improvement. This is done by wrapping a standard deviation function written in C into a Perl module. Wrapping the C function allows us to keep the majority of the program in Perl but allows us to mix in C and C++ where appropriate.

Writing a Perl wrapper over an existing C or C++ interface requires using XS. XS is a tool that is distributed with the Perl package, and it is documented in the perlxs Perl document. You also need some of the information located in the perlguts document. Using XS, I created a Perl package called OAFastStats containing a standard deviation function implemented in C. This function, shown in Listing 5, can then be called directly from Perl. For comparison purposes, this standard deviation function will be called std_dev_OAFast.

Listing 5. The XS Implementation

```
double
std_dev(sv)
    INPUT:
        SV *            sv
    CODE:
        double sum = 0;
        double sumsq = 0;
        double mean = 0;

        /* Dereference a scalar to retrieve
           an array value */
        AV* data = (AV*)SvRV(sv);

        /* Determine the length of the array */
        I32 arrayLen = av_len(data);

        if(arrayLen > 0)
        {
          for(I32 i = 0; i <= arrayLen; i++)
            {
              /* Fetch the scalar located at i
                 from the array.*/
              SV** pvalue = av_fetch(data,i,0);

              /* Dereference the scalar into
                 a numeric value. */
              double value = SvNV(*pvalue);

              /* collect the sum and the
                 sum of squares. */
              sum += value;
              sumsq += value * value;

            }
          mean = (sum/(arrayLen+1));
          RETVAL = sqrt((sumsq/(arrayLen+1)) -
                        (mean * mean));
        }
        else
        {
          RETVAL = 0;
        }

    OUTPUT:
            RETVAL
```

The comparison between the baseline standard deviation function and the C function wrapped with XS is presented in Table 2, showing a significant speedup. The C function (std_dev_ref_OAFast) is 1,175% faster than the baseline function (std_dev), and it is 395% faster than the best Perl implementation (std_dev_ref_sum).

Table 2. Baseline and Fastest Perl Implementations Compared with C

	s/iter
std_dev	3.53
std_dev_ref_sum	1.37
std_dev_OAFast	0.277

Conclusions

During this process I identified a function that probably wasn't performing as well as it could. I was able to achieve several modest performance gains by refining the logic of the calculation in Perl. I also tried using an open-source package, only to find that it was 48% worse than my original function. Finally, I implemented the standard deviation function in C and exposed it to Perl through an XS layer. The C version showed a 1,175% speedup compared to the original Perl version. Improvements are summarized in Figure 1.

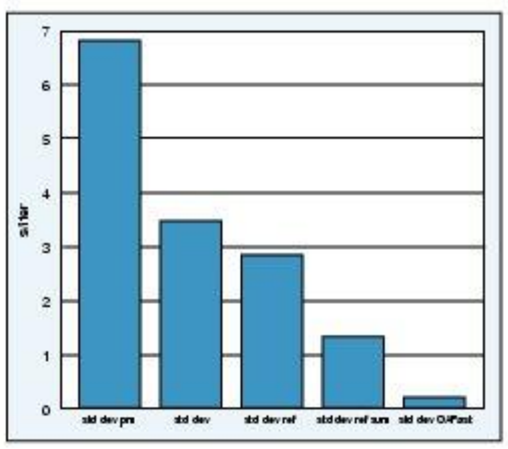


Figure 1. Comparison of All Implementations

In most cases, I have seen Perl performance that rivals C; however, this obviously isn't one of those cases. Perl is a good general-purpose language, and one of its benefits is the ability to step out of the language and implement code in a lower-level language. Don't be afraid of language mix-ins when you really need to improve performance, as long as you understand that there is a maintenance cost. The disadvantage of introducing additional languages is that it will increase the burden for those that must maintain the application in the future. They will need to know C and understand XS functions. However, in our case, the improved performance significantly outweighed the impact of supporting XS.

Bruce W. Lowther (blowther@micron.com) is a software engineer for Micron Technology, Inc., in Boise, Idaho. He has worked at Micron for nine years and has spent the past five years there working on tools to help integrate semiconductor equipment into the Micron manufacturing process. He received his undergraduate and Master's degrees in Computer Science from the University of Idaho.

Archive Index Issue Table of Contents

Advanced search

Asterisk Open-Source PBX System

Brett Schwarz

Issue #118, February 2004

Use one system to manage voice over IP and conventional phone lines, manage voice mail and run CGI-like applications for phone users.

So, you need to deploy a Private Branch eXchange (PBX) system for your small office. Or, maybe you want a voice-mail system running on your Linux box at home. What about an interactive voice response (IVR) system for home automation? Voice over IP (VoIP) capabilities would be nice too. How do you do it? One very interesting and powerful solution is Asterisk, a GPLed PBX system built on Linux that bridges the gap between traditional telephony, such as your telephone line, and VoIP. Asterisk also supports a host of other features that make it an attractive solution. In this article, I touch on some of these features and give you enough information to get started without having to buy any special hardware.

Background

Asterisk is an open-source project sponsored by Digium. The primary maintainer is Mark Spencer, but numerous patches have been contributed from the community. As of this writing, it runs only on Linux for Intel, although there was some success in the past with Linux PPC, and an effort is underway to port Asterisk to *BSD. Digium also sells various hardware components that operate with Asterisk (see Resources). These components are PCI cards that connect standard analog phone lines to your computer. Other hardware is supported as well, such as hardware from Dialogic and Quicknet. Asterisk has its own VoIP protocol, called IAX, but it also supports SIP and H.323. This leads us to one of Asterisk's most powerful features: its ability to connect different technologies within the same feature-rich environment. For example, you could have IAX, SIP, H.323 and a regular telephone line connecting through Asterisk (see Figure 1—courtesy of Digium).

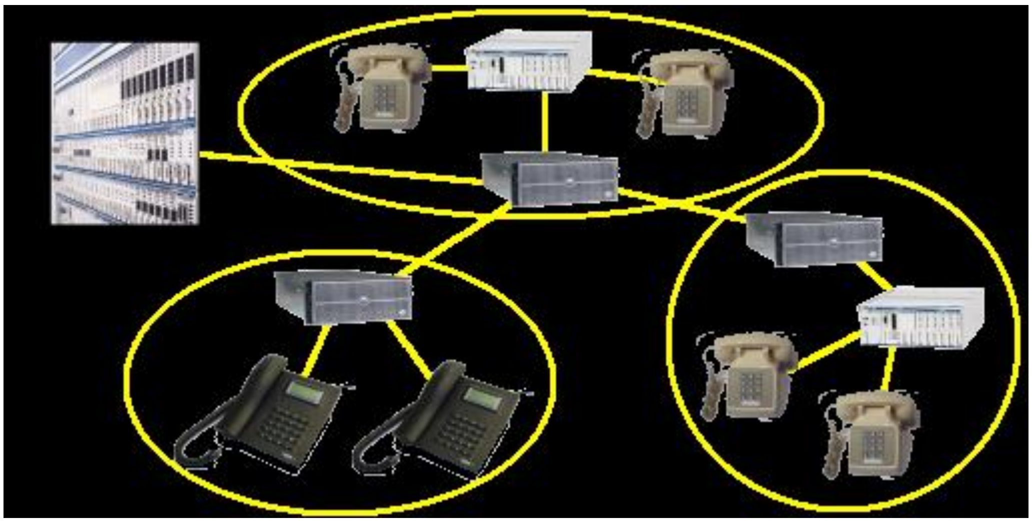


Figure 1. Asterisk can connect regular telephone lines and multiple VoIP standards.

The developer can extend Asterisk by working with the C API or by using AGIs, which are analogous to CGI scripts. AGIs can be written in any language and are executed as an external process. They are the easiest and most flexible way to extend Asterisk's capabilities (see Listing 1).

Listing 1. Example Caller ID AGI Script

```
#!/bin/sh
# \
exec tclsh "$0" ${1+"$@"}

set port 10000
set hosts [list 192.168.123.166 192.168.123.168]

##
##  Sends the info to the hosts
##
proc sendInfo {ip_ port_ callerid_} {

  if {[catch {socket $ip_ $port_} sock]} {
    return
  }

  fconfigure $sock -buffering line
  puts $sock $callerid_
  close $sock

  return
}

##
##  We get all of the variables from stdin;
##  they start with "agi_"; and populate
##  an array with the values.
##
while {[gets stdin l] > 0} {
  if {[regexp {^agi_([\w]+):[\s]+(.*)} $l -> k v]} {
    set AGI($k) $v
  }
}

##
##      Send the callerid info to each host
##      that we have listed
##
foreach H $hosts {
  sendInfo $H $port $AGI(callerid)
}
```

Getting Started

An official release hasn't happened for quite a while, but there is talk of one coming. Currently, the best way to get Asterisk is by CVS:

```
export CVSROOT=\
:pserver:anoncvs@cvs.digium.com:/usr/cvsroot
cvs login (password is "anoncvs")
cvs co asterisk
```

If you plan on using a PCI card from Digium, you should look at zaptel as well. If you plan on having connectivity, you need to check out libpri.

There is no configure script, so you simply use make. You also need readline, OpenSSL and Linux 2.4.x with the kernel sources installed in order to compile Asterisk properly:

```
cd asterisk
make clean install samples
```

This compiles Asterisk, installs it and also installs the sample configuration files. The last target overwrites any existing configuration files, so either skip this target or back up any existing configuration files if you want to preserve them. If you are using zaptel or ISDN, compile those before compiling Asterisk. Asterisk is installed in /usr/sbin/ with the configuration files in /etc/asterisk/ by default. Voice-mail messages are stored in /var/spool/asterisk/voicemail/. CDRs for billing and log files are located under /var/log/asterisk/.

You can start Asterisk by typing `asterisk` at the command line. However, the best way to use Asterisk during the testing phase is to run it with the -vvvc options. The -vvv option is extra-verbose output, and the -c option gives you a console prompt, which allows you to interact with the Asterisk process. For example, you can submit commands to Asterisk, such as management and status commands.

Asterisk's operation and functionality relies on several configuration files. We discuss three of them in this article, but several others exist. Here, we set up Asterisk so that users can call each other through IAX. We also set up voice mail and give users a way to manage their voice-mail messages.

The Dialplan

Before getting into the setup of Asterisk, we should have a general understanding of the dialplan. It is flexible and powerful but also can be confusing. The dialplan is used to define number translations and routing and, therefore, is the heart of Asterisk. The dialplan defines contexts, which are containers for extensions (digit patterns) that provide specific functionality. For instance, you may want to provide a context for people who are in your office or home, so that they have certain dialing privileges. You also could set up an external or guest context that allows only limited dialing capabilities, such as no long distance. Context names are enclosed by brackets ([]). The extensions associated with the context follow the name.

Each extension can have several steps (priorities) associated with it. The call flow continues sequentially unless an application returns -1, the call is terminated or the application redirects the call flow. The syntax of an extension entry looks like this:

```
exten => <exten>,<priority>,<application(args)>
```

Below are a couple of examples:

```
exten => 9911,1,Wait(1)
exten => 9911,2,Dial(Zap/1/${EXTEN:1})
```

An extension is denoted by using `exten =>`. In this example, 9911 is the extension; 1 and 2 are the priorities or step numbers (these need to be sequential); and Wait and Dial are the applications. Asterisk uses applications to process each step within an extension. You can get help for the different applications from the Asterisk console by typing `show applications` to list the supported applications and `show application <application>` to display the help message.

Extension matching can be done on the dialed number as well as the calling number. This allows for greater flexibility when processing calls. Patterns also can be used, and these are preceded with an underscore (_):

- N—a single digit between 2 and 9.
- X—a single digit between 0 and 9.
- [12-4]—any digit within the brackets.
- . —wild card.

For example, the extension _NXX5551212 would match any information number, regardless of area code.

Extensions can be any alphanumeric string. Some special characters are built-in:

- s—start here when no dialed digits are received, as from an incoming call from an analog line.
- t—used when a timeout occurs.
- i—used for invalid dialed digits.
- o—operator extension.
- h—hangup extension.

Creating IAX Users

The first file we create is the iax.conf file (see Listing 2). This file controls the operation of the IAX protocol and defines users of the protocol. The protocol has two versions. The old one is IAX, and the new one is IAX2.

Listing 2. iax.conf File

```
[general]
port=5036
bindaddr=0.0.0.0
amaflags=default
accountcode=home

[brett]
type=friend
host=dynamic
secret=brettsecret
context=cg1
callerid="brett <111>"

[maria]
type=friend
host=dynamic
secret=mariasecret
context=cg1
callerid="maria <222>"

[niko]
type=friend
host=dynamic
secret=nikosecret
context=cg2
accountcode=external
callerid="Niko <333>"
```

The first section of the configuration file is the general section, which defines parameters for the IAX protocol. Four parameters are listed, but others can be defined as well. The port parameter is the port number over which IAX will communicate. It defaults to 5036, so strictly speaking, that entry is not needed. You can use the bindaddr parameter to tell Asterisk to bind to a particular IP address—for machines with multiple Ethernet cards. A bindaddr of 0.0.0.0 attempts to bind to all IP addresses. The parameters amaflags and accountcode are used for CDRs. When they are defined in the general section, they are used as the default values. You also can define them on a per-user basis. The values that amaflags can accept are billing, documentation, omit and default. accountcode can be an arbitrary value. For this setup, I use home for users local to my LAN and external for users outside of my LAN. Several other parameters have been omitted, but most of them are performance parameters.

The remaining sections are user definitions. I have three users: brett, maria and niko. The type definition has three possible values: a peer can receive calls, a user can place calls and a friend can do both. I have defined all of them as type friend. I defined all of the hosts as being dynamic, but if any host has a static IP address, you can specify that instead. secret is the password the user must provide when connecting to this Asterisk server. Two contexts are used in this file for users: [cg1] and [cg2]. I explain these in more detail when discussing the extensions.conf file, but effectively, these contexts enable the dialing privileges for the user.

Setting Up Voice Mail

The next file is voicemail.conf (Listing 3). Again, it has a general section that deals with general or global parameters for voice mail. The first parameter, format, lists the audio format of the messages. The next two parameters are used for e-mail notification: serveremail is the source e-mail address (from field), and attach instructs Asterisk to attach the message to the e-mail. In our example, we do not want the message attached. Again, some parameters have been omitted.

<mbox> is the number used to save and access messages for the user. This is also used in extensions.conf for directing the call flow to the proper voice-mail box. The <passwd> parameter is needed when checking messages. <name> is the name of the user. <email> and <pager> are e-mail addresses that are used to send message notifications. The pager e-mail has a shorter message, because it needs to be read on smaller devices (pagers and cell phones). Many mobile and pager providers have e-mail gateways that can deliver the message to the device.

Listing 3. Voicemail.conf File

```
[general]
format=gsm|wav49|wav
serveremail=asterisk
attach=no
maxmessage=180
```

```
maxgreet=60

;
;  Voicemail box definitions.
;  mbox# => password,name,email,pager/mobile
;
[cg1]
111 => 1111,Brett,brett_schwarz@yahoo.com
222 => 2222,Maria,maria@foo.com,4255551212@mob.net
```

Defining Extensions

The last file that we examine here is the extensions.conf file (Listing 4). This is one of the most involved files because it contains the dialplan. The dialplan in my example is rather simple compared to its capabilities. This file has a general and global section. The general section is similar to the general section in the previous files; it defines general parameters. I don't define any general parameters in this example. The global section is used to define global variables. These variables can be accessed in the dialplan by using the syntax `${VARIABLE}`. I have defined one variable: TIMEOUT is the answer timeout. Built-in variables also can be used within the dialplan, such as, CONTEXT, EXTEN and CALLERID.

Listing 4. extensions.conf File

```
[globals]
TIMEOUT=12

[misc]
exten => t,1,PlayBack(timeout)
exten => t,2,Hangup()
exten => i,1,PlayBack(invalid)
exten => i,2,Hangup()

;  voicemail management
[voicemail]
include => misc
exten => 6245,1,VoiceMailMain2()
exten => 6245,2,Hangup

[iax]
include => misc
exten => 111/222,1,SetCIDName("it's your wife!")
exten => 111/222,2,agi(callerid.agi)
exten => 111/222,3,Dial(IAX/brett/s,${TIMEOUT})
exten => 111/222,4,Voicemail2(111)
exten => 111,1,agi(callerid.agi)
exten => 111,2,Dial(IAX/brett/s,${TIMEOUT})
exten => 111,3,Voicemail2(111)
exten => 222,1,Dial(IAX/maria/s,${TIMEOUT})
exten => 222,2,Voicemail2(222)
exten => maria,1,Goto(iax,222,1)
exten => 333,1,Dial(IAX/niko/s,${TIMEOUT})

[afterhours]
include => misc
exten => _.,1,Wait(1)
exten => _.,2,Answer
exten => _.,3,Background(vm-menu)

exten => 1,1,Voicemail2(111)
exten => 2,1,Voicemail2(222)
exten => 3,1,Voicemail2(333)
```

```
[cg1]
include => iax
include => voicemail

[cg2]
include => afterhours|1:00-6:00|*|*|*
include => iax
```

All of the other sections are context definitions. A context is simply a grouping of digit patterns. Here I have defined several contexts that define dialing scenarios: voicemail, iax and afterhours. Think of these as individual or mini-dialplans. I then define two contexts that I assign to the users. These inherit the capabilities of the other contexts I already have defined by using the include keyword.

The first context, voicemail, lists the digit patterns that allow users to access their voice-mail messages. Users can dial 6245, and the application VoicemailMain2 prompts them for the mailbox number and password. Users then can manage (listen to, delete and so on) the messages in their mailbox.

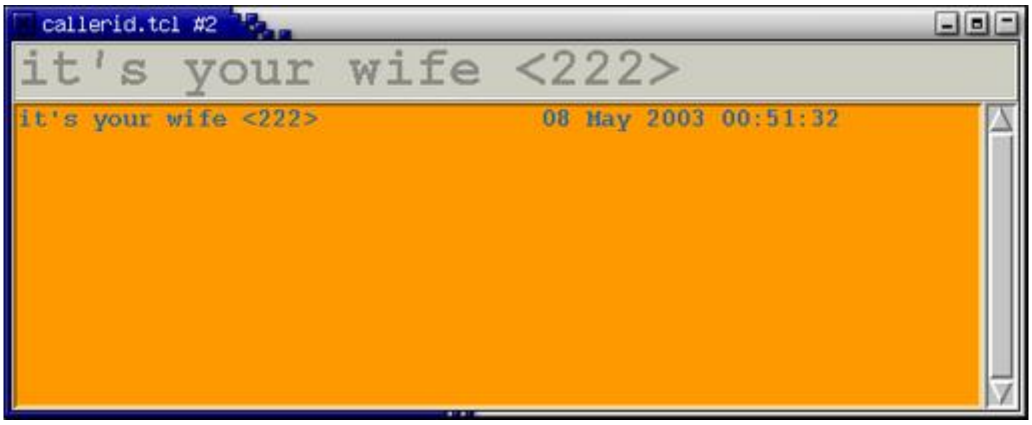
The iax context is used for PBX dialing between IAX users. We have defined various extensions for each of the users. An entry with the name of the user (maria) redirects to the extension number entry. For the 111 extension, I also match on callerid. If the callerid matches, I change the callerid name so it has a relative meaning. For example, if the extension dialed is 111, and the callerid is 222, the callerid name is changed to “it's your wife!”. This message shows up on my client whenever my wife calls me (I won't get into how I use this to my advantage).

The last digit pattern context is used for calls that arrive during late hours. Because I don't want to be disturbed at night by external users, I match on any dialed number (_.). It waits for one second and then answers the call. After it answers, it plays a background message so the caller can choose for which person to leave a message (“for brett, press 1”). So, if the caller presses 1, the call proceeds to the 1,1,Voicemail2(111) entry, which sends the user to the 111 mailbox. This is a simple illustration of how you could construct an IVR system.

The [cg1] and [cg2] contexts include functionality I already have defined inside other contexts. This allows me to create different user groups easily. For example, [cg1] has all of the capabilities I have defined, but [cg2] has only the iax capabilities and gets directed to voice mail during late hours. Powerful dialing capabilities can be constructed by utilizing the flexibility of Asterisk's dialplan. My example has shown only a glimpse of the possibilities. You also can simplify the dialplan by using macros, but I leave that as an exercise for the reader.

Using AGI

In the extensions.conf file, an entry called callerid.agi calls an AGI script. This is a simple example illustrating the AGI interface. The script is placed in the /var/ lib/ asterisk/agi-bin/ directory and is invoked by Asterisk as an external process. AGI and Asterisk communicate through stdin, stdout and stderr. Variables are passed in to the AGI through stdin, and the AGI can pass information back to Asterisk through stdout. Messages destined for the Asterisk console are written to stderr. Two parameters always are passed to the AGI: the full path to the AGI and the arguments that are passed to the AGI through the exten entry. The AGI collects the callerid and sends it to a GUI application running on another machine. The GUI application can be retrieved from my Web site (see Figure 2). AGI scripts also can be used to retrieve information. If you need to query a database for information about the call or the user, you can use the AGI interface as well.



```
it's your wife <222>
it's your wife <222>                08 May 2003 00:51:32
```

Figure 2. A GUI application shows the caller ID information for incoming calls.

Making a Call

So, what can we do now? After creating the configuration files above and starting Asterisk (`asterisk -vvvc`), we can try some calls. Currently, the availability of IAX soft clients is limited. SIP soft clients, including kphone and xten, and hard clients from Cisco, SNOM and other vendors also are available that will work with Asterisk, but I concentrate on using IAX in this article. Gnophone (Figure 3) is the oldest client and was developed by Digium. Work also is being done on a cross-platform client, as well as a Windows client. Another client is available that belongs to the tel Project at SourceForge. I have modified the user interface to that client (Figure 4). It is still alpha software, but it's functional. In fact, I used this client to establish a call between Germany (Reinhard Max), Australia (Steve Landers) and the US (me). Whichever client you choose, you need to define your user name, password and context for each Asterisk server with which you want to connect. Then, you can call anyone defined in the iax.conf file (if the dialplan is set up correctly). So, if I want to call my wife, I simply dial 222, or I can type `maria` (because I have defined this in the dialplan). If I want to check my voice-mail messages, I can dial 6245.

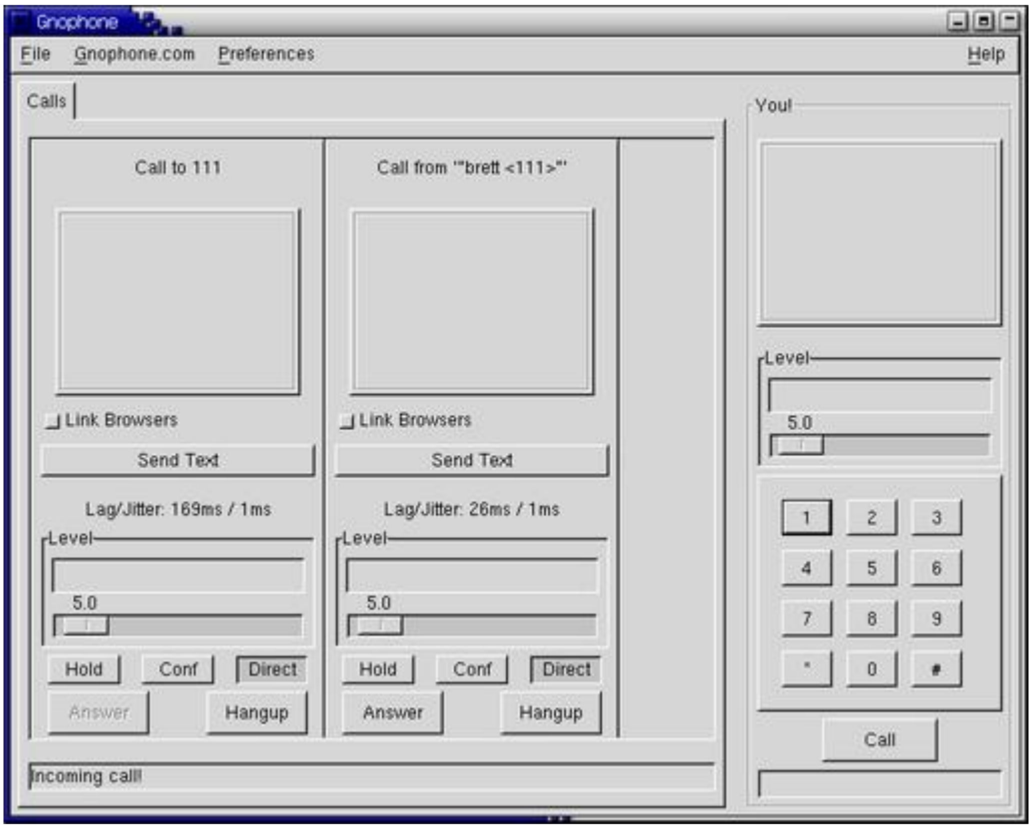


Figure 3. Digium's Gnophone is a software phone client you can use with Asterisk.

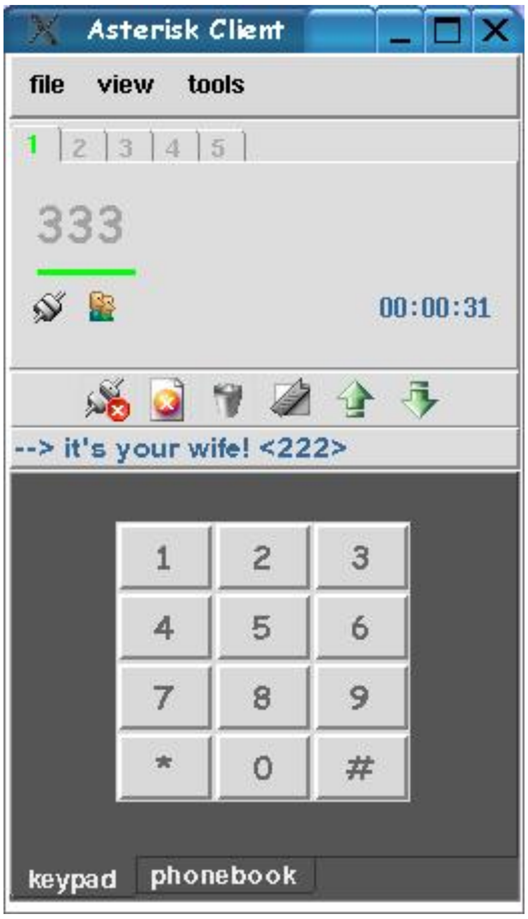


Figure 4. Alpha but working software: a modified version of the client from the tel Project.

Conclusion

I have touched on only a few of Asterisk's capabilities, but this article should give the reader a glimpse of Asterisk's potential. Asterisk scales well from small setups to larger and more complex configurations. For example, Asterisk servers in different locations can be connected through the IAX protocol, creating a virtual PBX. Because Asterisk runs on Linux you can leverage existing tools to help interface and manage Asterisk. For instance, you could have Web access to the CDRs, configuration files and voice mail. In fact, a CGI script comes with Asterisk that allows you to access your voice-mail messages with a Web browser. I encourage readers to explore Asterisk further and leverage its powerful features.

Acknowledgements

I would like to thank Digium, Reinhard Max and Steve Landers for their assistance with this article.

Resources

AGI Information: home.cogeco.ca/~camstuff/agi.html

“Asterisk: A Bare-Bones VoIP Example”, by John Todd (Asterisk and SIP Setup): www.onlamp.com/pub/a/onlamp/2003/07/03/asterisk.html

Asterisk Client: tel.sf.net

Asterisk Forum: www.pbxtech.info/forumdisplay.php?f=113

Asterisk HOWTO (Beta): megaglobal.net/docs/asterisk/html

Asterisk Wiki: www.voip-info.org/wiki-Asterisk

Brett's Web Site: www.bschwarz.com

Cross-Platform IAX Client and IAXPhone: iaxclient.sf.net

Digium (Documentation and Hardware): www.digium.com

Getting Started with Asterisk: www.automated.it/guidetoasterisk.htm

Gnophone: www.gnophone.com

Notes on Asterisk: asterisk.drunkcoder.com

Perl Modules for Asterisk and Other Information: asterisk.gnuinter.net

Windows IAX Client: laser.com/dante/diax/diax.html

Brett Schwarz lives near Seattle, Washington, with his wife, son and dog. Although he is familiar with multiple platforms, his platform of choice is Linux. He has many years of experience working on both computer and telecom systems. He can be contacted through his home page at www.bschwarz.com.

Archive Index Issue Table of Contents

Advanced search

A Guided Tour of Ethereal

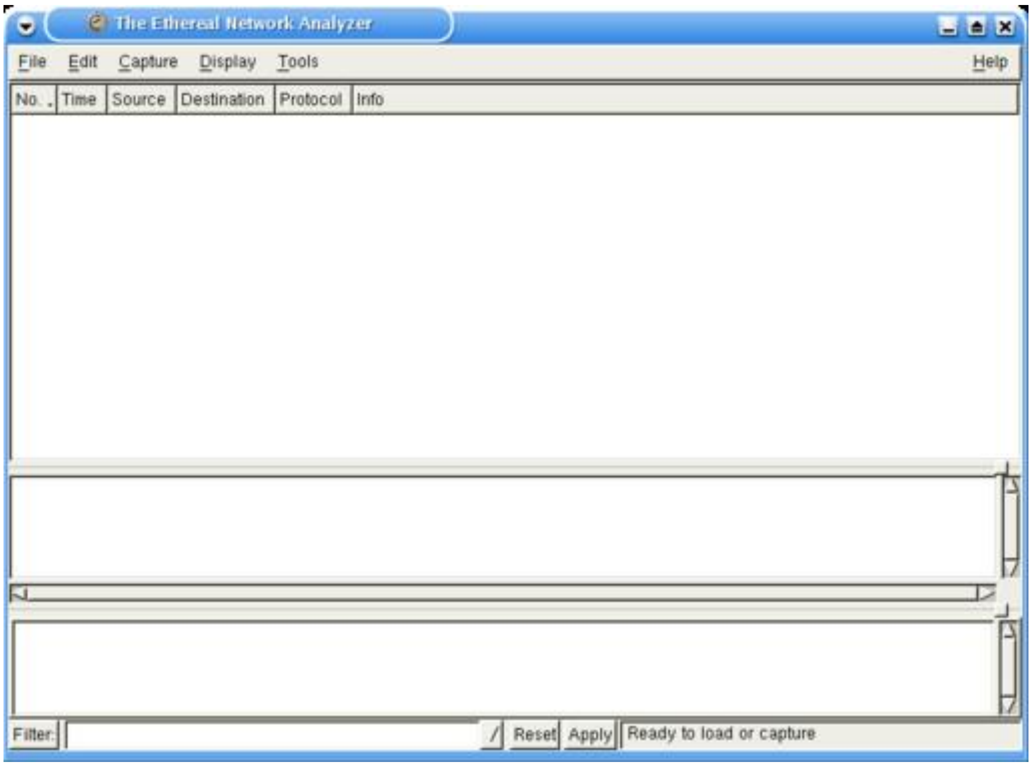
Brad Hards

Issue #118, February 2004

Learn exactly what's in all those packets flying by on your network with this essential development and administration tool.

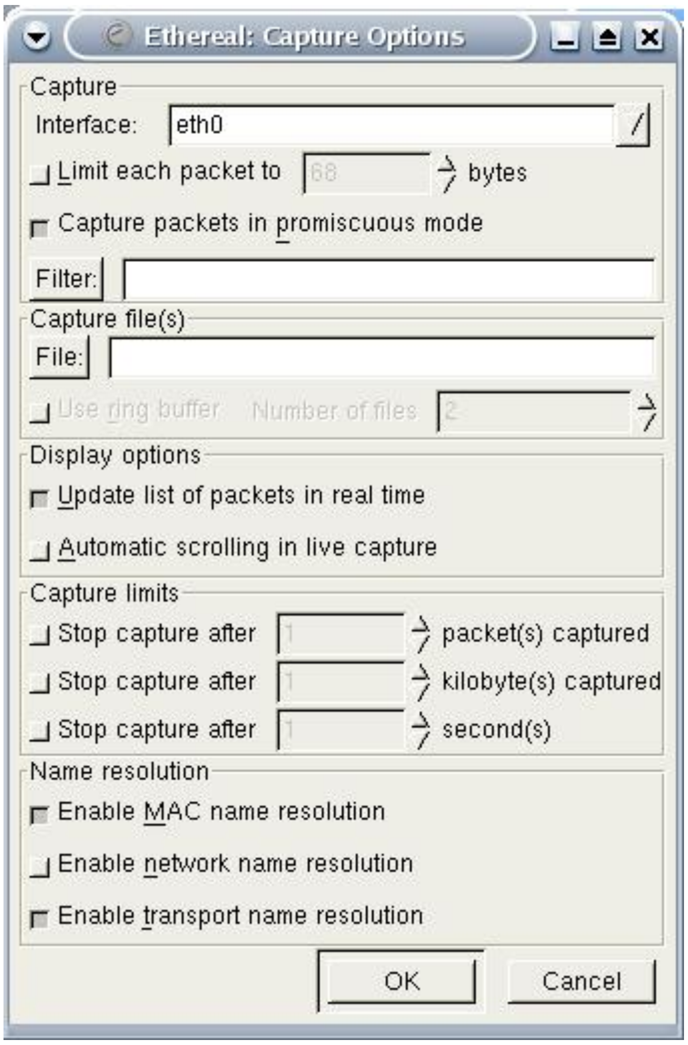
I recently started using a network tool called Ethereal. For those familiar with tcpdump, think of Ethereal as a GUI form of tcpdump that shows you the whole packet and can break down the packet to show individual fields. For those who haven't used tcpdump or similar packet sniffers, it might be best to show the capabilities of Ethereal through a few examples.

When you start Ethereal, it looks like the graphic shown in Figure 1. Typically, you want to capture some data from the network attached to your workstation; do this by selecting Capture→Start..., which brings up the dialog shown in Figure 2. When you've captured the data you need, stop the capture and examine it. Figure 3 shows a capture of some IPv6 traffic, where I've selected an ICMPv6 packet (in the top frame) and expanded the IPv6 and ICMPv6 contents to select the IPv6 source address (in the middle frame). Ethereal automatically highlights the raw bytes corresponding to the selected field—in this case, source address—within the packet in the bottom frame. This type of functionality makes Ethereal useful for understanding various network protocols, and I definitely recommend its use as a teaching or self-education aid in conjunction with networking RFCs. Ethereal also is useful for educating users and management about the dangers of using protocols that send data in clear text, as shown for File Transfer Protocol in Figure 4.


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
No. Time Source Destination Protocol Info
Filter: Reset Apply

Ready to load or capture

Figure 1. The Ethereal Main Window



Capture

Interface: eth0

☐ Limit each packet to 68 bytes

☑ Capture packets in promiscuous mode

Filter:

Capture file(s)

File:

☐ Use ring buffer Number of files 2

Display options

☑ Update list of packets in real time

☐ Automatic scrolling in live capture

Capture limits

☐ Stop capture after 1 packet(s) captured

☐ Stop capture after 1 kilobyte(s) captured

☐ Stop capture after 1 second(s)

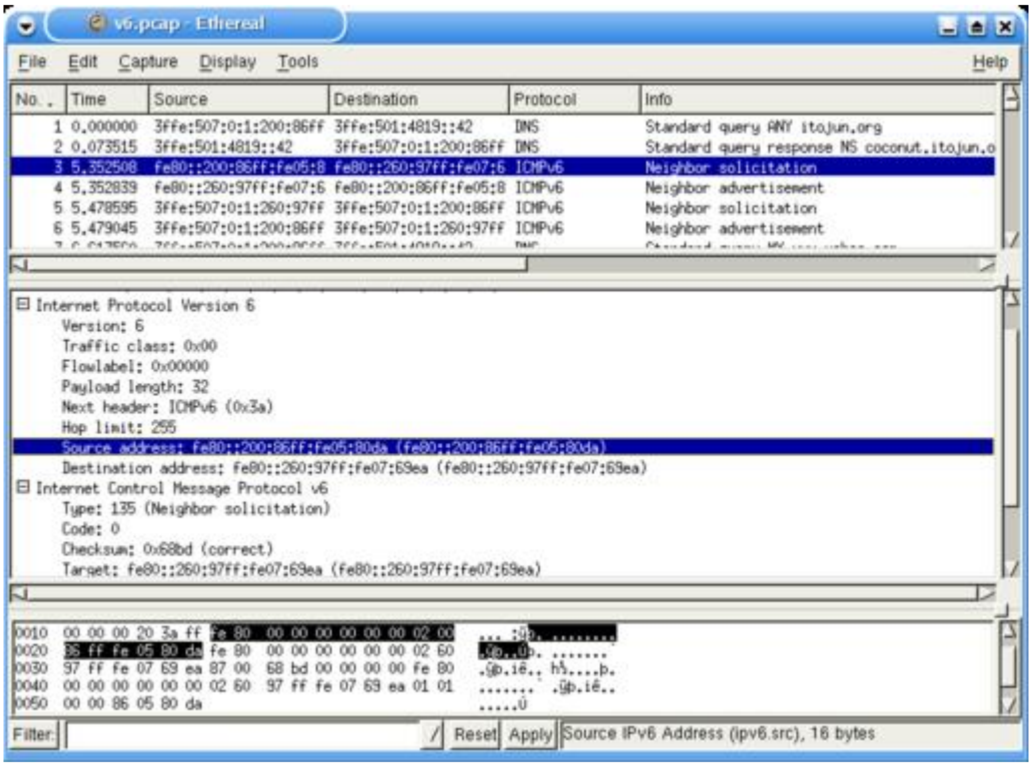
Name resolution

☑ Enable MAC name resolution

☐ Enable network name resolution

☑ Enable transport name resolution

Figure 2. Capture Dialog



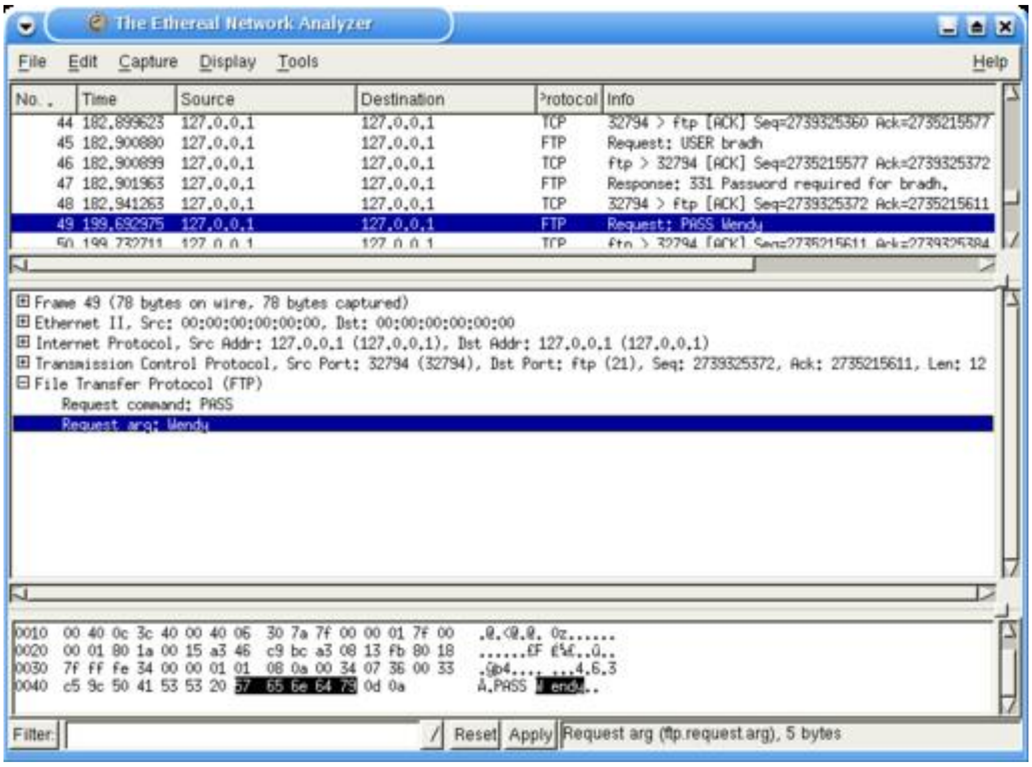
```
No. . Time      Source                Destination           Protocol  Info
    1 0.000000  3ffe:507:0:1:200:86ff 3ffe:501:4819::42     DNS       Standard query ANY itojun.org
    2 0.073515  3ffe:501:4819::42     3ffe:507:0:1:200:86ff DNS       Standard query response NS coconut.itojun.o
    3 5.352508  fe80::200:86ff:fe05:8 fe80::260:97ff:fe07:6 ICMPv6    Neighbor solicitation
    4 5.352839  fe80::260:97ff:fe07:6 fe80::200:86ff:fe05:8 ICMPv6    Neighbor advertisement
    5 5.478595  3ffe:507:0:1:260:97ff 3ffe:507:0:1:200:86ff ICMPv6    Neighbor solicitation
    6 5.479045  3ffe:507:0:1:200:86ff 3ffe:507:0:1:260:97ff ICMPv6    Neighbor advertisement
```

```
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source address: fe80::200:86ff:fe05:80da (fe80::200:86ff:fe05:80da)
    Destination address: fe80::260:97ff:fe07:69ea (fe80::260:97ff:fe07:69ea)
⊟ Internet Control Message Protocol v6
    Type: 135 (Neighbor solicitation)
    Code: 0
    Checksum: 0x68bd (correct)
    Target: fe80::260:97ff:fe07:69ea (fe80::260:97ff:fe07:69ea)
```

```
0010  00 00 00 20 3a ff fe 80  00 00 00 00 00 00 02 00   ... :ÿþ. ........
0020  86 ff fe 05 80 da fe 80  00 00 00 00 00 00 02 60   .ÿþ..Úþ. .......`
0030  97 ff fe 07 69 ea 87 00  68 bd 00 00 00 00 fe 80   .ÿþ.iê.. h½....þ.
0040  00 00 00 00 00 00 02 60  97 ff fe 07 69 ea 01 01   .......` .ÿþ.iê..
0050  00 00 86 05 80 da                                  .....Ú
```

Source IPv6 Address (ipv6.src), 16 bytes

Figure 3. ICMPv6 Capture and Dissection


The Ethereal Network Analyzer
File Edit Capture Display Tools Help
Filter: Reset Apply

```
No. . Time        Source      Destination   Protocol Info
   44 182.899623  127.0.0.1   127.0.0.1     TCP      32794 > ftp [ACK] Seq=2739325360 Ack=2735215577
   45 182.900880  127.0.0.1   127.0.0.1     FTP      Request: USER bradh
   46 182.900899  127.0.0.1   127.0.0.1     TCP      ftp > 32794 [ACK] Seq=2735215577 Ack=2739325372
   47 182.901963  127.0.0.1   127.0.0.1     FTP      Response: 331 Password required for bradh.
   48 182.941263  127.0.0.1   127.0.0.1     TCP      32794 > ftp [ACK] Seq=2739325372 Ack=2735215611
   49 199.692975  127.0.0.1   127.0.0.1     FTP      Request: PASS Wendy
   50 199.732711  127.0.0.1   127.0.0.1     TCP      ftp > 32794 [ACK] Seq=2735215611 Ack=2739325384
```

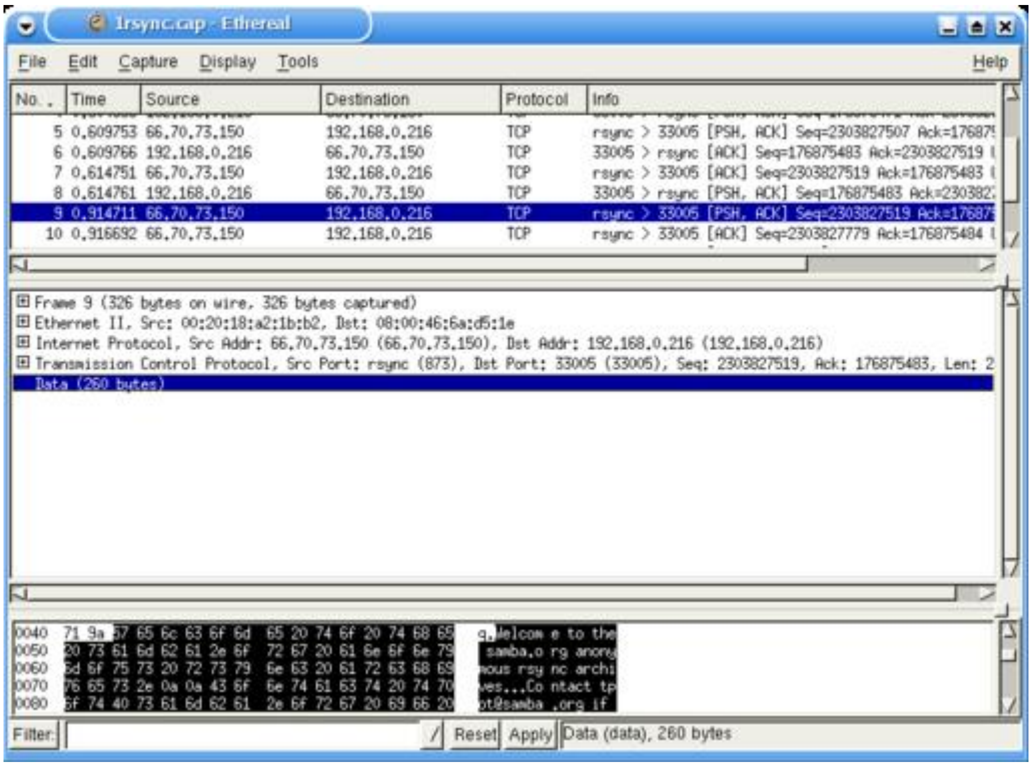
```
⊞ Frame 49 (78 bytes on wire, 78 bytes captured)
⊞ Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
⊞ Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
⊞ Transmission Control Protocol, Src Port: 32794 (32794), Dst Port: ftp (21), Seq: 2739325372, Ack: 2735215611, Len: 12
⊟ File Transfer Protocol (FTP)
    Request command: PASS
    Request arg: Wendy
```

```
0010  00 40 0c 3c 40 00 40 06  30 7a 7f 00 00 01 7f 00   .@.<@.@. 0z......
0020  00 01 80 1a 00 15 a3 46  c9 bc a3 08 13 fb 80 18   ......£F É¼£..û..
0030  7f ff fe 34 00 00 01 01  08 0a 00 34 07 36 00 33   .ÿþ4.... ...4.6.3
0040  c5 9c 50 41 53 53 20 57  65 6e 64 79 0d 0a         Å.PASS W endy..
```

Request arg (ftp.request.arg), 5 bytes

Figure 4. FTP Capture and Dissection, Showing Password

Ethereal also is useful for investigating proprietary protocols or other networking protocols that are not well documented. Figure 5 shows a somewhat contrived example—rsync. This protocol is in widespread use because of its ability to save significant bandwidth but is essentially defined by the source code to the application. I used Ethereal to capture a number of rsync transactions and figured out how the protocol works—at least enough to write an rsync protocol dissector for Ethereal. I understand the Samba team uses Ethereal and a number of other tools to develop clients and servers that interoperate with the Microsoft CIFS implementations, because the Microsoft documentation for these protocols is incomplete or incorrect.



```
No. . Time     Source          Destination     Protocol Info
   5 0.609753 66.70.73.150    192.168.0.216   TCP      rsync > 33005 [PSH, ACK] Seq=2303827507 Ack=17687
   6 0.609766 192.168.0.216   66.70.73.150    TCP      33005 > rsync [ACK] Seq=176875483 Ack=2303827519 
   7 0.614751 66.70.73.150    192.168.0.216   TCP      rsync > 33005 [ACK] Seq=2303827519 Ack=176875483 
   8 0.614761 192.168.0.216   66.70.73.150    TCP      33005 > rsync [PSH, ACK] Seq=176875483 Ack=230382
   9 0.914711 66.70.73.150    192.168.0.216   TCP      rsync > 33005 [PSH, ACK] Seq=2303827519 Ack=17687
  10 0.916692 66.70.73.150    192.168.0.216   TCP      rsync > 33005 [ACK] Seq=2303827779 Ack=176875484 

⊞ Frame 9 (326 bytes on wire, 326 bytes captured)
⊞ Ethernet II, Src: 00:20:18:a2:1b:b2, Dst: 08:00:46:6a:d5:1e
⊞ Internet Protocol, Src Addr: 66.70.73.150 (66.70.73.150), Dst Addr: 192.168.0.216 (192.168.0.216)
⊞ Transmission Control Protocol, Src Port: rsync (873), Dst Port: 33005 (33005), Seq: 2303827519, Ack: 176875483, Len: 2
  Data (260 bytes)

0040  71 9a 57 65 6c 63 6f 6d  65 20 74 6f 20 74 68 65   q.Welcom e to the
0050  20 73 61 6d 62 61 2e 6f  72 67 20 61 6e 6f 6e 79    samba.o rg anony
0060  6d 6f 75 73 20 72 73 79  6e 63 20 61 72 63 68 69   mous rsy nc archi
0070  76 65 73 2e 0a 0a 43 6f  6e 74 61 63 74 20 74 70   ves...Co ntact tp
0080  6f 74 40 73 61 6d 62 61  2e 6f 72 67 20 69 66 20   ot@samba .org if
```

Figure 5. Ethereal Capturing rsync (Now Supported)

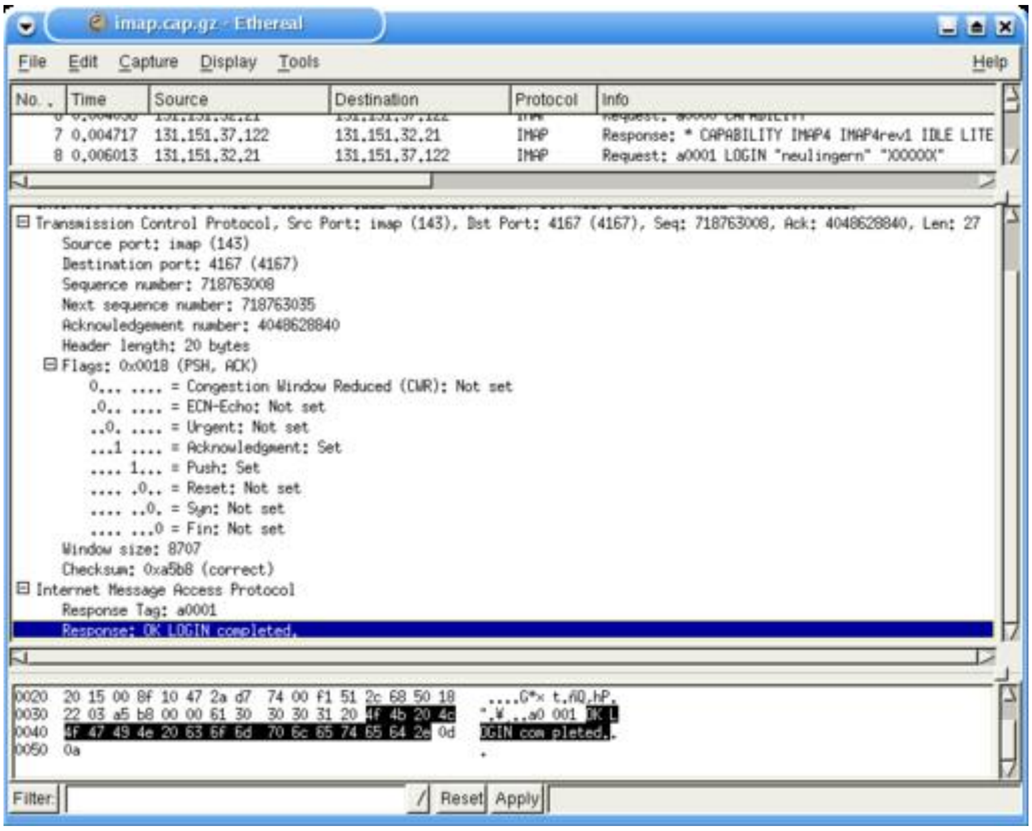
I also have used Ethereal as a part of network application testing (on zcip and Service Location Protocol) to assess correctness and response times. Ethereal time-tags each transaction, so you easily can see the relationship between packets.

How Ethereal Works

Ethereal works by capturing packets through a reasonably portable library called libpcap, which on Linux accesses the packets on the network through using a kernel mechanism called packet socket. It is possible to disable this option under Linux, although probably all vendor kernels have it enabled, and it is enabled in the default kernel configuration for most architectures on Linux kernels. Other operating systems have different interfaces, but libpcap abstracts this away and provides a common API.

Having received a copy of the network packets, Ethereal builds an internal linked list and saves the packets to a file. It then determines what protocol the packet is carrying based on the port numbers, type fields in the supporting protocols or a heuristic that guesses the protocol based on the contents of the field. It is worth noting that this approach essentially is informed guesswork and is by no means infallible. For example, traffic to port 53 probably is DNS, but there is no reason why a network administrator could not choose to run another service on that port. In addition, Ethereal supports an option to interpret a particular packet as a different protocol, using Tools→Decode As.

Based on the guessed protocol, Ethereal decodes (dissects, in Ethereal nomenclature) the packet. Each protocol supported by Ethereal is handled through a bit of code known as a dissector. At the time of this writing, 333 dissectors are built in to Ethereal, some of which handle more than one protocol. Protocols also can be provided as plugins, which are loaded dynamically. Depending on the protocol and the level of sophistication provided by the dissector code, the packets can be broken down for analysis of individual bits or they can be presented at a very high level. Both options are depicted in Figure 6, where the TCP dissector shows the individual bits set in the flags, but the IMAP dissector breaks out only two fields. It is worth noting that IMAP is a text-based protocol, so a simple ASCII dump of the packet contents is an appropriate way to show them.



```
No. . Time      Source          Destination     Protocol Info
    7 0.004717  131.151.37.122  131.151.32.21   IMAP     Response: * CAPABILITY IMAP4 IMAP4rev1 IDLE LITE
    8 0.006013  131.151.32.21   131.151.37.122  IMAP     Request: a0001 LOGIN "neulingern" "XXXXXX"
```

```
⊟ Transmission Control Protocol, Src Port: imap (143), Dst Port: 4167 (4167), Seq: 718763008, Ack: 4048628840, Len: 27
    Source port: imap (143)
    Destination port: 4167 (4167)
    Sequence number: 718763008
    Next sequence number: 718763035
    Acknowledgement number: 4048628840
    Header length: 20 bytes
  ⊟ Flags: 0x0018 (PSH, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 1... = Push: Set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 8707
    Checksum: 0xa5b8 (correct)
⊟ Internet Message Access Protocol
    Response Tag: a0001
    Response: OK LOGIN completed.
```

```
0020  20 15 00 8f 10 47 2a d7  74 00 f1 51 2c 68 50 18   ....G*× t.ñQ,hP.
0030  22 03 a5 b8 00 00 61 30  30 30 31 20 4f 4b 20 4c   ".¥...a0 001 OK L
0040  4f 47 49 4e 20 63 6f 6d  70 6c 65 74 65 64 2e 0d   OGIN com pleted..
0050  0a                                                 .
```

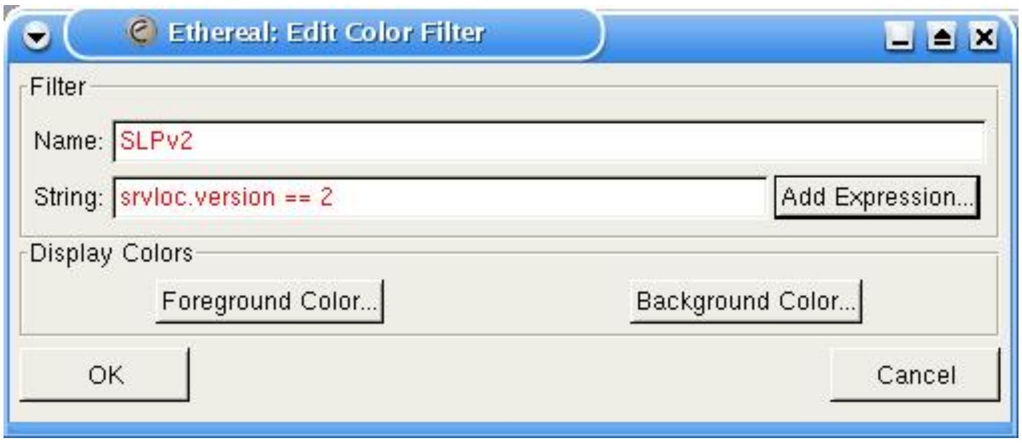
Figure 6. Two Variations on Dissection—TCP and IMAP

Key Features

From my point of view, the key features of Ethereal are its ability to capture and analyze network traffic within a single application and the sophistication of its display and filtering code.

Although we looked earlier in this article at how capturing network traffic is done, Ethereal can capture more than Ethernet traffic. Ethereal typically can (at least on Linux) capture data from Ethernet, Token-Ring, FDDI, serial (PPP and SLIP), 802.11 wireless LAN, ATM connections and all networking devices at the same time. Called the “any” device in the Ethereal capture dialog, this feature only works in Linux. Of course, suitable networking hardware and kernel drivers need to be enabled to get the packets.

On a busy network, you may have thousands of packets in a capture file and be interested in only some of them. To make it easier to interpret the Ethereal display, which can get pretty busy, you can use colors. From the Display→Colorize Display... option, you can select display packets in various colors; Figure 7 shows how the filter is specified. In this case, I'm filtering on only a single field (the version number for Service Location Protocol), but you can build sophisticated filters with Boolean logic. Figure 8 shows a typical example with a few filters, and Figure 9 shows the working display (with Service Location Protocol Version 2 in red, DNS in green and ARP in blue). You can use a wide range of text colors as well as background coloring to separate out the various protocols.

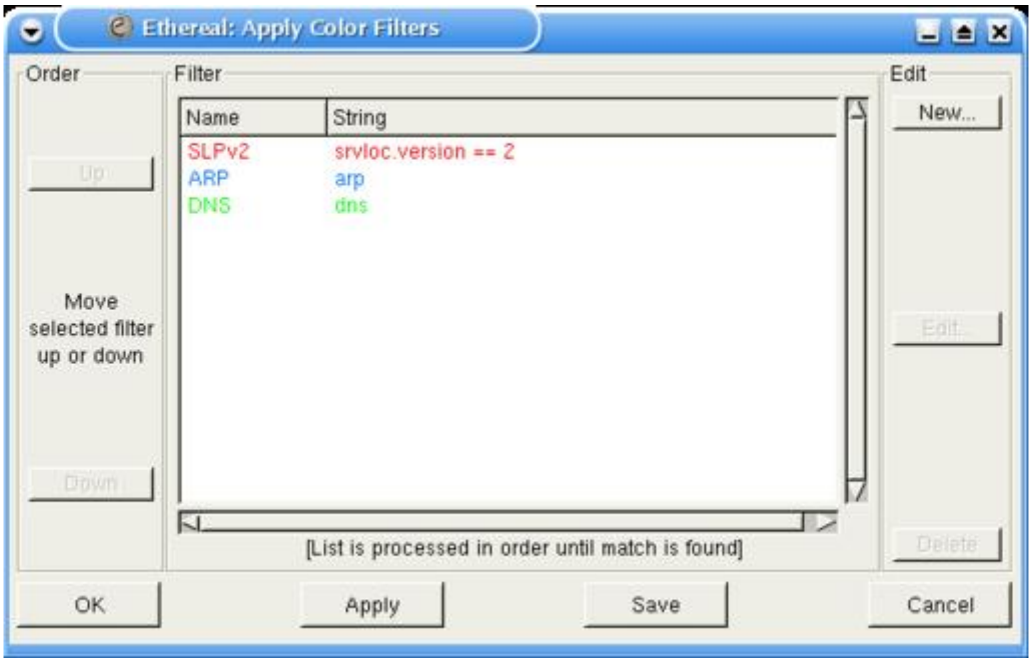

Ethereal: Edit Color Filter

Filter

Name: SLPv2

String: srvloc.version == 2

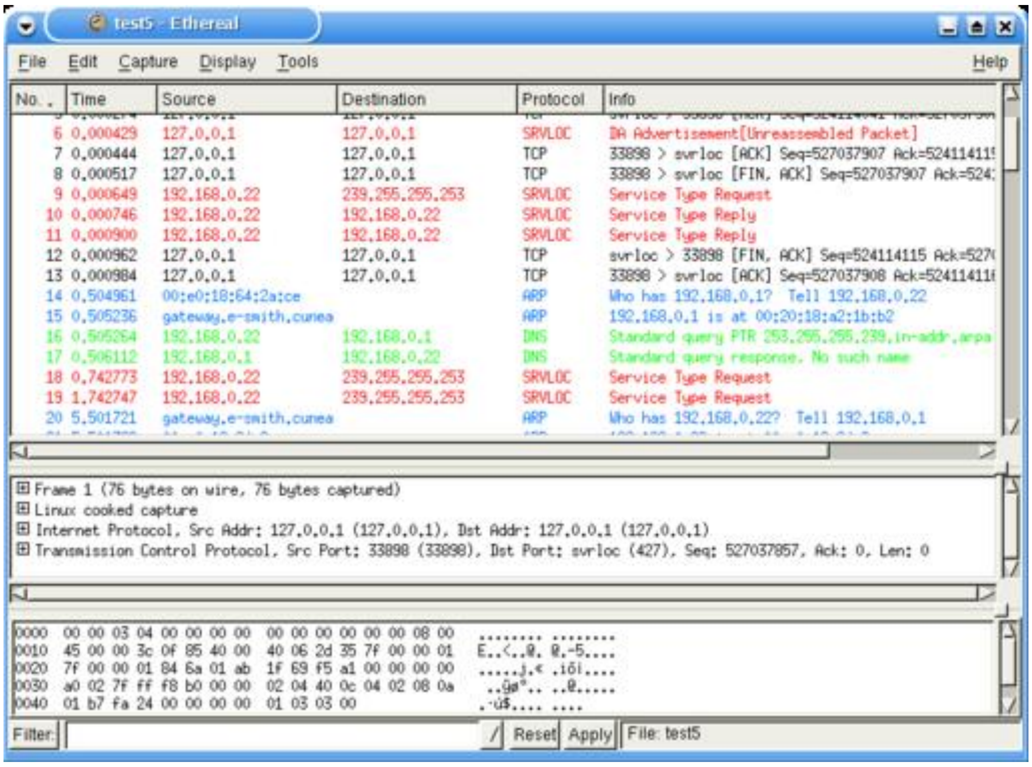
Figure 7. Specifying an Ethereal Color Filter


Ethereal: Apply Color Filters
Order
Up
Move selected filter up or down
Down
Filter
Edit
New...
Edit...
Delete
OK
Apply
Save
Cancel

Name	String
SLPv2	srvloc.version == 2
ARP	arp
DNS	dns

[List is processed in order until match is found]

Figure 8. Ethereal Supports Multiple Color Filters



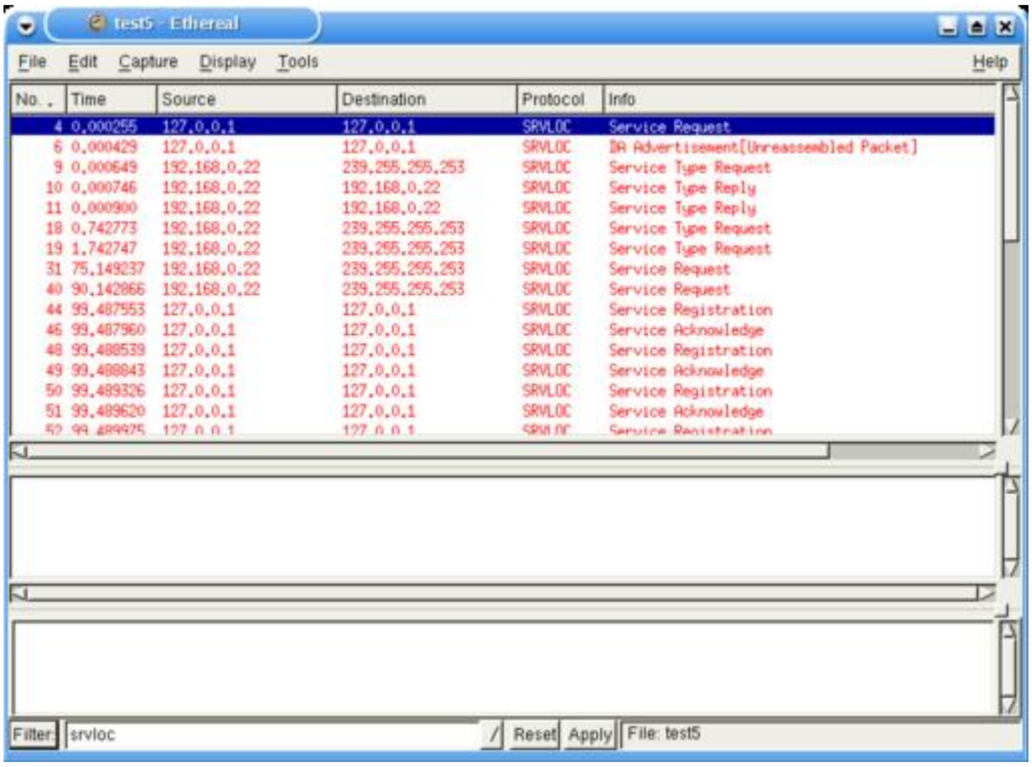
```
No. Time     Source                Destination       Protocol Info
 6 0.000429  127.0.0.1             127.0.0.1         SRVLOC   DA Advertisement[Unreassembled Packet]
 7 0.000444  127.0.0.1             127.0.0.1         TCP      33898 > svrloc [ACK] Seq=527037907 Ack=52411411
 8 0.000517  127.0.0.1             127.0.0.1         TCP      33898 > svrloc [FIN, ACK] Seq=527037907 Ack=524
 9 0.000649  192.168.0.22          239.255.255.253   SRVLOC   Service Type Request
10 0.000746  192.168.0.22          192.168.0.22      SRVLOC   Service Type Reply
11 0.000900  192.168.0.22          192.168.0.22      SRVLOC   Service Type Reply
12 0.000962  127.0.0.1             127.0.0.1         TCP      svrloc > 33898 [FIN, ACK] Seq=524114115 Ack=527
13 0.000984  127.0.0.1             127.0.0.1         TCP      33898 > svrloc [ACK] Seq=527037908 Ack=52411411
14 0.504961  00:e0:18:64:2a:ce                       ARP      Who has 192.168.0.1?  Tell 192.168.0.22
15 0.505236  gateway.e-smith.cunea                   ARP      192.168.0.1 is at 00:20:18:a2:1b:b2
16 0.505264  192.168.0.22          192.168.0.1       DNS      Standard query PTR 253.255.255.239.in-addr.arpa
17 0.506112  192.168.0.1           192.168.0.22      DNS      Standard query response, No such name
18 0.742773  192.168.0.22          239.255.255.253   SRVLOC   Service Type Request
19 1.742747  192.168.0.22          239.255.255.253   SRVLOC   Service Type Request
20 5.501721  gateway.e-smith.cunea                   ARP      Who has 192.168.0.22?  Tell 192.168.0.1
```

```
⊞ Frame 1 (76 bytes on wire, 76 bytes captured)
⊞ Linux cooked capture
⊞ Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
⊞ Transmission Control Protocol, Src Port: 33898 (33898), Dst Port: svrloc (427), Seq: 527037857, Ack: 0, Len: 0
```

```
0000  00 00 03 04 00 00 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 3c 0f 85 40 00  40 06 2d 35 7f 00 00 01   E..<..@. @.-5....
0020  7f 00 00 01 84 6a 01 ab  1f 69 f5 a1 00 00 00 00   .....j.« .iõ¡....
0030  a0 02 7f ff f8 b0 00 00  02 04 40 0c 04 02 08 0a    ..ÿø°.. ..@.....
0040  01 b7 fa 24 00 00 00 00  01 03 03 00               .·ú$.... ....
```

Figure 9. A Typical Colorized Capture Session

After coloring the display, the next step is to remove packets of no interest, a task Ethereal handles through display filtering. A simple example is shown in Figure 10, where adding a srvloc filter (in the bottom left of the window) has removed all the other protocols, leaving only the Service Location Protocol. If this still is too complex, you could choose to change the coloring again, this time showing packets from particular hosts in separate colors or packets containing particular types of client requests or server responses in particular colors.

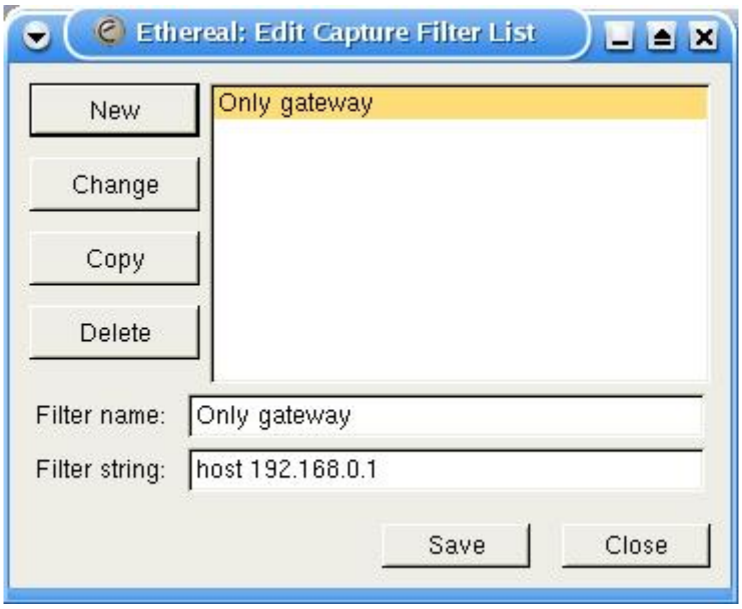


No.	Time	Source	Destination	Protocol	Info
4	0.000255	127.0.0.1	127.0.0.1	SRVLOC	Service Request
6	0.000429	127.0.0.1	127.0.0.1	SRVLOC	DA Advertisement[Unreassembled Packet]
9	0.000649	192.168.0.22	239.255.255.253	SRVLOC	Service Type Request
10	0.000746	192.168.0.22	192.168.0.22	SRVLOC	Service Type Reply
11	0.000900	192.168.0.22	192.168.0.22	SRVLOC	Service Type Reply
18	0.742773	192.168.0.22	239.255.255.253	SRVLOC	Service Type Request
19	1.742747	192.168.0.22	239.255.255.253	SRVLOC	Service Type Request
31	75.149237	192.168.0.22	239.255.255.253	SRVLOC	Service Request
40	90.142866	192.168.0.22	239.255.255.253	SRVLOC	Service Request
44	99.487553	127.0.0.1	127.0.0.1	SRVLOC	Service Registration
46	99.487960	127.0.0.1	127.0.0.1	SRVLOC	Service Acknowledge
48	99.488539	127.0.0.1	127.0.0.1	SRVLOC	Service Registration
49	99.488843	127.0.0.1	127.0.0.1	SRVLOC	Service Acknowledge
50	99.489326	127.0.0.1	127.0.0.1	SRVLOC	Service Registration
51	99.489620	127.0.0.1	127.0.0.1	SRVLOC	Service Acknowledge
52	99.489975	127.0.0.1	127.0.0.1	SRVLOC	Service Registration

Filter: srvloc

Figure 10. Display Filtering on Same Session as Shown in Figure 9

Another option is to not capture the unwanted packets in the first place. To do this, Ethereal supports the same capture filter syntax that tcpdump uses. An example of this syntax is shown in Figure 11, where the dialog captures only the packets going to or from the machine with IP 192.168.0.1. Unfortunately, the syntax used in capture filters is different from that used in the display filters, a fact that makes capture filtering much less accessible to occasional users.



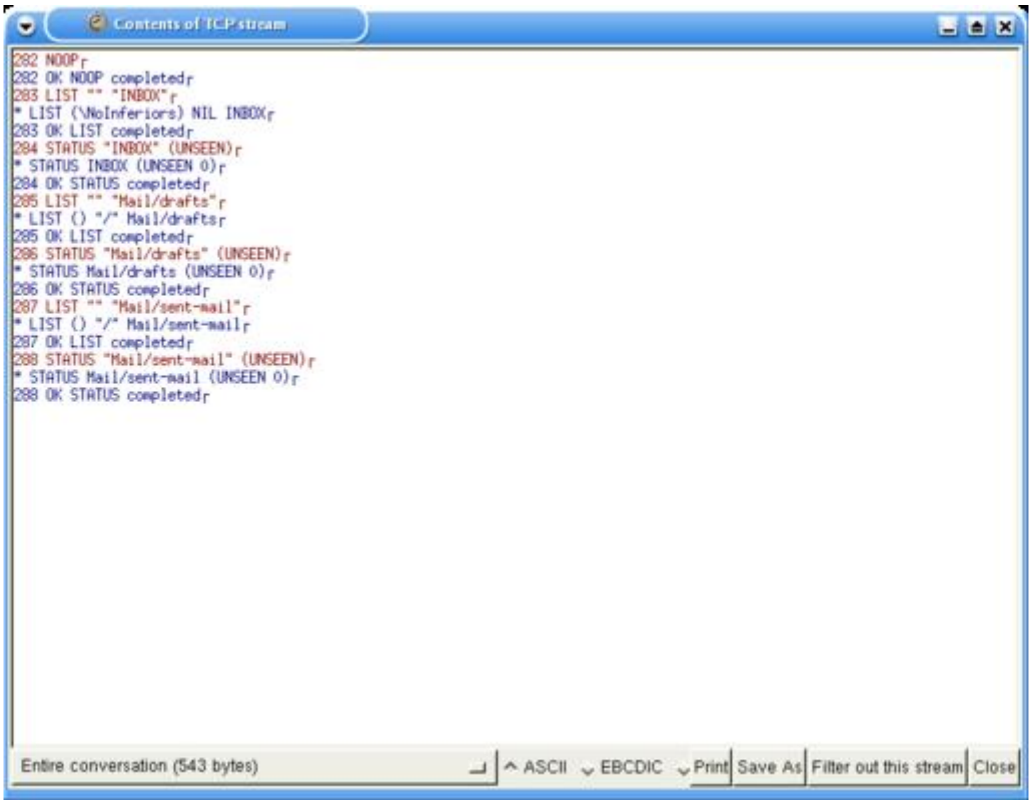
Only gateway

Filter name: Only gateway

Filter string: host 192.168.0.1

Figure 11. Capture Filtering Dialog

Another feature that some people find useful is the Follow TCP Stream... tool, which presents a text representation of the conversation. I personally don't use this feature often, but it is a powerful tool for looking at text-based protocols such as IMAP (Figure 12).



```
282 NOOP┌
282 OK NOOP completed┌
283 LIST "" "INBOX"┌
* LIST (\NoInferiors) NIL INBOX┌
283 OK LIST completed┌
284 STATUS "INBOX" (UNSEEN)┌
* STATUS INBOX (UNSEEN 0)┌
284 OK STATUS completed┌
285 LIST "" "Mail/drafts"┌
* LIST () "/" Mail/drafts┌
285 OK LIST completed┌
286 STATUS "Mail/drafts" (UNSEEN)┌
* STATUS Mail/drafts (UNSEEN 0)┌
286 OK STATUS completed┌
287 LIST "" "Mail/sent-mail"┌
* LIST () "/" Mail/sent-mail┌
287 OK LIST completed┌
288 STATUS "Mail/sent-mail" (UNSEEN)┌
* STATUS Mail/sent-mail (UNSEEN 0)┌
288 OK STATUS completed┌
```

Figure 12. Following a TCP Stream—IMAP

Misfeatures and Omissions

Apart from the different syntaxes required for capturing and displaying filters, I've come across a few other issues in the time I've been using Ethereal. Some of these have to do with personal preferences, and others have been gleaned from monitoring the Ethereal mailing lists.

At the time of this writing, my biggest issue is with the quality of the support documentation, especially the User's Guide, which is incomplete and outdated. Also, a significant amount of the User's Guide, about the last 80%, is generated automatically and is not user-friendly. In addition, the version on the Web site has not been regenerated in some time. I personally found the GUI a little difficult to get used to, although as I became more familiar with the various menus, I became more productive with Ethereal. Perhaps some better documentation would have helped with this. There is also limited developer documentation, although I see this as a less important issue, given the large number of examples from which you can work.

Various users occasionally ask “when will such and such a protocol be supported?” Where I have found a few protocols not supported by Ethereal (rsync, distcc and ACAP), I've generally needed to code support myself. This is fairly easy to do with Ethereal. If you need support for a particular protocol, however, and it is not supported by Ethereal at the moment, you should allow for some development effort (either as an in-house development or on a contract basis) before committing to Ethereal. If you do develop additional dissectors or enhance an existing one, I strongly recommend that you have it incorporated into the Ethereal source tree to ensure it remains up to date.

Another feature supported by other packet analysis tools is the ability to capture data on a remote host and display it locally. If you can run Ethereal on the remote host, this scenario is possible, but often you want to capture data on a machine acting as a router or a server, where a full-blown X environment is undesirable. This lack may be overcome in a future version or it may not be particularly important, depending on your environment.

The only other issue worth mentioning is that a substantial number of the queries on the user-support mailing list seem to be from Windows users experiencing a wide range of problems. I personally haven't run the Windows version, so I don't know if the difficulties are associated with the underlying tools (especially WinPcap), Windows itself or the skill levels of the users.

Resources

For more information on Ethereal, start with www.ethereal.com. This page includes links to the Ethereal manual, downloads and mailing lists.

To understand better what Ethereal is showing you, you need the appropriate documentation on the network protocol. Those protocols, codified by the Internet Engineering Task Force, are available at www.rfc-editor.org.

rsync is an efficient network file transfer application originally developed by Andrew Tridgell (of Samba fame). See rsync.samba.org.

zcip is a tool for automatic assignment (zeroconf) of IPv4 addresses, without needing a DHCP server. See zeroconf.sourceforge.net.

Service Location Protocol is a way for clients to find servers in a network-efficient way. See www.srvloc.org for more details on the protocol, or refer to RFC 2608, RFC 2609, RFC 2610 and RFC 2614, available from www.rfc-editor.org. A free implementation, mainly developed by Matt Peterson, is available at www.openslp.org.

The capturing capabilities of Ethereal depend on libpcap, developed by the TCPDUMP Group. You need libpcap to build Ethereal, although most distributions ship with libpcap packages. See www.tcpdump.org.

The Windows version of libpcap is WinPcap. See winpcap.polito.it for more information and to download the installer package.

IMAP (Internet Message Access Protocol) is a server-based e-mail protocol, in many ways superior to the Post Office Protocol (POP) that is widely used. For more details, get RFC 2060 from www.rfc-editor.org.

distcc is a distributed compilation application developed by Martin Pool that uses various network machines to participate in building C code. See distcc.samba.org.

Brad Hards is the technical director for Sigma Bravo, a small professional services company in Canberra, Australia. In addition to Linux, his technical foci include aircraft system integration and certification, GPS and electronic warfare. Comments on this article may be sent to bradh@frogmouth.net.

LinuxBIOS at Four

Ronald G. Minnich

Issue #118, February 2004

LinuxBIOS is more than a way to boot your Linux box in a few seconds. The new in-demand software for Linux cluster sites also offers a fallback mode that can save your system if a power failure strikes during a BIOS upgrade.

LinuxBIOS is a GPLed program that replaces the BIOS found on many computers, including AMD64, x86, Alpha and PowerPC systems. LinuxBIOS is a vendor-independent, architecture-neutral BIOS, more than 95% of which is written in C. LinuxBIOS is four years old. Some of the largest Linux clusters in the world use LinuxBIOS, and some of the smallest embedded systems in the world do too. LinuxBIOS has been used in robots searching for survivors in the World Trade Center, as well as robots used in Afghanistan and Iraq. LinuxBIOS is supported by many vendors, including AMD and Tyan. It now is possible, for example, to order LinuxBIOS motherboards from Tyan.

In this article I describe the basic structure of LinuxBIOS, the origins of LinuxBIOS and how it evolved to its current state. I also cover the platforms it supports and the lessons we have learned about trying to marry a GPL project to some of the lowest-level, most heavily guarded secrets that vendors possess.

LinuxBIOS Structure

Before we can explain LinuxBIOS structure we need to provide a quick overview of modern PC architectures. PCs consist of a set of chips, including the CPU, graphics and keyboard controller, all connected by buses. A bus is a set of one or more wires that can be used to interconnect two or more chips. Some buses have two wires, signal and ground, and other buses have tens or hundreds of wires.

A highly simplified diagram of PC architecture is shown in Figure 1. The different types of buses cannot be wired to one another directly, so chips known as bridges are used to connect one bus to another. The first bus is the front-side bus, and on most PCs it connects CPUs to one another and to the north bridge. The north bridge connects CPUs to both the memory bus and the PCI bus. In our diagram we show only one north bridge, but there are many variations on this theme. The AMD Opteron, for example, uses a north bridge for each CPU, and the front-side bus connects only each Opteron CPU to its own north bridge. In other words, there is no shared front-side bus on the Opteron. Nevertheless, the north bridge is an identifiable device in the Opteron chipset.

The south bridge, which almost always resides on PCI bus 0, is the next bridge in line. The south bridge interfaces from the PCI bus to legacy devices, namely the set of devices found on PCs ca. 1981. The south bridge also drives the BIOS Flash part.

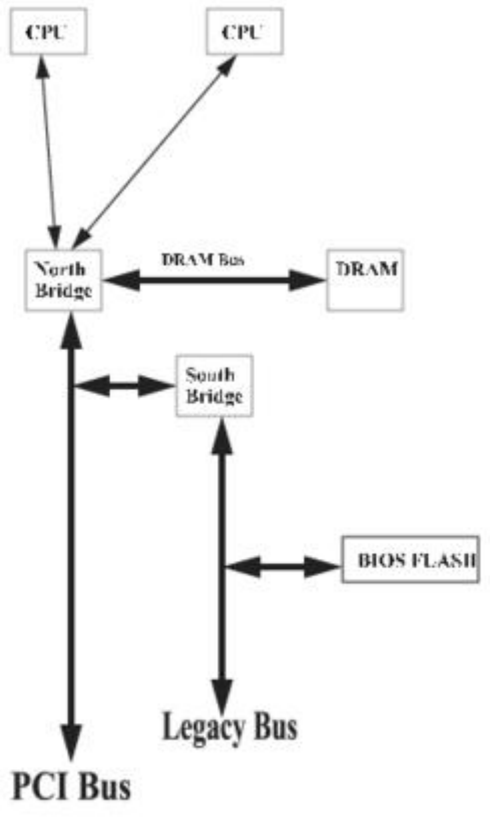


Figure 1. A simplified view of a basic PC architecture. Bridges are chips that connect one bus to another.

When the PC is turned on or reset, the CPUs start fetching from a known address, which traditionally has been from the top of memory (TOM) minus 16 bytes. In the original 8086, this was address 0xffff0; on newer PCs, it is address 0xfffffff0. This initial instruction fetch has to be supported by the hardware somehow, even before it has been configured. A lot of the hardware has to work for that first instruction fetch.

Nevertheless, when first turned on the PC hardly is ready to run C code and barely is ready to run assembly code. The motherboard has to be brought to life in stages. As a result, LinuxBIOS has a sequence of bootstraps, each bootstrap being invoked when additional CPU resources are activated. Each bootstrap assumes that certain resources have been enabled and that the machine has a well-defined set of resources available.

These LinuxBIOS pieces are:

- The first 10 or 15 instructions that enable the CPU, enable a minimal virtual memory capability (at minimum, 32-bit addresses) and enable other resources needed to turn on memory (such as the I2C bus). They also set the internal CPU state to clean up things, such as instruction pipelines.
- Memory startup code, which requires a sane CPU and a working I2C bus for interrogating memory parameters.
- Code that loads object code originally written in C from Flash to memory. The object code optionally can be compressed.
- Code that can be run once memory is working. This code scans all the hardware resources and initializes them.
- One or more payloads that perform any custom final configuration work and boot an OS.

We show all the phases in Figure 2.

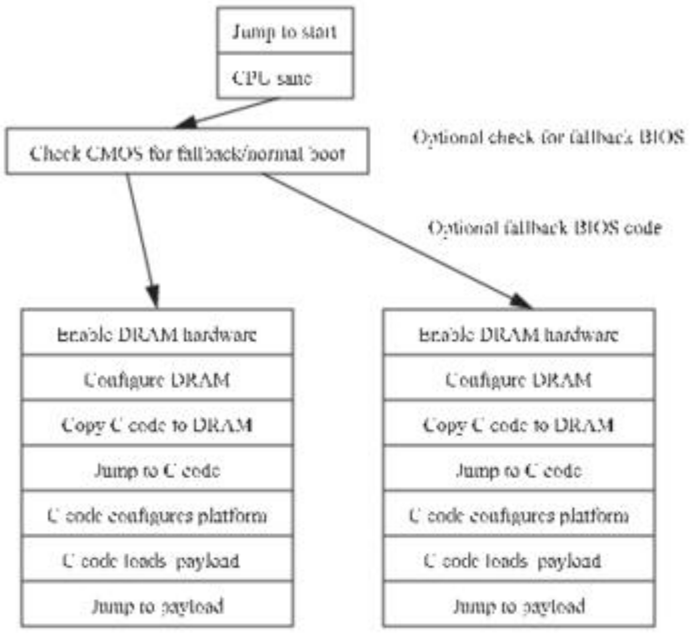


Figure 2. Phases of LinuxBIOS

LinuxBIOS supports an optional fallback BIOS in the event of BIOS problems. The fallback support is built in to the BIOS when it is compiled. Additional code checks flags in the CMOS and determines whether the CMOS is corrupted, whether the previous BIOS failed to start correctly or whether the user wishes to boot in to the fallback CMOS. The fallback BIOS is a complete LinuxBIOS image, and its capabilities are not limited in any way.

The fallback capability is useful for unattended BIOS updates. Consider the case of updating the BIOS on 1,024 or more nodes—what if it fails halfway through? For most systems, you now have a very expensive, very heavy paperweight. With LinuxBIOS, one simply resets the nodes and they come back up automatically in fallback mode.

Origins and Evolution of LinuxBIOS

I started the LinuxBIOS Project at Los Alamos National Lab (LANL) in September 1999. For the prior eight years, I had been building clusters of all kinds and had built my first PC cluster in 1994. In all this time, the BIOS had been a stumbling block in constructing larger clusters.

In 1997, I built the 144-node Cyclone cluster at the Sarnoff Corporation. As an experiment, we had only 16 nodes with video. The experiment was not successful; PCs using the standard BIOS simply are too unreliable to have the video removed, because PC failure recovery always requires interaction with the BIOS. It was clear that if we were to move to ever-larger PC clusters, we needed to resolve the problems of the BIOS.

We decided the ideal PC cluster node would have the following capabilities: boots directly into an OS from some onboard, nonvolatile RAM; configures all the network interfaces but configures no other hardware; connects to a control node using any working network interface; and takes action only at the direction of the control node.

Private industry was not the place to move on this kind of work, however, so we never were able to take these ideas past the talking stage.

Once I got to LANL, I had the ability to pursue these ideas. Several technology trends also made 1999 a far better year than 1997 to look at this problem. In 1999, motherboards with 1MB of Flash were appearing, and the self-describing PCI bus had replaced the older EISA and ISA buses completely. Also important, Linux was becoming much better at doing more configuration, as exemplified by the SGI Visual Workstation, which didn't even have a standard BIOS.

It seemed clear that if we could put Linux in the BIOS part, we could achieve our goals. Linux can do a far better job of running the hardware than any BIOS we have seen. What we needed was a simple hardware bootstrap that loaded Linux from Flash into memory; Linux would do the rest. Hence, our early motto, “Let Linux do it!”

Before we got the LinuxBIOS Project going full steam, we needed to ensure that Linux could be used as an OS bootstrap, which meant that Linux had to be able to boot Linux. By December 1999, we had demonstrated Linux booting Linux with the LOBOS work.

The easiest way to get work done in an Open Source world is to let somebody else do it for you, so the next step in LinuxBIOS was to look for somebody else's software. James Hendricks and Dale Webster found such a system in the OpenBIOS Project. In the space of five days, starting with the OpenBIOS source, they wrote and built a test system on our Intel L440GX+ motherboards that could boot the system from reset—not power on, but reset. Starting from power on would take another five months to figure out, but it wasn't bad work for five vacation days.

We realized early on that assembly code could not be the future of LinuxBIOS. OpenBIOS was a lot of assembly code, with a difficult-to-master build structure. Our small community began a search for a better foundation for LinuxBIOS. Jeff Garzik found a new BIOS and learned that STPC, which had written it, was willing to open source it. The STPC BIOS became the code base for the new LinuxBIOS. The STPC code required substantial reorganization so it could support multiple motherboards and chipsets, but it did provide a good starting point.

The next six months were spent getting a few platforms to run LinuxBIOS. Our first non-graphical platform was an Intel L440GX+ motherboard, followed by an SiS 630 motherboard. With the SiS, we got our first corporate involvement. SiS supplied data books, schematics, assembly code and technical support, all aimed at getting LinuxBIOS running on its platform.

We learned what Linux could and could not do. At the time, we were working with kernel version 2.2. We learned that Linux could not configure a PCI bus from scratch—LinuxBIOS had to do that. We were able to take the PCI code from Linux and, with modifications, use it directly in LinuxBIOS, while adding the extensions we needed for true PCI configuration. We learned that LinuxBIOS came up so fast, the IDE drives were not spun up. We continue to support a patch for Linux to work around this problem. These and a host of other lessons required some unexpected changes in our “Let Linux do it!” philosophy.

By the nine-month mark, we had LinuxBIOS working well on two platforms, written mostly in C, and we had the beginnings of corporate interest. VIA and Acer contributed data books that allowed us to port to their new chipsets. That summer James Hendricks began work on SMP support, and in “Let Linux do it!” mode, that support was written as patches to the Linux kernel, not as extensions to LinuxBIOS. At one point, with our patches, a Linux kernel could come up as a uniprocessor and enable the additional processors from scratch —something that heretofore only the BIOSes knew how to do.

That summer, Linux NetworX joined the effort, and to our good fortune, Eric Biederman got involved. Eric's most important early work was the Alpha port. Eric also cleaned up the memory startup code significantly. Our collaboration continues to this day; Linux NetworX is the largest reseller of LinuxBIOS-based systems, and Eric has spearheaded the creation and architecture of version 2 of LinuxBIOS.

That fall, we presented talks at Atlanta Linux Showcase 2000, and while there met Steve James from Linux Labs. This partnership allowed us, in the space of less than a month, to realize our dream: we built a 13-node LinuxBIOS-based cluster for Supercomputing 2000. The cluster booted to full operational status in about 13 seconds.

By 2001, Linux NetworX had completed the Alpha port for the DS10. We then built a cluster with 104 DS10s, all running LinuxBIOS. The DS10 booted more slowly than the Pentium systems, so it took this cluster 50 or so seconds to come to full operational status, a speed that still was quite acceptable. We were used to BIOSes that took 50 seconds simply to test memory.

The Alpha port demonstrated that LinuxBIOS was portable. Little if any of the code changed, and yet LinuxBIOS worked fine as a 64-bit BIOS or as a 32-bit BIOS.

Since 2001, we have added developers (there are now 11) and continued to port to more platforms, the most recent being the AMD Opteron. We envisioned LinuxBIOS as purely for clusters, but now non-cluster use far outstrips LinuxBIOS use in clusters. We thought Linux could do everything hard; LinuxBIOS does a lot now, including SMP startup. We would have preferred to “Let Linux do it”, but the design of the AMD K7 SMP hardware requires that SMP startup be done in the BIOS.

We thought vendors would jump in. It has taken four years, but in this fifth year of LinuxBIOS development, we now are finding some of the largest computer vendors in the world expressing interest. We simply were a little optimistic on the time frame. Once vendors see the business case, however, they get involved. Vendors sold at least $30 million US worth of LinuxBIOS-based systems in 2003, up from $0 million in 2000.

Platforms

LinuxBIOS runs on a wide range of platforms. Fifty supported motherboards are in the source tree, but we have found that many motherboards are so similar that a LinuxBIOS for one motherboard can work on another. Companies build code for one motherboard, run it on another motherboard and do not always get around to telling us.

LinuxBIOS works on 64-bit and 32-bit CPUs. CPUs supported include the Alpha, K8, K7, PowerPC, P4, PIII, PII, Cyrix (VIA), Geode (now AMD) and SC520 (AMD). Chipsets are too numerous to list. Form factors of mainboards range from the smallest PC/104 systems to the largest K8 systems. An IBM PPC 970 port is in progress.

Chipset Secrets

One of the most common phrases we heard from chip vendors in the first few years was “we'll never tell you that.” “That” being CPU information, chipset information, motherboard information or any combination of the three. The designs for these three systems constitute highly guarded secrets. It seems amazing, even now, that vendors are able to let us build a GPLed BIOS that by its nature exposes some of these secrets.

How was it possible for us to get this type of information? Simple, businesses are not charities. If there is no business case for releasing this information to us, they do not do it. If, however, there is a business case, then it happens—sometimes with astonishing speed.

From what we can see, the two factors in our success were competition and the creation of a market. Competition gave us a wide variety of choices as to motherboard, chipset and CPU. Once there was a reasonable market, vendors were concerned about being left out.

The experience at LANL is revealing. LANL's last two large cluster RFPs have specified LinuxBIOS as a mandatory requirement. Spending on these RFPs has come in at over $19 million US. Companies that had decided not to become involved in LinuxBIOS could not respond to these RFPs. Companies that had the foresight to get involved in LinuxBIOS early in the game were equipped to respond. Foresight, in this case, conferred a competitive advantage.

Conclusions

LinuxBIOS has come a long way in four years—as one person put it, from “I'm Possible” to “In Production”. LinuxBIOS is used on everything from the largest Linux clusters yet built to the small—test instruments, MP3 players and portable clusters.

LinuxBIOS makes it possible to build systems without PC hardware baggage. The systems can be optimized for Linux and thus can be more compact and simpler. There is increasingly a business case for such systems.

LinuxBIOS is now in its second version, with four years, at least six CPUs and over 50 motherboards' worth of experience behind it. It now takes only days in some cases to do a port to a new system; originally, it took months. LinuxBIOS' impact on the world of computing is only beginning.

Acknowledgements

So many people have contributed to LinuxBIOS that it is easy to slight them by listing some and not all. Nevertheless, a few contributors stand out as having made LinuxBIOS possible. First, of course, is Stefan Reinauer and the OpenBIOS effort; Jeff Garzik, who got the STPC BIOS Project set up on SourceForge as FreeBIOS; Ollie Lho, who did so much to get our first workstation platforms going in 2000; Steve James and Linux Labs, who worked with us and expedited the shipment in 2000 of our first LinuxBIOS cluster; Greg Watson, who did the PowerPC port; and Eric Biederman, who has done so much to get our really hard platforms up and stable and who has done so much to create version 2.

This paper is released under LAUR 03-8165. This research was funded by the Mathematical Information and Computer Sciences (MICS) Program of the DOE Office of Science and the Los Alamos Computer Science Institute (ASCI). Los Alamos National Laboratory is operated by the University of California for the National Nuclear Security Administration of the United States Department of Energy under contract W-7404-ENG-36.

Ronald G. Minnich has been working in high-performance computing and clustering for 15 years. He recently realized that one of his first clusters, a 16-node SPARC cluster, has a total power equivalent to one-fourth of one of the 2,048 processors in his newest cluster; his new cluster has 10,000 times the power of his first one. Ron started working with UNIX in 1976, with Linux in 1993 and built his first PC cluster in 1994.

I2C Drivers, Part II

Greg Kroah-Hartman

Issue #118, February 2004

Here's what has to happen to read the sensors that report the temperature, fan speed and other important system health information.

In my last column [*LJ*, December 2003], we discussed how I2C bus drivers and I2C algorithm drivers work. We also described how to make a tiny dummy I2C bus driver. This month, we discuss how an I2C chip driver works and provide an example of one in action.

An I2C chip driver controls the process of talking to an individual I2C device that lives on an I2C bus. I2C chip devices usually monitor a number of different physical devices on a motherboard, such as the different fan speeds, temperature values and voltages.

The struct i2c_driver structure describes a I2C chip driver. This structure is defined in the include/linux/i2c.h file. Only the following fields are necessary to create a working chip driver:

- struct module *owner; — set to the value THIS_MODULE that allows the proper module reference counting.
- char name[I2C_NAME_SIZE]; — set to a descriptive name of the I2C chip driver. This value shows up in the sysfs file name created for every I2C chip device.
- unsigned int flags; — set to the value I2C_DF_NOTIFY in order for the chip driver to be notified of any new I2C devices loaded after this driver is loaded. This field probably will go away soon, as almost all drivers set this field.
- int (*attach_adapter)(struct i2c_adapter *); — called whenever a new I2C bus driver is loaded in the system. This function is described in more detail below.

- int (*detach_client)(struct i2c_client *); — called when the i2c_client device is to be removed from the system. More information about this function is provided below.

The following code is from an example I2C chip driver called tiny_i2c_chip.c., which is available from the *Linux Journal* FTP site [ftp.linuxjournal.com/pub/lj/listings/issue118/7252.tgz]. It shows how the struct i2c_driver structure is set up:

```
static struct i2c_driver chip_driver = {
    .owner          = THIS_MODULE,
    .name           = "tiny_chip",
    .flags          = I2C_DF_NOTIFY,
    .attach_adapter = chip_attach_adapter,
    .detach_client  = chip_detach_client,
};
```

Registering a Chip Driver

To register this I2C chip driver, the function i2c_add_driver should be called with a pointer to the struct i2c_driver:

```
static int __init tiny_init(void)
{
    return i2c_add_driver(&chip_driver);
}
```

To unregister the I2C chip driver, the i2c_del_driver function should be called with the same pointer to the struct i2c_driver:

```
static void __exit tiny_exit(void)
{
    i2c_del_driver(&chip_driver);
}
```

After the I2C chip driver is registered, the attach_adapter function callback is called when an I2C bus driver is loaded. This function checks to see if any I2C devices are on this I2C bus to which the client driver wants to attach. Almost all I2C chip drivers call the core I2C function i2c_detect to determine this. For example, the tiny_i2c_chip.c driver does this:

```
static int
chip_attach_adapter(struct i2c_adapter *adapter)
{
    return i2c_detect(adapter, &addr_data,
                      chip_detect);
}
```

The i2c_detect function probes the I2C adapter, looking for the different addresses specified in the addr_data structure. If a device is found, the chip_detect function then is called.

If you look closely at the source code, you cannot find the addr_data structure anywhere. The reason for this is it is created by the SENSORS_INSMOD_1 macro. This macro is defined in the include/linux/i2c-sensor.h file and is quite complicated. It sets up a static variable called addr_data based on the number of different types of chips that this driver supports and the addresses at which these chips typically are present. It then provides the ability to override these values by using module parameters. An I2C chip driver must provide the variables normal_i2c, normal_i2c_range, normal_isa and normal_isa_range. These variables define the i2c smbus and i2c isa addresses this chip driver supports. They are an array of addresses, all terminated by either the special value I2C_CLIENT_END or I2C_CLIENT_ISA_END. Usually a specific type of I2C chip shows up in only a limited range of addresses. The tiny_i2c_client.c driver defines these variables as:

```
static unsigned short normal_i2c[] =
  { I2C_CLIENT_END };
static unsigned short normal_i2c_range[] =
  { 0x00, 0xff, I2C_CLIENT_END };
static unsigned int normal_isa[] =
  { I2C_CLIENT_ISA_END };
static unsigned int normal_isa_range[] =
  { I2C_CLIENT_ISA_END };
```

The normal_i2c_range variable specifies that we can find this chip device at any I2C smbus address. This allows us to test this driver on almost any I2C bus driver.

What to Do When the Chip Is Found

In the tiny_i2c_chip.c driver, when an I2C chip device is found, the function chip_detect is called by the I2C core. This function is declared with the following parameters:

```
static int
chip_detect(struct i2c_adapter *adapter,
            int address, int kind);
```

The adapter variable is the I2C adapter structure on which this chip is located. The address variable contains the address where the chip was found, and the kind variable indicates what kind of chip was found. The kind variable usually is ignored, but some I2C chip drivers support different kinds of I2C chips, so this variable can be used to determine the type of chip present.

This function is responsible for creating a struct i2c_client structure that then is registered with the I2C core. The I2C core uses that structure as an individual I2C chip device. To create this structure, the chip_detect function does the following:

```
struct i2c_client *new_client = NULL;
struct chip_data *data = NULL;
int err = 0;

new_client = kmalloc(sizeof(*new_client),
                     GFP_KERNEL);
if (!new_client) {
    err = -ENOMEM;
    goto error;
}
memset(new_client, 0x00, sizeof(*new_client));

data = kmalloc(sizeof(*data), GFP_KERNEL);
if (!data) {
    err = -ENOMEM;
    goto error;
}
memset(data, 0x00, sizeof(*data));

i2c_set_clientdata(new_client, data);
new_client->addr = address;
new_client->adapter = adapter;
new_client->driver = &chip_driver;
new_client->flags = 0;
strncpy(new_client->name, "tiny_chip",
        I2C_NAME_SIZE);
```

First, the struct i2c_client structure and a separate local data structure (called struct chip_data) are created and initialized to zero. It is important that the i2c_client structure is initialized to zero, as the lower levels of the kernel driver core require this in order to work properly. After the memory is allocated successfully, some fields in the struct i2c_client are set to point to this specific device and this specific driver. Notably, the addr, adapter and driver variables must be initialized. The name of the struct i2c_client also must be set if it is to show up properly in the sysfs tree for this I2C device.

After the struct i2c_client structure is initialized, it must be registered with the I2C core. This is done with a call to the i2c_attach_client function:

```
/* Tell the I2C layer a new client has arrived */
err = i2c_attach_client(new_client);
if (err)
    goto error;
```

When this function returns, with no errors reported, the I2C chip device is set up properly in the kernel.

I2C and sysfs

In the 2.0, 2.2 and 2.4 kernels, the I2C code would place the I2C chip devices in the /proc/bus/i2c directory. In the 2.6 kernel, all I2C chip devices and adapters show up in the sysfs filesystem. I2C chip devices can be found at /sys/bus/i2c/ devices, listed by their adapter address and chip address. For example, the tiny_i2c_chip driver loaded on a machine might produce the following sysfs tree structure:

```
$ tree /sys/bus/i2c/
/sys/bus/i2c/
|-- devices
|   |-- 0-0009 -> ../../../devices/pci0000:00/0000:00:06.0/i2c-0/0-000
|   |-- 0-000a -> ../../../devices/pci0000:00/0000:00:06.0/i2c-0/0-000
|   |-- 0-000b -> ../../../devices/pci0000:00/0000:00:06.0/i2c-0/0-000
|   `-- 0-0019 -> ../../../devices/pci0000:00/0000:00:06.0/i2c-0/0-001
`-- drivers
    |-- i2c_adapter
    `-- tiny_chip
        |-- 0-0009 -> ../../../../devices/pci0000:00/0000:00:06.0/i2c-
        |-- 0-000a -> ../../../../devices/pci0000:00/0000:00:06.0/i2c-
        |-- 0-000b -> ../../../../devices/pci0000:00/0000:00:06.0/i2c-
        `-- 0-0019 -> ../../../../devices/pci0000:00/0000:00:06.0/i2c-
```

This shows four different I2C chip devices, all controlled by the same tiny_chip driver. The controlling driver can be located by looking at the devices in the / sys/bus/i2c/drivers directory or by looking in the directory of the chip device itself and reading the name file:

```
$ cat /sys/devices/pci0000\:00/0000\:00\:06.0/i2c-0/0-0009/name
tiny_chip
```

All I2C chip drivers export the different sensor values through sysfs files within the I2C chip device directory. These filenames are standardized, along with the units in which the values are expressed, and are documented within the kernel tree in the file Documentation/i2c/sysfs-interface (Table 1).

Table 1. Sensor Values Exported through sysfs Files

temp_max[1-3]	Temperature max value. Fixed point value in form XXXXX and should be divided by 1,000 to get degrees Celsius. Read/Write value.

temp_min[1-3]	Temperature min or hysteresis value. Fixed point value in form XXXXX and should be divided by 1,000 to get degrees Celsius. This is preferably a hysteresis value, reported as an absolute temperature, *not* a delta from the max value. Read/Write value.
temp_input[1-3]	Temperature input value. Read-only value.

As the information in Table 1 shows, there is only one value per file. All files are readable and some can be written to by users with the proper privileges.

The tiny_i2c_chip.c driver emulates an I2C chip device that can report temperature values. It creates the files, temp_max1, temp_min1 and temp_input1 in sysfs. The values it returns when these files are read from is incremented every time the file is read to show how to access different unique chip values.

In order to create a file in sysfs, the DEVICE_ATTR macro is used:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
static DEVICE_ATTR(temp_min, S_IWUSR | S_IRUGO,
                   show_temp_hyst, set_temp_hyst);
static DEVICE_ATTR(temp_input, S_IRUGO,
                   show_temp_input, NULL);
```

This macro creates a structure that then is passed to the function device_create_file at the end of the chip_detect function:

```
/* Register sysfs files */
device_create_file(&new_client->dev,
                   &dev_attr_temp_max);
device_create_file(&new_client->dev,
                   &dev_attr_temp_min);
device_create_file(&new_client->dev,
                   &dev_attr_temp_input);
```

That call creates the sysfs files for the device:

```
/sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009
|-- detach_state
|-- name
|-- power
|   `-- state
|-- temp_input
|-- temp_max
`-- temp_min
```

The file name is created by the I2C core, and the files detach_state and power/state are created by the driver core.

But, let's go back to the DEVICE_ATTR macro. That macro wants to know the name of the file to be created, the mode of the file to be created, the name of the function to be called when the file is read from and the name of the function to be called when the file is written to. For the file temp_max, this declaration was:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
```

The function called when the file is read from is show_temp_max. This is defined, as are many sysfs files, with another macro that creates a function:

```
#define show(value) \
static ssize_t \
show_##value(struct device *dev, char *buf)        \
{                                                   \
    struct i2c_client *client = to_i2c_client(dev);\
    struct chip_data *data =                        \
        i2c_get_clientdata(client);                 \
                                                    \
    chip_update_client(client);                     \
    return sprintf(buf, "%d\n", data->value);       \
}
show(temp_max);
show(temp_hyst);
show(temp_input);
```

The reason this function is created with a macro is that it is quite simple to create other sysfs files that do almost the same thing, with different names and that read from different variables, without having to duplicate code. This single macro creates three different functions to read from three different variables from the struct chip_data structure.

In this function, the struct device * is converted into a struct i2c_client *. Then the private struct chip_data * is obtained from the struct i2c_client *. After that the chip data is updated with a call to chip_update_client. From there, the variable that has been asked for is printed into a buffer and returned to the driver core, which then returns it to the user:

```
$ cat /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_input
1
```

The chip_update_client increments all values by one every time it is called:

```
static void
```

```
chip_update_client(struct i2c_client *client)
{
    struct chip_data *data =
        i2c_get_clientdata(client);

    down(&data->update_lock);
    dev_dbg(&client->dev, "%s\n", __FUNCTION__);
    ++data->temp_input;
    ++data->temp_max;
    ++data->temp_hyst;
    data->last_updated = jiffies;
    data->valid = 1;
    up(&data->update_lock);
}
```

So, all subsequent requests for this value are different:

```
$ cat /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_input
2
$ cat /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_input
3
```

The set_temp_max function also is created from a macro to allow variables to be written to:

```
#define set(value, reg) \
static ssize_t                                     \
set_##value(struct device *dev,                    \
            const char *buf, size_t count)         \
{                                                  \
    struct i2c_client *client = to_i2c_client(dev);\
    struct chip_data *data =                       \
        i2c_get_clientdata(client);                \
    int temp = simple_strtoul(buf, NULL, 10);      \
                                                   \
    down(&data->update_lock);                      \
    data->value = temp;                            \
    up(&data->update_lock);                        \
    return count;                                  \
}
set(temp_max, REG_TEMP_OS);
set(temp_hyst, REG_TEMP_HYST);
```

Just like the show functions, this function converts the struct device * to a struct i2c_client *, and then the private struct chip_data * is found. The data the user provides then is turned into a number with a call to simple_strtoul and is saved into the proper variable:

```
$ cat /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_max
1
$ echo 41 > /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_max
$ cat /sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009/temp_max
42
```

Cleaning Up

When the I2C chip device is removed from the system, either by the I2C bus driver being unloaded or by the I2C chip driver being unloaded, the I2C core calls the detatch_client function specified in the struct i2c_driver structure. This usually is a simple function, as can be seen in the example driver's implementation:

```
static int chip_detach_client(struct i2c_client *client)
{
    struct chip_data *data = i2c_get_clientdata(client);
    int err;

    err = i2c_detach_client(client);
    if (err) {
        dev_err(&client->dev,
                "Client deregistration failed, "
                "client not detached.\n");
        return err;
    }
    kfree(client);
    kfree(data);
    return 0;
}
```

As the i2c_attach_client function was called to register the struct i2c_client structure with the I2C core, the i2c_detach_client function must be called to unregister it. If that function succeeds, the memory the driver has allocated for the I2C device then needs to be freed before returning from the function.

This example driver does not specifically remove the sysfs files from the sysfs core. This step is done automatically in the driver core within the i2c_detach_client function. But if the author would like, the file can be removed manually by a call to device_remove_file.

Conclusion

This two-part series of articles has explained the basics of how to write a kernel I2C bus driver, I2C algorithm driver and I2C chip driver. A lot of good information on how to write I2C drivers can be found in the Documentation/i2c directory in the kernel tree and on the Lm_sensors Web site (secure.netroedge.com/~lm78).

Greg Kroah-Hartman currently is the Linux kernel maintainer for a variety of different driver subsystems. He works for IBM, doing Linux kernel-related things, and can be reached at greg@kroah.com.

Archive Index Issue Table of Contents

Advanced search

Kernel Korner

I/O Schedulers

Robert Love

Issue #118, February 2004

Here's how I/O schedulers contribute to disk performance in Linux and the improvements you can get from the new I/O schedulers in the 2.6 kernel.

Although most Linux users are familiar with the role of process schedulers, such as the new O(1) scheduler, many users are not so familiar with the role of I/O schedulers. I/O schedulers are similar in some aspects to process schedulers; for instance, both schedule some resource among multiple users. A process scheduler virtualizes the resource of processor time among multiple executing processes on the system. So, what does an I/O scheduler schedule?

A naïve system would not even include an I/O scheduler. Unlike the process scheduler, the I/O scheduler is not a mandatory component of the operating system. Instead, performance is the I/O scheduler's sole *raison d'être*.

To understand the role of an I/O scheduler, let's go over some background information and then look at how a system behaves without an I/O scheduler. Hard disks address their data using the familiar geometry-based addressing of cylinders, heads and sectors. A hard drive is composed of multiple platters, each consisting of a single disk, spindle and read/write head. Each platter is divided further into circular ring-like tracks, similar to a CD or record. Finally, each track is composed of some integer number of sectors.

To locate a specific unit of data in a hard drive, the drive's logic requires three pieces of information: the cylinder, the head and the sector. The cylinder specifies the track on which the data resides. If you lay the platters on top of one another (as they are in a hard disk), a given track forms a cylinder through each platter. The head then identifies the exact read/write head (and thus the exact platter) in question. The search now is narrowed down to a single track on a single platter. Finally, the sector value denotes the exact sector on the track. The search is complete: the hard disk knows what sector, on what track, on what platter the data resides. It can position the read/write head of the correct platter over the correct track and read the proper sector.

Thankfully, modern hard disks do not force computers to communicate with them in terms of cylinders, heads and sectors. Instead, modern hard drives map a unique block number over each cylinder/head/sector triplet. The unique number identifies a specific cylinder/head/sector value. Modern operating systems then can address hard drives using this block number—called logical block addressing—and the hard drive translates the block number into the correct cylinder/head/sector value.

One thing of note about this block number: although nothing guarantees it, the physical mapping tends to be sequential. That is, logical block n tends to be physically adjacent to logical block n+1. We discuss why that is important later on.

Now, let's consider the typical UNIX system. Applications as varied as databases, e-mail clients, Web servers and text editors issue I/O requests to the disk, such as read this block and write to that block. The blocks tend to be located physically all over the disk. The e-mail spool may be located in an entirely different region of the disk from the Web server's HTML data or the text editor's configuration file. Indeed, even a single file can be strewn all over the disk if the file is fragmented, that is, not laid out in sequential blocks. Because the files are broken down into individual blocks, and hard drives are addressed by block and not the much more abstract concepts of files, reading or writing file data is broken down into a stream of many individual I/O requests, each to a different block. With luck, the blocks are sequential or at least physically close together. If the blocks are not near one another, the disk head must move to another location on the disk. Moving the disk head is called seeking, and it is one of the most expensive operations in a computer. The seek time on modern hard drives is measured in the tens of milliseconds. This is one reason why defragmented files are a good thing.

Unfortunately, it does not matter if the files are defragmented because the system is generating I/O requests for multiple files, all over the disk. The e-mail client wants a little from here and the Web server wants a little from there—but wait, now the text editor wants to read a file. The net effect is that the disk head is made to jump around the disk. In a worst-case scenario, with interleaved I/O requests to multiple files, the head can spend all of its time jumping around from one location to another—not a good thing for overall system performance.

This is where the I/O scheduler comes in. The I/O scheduler schedules the pending I/O requests in order to minimize the time spent moving the disk head. This, in turn, minimizes disk seek time and maximizes hard disk throughput.

This magic is accomplished through two main actions, sorting and merging. First, the I/O scheduler keeps the list of pending I/O requests sorted by block number. When a new I/O request is issued, it is inserted, block-wise, into the list of pending requests. This prevents the drive head from seeking all around the disk to service I/O requests. Instead, by keeping the list sorted, the disk head moves in a straight line around the disk. If the hard drive is busy servicing a request at one part of the disk, and a new request comes in to the same part of the disk, that request can be serviced before moving off to other parts of the disk.

Merging occurs when an I/O request is issued to an identical or adjacent region of the disk. Instead of issuing the new request on its own, it is merged into the identical or adjacent request. This minimizes the number of outstanding requests.

Let's look at an example. Consider the case where two applications issue requests to the following block numbers, such that they arrived in the kernel in this order: 10, 500, 12, 502, 14, 504 and 12. The I/O scheduler-less approach would service these blocks in the given order. That is seven long seeks, back and forth between two parts of the disk. What a waste! If the kernel sorted and merged these requests, however, and serviced them in that order, the results would be much different: 10, 12, 14, 500, 502 and 504. Only a single far-off seek and one less request overall.

In this manner, an I/O scheduler virtualizes the resources of disk I/O among multiple I/O requests in order to maximize global throughput. Because I/O throughput is so crucial to system performance and because seek time is so horribly slow, the job of an I/O scheduler is important.

The Linus Elevator

The I/O scheduler found in the 2.4 Linux kernel is named the Linus Elevator. I/O schedulers often are called elevator algorithms, because they tackle a problem similar to that of keeping an elevator moving smoothly in a large building. The Linus Elevator functions almost exactly like the classic I/O scheduler described above. For the most part, this was great because simplicity is a good thing and the 2.4 kernel's I/O scheduler just worked. Unfortunately, in the I/O scheduler's quest to maximize global I/O throughput, a trade-off was made: local fairness—in particular, request latency—can go easily out the window. Let's look at an example.

Consider a stream of requests to logical disk blocks such as 20, 30, 700 and 25. The I/O scheduler's sorting algorithm would queue and issue the requests in the following order (assuming the disk head currently is at the logical start of the disk): 20, 25, 30 and 700. This is expected and indeed correct. Assume, however, that halfway through servicing the request to block 25, another request comes in to the same part of the disk. And then another. And yet another. It is entirely feasible that the request way over to block 700 is not serviced for a relatively long time.

Worse, what if the request was to read a disk block? Read requests generally are synchronous. When an application issues a request to read some data, it typically blocks and waits until the kernel returns the data. The application must sit and wait, twiddling its thumbs, until that request way over at block 700 finally is serviced. Writes, on the other hand, typically are not synchronous—they are asynchronous. When an application issues a write, the kernel copies the data and metadata into the kernel, prepares a buffer to hold the data and returns to the application. The application does not really care or even know when the data actually hits the disk.

It gets worse for our friend the read request, however. Because writes are asynchronous, writes tend to stream. That is, it is common for a large writeback of a lot of data to occur. This implies that many individual write requests are submitted to a close area of the hard disk. As an example, consider saving a large file. The application dumps write requests on the system and hard drive as fast as it is scheduled.

Read requests, conversely, usually do not stream. Instead, applications submit read requests in small one-by-one chunks, with each chunk dependent on the last. Consider reading all of the files in a directory. The application opens the first file, issues a read request for a suitable chunk of the file, waits for the returned data, issues a read request for the next chunk, waits and continues likewise until the entire file is read. Then the file is closed, the next file is opened and the process repeats. Each subsequent request has to wait for the previous, which means substantial delays to this application if the requests are to far-off disk blocks. The phenomenon of streaming write requests starving dependent read requests is called writes-starving-reads (see Sidebar “Test 1. Writes-Starving-Reads”).

Test 1. Writes-Starving-Reads

In the background, perform a streaming write, such as:

```
while true
do
        dd if=/dev/zero of=file bs=1M
done
```

Meanwhile, time how long a simple read of a 200MB file takes:

```
time cat 200mb-file > /dev/null
```

This test demonstrates the writes-starving-reads problem.

The possibility of not servicing some requests in a reasonable amount of time is called starvation. Request starvation results in unfairness. In the case of I/O schedulers, the system explicitly has decided to trade fairness for improved global throughput. That is, the system attempts to improve the overall performance of the system at the possible expense of any one specific I/O request. This is accepted and, indeed, desired—except that prolonged starvation is bad. Starving read requests for even moderate lengths of time results in high application latency for applications issuing read requests during other disk activity. This high read latency adversely affects system performance and feel (see Sidebar "Test 2. Effects of High Read Latency").

Test 2. Effects of High Read Latency

Start a streaming read in the background:

```
while true
do
        cat big-file > /dev/null
done
```

Meanwhile, measure how long it takes for a read of every file in the kernel source tree to complete:

```
time find . -type f -exec cat '{}' ';' > /dev/null
```

This measures the performance of a series of small dependent reads during a large streaming read.

The Deadline I/O Scheduler

Preventing the starvation of requests in general, and read requests in particular, was a goal of the new 2.6 I/O schedulers.

The Deadline I/O Scheduler was introduced to solve the starvation issue surrounding the 2.4 I/O scheduler and traditional elevator algorithms in general. As discussed, the Linus Elevator maintains the sorted list of pending I/O requests in a single queue. The I/O request at the head of the queue is the next one to be serviced. The Deadline I/O Scheduler continues to keep this queue, but kicks things up a notch by introducing two additional queues—the read FIFO queue and the write FIFO queue. The Deadline I/O Scheduler keeps the items in each of these queues sorted by submission time (effectively, first in is first out). The read FIFO queue, as its name suggests, contains only read requests. The write FIFO queue, likewise, contains only write requests. Each FIFO queue is assigned an expiration value. The read FIFO queue has an expiration time of 500 milliseconds. The write FIFO queue has an expiration time of five seconds.

When a new I/O request is submitted, it is insertion-sorted into the standard queue and placed at the tail of its respective (either read or write) FIFO queue. Normally, the hard drive is sent I/O requests from the head of the standard sorted queue. This maximizes global throughput by minimizing seeks, as the normal queue is sorted by block number, as with the Linus Elevator.

When the item at the head of one of the FIFO queues, however, grows older than the expiration value associated with its queue, the I/O scheduler stops dispatching I/O requests from the standard queue. Instead, it services the I/O request at the head of the FIFO queue, plus a couple extra for good measure. The I/O scheduler needs to check and handle only the requests at the head of the FIFO queues, as those are the oldest requests in the queue.

Remember our old friend, the request to block 700? Despite the flood of write requests to the faraway blocks, after 500 milliseconds the Deadline I/O Scheduler would stop servicing those requests and service the read over at block 700. The disk would seek to block 700, service the read request and then continue servicing the other outstanding requests.

In this manner, the Deadline I/O Scheduler can enforce a soft deadline on I/O requests. Although it makes no promise that an I/O request is serviced before the expiration time, the I/O scheduler generally services requests near their expiration times. Thus, the Deadline I/O Scheduler continues to provide good global throughput without starving any one request for an unacceptably long time. Because read requests are given short expiration times, the writes-starving-reads problem is minimized.

Anticipatory I/O Scheduler

This is all well and good, but it's not a perfect solution. Consider what happens with our fictional request to block 700, which presumably is the first of many dependent reads to that area of the disk. After servicing the read request, the Deadline I/O Scheduler continues servicing the write requests to the earlier blocks. This is fine, until the reading application submits its next read request (say, to block 710). In 500 milliseconds, that request expires and the disk seeks over to block 710, services the request, seeks back to where it was before and continues servicing the streaming write. And then another read arrives.

The problem again stems from those darn dependent reads. Because reads are issued in dependent chunks, the application issues the next read only when the previous is returned. But by the time the application receives the read data, is scheduled to run and submits the next read, the I/O scheduler has moved on and begun servicing some other requests. This results in a wasted pair of seeks for each read: seek to the read, service it and seek back. If only there was some way for the I/O scheduler to know—nay, to anticipate—that another read would soon be submitted to the same part of the disk. Instead of seeking back and forth, it could wait in anticipation for the next read. Saving those awful seeks certainly is worth a few milliseconds of waiting; we save two seeks.

This is, of course, exactly what the Anticipatory I/O Scheduler does. It began as the Deadline I/O Scheduler; it implements the same deadline-based scheduling. But it was gifted with the addition of an anticipation mechanism. When a read request is submitted, the Anticipatory I/O Scheduler services it within its deadline, as usual. Unlike the Deadline I/O Scheduler, however, the Anticipatory I/O Scheduler then sits and waits, doing nothing, for up to six milliseconds. Chances are good that the application will issue another read to the same part of the filesystem during those six milliseconds. If so, that request is serviced immediately, and the Anticipatory I/O Scheduler waits some more. If six milliseconds go by without a read request, the Anticipatory I/O Scheduler guessed wrong and returns to whatever it was doing before.

If even a moderate number of requests are anticipated correctly, a great deal of time (two expensive seeks, each) is saved (Table 1). Because most reads are dependent, the anticipation pays off most of the time. To further improve the odds of a correct anticipation, the Anticipatory I/O Scheduler uses a heuristic to better guess for which processes to wait. To this end, the scheduler maintains I/O statistics about each process to keep track of its behavior. Because of these statistics and intelligent heuristics, the Anticipatory I/O Scheduler correctly anticipates the actions of applications a sufficiently large amount of the time to be well worth the overhead.

Table 1. The Results

I/O Scheduler and Kernel	Test 1	Test 2
Linus Elevator on 2.4	45 seconds	30 minutes, 28 seconds
Deadline I/O Scheduler on 2.6	40 seconds	3 minutes, 30 seconds
Anticipatory I/O Scheduler on 2.6	4.6 seconds	15 seconds

By minimizing unneeded seeks and more quickly servicing read requests, in many workloads the Anticipatory I/O Scheduler provides both improved request latency and global throughput over the Deadline I/O Scheduler and the Linus Elevator. Unsurprisingly, the Anticipatory I/O Scheduler is the default I/O scheduler in the 2.6 kernel. And rightly so, it rocks.

Acknowledgement

Andrea Arcangeli and Jens Axboe are the primary authors of the Linus Elevator. Jens Axboe is the primary author of the Deadline I/O Scheduler. Nick Piggin is the primary author of the Anticipatory I/O Scheduler.

Robert Love (rml@tech9.net) is a kernel hacker at MontaVista Software and a student at the University of Florida. He is the author of *Linux Kernel Development*. Robert loves O.A.R. and lives in Gainesville, Florida.

Archive Index Issue Table of Contents

Advanced search

Cooking with Linux

The Customer Is Always Served

Marcel Gagné

Issue #118, February 2004

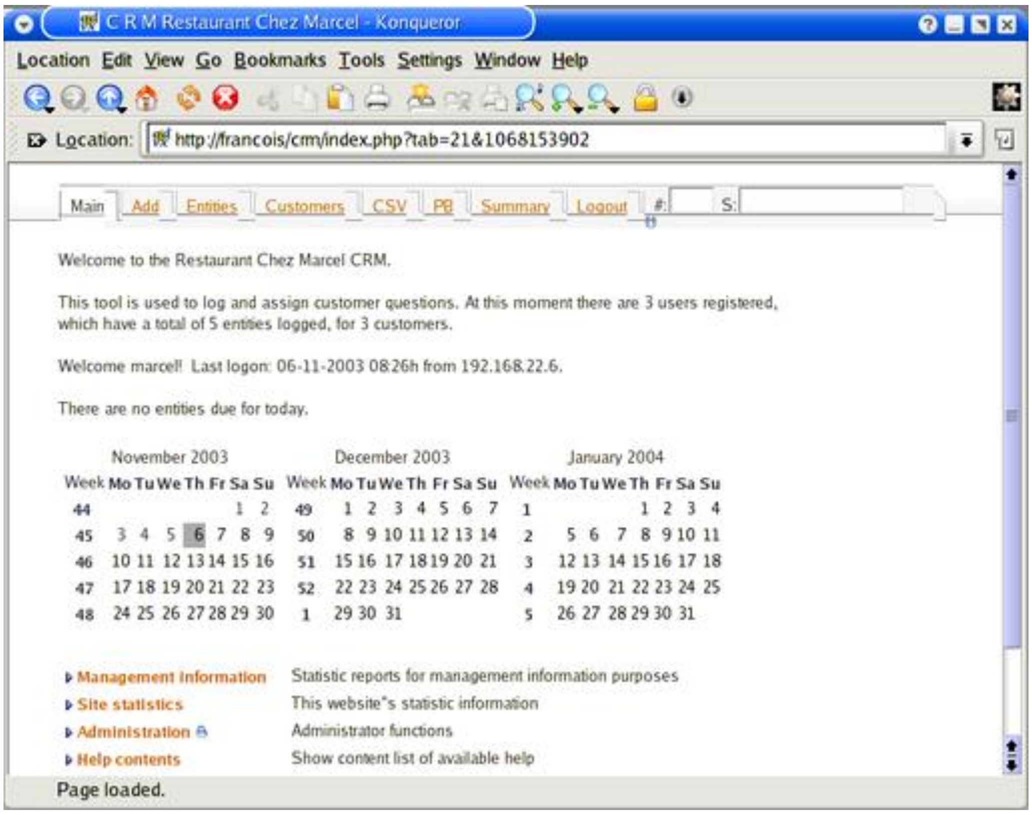
Sometimes it takes more than wine to keep customers happy. Keep track of your customers' needs, including pre-sales information, support and meetings.

Web-based intelligence, François, is the theme of this issue. As I planned the menu for today, I was caught by the sheer number of possibilities that term brought to mind. You too, *mon ami*? True, it could be Web-based artificial intelligence programs, but I was thinking of simpler things. Encyclopedias, business-to-business resources, dictionaries, new sites, search engines and even electronic mail. Yes, François, I agree with you. With so much spam these days, the intelligence in e-mail is becoming questionable. Nevertheless, e-mail still is an indispensable tool for modern communication. Consider that closely tied to e-mail, groupware solutions continue to grow, and you can begin to understand why I chose today's menu.

We must continue this later, François, for our guests have arrived. *Bonjour, mes amis*, and welcome to *Chez Marcel*, the Linux world's finest restaurant and finest wine cellar. Please sit and make yourselves comfortable. François will fetch the wine *immédiatement*. To the wine cellar, François, south wing. Something Spanish is in order—the Bierzo Corullín 2000 is a perfect choice for today's menu.

Right before you arrived, François and I were discussing Web-based intelligence. From a business perspective, I tend to think that the kind of intelligence that matters most is entirely customer-related. Without customers, there is no business. Consequently, a great deal of energy goes into systems that help in the management of customer relationships. This is where CRM (customer relationship management) software comes into play. This type of software gathers all sorts of customer information from the obvious contact details to sales, marketing and support. With that information, companies can assign tasks or reminders related to each customer's need. This could be calling back on a specific product, supporting a request, dealing with a complaint, setting up a meeting date or anything related to your relationship with a particular customer. CRM systems run the gamut from very simple to highly complex, corporate-wide, data-mining behemoths. Prices vary as well from no or little cost to hundreds of thousands of dollars. That figure alone should give you an idea of the importance businesses place on customer relationships.

Here in our Linux kitchen, with the help of open-source programmers, we can sample a number of CRM systems while spending nothing more than a little time.


C R M Restaurant Chez Marcel - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://francois/crm/index.php?tab=21&1068153902
Main Add Entities Customers CSV PB Summary Logout #: S:

Welcome to the Restaurant Chez Marcel CRM.

This tool is used to log and assign customer questions. At this moment there are 3 users registered, which have a total of 5 entities logged, for 3 customers.

Welcome marcel! Last logon: 06-11-2003 08:26h from 192.168.22.6.

There are no entities due for today.

November 2003

Week	Mo	Tu	We	Th	Fr	Sa	Su
44						1	2
45	3	4	5	6	7	8	9
46	10	11	12	13	14	15	16
47	17	18	19	20	21	22	23
48	24	25	26	27	28	29	30

December 2003

Week	Mo	Tu	We	Th	Fr	Sa	Su
49	1	2	3	4	5	6	7
50	8	9	10	11	12	13	14
51	15	16	17	18	19	20	21
52	22	23	24	25	26	27	28
1	29	30	31				

January 2004

Week	Mo	Tu	We	Th	Fr	Sa	Su
1				1	2	3	4
2	5	6	7	8	9	10	11
3	12	13	14	15	16	17	18
4	19	20	21	22	23	24	25
5	26	27	28	29	30	31	

Management information	Statistic reports for management information purposes
Site statistics	This website"s statistic information
Administration	Administrator functions
Help contents	Show content list of available help

Page loaded.

Figure 1. CRM-ctt's main screen offers admin and configuration options.

The first CRM package I would like to look at is called CRM customer tracking system (CRM-ctt). Its basic Web-based interface might make you look elsewhere because it appears so simple—I confess I almost did the same thing. Hidden underneath this simplicity, however, is a capable CRM system. With options for multiple users, multiple languages, extensive customizations, security, prioritization, e-mail notification and PDF reporting, CRM-ctt starts to look rather impressive (Figure 1). Pick up a copy of the program at crm-ctt.sourceforge.net.

To get the system up and running, you need a running Apache server, with PHP and MySQL support. If you want graphical support for CRM-ctt, you also need to have the php-gd package installed. The installation procedure takes a few steps, but it is not complicated. Extract the package into your Web server's document root. You also may want to change the name of the directory at this point:

```
cd /var/www/html
tar -xzvf crm-1.9.2-19102003.tar.gz
mv crm-1.9.2-19102003 crm
```

Before you continue, change the ownership of the CRM directory to that of your Web server's user and group. In my case, that meant running `chown -R apache.apache crm`.

Let's move on to the actual configuration of the software. Start by pointing your browser to the location where the application is installed (something like http://www.webserver.dom/crm). You then are led through a number of setup screens, the last of which involves the creation of your company repository. From here, follow the steps—enter your name, the name and e-mail address of the administrator and your chosen admin password. There are four Web-based steps in all.

The final step in the installation procedure is to write out the configuration file. The suggested method here is to change the permissions on the header.inc.php file temporarily:

```
cd /var/www/html/crm
chmod 777 header.in.php
```

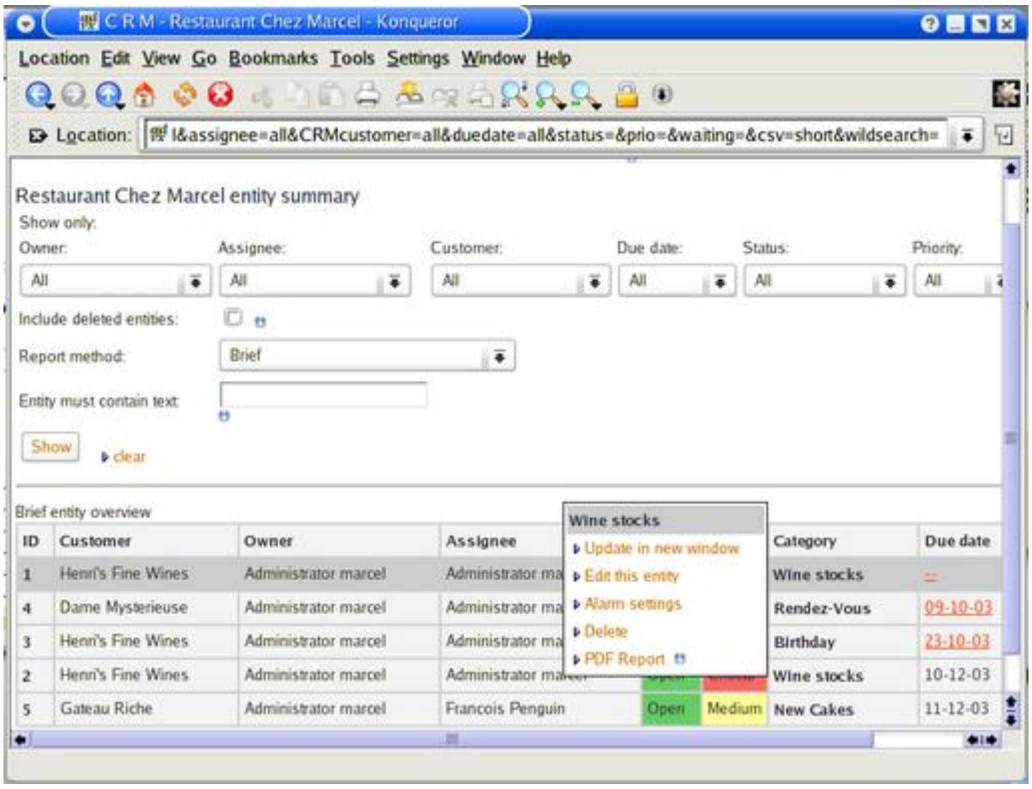
Return to your installation page and click Try Now. If all goes well, you should be presented with a successful completion message. You also can choose to generate the file and copy it manually to the directory where CRM is installed. Once you have completed this step, change the permission of the header file from 777 back to 755. Now, go to the bottom of the page. Click the highlighting where it says “When done, point your browser here”, and you are taken to the main login screen. Log in with your admin user name and password.

A system like this isn't particularly useful without customer information, so start adding. Click the Customers tab, then click Add a customer on the screen that follows. Repeat this for all of your customers. Once you have customer data entered, you can create Entities against those customers. An entity is anything you have to do for that customer, whether it is a pending order, an open trouble ticket, a friendly follow-up call or a note to send flowers on someone's birthday. CRM-ctt even can send you an e-mail to remind you of those things. As you explore CRM-ctt, you'll find great reporting features, including a one-click export to PDF option for quick, professional reports.

Because CRM can remind employees of people they need to contact and when, you also should add an entry to cron so it can send these e-mail notifications. What's interesting here is the server doing the mail-outs can be a machine other than the one running CRM-ctt. Although I show you only one entry (for 8:00 am), you could have the notify process run as often as you deem necessary for your organization. The yourCRONpwd in the following example is set in the Administration menu on CRM-ctt's main page, under Change global system's values.

```
# CRM Alarm date manager
0 8 * * * wget http://www.webserver.dom/crm/
↪duedate-notify-cron.php?password=yourCRONpwd
↪\&reposnr=XXX 1> /dev/null 2> /dev/null
```

As an administrator, you can add other accounts so you can delegate tasks to other members of your organization. From the administration screen, you also can generate management reports for your entire customer base, another task that can be exported to PDF reports. To help you along, CRM-ctt includes an on-line manual.


C R M - Restaurant Chez Marcel - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: l&assignee=all&CRMcustomer=all&duedate=all&status=&prio=&waiting=&csv=short&wildsearch=
Show
clear
Update in new window
Edit this entity
Alarm settings
Delete
PDF Report

Restaurant Chez Marcel entity summary

Show only:

Owner:	Assignee:	Customer:	Due date:	Status:	Priority:
All	All	All	All	All	All

Include deleted entities: ☐

Report method: Brief

Entity must contain text:

Brief entity overview

ID	Customer	Owner	Assignee			Category	Due date
1	Henri's Fine Wines	Administrator marcel	Administrator ma			Wine stocks	--
4	Dame Mysterieuse	Administrator marcel	Administrator ma			Rendez-Vous	09-10-03
3	Henri's Fine Wines	Administrator marcel	Administrator ma			Birthday	23-10-03
2	Henri's Fine Wines	Administrator marcel	Administrator marcel			Wine stocks	10-12-03
5	Gateau Riche	Administrator marcel	Francois Penguin	Open	Medium	New Cakes	11-12-03

Figure 2. CRM-ctt provides easy reporting and one-click PDF exporting.

Many CRM packages are available for Linux, and all of them track customer relationships as the core of what they do. If you look closely, though, you might think that CRM packages look a lot like some of the new Web-based groupware suites. And you might be right. Many of the ideas behind a good CRM system also exist in a good groupware system. What sets groupware apart is the scope of the Web-based intelligence it provides and the collaborative possibilities it opens. Groupware suites can include centralized electronic mail, calendars, address books, discussion forums, call tracking and any number of other applications. One Web-based groupware suite that I think is well worth checking into is eGroupWare.

eGroupWare is a GPL suite with an open-source API, so applications can be created and added to the system easily. Among the applications included are e-mail, calendaring, an infolog for tracking customer calls and setting up to-do lists, a trouble ticket system, forums, personal and corporate address books, a knowledge base, a wiki documentation system and more. There's also a site manager feature, so individual users can create and deploy their own Web sites. eGroupWare does all this while sporting a slick-looking business-ready face (Figure 3). At the time of this writing (early November 2003), eGroupWare was about two weeks away from its 1.0 release.

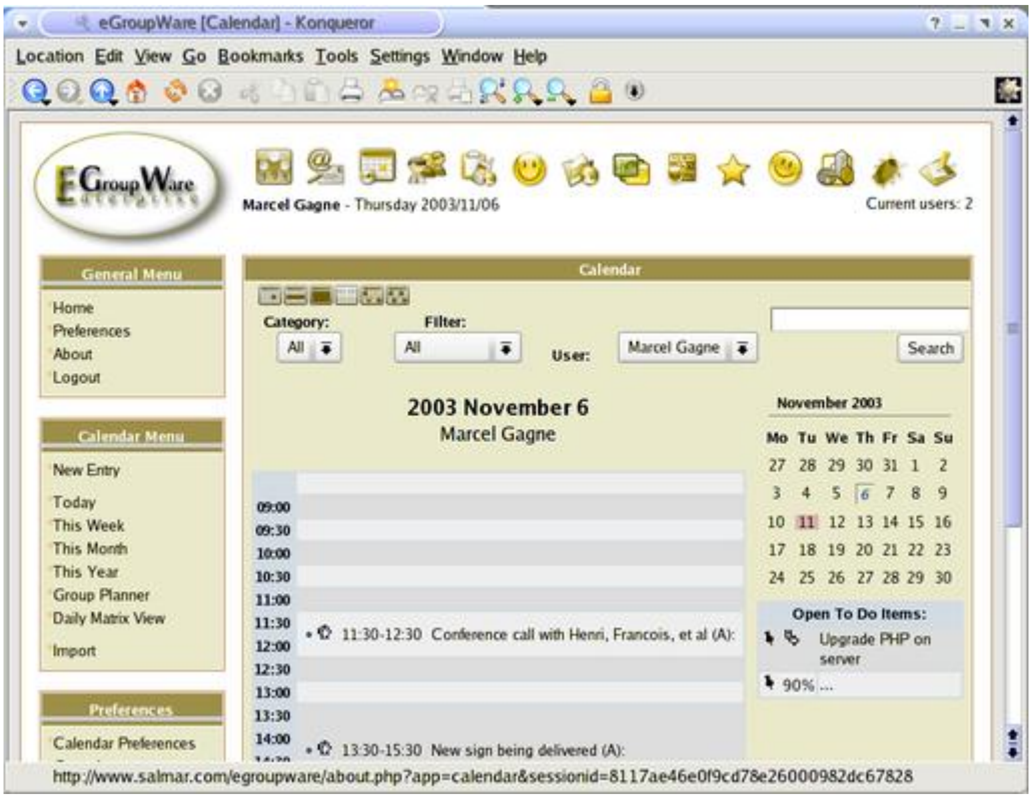


Figure 3. eGroupWare offers a calendar, trouble ticket system and management for user Web sites.

eGroupWare supports the two most popular open-source database formats, MySQL and PostgreSQL. In my test, I chose to run it with PostgreSQL, but both formats are easy to set up. To get in on the eGroupWare action, download the package from www.egroupware.org. Tarballs and binary packages both are available.

To install from the tarred and gzipped bundle, extract the package into your Web server's document root. For instance, a default Apache installation should have document root at /usr/local/apache/htdocs, whereas an RPM-based install (such as Red Hat) often has it at /var/www/html:

```
cd /var/www/html
tar -xzvf eGroupWare-relnum.tar.gz
```

This step creates a directory called egroupware. Change the permissions on this directory tree like this:

```
chown -R apache.apache egroupware
```

Assuming a PostgreSQL installation, your next step is to create a PostgreSQL user to access the database. Do this by switching to your postgres user:

```
$ su - postgres
$ createuser egroupware
Shall the new user be allowed to create
databases? (y/n) y
Shall the new user be allowed to create more new
users? (y/n) n
CREATE USER
```

When asked whether this user is allowed to create other databases, say yes. When asked whether this user can create other users, choose no. All that's left to do is create a database for eGroupWare. Still logged in as the postgres user, type the following:

```
$ createdb -U egroupware egroupware_db
CREATE DATABASE
```

If you don't like the idea of calling it egroupware, you could use basically any name you want for the database or modify the name slightly, as I did above. After this, you are done with the command-line work, so fire up your browser and finalize eGroupWare's settings. Open up Mozilla, Konqueror or any JavaScript-enabled browser and point your Web server to http://yourwebserver/egroupware/setup.

On this screen, enter the relevant information for your setup to create your header configuration file (header.inc.php). If you changed your DB user from egroupware to something else, make sure you identify it here. The same holds true for the database name. You also should assign a header password and an administration password. The header password lets you modify or recreate the configuration file you are building now.

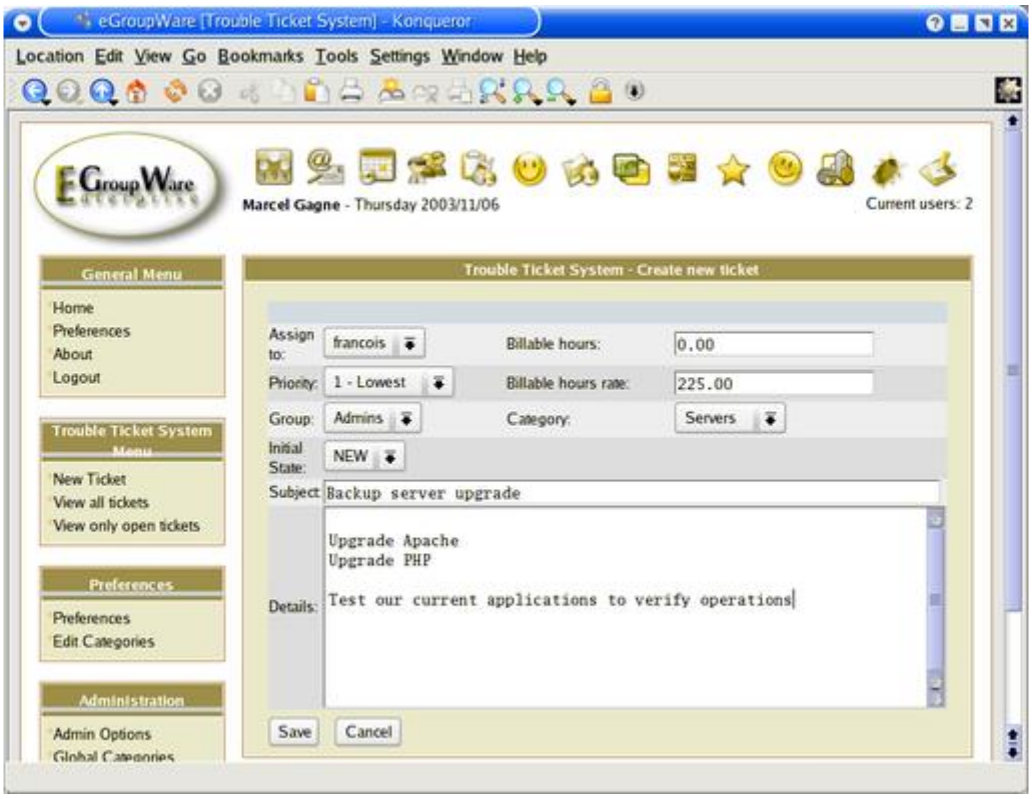
When you are finished, you are taken to the setup/header login screen. You already have created the header file, so chances are you do not wish to do it all over again, *non*? Your concern now is the actual eGroupWare setup. Before we move to this step, I have noticed that many of you have emptied your glasses. François, if you would kindly do the honor of refilling them—*merci*.

Once you have logged in using the admin password, the setup checks to see if your database has been created properly and if the appropriate user ID defined in the header creation step is used. If everything has gone well up to this point, you should be at Step 1 of the local configuration. Click Install to create the application tables and install the eGroupWare suite of applications. The system chugs along for a couple of minutes while it does this.

When everything has been completed, check the browser's screen to make sure no error messages have been reported, and click the Recheck my Installation button. If all has gone well, you can go on to Step 2 and create your admin account. The option also exists to create three demo accounts, but you do not have to do this. Step 3 lets you define the default language to be used, and Step 4 is for individual application management. From this dialogue, you can specify whether you want all the applications (this is the default) or only some of them. When you log out from here, your installation is complete.

Now it's time to start doing things with eGroupWare; begin by logging in with your admin account. This most likely means pointing your browser to http://your.server.dom/egroupware. Along the top of the screen, you should see a number of icons representing the various groupware services. To the left, menus appear based on the functions of the current application, although a smaller menu with Home, Preferences, About and Logout always is there. The look and feel can be modified to suit your personal tastes by clicking Preferences and making adjustments.

If you are the administrator, you can make changes for the entire organization. You even can force some defaults and prevent users from changing them, a useful feature for the corporate administrator. User accounts can be created with predefined applications delivered to their specific login based on groups. For instance, the support group may need access to the trouble ticket system (Figure 4). Using this group-based approach provides a consistent set of tools for your users. Create your groups first, decide what applications they need to access and create your users based on those groups.



Trouble Ticket System - Create new ticket

Assign to:	francois	Billable hours:	0.00
Priority:	1 - Lowest	Billable hours rate:	225.00
Group:	Admins	Category:	Servers
Initial State:	NEW		
Subject:	Backup server upgrade		

Details:

```
Upgrade Apache
Upgrade PHP

Test our current applications to verify operations
```

Figure 4. Entering a Trouble Ticket in eGroupWare

In terms of the future and ongoing support, eGroupWare has an active community of users and developers. Several mailing lists and an IRC channel are available, should you find yourself needing answers to your questions.

Mon Dieu! We are running out of time once again. Space and time do not permit me to cover anything else in detail today, but many excellent packages out there are worthy of your consideration. Although we all enjoy cooking with Linux, I humbly suggest that our customer/restaurateur relationship needs no software to manage it. Instead, I shall continue providing you comfortable chairs at your favorite table, and François shall keep your glasses filled. Sometimes, simpler is better. François, if you would do the honors, please. Until next time, *mes amis*, let us all drink to one another's health. *A votre santé Bon appétit!*

Resources

CRM-ctt: crm-ctt.sourceforge.net

eGroupWare: www.egroupware.org

Marcel's Wine Page: www.marcelgagne.com/wine.html

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of the newly published *Moving to Linux: Kiss the Blue Screen of Death*

Goodbye! (ISBN 0-321-15998-5) from Addison Wesley. His first book is the highly acclaimed *Linux System Administration: A User's Guide* (ISBN 0-201-71934-7). In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm.

Archive Index Issue Table of Contents

Advanced search

Paranoid Penguin

Seven Top Security Tools

Mick Bauer

Issue #118, February 2004

Simply installing more software won't make your systems more secure. But with these seven packages, you can learn to set up a security policy and test that the other software on your system complies with it.

Linux supports a wealth of outstanding free and open-source security tools—enough, obviously, to write a monthly column on the topic. But whereas I usually focus on one or two particular tools or techniques in-depth, this month I'd like to discuss, at a high level, a variety of my favorite security tools for Linux.

If you're new to Linux or to network security, this may be your first exposure to these particular software packages, and I hope this column nudges you in the direction to learn more. If you're familiar with a couple of them but not the others, I hope this article helps you to augment your toolkit. But even if all of this is old hat for you, I hope you find it amusing to see which of the t00lz on my laptop have been getting the most CPU time lately. So without further ado, I bring you the Paranoid Penguin's Choice.

Netfilter/iptables

We begin with the most ubiquitous of our featured tools, Netfilter, the Linux kernel's built-in firewall code. To be precise, the collection of modules in question officially is called Netfilter—iptables is merely the user-space command we use to configure the Netfilter kernel modules. The two names can be used interchangeably most of the time except, of course, when you're issuing iptables commands or talking to kernel developers.

Netfilter was the winner in the Best Security Tool category of our 2003 Editors' Choice Awards. As I explained then, Netfilter is responsible for moving Linux firewalls out of the primordial soup of dumb, stateless packet filtering and into the modern era of stateful packet filters. What this means for non-security geeks is Netfilter allows Linux firewalls to inspect network packets statefully in relation to one another, that is, by associating them with established connections, identifying them as beginning new transactions and so on. In contrast, in pre-2.4 kernels Linux treated each packet as a standalone entity, filtering it based strictly on where it came from and where it was headed. For example, all the packets in an HTTP transaction were filtered separately rather than being treated as a group, but no more.

This new packet power and intelligence has ramifications that extend beyond Linux's usefulness as a network firewall. Netfilter is as useful for local security on servers and even on workstations as it is on proper network firewalls—I explain precisely how and include code examples in my article "Using iptables for Local Security", *LJ*, August 2002, and also in Chapter 3 of my book *Building Secure Servers With Linux*.

The command iptables is, for many people, simple to use after spending some time with the iptables(8) man page. Besides my own material on that topic, I also recommend Robert Ziegler's book *Linux Firewalls, 2nd Ed.* (New Riders, 2002). iptables is eminently scriptable, and the aforementioned sources and the Internet abound with example scripts you can adapt for your own use.

But what if you prefer to insulate yourself from the inner workings of packet filtering and instead want a GUI front end that speaks plain English to you? You're in luck: many quality third-party front ends for Netfilter exist. One of the best is Firewall Builder (www.fwbuilder.org), which allows you to create firewall rules with reusable objects and with wizards. I covered Firewall Builder in-depth in my two-part series "Using Firewall Builder" (*LJ*, May and June 2003).

Another popular iptables helper is Mason, which automatically builds iptables scripts by passively observing normal system use. This is useful especially for personal firewall setups on workstations. Mason is available at users.dhp.com/~whisper/mason. Yet another increasingly popular tool is Shorewall, which generates iptables scripts based on how you configure a few simple text files in the directory /etc/shorewall. Shorewall's home page is shorewall.net.

Finally, I'd be remiss if I didn't mention that many Linux distributions have their own (distribution-specific) packages for using iptables. SuSE 8.2, for example, has SuSEfirewall2, which automatically generates and runs iptables commands based on simple parameters you set in the file /etc/sysconfig/SuSEfirewall2. If your preferred distribution has such a tool, it's worth checking out—it already may be installed on your system.

By the way, in case you're wondering what I myself prefer, I usually write my own iptables scripts by hand. For me that's the simplest and most direct way; then again I'm a professional firewall engineer—your needs and skills may vary.

Bastille

Bastille, the brainchild of Jay Beale and Jon Lasser, is in a class by itself. It's a script that performs a comprehensive lockdown of your Linux system, based entirely on questions it asks you. What really sets it apart from other hardening scripts is all the questions it asks are annotated copiously. Of all the security tools I've seen, none does more to educate its users than Bastille. For this reason, I especially recommend Bastille to newbies.

When I wrote a *Linux Journal* article on Bastille a couple of years ago ("Battening Down the Hatches with Bastille" *LJ*, April 2001), I asked Jay Beale a few questions over e-mail that, after meeting face-to-face soon after, led to an enduring friendship. Bastille benefits greatly from Jay's outgoing personality, and he uses direct and even entertaining language to enable you to help Bastille tweak your system into a more secure state.

Bastille is supported officially on Red Hat, Mandrake and Debian GNU/Linux. It's even been ported to HP-UX and Mac OS X. You can get Bastille at www.bastille-linux.org.

Nmap

Netfilter and Bastille are strictly defensive tools, but what if you want to test your Linux box's current state of security? One way is to run a port scanner and enumerate the listening ports on it, for the purpose of deducing which network applications are running.

In a site-wide security audit, automated port scanners are invaluable in determining how carefully and consistently hosts have been secured. If you run a port scanner against hosts protected by a firewall, it can validate the firewall's configuration. And at the most tactical level, a good port scanner tells you the precise points of entry attackers can see on each host it runs against.

Nmap (Listing 1) is the undisputed king of port scanners: it's fast, low-profile, free and feature-rich. Nmap offers a variety of scanning methodologies, from the fast but noisy TCP Connect method to arcane but stealthful approaches, such as Xmas Tree scanning. Nmap even comes with a GUI, NmapFE, though it's quite easy to use from the command prompt as well. You can get the latest version of Nmap from www.insecure.org, but your Linux distribution of choice probably has its own reasonably current package. You most likely needn't look any further than your Linux CDs to get Nmap.

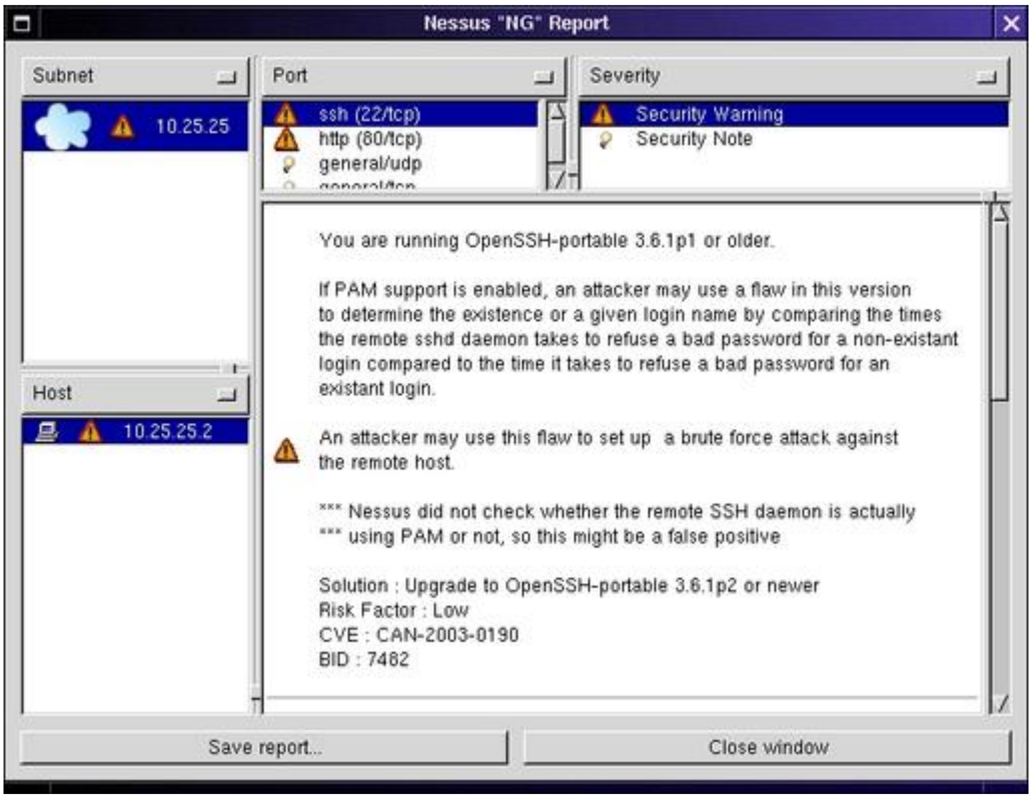
Listing 1. Nmap reveals which network services are available on a host.

```
tamarin:/usr/src # nmap -sS -F -P0 -O 10.1.2.123

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
caught SIGINT signal, cleaning up
tamarin:/usr/src # nmap -sS -F -P0 10.1.2.123

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on wuxia.wiremonkeys.org
(10.1.2.123):
(The 1134 ports scanned but not shown below are in
state: filtered)
Port       State       Service
21/tcp     closed      ftp
22/tcp     open        ssh
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
119/tcp    closed      nntp
389/tcp    open        ldap
418/tcp    closed      hyper-g
443/tcp    open        https
636/tcp    open        ldapssl
873/tcp    closed      rsync
993/tcp    open        imaps
3389/tcp   closed      ms-term-serv
6666/tcp   closed      irc-serv
8080/tcp   closed      http-proxy
11371/tcp  closed      pksd
```

Nessus

Whereas port scanners simply enumerate listening ports, security scanners attempt to connect to open ports and find out as much as possible about the applications doing the listening. At its simplest, this can amount to banner grabbing, which is logging the text message the application prints upon successful connection. Many applications identify themselves by name and some even by version.

But professional-grade security scanners go much further than banner grabbing. Once they identify which application is running on a given port, they try to determine whether various known vulnerabilities can be exploited against that application, sometimes by actually beginning but not following through with penetration methods. Nessus (Figure 1) is a professional-grade security scanner, but it's a free and 100% customizable one.



```
You are running OpenSSH-portable 3.6.1p1 or older.

If PAM support is enabled, an attacker may use a flaw in this version
to determine the existence or a given login name by comparing the times
the remote sshd daemon takes to refuse a bad password for a non-existant
login compared to the time it takes to refuse a bad password for an
existant login.

An attacker may use this flaw to set up  a brute force attack against
the remote host.

*** Nessus did not check whether the remote SSH daemon is actually
*** using PAM or not, so this might be a false positive

Solution : Upgrade to OpenSSH-portable 3.6.1p2 or newer
Risk Factor : Low
CVE : CAN-2003-0190
BID : 7482
```

Figure 1. This sample Nessus report reveals a vulnerable SSH dæmon.

As with Nmap, the value of Nessus to professional security engineers is immeasurable; I use both in my work all the time. But even civilians can benefit from, for example, testing their hobby Web servers with Nessus. As with Bastille, Nessus includes user education in its design goals. If you read a report carefully, you can learn a thing or two not only about the vulnerabilities it identifies but what to do to fix them.

Scanning, Probing and Fuzzing: Caution

I enjoy using and writing about port scanners, security scanners and other offense-oriented security tools. In the hands of a careful and responsible user, they serve an important role in validating system and network security.

They also carry significant potential for abuse, however, so much so that if someone unexpectedly discovers you using such tools against their system, they probably won't assume you're trying to help them. Never scan any host you haven't been authorized explicitly to scan.

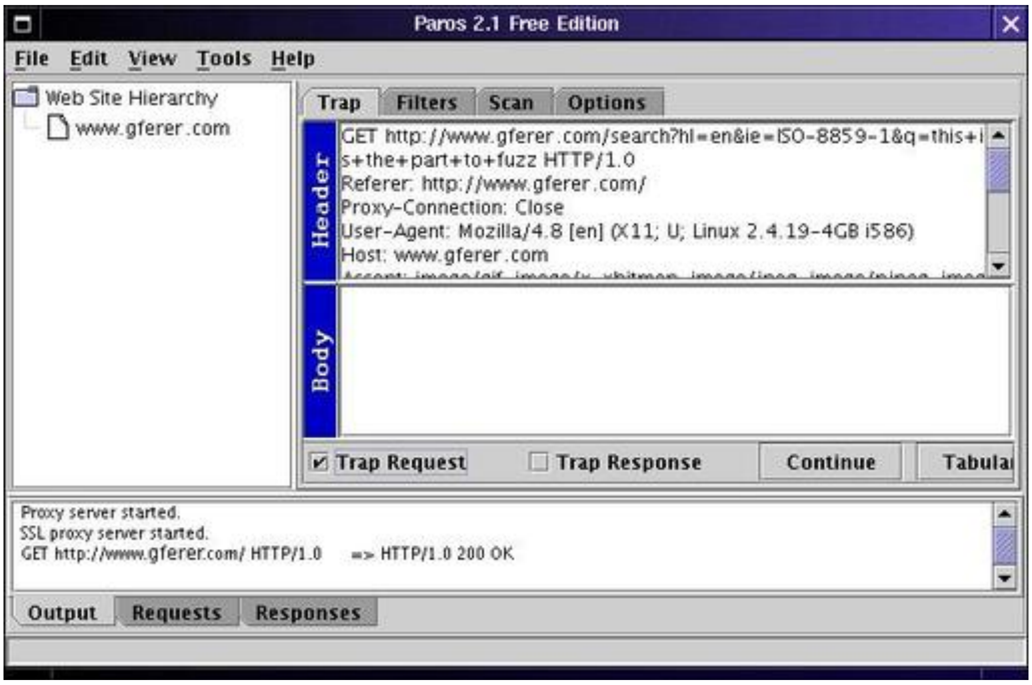
Also, never install a security scanner on a bastion host (a hardened, publicly accessible server). Such hosts are at higher-than-average risk of being compromised by outsiders, so they're the last place to keep security-probing tools. If you need to do your scans on the same LANs as your target systems, get a laptop computer. A used laptop capable of running Nmap, Nessus and other tools shouldn't cost you more than $350 US, and I do much of my own scanning and penetration-testing with such a system.

Fuzzing with Paros

Before we leave the realm of security validation checking, let's consider Web application security. Web applications constitute the single largest area of growth both in Internet-accessible services and in externally exploitable system vulnerabilities. So how do we test the security of our Web applications?

You might think that Nessus is a good start, and it is, but mainly for generic Web dæmon security. Most of what Nessus tells us about Web services applies to the server dæmon itself, such as Apache, not to the actual Web content it serves up. It doesn't tell us whether our custom Web applications do proper input validation, whether they're vulnerable to cross-site scripting vulnerabilities, whether they're vulnerable to fuzzing attacks (in which expected parameters are altered or fuzzed) and so forth. That's where tools like Paros come in.

Paros (Figure 2) is a free tool released under the Clarified Artistic License, and it's written in Java. You need the Java Runtime Environment installed in order to use Paros. You can download both Paros' executable JAR file and its complete source code from www.proofsecure.com.



```
GET http://www.gferer.com/search?hl=en&ie=ISO-8859-1&q=this+i
s+the+part+to+fuzz HTTP/1.0
Referer: http://www.gferer.com/
Proxy-Connection: Close
User-Agent: Mozilla/4.8 [en] (X11; U; Linux 2.4.19-4GB i586)
Host: www.gferer.com
```

```
Proxy server started.
SSL proxy server started.
GET http://www.gferer.com/ HTTP/1.0     => HTTP/1.0 200 OK
```

Figure 2. Paros is a free tool for testing Web applications.

Paros works on a principle common to the new generation of Web security tools. You run it as a local proxy on your scanning workstation, and all the interaction between your local browser and the target Web server is brokered by the security tool. In this way, you can capture an outbound query, alter or fuzz it and then send it along to the server. For example, suppose your Web application uses a form with pull-down menus, and you want to make sure that the application is validating input properly. With Paros, you could replace pull-down menu options arbitrarily with random strings—blozzle instead of monday—and see that query's effect on the application.

Paros also supports several scanning-type features, such as directory traversal. Although the JRE can be taxing on older systems, overall Paros is a flexible and user-friendly tool. Furthermore, being Java-based, it's cross-platform. I've also used it on several different flavors of Windows.

Paros isn't the only free fuzzing proxy. I also should mention Dave Aitel's SPIKE Proxy. It too acts as a local proxy but has the ability to run automated fuzzing attacks based on things it learns about the target site by watching you interact with it. SPIKE has the added advantage of being written in Python, which means much less CPU and memory overhead than Paros requires.

F.I.R.E.

I'm closing my little survey with a forensics tool: William Salusky's F.I.R.E., the Forensics and Incident Response Environment. It's unpleasant to contemplate, but no matter how careful and proactive you are, you may nonetheless someday experience a system compromise. If you do and you want to understand how and why, F.I.R.E. can help.

F.I.R.E. is a single CD-ROM Linux distribution geared toward analyzing compromised systems and recovering data from them. You can use it either by rebooting the compromised system with the F.I.R.E. CD-ROM or by mounting the CD-ROM in a running but feared-compromised Linux system and running tools directly off the CD. The latter technique is useful particularly when you don't trust the system's binaries, as when you fear they've been replaced by rootkit or trojaned versions, but can't take the system off-line just yet.

Besides analysis, F.I.R.E. makes it easy to copy data from the compromised system to other hosts on your network. F.I.R.E. also includes the X Windows System and a variety of both command-line and X-based security tools (including Nmap and Nessus). You can use F.I.R.E. to transform an ordinary Windows laptop into an awesome penetration-testing juggernaut. And at no extra charge, F.I.R.E.'s major functions can be accessed from a menu system comprehendable even by those of us who aren't full-time computer forensics specialists. You can learn all about and obtain F.I.R.E. at fire.dmzs.com.

Conclusions

This article is by no means a comprehensive list of the many, many excellent security tools available for Linux. I would have loved to devote blurbs to Tripwire, AIDE, Nikto, GnuPG, FreeS/WAN, Snort, PSAD, Stunnel, OpenSSL and a score of other worthy tools. But I've had fun showing off some of my favorite tools, and I hope you've found it useful. Remember, many of these tools bestow awesome powers upon their bearer: use them responsibly, ethically and carefully. But so long as you do, have fun.

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant in Minneapolis, Minnesota. He's the author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

EOF

Linux vs. SCO—A Foregone Conclusion

Jim Ready

Issue #118, February 2004

Failure to embrace disruptive technology leads to bad behavior and worse business.

When a disruptive technology appears in an existing marketplace, established players initially take pains, ostrich-like, to ignore it. Disruptive technologies usually start out as underpowered or as answers in search of a question, making it easy to belittle and discount the interloper. However, once a disruptive technology gains traction and starts making in-roads to established territory, the powers-that-be wake to the fact with one of two responses—they either embrace the disrupter or attempt to crush it.

Linux is a disruptive technology par excellence. Linux began its life as a hobbyist novelty and a graduate school project. Even after it made its way into the world over the Internet, it lagged behind UNIX systems in terms of capabilities, documentation, support and the ability to leverage the hardware platforms on which it ran. What made (and still makes) Linux disruptive is that it works. The open-source development model, incremental but steady as the proverbial tortoise, relentlessly advanced the GNU/Linux operating system and tools to reach and then surpass the startled proprietary hare. The cooperative spirit of the Open Source community and the assurances provided by licenses such as the GPL, LGPL, BSD copyright, NPL/MPL and others help prevent the fragmentation endemic in proprietary UNIX and provide the legal basis for the open-source process.

The divide between the powers that embrace Linux and those that would crush it is wide and deep. Long-standing leaders, such as IBM, HP and Oracle, had the vision to build new businesses with and on the Linux platform, despite or because of their investment in proprietary software, including proprietary UNIX. Foremost in the opposite camp is of course Microsoft, longtime foe of

UNIX and natural-born enemy of Linux, for both its open pedigree and the open-source model.

Poor misguided SCO meanders somewhere in the chasm between the two extremes. The SCO Group, reborn from Linux distributor Caldera and reminted with the SCO name, now feels rejected as its base of UNIX licensees migrate to Linux. With its claims against IBM and attempt to extort monies from Linux users, SCO is trying to turn Caldera's earlier enthusiastic embrace into a crushing clinch.

Outpaced technology companies have a long and sorry tradition of seeking in court what they cannot achieve in the open marketplace. Legal remedy in lieu of a successful business strategy always invokes the doctrine of unintended consequences. SCO's attempts to recover lost business from IBM AIX royalties will lead IBM and other licensees only to curtail their already declining proprietary UNIX shipments. Attempting to extract license fees from enterprise Linux users and embedded Linux deployers may delay Linux deployment in the short term, thereby stalling SCO's attempt to generate license revenues.

Self-deception is a typical response to disruptive technology. SCO's stated Linux Licensing Fees of $699 US for enterprise use and $32 US for embedded deployment represent a fantasy worldview that is out of line with the pricing and practices of both markets. On the enterprise side, this tidy sum is twice what Microsoft demands for its Windows products in comparable applications. For embedded systems, the SCO invoice outstrips typical low-volume royalties by a factor of at least three and is several orders of magnitude more for high-volume shipments in consumer electronics applications. For enterprise, the SCO fee quickly would exhaust available IT budgets; for embedded, it would overwhelm already slim margins on devices with bills of materials in the key $50–$200 US range.

None of these legal machinations would enhance the position of SCO UNIX one bit, nor would they garner Darl McBride any new Linux-based revenues. In SCO's imagined universe, where it prevails in the IBM suit or in its Linux licensing campaign, SCO still loses—does SCO imagine that it could capture, wound or even kill Linux as a commercial platform? It ain't gonna happen. A stuffed penguin in McBride's den would not lay any more golden eggs. Angry enterprise and embedded Linux users would not turn around and license SCO UNIX—it offers no technological advantage over Linux, even in high-end systems, and is completely inappropriate for embedded applications. Serious developers and users would wish a plague upon both SCO UNIX and SCO-licensed Linux. Diehards would stick with Linux and help to create a version scrubbed of any SCO detritus. Lesser souls might migrate to BSD or even end up in the clutches of Microsoft. None of these scenarios helps SCO, whose licensing-based revenue fantasy then reveals itself as a nightmare, first and foremost for SCO itself.

McBride and company need to realize that in the long term, SCO's claims would ensure that all SCO code, whether (temporarily) in Linux or in SCO UNIX, becomes untouchable and unmarketable, in any form.

So, McBride, Boies, et al. —go ahead, continue gunning for Linux and open source. Ready. Aim. Shoot yourself in the foot. Then, please hobble out of the way. Some of us are trying to do business.

James Ready is president and CEO of MontaVista Software, Inc., and has more than 25 years of technical and entrepreneurial experience. Cofounder of Ready Systems, he pioneered the development of the first commercially viable, real-time operating system (RTOS) product—the VRTX real-time kernel.

Archive Index Issue Table of Contents

Advanced search

AstroFlowGuard Appliance

José Nazario

Issue #118, February 2004

Along with a nice reporting system, this package delivers an integrated and easy-to-manage interface with a good feature set.

Product Information.

- Manufacturer: NetSoft and Offmyserver
- URL: www.netsoft.co.za and www.offmyserver.com
- Price: $6,495 US

The Good.

- Integrated appliance.
- Full-featured product.
- Cost.

The Bad.

- IDS is not well developed.
- Work flow could use improvement.
- Browser compatibility issues in Web UI.

The AstroFlowGuard appliance is a combined bandwidth management system, a VPN gateway, an IDS, a firewall and a NAT device. Along with a nice reporting system, this package delivers an integrated and easy-to-manage interface with a good feature set. Being an appliance, as opposed to a software distribution, it can be less error-prone—for a cost.

These boxes have been shipping for several months now, and the company has several customers both large and small. This means the company has been improving its product and proving itself in trials and deployments. Offmyserver and NetSoft teamed up to bring this appliance to market, with NetSoft doing the software and Offmyserver bundling it with the hardware. Offmyserver isn't that new, either, as it is an employee buy-out of iXsystems, formerly BSDi. Because of this, there's experience and market understanding behind this product, and it shows.

Setup Out of the Box

The AstroFlowGuard system ships as an appliance, so you get a box, a few cables, a manual and the system. The hardware is based on a Pentium 4 processor and should fit nicely into a 19" rack. Be warned, though; it's got a noisy fan, comparable to a medium- or large-sized router or enterprise switch, so this isn't for an open equipment room.

Initially, you have two big options to configure the system. The first is to use the LCD front panel to configure basic services. Here you can configure the basic IP networking parameters (address, netmask and gateway) along with the enabling or disabling of services. You navigate with a small number of easy-to-use buttons, almost like a network printer. Alternatively, you can hook up a PS/2 keyboard and a VGA monitor and use a curses-based configuration menu. You get the same basic menu items with this option that you do with the LCD screen. There isn't a command-line option, but most of the reporting is done better in the GUI. I was surprised a serial console interface wasn't included.

Once you have the basics set up, you can begin the final setup stages using your Web browser. This process isn't as easy as it sounds. I couldn't get the system to respond to HTTPS until the firewall was disabled, but after that I didn't have much difficulty. The login and product navigation is straightforward, so you don't need to consult the manual much except for a few tasks.

Hardware-wise, the box for the AstroFlowGuard should be enough to manage anyone's network. The system comes with four to six 10/100bT interfaces, which should work for most networks. Gigabit Ethernet is not an option at this time. AstroFlowGuard also lets you break out a DMZ network and a management network, all on one device.

A likely scenario for deployment would be to rack the box and configure the management address for the system. Once that's done, you would log in to the UI and configure the networks for the system to route. There, you can begin setting up your network management and enforcing that policy through the VPN (for secure Internet endpoints), the firewall and the bandwidth monitor.

The traffic shaping module is one of the more novel features in this class of device. With it, you can set up per-host and per-service bandwidth caps, which can help make the best use of a small network pipe. For example, you can configure a 50% maximum for Web traffic with an optional 10%, if needed, for short bursts. If you find peer-to-peer communications are hogging bandwidth, you can shape that down as well. Finally, if downloads from the outside world are consuming bandwidth from a server you run, you can back that off too. The UI makes all of this management relatively easy, and the reporting interface helps you make those decisions quickly.

Strengths

Under the hood is a Linux system, modified to boot without much issue or interaction, and various applications for network monitoring. These components include iptraf, rrdtool and Apache. This list probably gives the impression that you could build something like this for your own network, given an engineer or two for a few weeks. You probably could, but maintenance would be a consideration in this scenario.

Maintenance, then, is probably the biggest selling point for this product—AstroFlowGuard fairs very well in the build vs. buy comparison. Although it's based on open and available components, it would take some effort to build a system like this and work out the kinks, keeping it usable for a staff of administrators. Because of this, what at first appears to be free quickly consumes a lot of money and time.

AstroFlowGuard goes well beyond this point, however. By being an appliance through and through, it's a simple matter of loading the box in a rack and maintaining it from there. Even upgrades are painless. You simply select the upgrade option from the menu, it tells you what changed and you go to it—painless, and the upgrade to 1.002 happened without a hitch.

The price of AstroFlowGuard, under $6,500 US, puts it well below its competition. For a bandwidth appliance, you could use a Packeteer or similar product; there are various (and expensive) traffic monitors. VPN appliances also can be quite expensive. Firewalls have been known to be expensive at times, too, and finally, an IDS appliance typically costs this much without the other features. Although the price may seem a bit steep, for that amount of money you'd have difficulty finding an appliance that does one or two of these tasks.

One of those features typically found only in expensive commercial firewalls is the support for failover. Parallel AstroFlowGuard devices can communicate and detect when the other one has failed and begin routing around it. This is a very useful feature for networks that require high availability.

Overall, the feature list of the AstroFlowGuard makes sense as a network edge device. Most people deploy their IDS functionality here, and the other modules (bandwidth shaping and monitoring, VPN tunneling and firewalling) all make sense in a policy management device. This single box can meet the needs of various small- and medium-sized business networks in a single relatively easy-to-use package.

As of version 1.002, the on-line help for the product is solid and easy to navigate. It's task-based, as opposed to feature-based, so it's easy to use when you're actively trying to set up a new management rule.

Drawbacks, Big and Small

There are, of course, a handful of drawbacks. The biggest one at this time is the fact that this is a new product, still forming and working out some creases. Although the major components are done, it has room to grow. Given this package's price, I recommend you examine it closely in relation to your network's needs before you dive in to a purchase, but you probably will like this product.

The biggest drawback to the AstroFlowGuard's newness is the work flow within the application. The reporting interface is done well, and it allows you to drill down to various levels of detail. But, the configuration interface for adding bandwidth and firewall rules, for example, is in need of some maturity. The biggest complaint I had was figuring out the order in which various options should be configured—it's by adding classes and then specific rules.

A second complaint some may have is the Web UI uses several Microsoft Internet Explorer HTML and JavaScript extensions. This isn't a strict requirement, however, and my contact at NetSoft tells me they're working on changing that; expect this work to be done by the time you read this review. With a quick read of the source code to the pages, you can find the right entry points and use Mozilla on most pages without much difficulty.

One feature I found lacking is the IDS functionality. It seems to be a minimized feature in version 1.002; one that probably will receive an overhaul in the future. The configuration interface in this version was rather thin and didn't give much detail to the signatures within the IDS database, nor was there any way to configure new rules. When I enabled it on my home network, I received various alerts for traffic that didn't make much sense, but I didn't find the reporting interface for the IDS module very helpful either. I'd probably skip the IDS functionality at this point and hope it improves in future revisions.

What's Coming Next

Matt Olander, from Offmyserver, the company that distributes the AstroFlowGuard system, tells me that many of these issues will be addressed in the next revision of the software. The browser dependency will be removed. Secondly, the IDS functionality will be improved, allowing you to edit and escalate classes and events more significantly. And finally, the host management internals will be more automated, using automatic host detection on your local network. Combined, these new features significantly improve an already good product.

Conclusion

The AstroFlowGuard device certainly is a product worth looking at to bring a small network up to speed. Because it's an appliance, hardware and software configurations are kept at a minimum, meaning the staff can focus on other aspects and not have to worry about compatibility or installation issues. Currently at a 1.0 revision, some kinks need to be worked out, and not all of the features are mature at the time of this writing. Despite this, AstroFlowGuard compares favorably to other commercial offerings and beats them in terms of price.

José Nazario, PhD, works as a software engineer and security researcher for an unnamed Internet security company. He also develops on several open-source projects, has contributed to various Linux publications and likes to travel and give presentations.

Archive Index Issue Table of Contents

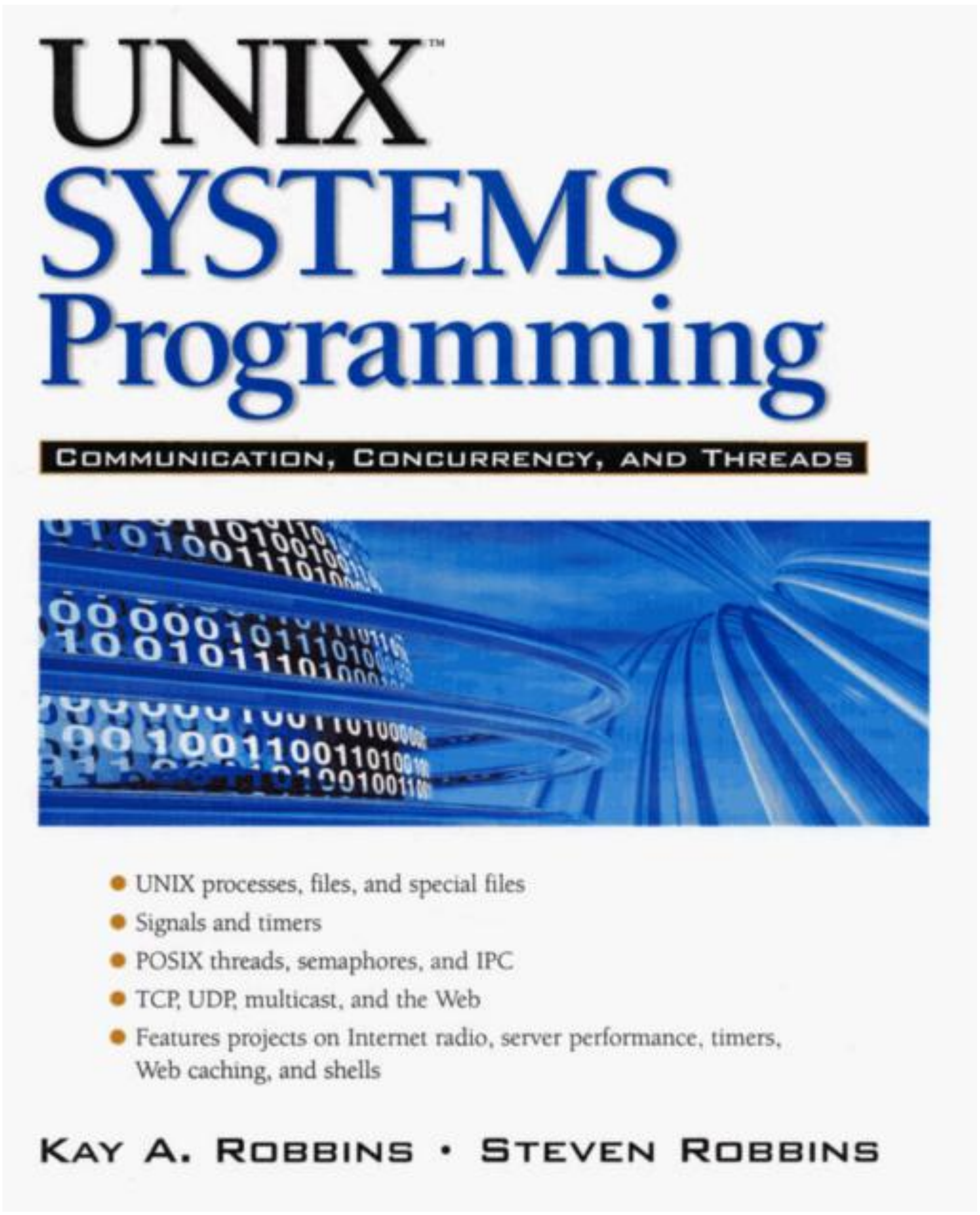
Advanced search

UNIX Systems Programming: Communication, Concurrency, and Threads by Kay A. Robbins and Steven Robbins

Ibrahim Haddad

Issue #118, February 2004

This updated second edition includes all-new chapters covering the Web and multicast, plus a completely revised and updated remote procedure call (RPC) chapter.


UNIX™
SYSTEMS
Programming
COMMUNICATION, CONCURRENCY, AND THREADS
UNIX processes, files, and special files
Signals and timers
POSIX threads, semaphores, and IPC
TCP, UDP, multicast, and the Web
Features projects on Internet radio, server performance, timers, Web caching, and shells
KAY A. ROBBINS • STEVEN ROBBINS

Prentice Hall, 2003

ISBN: 0-13-042411-0

$69.99 US

UNIX Systems Programming: Communication, Concurrency, and Threads is the successor to the 1995 *Practical UNIX Programming: A Guide to Communication, Concurrency, and Multithreading*. This updated second edition includes all-new chapters covering the Web and multicast, plus a completely revised and updated remote procedure call (RPC) chapter. Material on files, signals, semaphores, threads and client-server communication also has been updated and enhanced.

The book provides programming exercises for many fundamental concepts in process management, concurrency and communication. These programming exercises are similar to the exercises you would be doing as part of an operating systems course. Exercises are specified for systematic development, and many can be implemented in under 100 lines of code.

Another important feature of the book is compliance with the POSIX standards, the single UNIX specification adopted since the publication of the first edition.

The book provides everything you need to program with threads, TCP/IP and RPC. The authors explain the essentials of UNIX programming, concentrating on communication, concurrency and multithreading techniques, and why, when and how to use them in a tutorial manner. They provide a lot of reusable source code examples, all complete and ready to be compiled and run.

Another nice feature of the book is that it shows how to design complex software to get the best performance from a POSIX system. Many short examples are featured throughout the book, as are a number of hands-on projects that help readers expand their skill levels. The authors take a practical approach and use short code snippets to illustrate how to use system calls.

I highly recommend adding this book to your UNIX library if you want to learn UNIX system programming essentials with a concentration on communication, concurrency and multithreading techniques. It is *the book* that will keep you wondering how you were working without it.

Archive Index Issue Table of Contents

Advanced search

Letters

Readers sound off.

Laptop Vendor Catches the Cluetrain

As a Linux user and supporter, while I truly enjoy seeing evidence of businesses that get it, I also feel I have to share my experiences with a vendor that for a time seemed to have forgotten that one of the most important values we share, as buyers and sellers, users and hackers, is community, and that whenever we stray from open communication and straight dealing, we do so at our own peril.

Four months ago, I set out to purchase a new laptop to replace an aging VAIO that I had loaded with Linux manually, and in addition to the creeping obsolescence of the hardware, I always have been frustrated by certain incompatibilities, especially the IEEE 1394 chipset. I was looking forward to getting a beefier machine, from a Linux friendly vendor, so I finally could do some amateur video work with my Mini-DV camcorder. I decided to purchase my system from QLI, based on several recommendations I found in the on-line archives of this very magazine. I was excited at the prospect of receiving my new, more powerful, super-slim, pre-loaded with my fave distro, with 100% compatible hardware laptop. I initially was encouraged that my credit card was charged right away, assuming this meant the unit would ship soon. This quickly turned to confusion, frustration and eventually anger.

For the next two months, every time I tried to get information about my order, I was given a new variation of some story about how bad the supplier was. They don't speak English; they never send e-mail; their ETAs are always vague. I was understanding at first, but less so with each passing week, especially when an important trip came and went, and I still had to use my existing, ailing portable. At the two-month mark, I was facing the prospect of another trip, with more intermittent failures due to x-ray machines and increasingly poor battery life. Do you know how hard it is to find a usable outlet in some airports?

I canceled the order and was told the refund might take a while; the sales department could be slow at times. They were so slow that the original credit card expired, a fact even I had failed to notice. This merely made an already bad situation worse. Communication, which had never been very good, got worse. I had to send an e-mail every day, sometimes several times a day, to get a reply at the rate of about once a week. This time, the stories told were about problems with the order system and the credit-card processing company. More weeks passed without resolution and without any good explanation of why I was continuing to have to pay finance charges on a purchase that was never fulfilled. I was more irritated with the poor communication than anything else.

Finally, I lodged a complaint with the local Better Business Bureau and sent a draft of my letter to *Linux Journal*, in hopes that external intervention might draw enough attention to bring things to a satisfactory resolution. I am pleased to report that it did, although I am still disappointed it took as long as it did and that I had to take the actions I did.

Finally, the matter came to the attention of Ray Sanders, CTO and one of the founders of QLI. He helped clarify some of the internal issues that had caused this whole affair to drag on and also explained some of the steps that had been taken to correct the situation, including mandatory staff training, staff reviews for those involved and serious negotiations with the credit-card processing company who apparently were at least partly to blame, at one point attempting to issue the refund but only successfully debiting QLI, never crediting my account.

Ray's involvement finally resulted in a full refund, plus an unasked-for little extra to help defray the finance charges. Most importantly, Ray clearly communicated along the way what he was doing and when I should expect to see things happen and made himself directly available in case of a problem. He also was genuinely upset that it took his involvement to sort things out, another encouraging sign that someone in the organization gets it. I also see this as further proof that success can be dangerous, in that quick growth often dilutes important values no matter how strongly held at the outset.

Companies like QLI offer products and services that appeal to a market filled with savvier-than-usual consumers, so they must remember that trust and respect are as important to building and keeping goodwill as superior products and services at competitive prices. Problems such as mine should be seen as opportunities to prove commitments to the community through honesty, communication and courage.

—
Thomas Gideon

Porting US House of Representatives Software to Linux?

I started the Linux-pe mailing list over a year ago, with the purpose of providing a platform for Linux advocates in the US to *organize* to bring Linux into US governments at the local, state and national levels. My own personal project, as I live in Silver Spring, Maryland, a subway ride from Capitol Hill, is to get Linux usable by Congress. Now any member of Congress certainly can install Linux, but this will do a House of Representatives member very little good unless he or she can use the HIR (House Information Resources) software, which has not been ported to Linux, although there is nothing in principle preventing it from being ported.

Our Congressman here in the Maryland suburbs is Chris Van Hollen, a liberal Democrat. He is intelligent and energetic, but not at all IT-aware, nor is his staff. However, by dint of energetic lobbying—e-mails and visits by me and several other of his Linux-minded constituents—we have persuaded Mr Van Hollen to write a letter to the members of the Committee on House Administration, asking this committee, the one in charge of HIR, to investigate the issue of porting the client to Linux. I believe we should all congratulate Mr Van Hollen for taking this initiative and ask the members of the Committee on House Administration to act on Mr Van Hollen's proposal.

To join Linux-pe, go to www.tux.org/mailman/listinfo/linux-pe.

—
Alan McConnell

If you want to look up the members of a committee of Congress or find out your representative or senator's committees, you can do a search on Congress.org, which runs Apache on Linux. —Ed.

SE Linux from Outside the USA

I would like to correct a mistaken letter published in the November 2003 issue of *Linux Journal* [“SE Linux outside US?”]. I did my initial SE Linux work while living in The Netherlands and had no problem accessing any part of the NSA Web site using the Zon and KPN ISPs. I am now living in Australia and still have no problem accessing the NSA site. I am not aware of any country restrictions on who can access the site, and I am quite sure that there is no block on The Netherlands.

—
Russell Coker

Mandrake Automatically Recognizes Pen Drives

I just finished reading the December 2003 issue of *Linux Journal*, and I was pleased as usual. The article “Floppies for the New Millennium” by Rick Moen contained an excellent tip about using noatime for USB pen drives, which I have now adopted. As Moen pointed out, the problem of mounting his pen drive read/write seems to be with that brand. I have used a Universal Smart Drive for some time without any mounting problems. Using Linux-Mandrake 9.1, the first time I plugged in the pen drive, an icon was added to the KDE desktop and an entry was added to /etc/fstab (noatime added by me):

```
/dev/sdb1 /mnt/removable auto
user,iocharset=iso8859-1,kudzu,codepage=850,noauto,noatime,umask=0,exec
0 0
```

Right-clicking on the icon allows you to mount or unmount the pen drive. Of course, as a holdover from the old days, I also have a second entry:

```
/dev/sdb1 /mnt/usd vfat noauto,user,noatime 0 0
```

It follows the line added by Mandrake. Otherwise, the icon's load routine gets confused. Either mountpoint can be used from the command line. As Moen noted, leave the FAT format in place. It makes the pen drive usable on most machines (like that quarantined Windows box). Keep those articles coming!

—
Terry Vaughn

Traffic Shaping Advantage

I am no longer a newbie, but every time I explore Linux I discover that it is more than an ordinary operating system. I could not have prioritized my network bandwidth better than with the easily configured traffic shaper package shapecfg. Thanks to the community of developers and everyone who is involved in Linux. I am more than convinced that by using Linux you will save your company from all unnecessary expenses in securing a good networking environment and administer your network more effectively. It's nice discovering Linux.

—
Makinde Olojede
Nigeria

SATA Question on Ultimate Linux Box

As always, I enjoyed getting your most recent issue [December 2003]. In the Ultimate Linux Box article, it mentions that SATA is used. But nowhere does it say what had to be done to get SATA working. Were any kernel patches installed to get this to work with 2.4.19?

—
Eric

Glenn Stone replies: We did not use the regular Serial ATA interface on the motherboard, but instead used the SATA interfaces on the 3ware Escalade 8500 card; the 3w-xxxx driver we used was the stock driver that comes with SuSE Linux Enterprise Server 8. 3ware has been really good about seeing to it that its drivers made it into the main kernel tree the last two years or so.

Serial ATA support for the Silicon Image chipset is present (in the source, at least) in SuSE's 32-bit offerings in kernel 2.4.20. Linus' 2.4.22 tree adds support for the Intel ICH5. Those are the only two SATA chipsets that linux-ide.org advertises support for as of this writing.

Photo of the Month

Here's a picture of my daughter Madeline holding her new little brother, Griffin.

—
Greg Kroah-Hartman

If your photo is chosen as Photo of the Month, we'll give you a one-year subscription or extend your current subscription by one year. —Ed.

Errata

December 2003, “Ultimate Linux Box”: Credits for technical support and other assistance with the project were inadvertently omitted. The author would like to thank Trey Harris and Chuck Henderson of Monarch Computer Systems, along with Clay Deveny, Wanda Gillis and the rest of the Monarch staff; Paul Chang of Arima Computer Corp.; Mark Visconti and Susan Austin of NVIDIA Corp.; and Greg Kroah-Hartman of IBM for being instrumental in the success of this project.

Also, it should have been noted that the case used for the ULB Project is a Lian-Li product that is available only through Monarch Computer Systems (www.monarchcomputer.com/Merchant2/merchant.mv?Screen=PROD&Product_Code=100128).

Archive Index Issue Table of Contents

Advanced search

UpFront

diff -u, They Said It and more.

diff -u: What's New in Kernel Development

by Zach Brown

Last month I wrote about **digsig**, a security module that checks the checksum of a binary before executing it to ensure that no compromised code is run on a given system. Apparently, I was a bit more enthusiastic than various kernel developers, notably **Alexander Viro** and **Pavel Machek**. It turns out (and the digsig developers agree with this) that digsig is no more than a temporary security tactic, which script kiddies would be able to work around after a fairly short time. Historically, these sorts of temporary measures never have been adopted into the kernel, precisely because they don't provide a robust solution.

With the 2.6.0-test1 kernel released in early November 2003, **Linus Torvalds** announced a **stability freeze**, meaning patches would be accepted only if they fixed data corruption, system crashes or some other serious breakage. Even cleanup patches that don't actually change the behavior of the code are being rejected now. According to Linus, the goal is to start the final preparations for the official 2.6.0 release, at which point he has strongly hinted that he immediately will hand maintainership of the 2.6 tree to **Andrew Morton**. This is a change from 2.4, where he handed maintainership to **Marcelo Tosatti** only after several releases. There is still no indication of when the 2.7 tree will fork; it could occur at the same time as the hand-off to Andrew, though certainly not before.

Rusty Russell and **Tim Hockin** together have produced a patch to raise the number of supported groups under Linux to well over 200. In fact, the patch supports thousands of groups, something various folks (like some of Tim's Samba clients) sometimes want. Linus Torvalds felt that some incarnations of their patch were uglier than others, but apparently he's ready to consider support for that many groups, as long as the implementation doesn't make him retch too hard into his hand.

Ian Pratt and others from the University of Cambridge Computer Laboratory Systems Research Group have produced the first stable release of their **Xen** virtual machine monitor for the x86 architecture and have ported the 2.4.22 kernel to it as a guest OS. Xen is designed to run multiple operating systems simultaneously on a single computer. Once a kernel is ported to Xen, all user binaries apparently will run unchanged; so in theory any Linux distribution would run under Xen, with a drop-in kernel as the only modification. Currently, it's not possible to run Xen recursively, though various twisted kernel hackers have expressed such an interest.

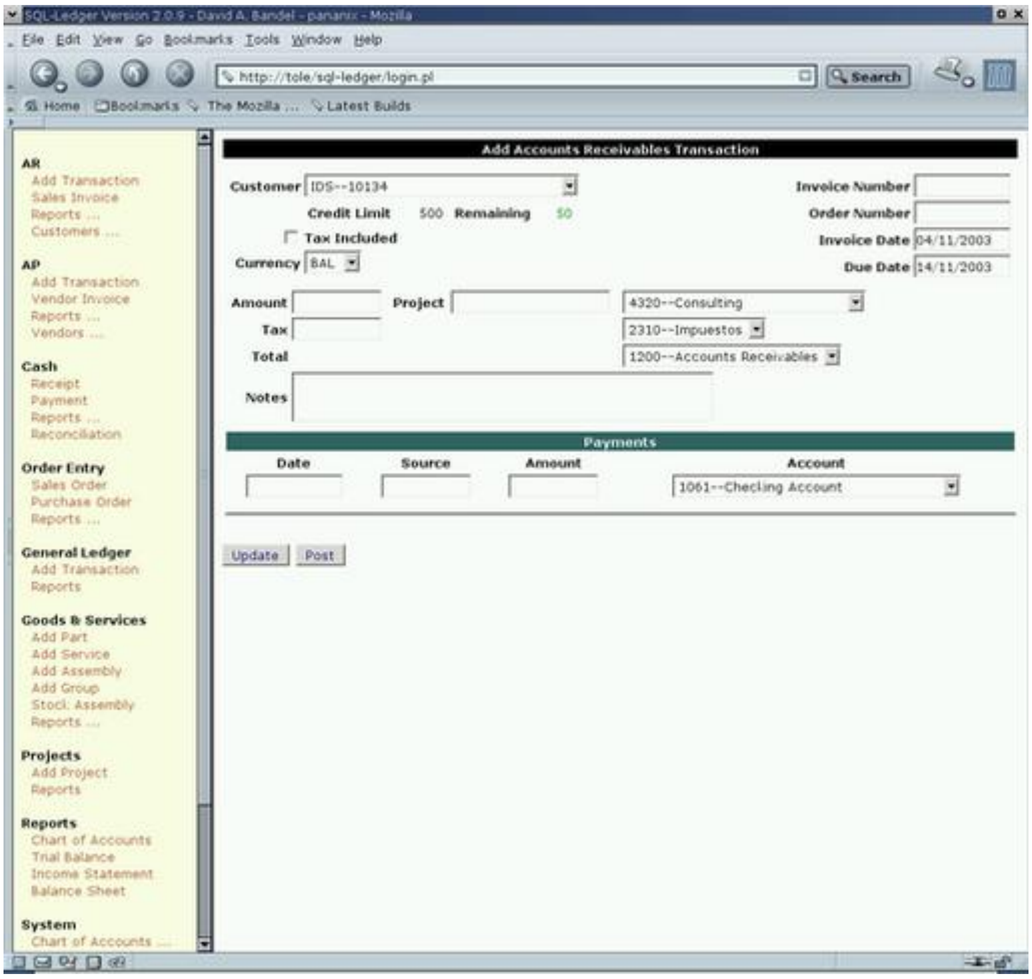
Eli Billauer has created **/dev/frandom**, a random number generator that is up to 50 times faster than /dev/urandom, under some tests. Although not intended to be random enough for cryptography (but it might one day become so), it nevertheless is good for scientific simulations, stress tests on algorithms and performing data-wipes. In spite of its various good qualities, it probably will have to live as an external patch for quite a while, as several kernel developers like **Nick Piggin** feel there is no compelling need to do /dev/frandom in kernel space, because it could be done well enough entirely in user space.

Roman Zippel announced in mid-October 2003 an implementation of the **iSCSI** specification. Although it's not a complete implementation and uses the deprecated /proc filesystem instead of SysFS for its interface, it apparently is already somewhat usable. Roman has promised to continue maintaining it if there is enough interest, but without help he says it may be slow-going.

SQL-Ledger: www.sql-ledger.com

by David Bandel

I don't think I've seen any project come as fast or as far as this one has in the three years I've been using it. This system offers full double-entry accounting for almost any business in a large number of languages. I'd be hard-pressed to find a better accounting package. Installation is extremely simple. If you have LaTeX installed, you can output PDF files directly or e-mail your bills or statements. The out-of-the-box configuration is based on the Canadian tax system but easily can be adopted to any other system. One of the best features is that the database uses PostgreSQL. If you need a full accounting package for a business, this is it. Requires: Perl, Perl modules DBI, DBD- (DBD-Pg or DBD-Oracle), PostgreSQL or Oracle, Web server capable of running Perl scripts, Web browser and LaTeX (optional).



Add Accounts Receivables Transaction

Customer IDS--10134
Credit Limit 500 Remaining 50
Tax Included
Currency BAL
Invoice Number
Order Number
Invoice Date 04/11/2003
Due Date 14/11/2003
Amount
Project
Tax
Total
Notes
4320--Consulting
2310--Impuestos
1200--Accounts Receivables

Payments

Date	Source	Amount	Account
			1061--Checking Account

They Said It

I do not believe that we should be willing to buy or use voting systems where the source code and design is not open for public review. I think there are companies that would be willing to work in this model, particularly if the contract provided some long-term commitments. This is not Britney Spears we're talking about here–the integrity of our voting system is a fundamental component of our government.

—Phil Windley, www.windley.com/2003/11/03.html#a893

eGovernment applications need to cost less, allow for rapid development, provide a user–friendly experience for constituents and offer enhanced security. Linux provides everything eGovernment initiatives need.

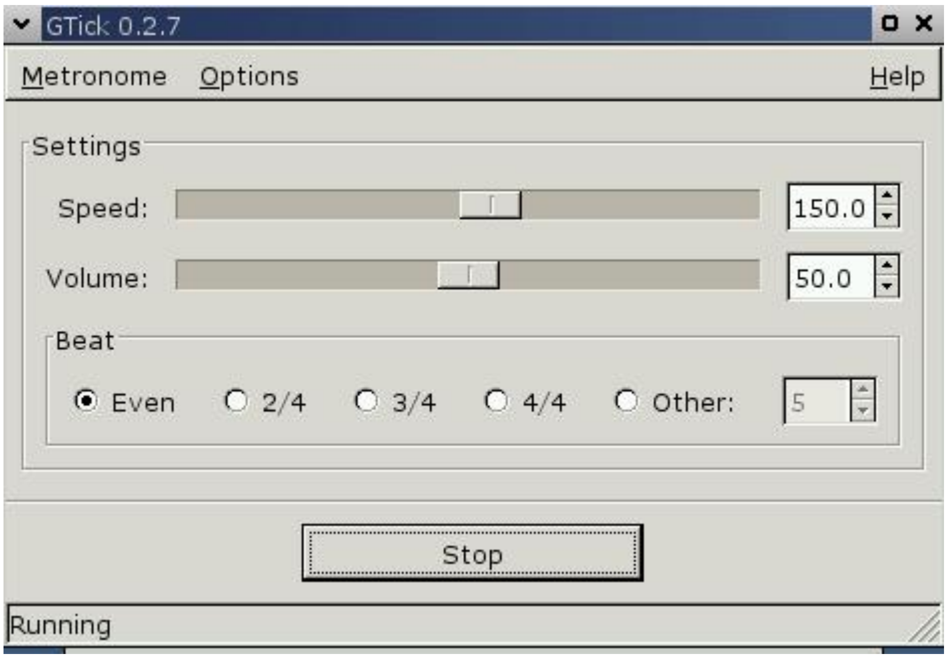
—Tom Adelstein, www.linuxjournal.com/article/7076

Interestingly, the most popular source of embedded Linux for future projects is “home grown (I build my own from GNU/Linux downloaded sources)”. That is, developers apparently prefer freely downloadable noncommercial sources–such as kernel.org, Debian or the uClinux project–over commercial embedded Linux distributions.

—Rick Lehrbaum, linuxdevices.com

gtick: www.antcom.de/gtick linuxdevices.com

Need a metronome? This metronome doesn't only give you a simple beat for your music, it also gives you selectable beats, such as 2/4, 3/4, 4/4 or a customizable beat. Variable speed and volume controls with both slider bars for coarse tuning and an incremental wheel for fine-tuning allow you a lot of flexibility. Requires: libgtk-x11-2.0, libgdk-x11-2.0, libatk-1.0, libgdk_pixbuf-2.0, libm, libpangoxft-1.0, libpangox-1.0, libpango-1.0, libgobject-2.0, libgmodule-2.0, libdl, libgthread-2.0, libglib-2.0, libpthread, glibc, libX11, libXi, libXext, libXft, libXrender, libfontconfig, libfreetype, libz and libexpat.



Speed: 150.0
Volume: 50.0
Beat: Even [✓] 2/4 3/4 4/4 Other: 5
Running

_LJ_ Index—February 2004

1. Minimum projected percentage TCO (total cost of ownership) savings for Sanchez Computer Associates' Linux-based core processing system for banks: 50
2. Millions of people using Linux in the world: 18
3. Top dollar price of Commercequest's software for helping companies meet the new Sarbanes-Oxley corporate compliance law: 500,000
4. Percentage of IT budgets that go to internal staff: 70
5. Annual percentage growth rate of Wi-Fi access-point shipments through 2008: 50
6. Millions of Wi-Fi access points expected to be shipped annually in 2008: 1
7. Projected percentage growth rate in Wi-Fi access-point shipments in 2005: 132
8. Rank of Linux among embedded Wi-Fi access-point operating systems: 1

9. Percentage of embedded Linux developers who preferred “home grown” as their “Linux source/vendor” during the past two years: 15
10. Rank of “home grown” as the preferred “Linux source/vendor” for embedded developers during the past two years: 2
11. Percentage of embedded Linux developers who preferred Red Hat as their “Linux source/vendor” during the past two years: 17
12. Rank of Red Hat as the preferred “Linux source/vendor” for embedded developers during the past two years: 1
13. Percentage of embedded Linux developers who prefer “home grown” as their “Linux source/vendor” during the next two years: 18
14. Rank of “home grown” as the preferred “Linux source/vendor” for embedded developers during the next two years: 1
15. Percentage of embedded Linux developers who prefer Red Hat as their “Linux source/vendor” during the next two years: 17
16. Rank of Red Hat as the preferred “Linux source/vendor” for embedded developers during the next two years: 2
17. Minimum percentage of future Linux embedded projects “attributable to essentially noncommercial sources”: 50
18. Millions of Apache Web servers surveyed by Netcraft for November 2003: 30.30
19. Percentage of all surveyed Web servers that run Apache: 67.41
20. Percentage gain by Apache over the prior month: 2.80
21. Percentage loss by Microsoft Web servers during the same period: 1.65

Sources

- 1: Sanchez Computer Associates
- 2: *Wired*
- 3–6: *Forbes*
- 7–17: ABI Research, Linux Devices
- 18–21: Netcraft

From the Editor

Web Technologies for Business Apps

Don Marti

Issue #118, February 2004

Remember this? “That's a nice demo application you've got working on the Linux box. Now spend ten times the money and twenty times the work to do it for real.”

No matter how long it took, the “real” version on a proprietary platform somehow never seemed to have the stability, performance or maintainability of the version you put together with Perl, Python or PHP. The good news is that now, even for the most conservative employers or clients, you no longer have to redo your Linux projects on an approved platform.

The tools you have been using to build Web sites that aren't business-critical now are seeing action in core business apps at more and more companies.

There's always someone who says that you can't possibly develop business apps on *that*—it doesn't have feature X. Soon, though, someone does write feature X for the new platform, or it turns out that you don't need feature X to do business apps anyway. Then new projects that could have gone with the legacy platform start going to the new one.

That's what's happening right now with the all-free, all-open source LAMP platform, consisting of Linux, Apache, MySQL and the “P” languages—Perl, Python and PHP.

Giovanni Organtini and Luciano M. Barone cover one large success story on page 36. Their LAMP-based work-flow management system, used in assembling a particle physics instrument with 500,000 parts, has replaced a proprietary system. The new system cuts CPU and memory loads, improves performance and, most important, slashes the amount of time that operators spend interacting with the system, giving them more time to work on the product.

On page 50, Tom Adelstein has some encouraging news: the LAMP platform also is the basis of US government IT projects at the Navy, the Department of Labor and other agencies. He reports that state and local governments are succeeding with Linux too. Selling to the government is hard, but the consulting firm gOSapps LLC has done more than 500 apps for 400 government bodies. Fewer of your tax dollars are being wasted on lock-in, but there's still work to be done.

Doc Searls has been following IT's biggest behind-the-scenes story for a year now. Customers are using the power of Linux and other open-source software to take control of their own technology decisions. Acronym alphabet soup and projected releases in 2005 or 2006 might make for entertaining reading, but when you've got a project to do, it's time to break out the tools that give you freedom. Doc reports on Linux successes at Morgan Stanley, Ticketmaster and Ernie Ball, on page 48.

Customer-facing Web sites have long taken advantage of Linux's performance, flexibility and low total cost of ownership. The record industry, however, hasn't been on the best of terms with the Web. Time to start over. On page 42, John Buckman explains how he is running a record company that treats both the audience and the artists with respect, not with Digital Rights Management or other such indignities. I'm listening to an album I bought from the site right now.

Whatever your business or your pleasure, there's something for you in this issue. It's not all about the Web, either. With the article by Brett Schwarz on page 72, you can build a custom phone system that saves money, integrates voice over IP and even gives you a pop-up warning of special callers. Have a productive and successful month, and see you next time.

Don Marti is editor in chief of *Linux Journal*.

Best of Technical Support

Our experts answer your technical questions.

USB Joystick?

I have a Logitech Wingman Force 3D, a USB joystick with force feedback. I'm running a 2.4.20-20.9 kernel (Red Hat 9) with USB support enabled. Is there a way to make USB joysticks work under Linux? I've done searches on Google and I found a few possible solutions, but none of them worked for me. I do not care so much for the force feedback component, but I would like to be able to make the joystick work.

—
Pierre Rochefort

prochefort@rogers.com

USB joysticks should work fine under Linux. To learn how to configure them, see the Linux USB Guide at www.linux-usb.org. If you still have questions about USB support, you can ask on the linux-usb-users mailing list.

—
Greg Kroah-Hartman

greg@kroah.com

According to Johann Deneux's Web page at user.it.uu.se/~johannd/projects/ff/index.shtml, he is developing a force-feedback driver for Linux. Complete instructions and downloadable code are provided on this page, so you can test whether this driver works with your device. You might find you need some code compiling and module installation skills, but it may be worth the learning experience. You also might want to try the Linux Input Driver (linuxconsole.sourceforge.net/input/hardware.html) or the Linux joystick driver (atrey.karlin.mff.cuni.cz/~vojtech/joystick), which also states that it supports your device.

—
Felipe Barousse Boué

fbarousse@piensa.com

Configuring a New Network Card

I need to configure my third network card on my Red Hat 6 system to start automatically when the system reboots. How can this be done?

—
Akaize

akaize@zmsn.com

Your Red Hat release is a bit old; I'd suggest you upgrade it because you will get many benefits, such as security, stability and more hardware drivers. Nevertheless, you can try the user-friendly approach by using the netcfg or netconfig utility (from a root shell) and following on-screen instructions and options. Alternatively, you can edit the /etc/modules.conf and the /etc/sysconfig/network-scripts/ifcfg-ethX (where X is the net card number) files manually. The modules.conf file names each card and relates that name to the corresponding module (driver) that you need to have in your system. The ifcfg-ethX file is the actual configuration file of the card; there is one of these files per network card. Follow the example of the first file, which must be named ifcfg-eth0. The Ethernet HOWTO contains more information: www.ibiblio.org/mdw/HOWTO/Ethernet-HOWTO-2.html#ss2.4.

—
Felipe Barousse Boué

fbarousse@piensa.com

Setting Up Wi-Fi Restricted Mode on SuSE

I successfully installed SuSE 9.0 on my laptop, a ThinkPad 600E that uses a Netgear MA401 wireless card. SuSE has detected my card properly, and I have 128-bit encryption enabled and working. The problem is I need to put the card in restricted mode manually to be able to communicate with the access point. Every time I reboot the laptop, I have to `su` into root and type `iwconfig wlan0 key xxxxxxxx restricted` to make the card work. I configured the card properly using YaST and included the encryption key, but it always defaults to open mode. Did I miss something here? How can I make iwconfig default to restricted mode?

—
Kevin Lisciotti

lisciotti@yahoo.com

Make a copy of the file /etc/sysconfig/network/ifcfg.template and edit it to set the parameters according to your proper setup. Also, take a look at portal.suse.com/sdb/en/2002/11/wavelan.html, which offers more detailed information on this subject.

—
Felipe Barousse Boué

fbarousse@piensa.com

Red Hat 7.2 and Ethernet Card

I recently installed the Red Hat 7.2 distribution (my first Linux installation) that came with the book *Linux Administration for Beginners*, which I borrowed from a friend. When I installed Red Hat 7.2, I did not get a network config screen, as was suggested and shown in the book. The installation is supposed to recognize my network or something, but it doesn't. Hence I have been unable to set up my Internet connection, which is key to more resources for learning other than man pages.

I am using a home PC with the following components: Intel 3 500MHz, 256MB of RAM, 20GB WD Caviar IDE and Realtek (also tried it with D-Link card) NIC. I am connected by a cable modem and have no network, although I plan on expanding to two computers soon.

Red Hat 7.2 does not seem to recognize eth0, although there is a constant connection. I have searched and searched for instructions. Most direct me to install drivers from floppies for Realtek, which already appear to come with the distro, but as I said, I am really new to this and have no clue. I am about to go buy Mandrake or Red Hat 9 to see if either proves to be a more useful installation.

—
Marcel

marcelnh4@hotmail.com

Without knowing specific error messages or seeing some log files, it is extremely difficult to guess what is going on with your system. Is your network card okay? Is it properly installed? Does it have any conflicts with other hardware? Instead of buying another Linux distribution, you probably should buy a new network card, one that you know is officially supported. Go to hardware.redhat.com/hcl and look for a network card that fits your budget and your system.

—
Felipe Barousse Boué

fbarousse@piensa.com

To rule out a hardware problem, try your system with Knoppix (www.knoppix.net), which lets you run a current Linux from CD without installing on your hard drive. You can download and burn Knoppix freely. If the card works under Knoppix, you'll have more fun and have more time to learn administration skills if you upgrade to a more current distribution. For your first Linux distribution, try to select one with which your local user group, or whatever source of support you use, is familiar.

—
Don Marti

info@linuxjournal.com

Reconfiguring Wireless Keys for Different Sites

I use Wi-Fi on my laptop (running Red Hat 6.2) at a number of different sites, each one with a different set of encryption keys. It's a hassle to have to edit /etc/pcmcia/config.opts every time. Is there an easier way to manage my keys?

—
Andreas Meyer

ysgdhio@yahoo.com

Jean Tourrilhes maintains an extensive list of wireless utilities that do things like manage configurations and monitor signal strength at www.hpl.hp.com/

personal/Jean_Tourrilhes/Linux/Tools.html. One of the tools listed is waproamd, which automatically sets up preconfigured WEP keys based on the ESSID of the wireless network you're on (0pointer.de/lennart/projects/waproamd). There are also two simpler ways to deal with this issue, as explained in Felipe's answer and my other answer.

—
Don Marti

info@linuxjournal.com

For a poor guy's quick-and-easy way with not much hassle: 1) write a small shell script in /usr/local/bin for each site. Name each wireless-sitename or something similar, list the corresponding appropriate iwconfig and ifconfig commands (including the keys, of course) and set the proper permissions and ownership. Whenever you get to the site, simply run the wireless-sitename script and you should be set. This allows individual turn on/off control of access in each of your preferred network sites.

—
Felipe Barousse Boué

fbarousse@piensa.com

The way I deal with this kind of thing is to make extra copies of the config file. For example, you can create config.opts.home and config.opts.cafe, then set up an alias or panel button for cafe to do this:

```
sudo cp /etc/pcmcia/config.opts.cafe \
/etc/pcmcia/config.opts \
&& sudo /etc/rc.d/init.d/pcmcia restart
```

and an alias or panel button for home to do the same thing with the home version of the file.

—
Don Marti

info@linuxjournal.com

Managing Desktop Users and Policies?

I have worked at several businesses that could use the power of Linux on the desktop, but so far I have been reluctant to suggest it. It's not that I don't think it's a superior product or that the relative cost savings are significant. It has to do with the fact that these organizations are not likely to want to replace conventional computers with dumb terminals or low-powered network computers tied to a mainframe or server. Tied into this is the issue of administering 20 machines, 100 machines or 2,000 machines. I have not found anything so far that lets me sit down and look at a domain tree of users and administer polices and profiles. My ignorance may be blatantly clear at this point. I am a Linux user at home, but I am a Windows administrator and like the apparent ease of managing a Windows network. Can you point me in the right direction on this? Maybe I don't understand the vision or current mindset of the community on system administration.

—
Aaron Sharp

baronvaile@mach500.net

Undoubtedly and disregarding the used technologies, you are talking about a complex network configuration. Many issues come up: user management, password management, directories, e-mail addresses, IP numbers, shared resources and security. Many system administration tools and efforts exist in the Free Software/Open Source community to deal with these issues. One of them that I have used several times is Webmin, www.webmin.com. Webmin is a powerful and extensible systems management tool; it even allows clustering and remote systems management.

—
Felipe Barousse Boué

fbarousse@piensa.com

A flexible way to manage user information is with LDAP. Craig Swanson and Matt Lung covered a unified system for shared address books, unified login and shared file storage in the December 2002 issue of *Linux Journal* (/article/6266). If you like the Webmin interface, you can use Webmin to manage an LDAP database.

—
Don Marti

info@linuxjournal.com

Advanced search

On the Web

New Product Hype

Heather Mead

Issue #118, February 2004

If you're considering switching to a new distribution or purchasing some new hardware, stop by our on-line product review section first.

Trying to keep up with all the new product announcements that come across our desks would be a full-time job. Luckily, we have a pool of reviewers who take on much of the responsibility for trying out new hardware, software, books, games and every type of gadget they can get their hands on. Because we have only enough print space to run one or two reviews per issue, the *Linux Journal* Web site has become our best source for reviews.

The much shorter time frame involved in getting articles published on the Web makes our Web site an ideal place to post reviews of new distribution releases. The final Fedora Core 1 release arrived early in November 2003, and within a week, Adam Jenkins' “Fedora at a Glance” was posted on our site (www.linuxjournal.com/article/7257). Adam shared his experiences with downloading and installing Fedora and then discussed Fedora's features—what it had, what it didn't have and what was on its way. As more users probably will be making the switch to Fedora in the coming months, Adam's article provides some insight into what they'll be getting when they make the move.

If you're thinking about giving Gentoo a try, Sean Bossinger's “Gentoo Linux” product review is worth a read (www.linuxjournal.com/article/7002). Sean focuses his review on Gentoo's installation process, which is more manual than the commercial distributions. The upside of the manual input is you can optimize the compiled code for the settings specific to your system, which is part of Gentoo's goal of being a highly customizable distribution. The downside is all that customization can take quite a bit of time to achieve.

If you're still riding the Opteron wave, check out Steve Hastings' review of the “Appro Rackmount Dual Opteron Server” (www.linuxjournal.com/article/6883). Or if you're looking for something a bit more fun, something in the way of a gadget, we recently reviewed two Linux handheld devices. Apparently the really cool PDAs aren't available in the US; luckily, you can order them on the Web from international resellers. The Yopy 3700 that Guylhem Aznar reviewed (www.linuxjournal.com/article/6933) was ordered through a French reseller. Although it looks good and has a better keyboard, Guylhem isn't sure the Yopy is a better purchase than a Zaurus. And Tony Steidler-Dennison's review of the Sharp Zaurus SL-C760 (www.linuxjournal.com/article/7162) certainly makes it sound like this is the coolest Zaurus and the coolest PDA ever. Its superior screen display and its ability to function in either landscape or desktop mode, thanks to a pivot hinge, might be enough to convince you that you need one.

I haven't even mentioned all the book reviews you can browse on our site. So before you lay out any money on new purchases, visit the *Linux Journal* Web site and click on Product Review or Book Reviews under Topics. If you're interested in joining our reviewers mailing list and helping us test all this stuff, send me an e-mail at info@linuxjournal.com.

Heather Mead is senior editor of *Linux Journal*.

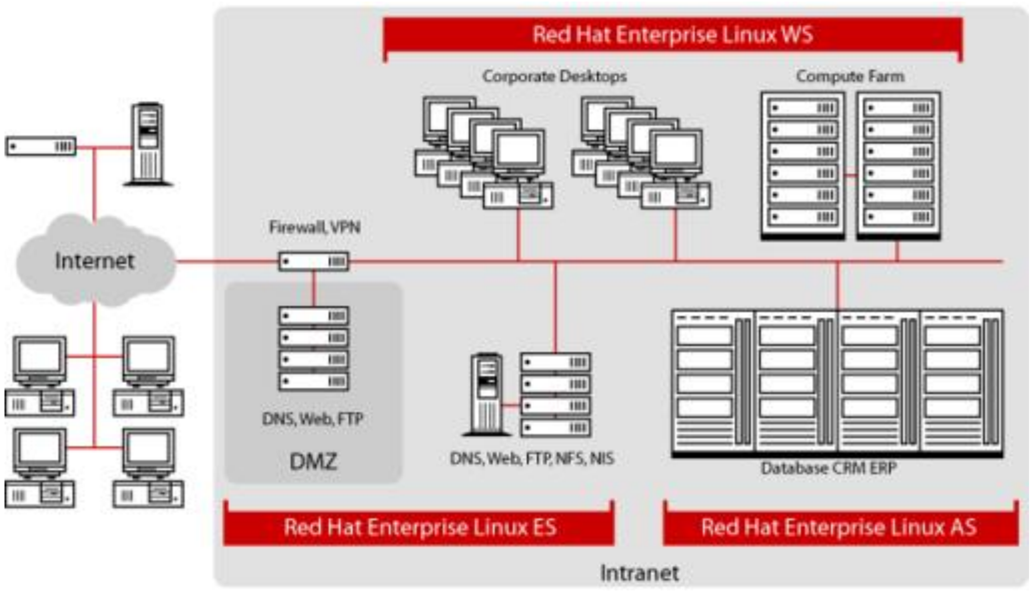
New Products

Red Hat Enterprise Linux 3, MontaVista DevRocket Development Environment and more.

Red Hat Enterprise Linux 3

Red Hat Enterprise Linux 3 now is available to support enterprise computing needs, from desktops to servers, on seven hardware architectures. Designed to offer a secure and consistent enterprise-wide platform, Red Hat Enterprise Linux 3 includes a native POSIX threading library for multithreading applications and a single code base. Other new features for version 3 include support for larger SMP, memory and I/O configurations; a 4-4 memory split to provide increased kernel and user address space for x86 systems; and Java implementations from BEA, IBM and Sun. The seven supported hardware architectures are x86, Itanium, AMD64 and IBM's zSeries, iSeries, pSeries and S/390.

Red Hat, Inc., PO Box 13588, RTP, North Carolina 27709, 919-754-3700, www.redhat.com.



MontaVista DevRocket Development Environment

MontaVista DevRocket 1.0 is a fully integrated and graphical development environment for embedded Linux. Based on Eclipse technology, it provides a common look-and-feel across development host platforms including Linux, Microsoft Windows and Solaris. MontaVista DevRocket is built on the latest Eclipse 2.1 base, letting customers and ISVs take full advantage of the Eclipse platform, including third-party development work contributed by the Eclipse community, as well as a multitude of Eclipse-supported tools. In addition to integrating core development capabilities such as compilation, editing and debugging, DevRocket provides easy-to-use project wizards designed to automate common embedded development activities.

MontaVista Software, 1237 East Arques Avenue, Sunnyvale, California 94085, 408-328-9200, www.mvista.com.

Announcing Professional Edition 3.1
MontaVista DevRocket Launches

NIOS Platform

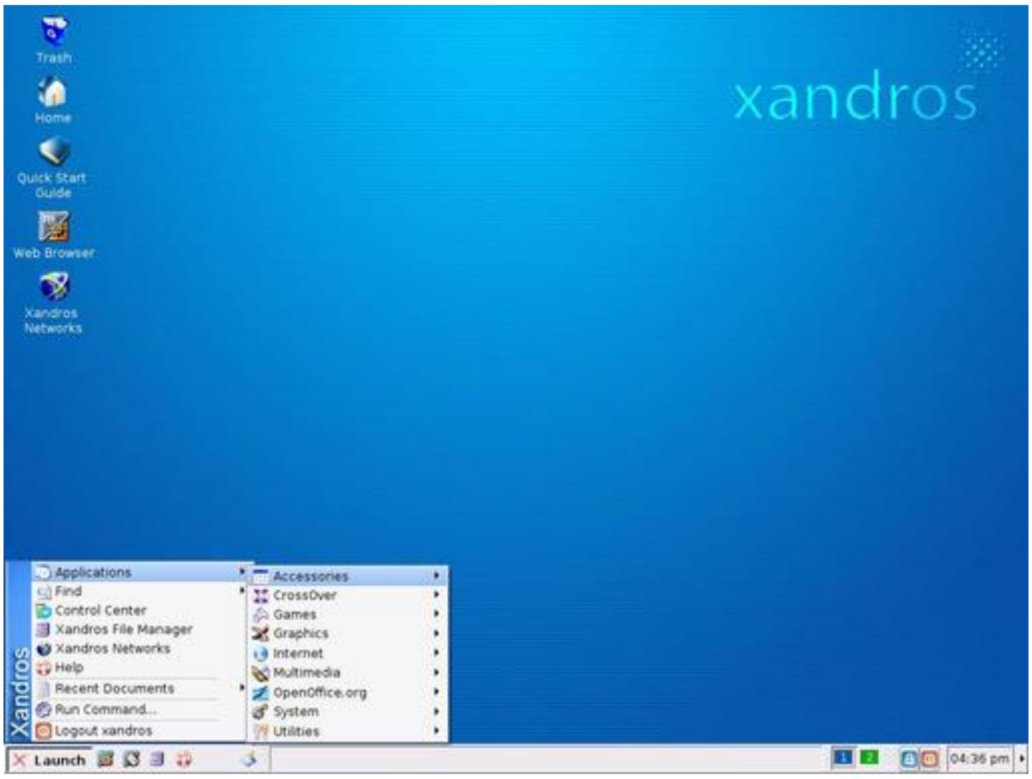
Net Integration Technologies offers an autonomic computing network platform based on its Net Integrator Operating System (NIOS), which is based on Linux. Designed to be self-repairing and self-maintaining, the NIOS Platform uses off-the-shelf hardware and is geared toward small- and mid-sized businesses. NIOS itself is built from open-source software and is 16MB in size. Included in the NIOS Platform is NetIntelligence, an artificial intelligence module that uses autonomic features to deploy, install and maintain system components and internal subsystems, including firewall and DHCP parameters and DNS records. The SystemER program enables system recovery from catastrophic failure in under two minutes. Other components include ExchangeIt!, a collaboration server; TunnelVision, an intelligent VPN solution that works without static IP addresses; Expression Desktop; and DoubleVision, a redundant Internet connectivity technology designed to connect multiple high-speed interfaces to NIOS-powered servers.

Net Integration Technologies, Inc., 7300 Warden Avenue, Suite 106, Markham, Ontario, Canada L3R 9Z6, 866-384-8324, www.net-itech.com.

Xandros Desktop 2.0

Xandros Desktop 2.0 now is available and offers an easy-to-use graphical environment that installs with four mouse-clicks. Based on the Sarge version of Debian and a Xandros-enhanced version of KDE 3.1.4, key Xandros features include a four-click installation process with automatic disk partitioning, drag-and-drop CD burning within the File Manager and file and resource sharing with Windows networks. Desktop 2.0 offers standards-compliant Web browsing of multiple sites in a single tabbed window; a mail reader with automatic spam filtering and the ability to turn off pop-up ads and banners; an instant messaging client that is compatible with MSN, Yahoo, AOl, ICQ and IRC; and OpenOffice.org 1.1. Xandros Desktop 2.0 comes in Standard Edition and Deluxe Edition. Deluxe Edition includes CrossOver Office 2.1; a 350-page user guide; extra games, applications and tools; and 60 days of e-mail support.

Xandros, Inc., 41 East 11th Street, 11th Floor, New York, New York 10003, 613-842-3494, www.xandros.com.

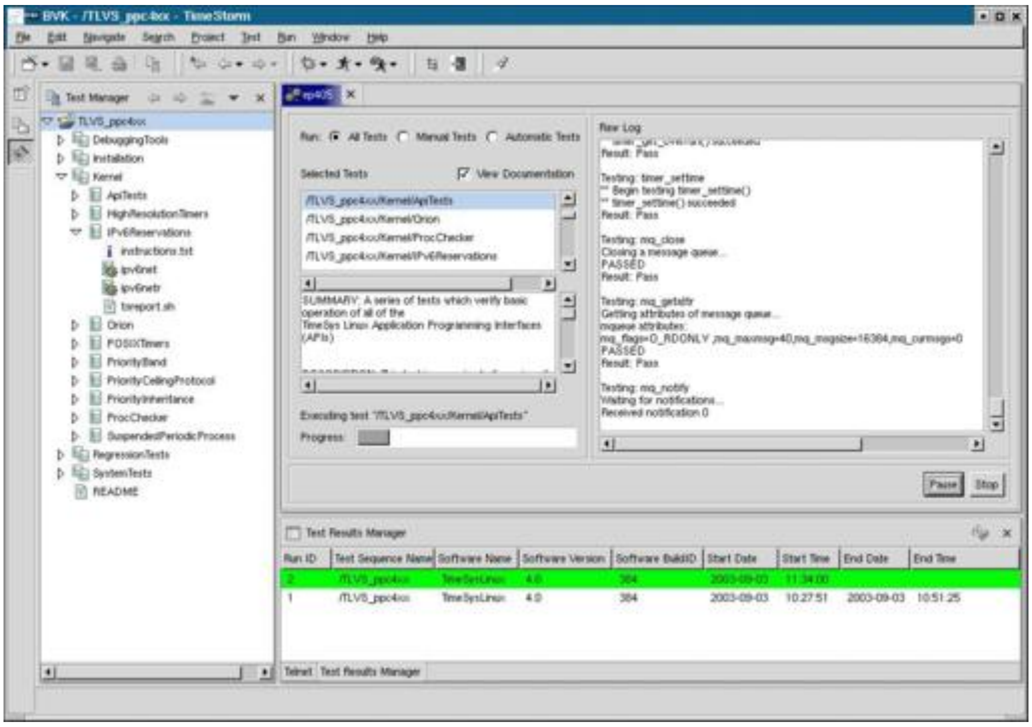


TimeStorm Linux Tool Suite

TimeSys' new TimeStorm Linux Tool Suite contains development tools that support the entire embedded Linux development cycle, no matter what type of embedded OS is being used. Built on the Eclipse Platform, the TimeStorm suite includes tools to handle kernel porting, hardware integration and a full range of testing and validation requirements. Delivered as plugins to TimeSys' TimeStorm IDE, the Tool Suite works with any embedded Linux distribution, homemade or commercial. The Tool Suite includes the TimeStorm Linux

Verification Suite, a framework that automates the testing and validation of an entire embedded distribution and its applications at each step in the development process. More than 1,150 open-source tests are available. The Linux Development Suite provides tools to help developers build and port a custom Linux OS to target hardware. The Linux Hardware-Assisted Debug assists in hardware debugging, initialization and Linux bring-up by providing an interface between the TimeStorm IDE and JTAG and on-chip debuggers using GDB.

TimeSys, 925 Liberty Avenue, 6th Floor, Pittsburgh, Pennsylvania 15222, 412-232-3250, www.timesys.com.


BVK - /TLVS_ppc4xx - TimeStorm
File Edit Navigate Search Project Test Run Window Help
Test Manager
Run: All Tests Manual Tests Automatic Tests
Selected Tests
View Documentation
Pause
Stop
Telnet Test Results Manager

```
Raw Log:
Result: Pass

Testing: timer_settime
** Begin testing timer_settime()
** timer_settime() succeeded
Result: Pass

Testing: mq_close
Closing a message queue...
PASSED
Result: Pass

Testing: mq_getattr
Getting attributes of message queue...
mqueue attributes:
mq_flags=O_RDONLY ,mq_maxmsg=40,mq_msgsize=16384,mq_curmsgs=0
PASSED
Result: Pass

Testing: mq_notify
Waiting for notifications...
Received notification 0
```

Executing test "/TLVS_ppc4xx/Kernel/ApiTests"

Test Results Manager

Run ID	Test Sequence Name	Software Name	Software Version	Software BuildID	Start Date	Start Time	End Date	End Time
2	/TLVS_ppc4xx	TimeSysLinux	4.0	384	2003-09-03	11:34:00		
1	/TLVS_ppc4xx	TimeSysLinux	4.0	384	2003-09-03	10:27:51	2003-09-03	10:51:25

Global Navigator 5.0

Global Navigator 5.0, from NEC, is a monitoring solution for tracking call activity and agent performance across single or mulitple contact centers. Global Navigator provides enterprise-wide call center management, as well as added control on a single or multisite network level. The application uses Infocast, real-time information delivery software that sends contact center statistics to a small window at the agent's workstation. New for version 5.0 is the migration of Global Navigator to Linux from SCO UNIX and the use of MySQL as the database management system.

NEC America, Inc., 6555 North State Highway 161, Irving, Texas 75039, 800-338-9549, www.cng.nec.com/cng.

Archive Index Issue Table of Contents

Advanced search