LINUX JOURNAL

Advanced search

Linux Journal Issue #122/June 2004

12 Soundfile Editors Compared
LINUX JOURNAL
Unlimited free updates and CrossOver Office too
Xandros Desktop Deluxe 2.0
FIRE DRILL
simulations at the New York Fire Department
Getting Justice: state and local courts stand up for open source
Hacking Democracy: the Web gets political
GNU Radio: FM, ham and HDTV on the same hardware
Develop safe user-space device drivers

Simulators for Training Firefighters

Douglas Maxwell

Issue #122, June 2004

The US Navy and the New York Fire Department are teaming up to create a safe, realistic 3-D training environment.

According to the Federal Emergency Management Agency (FEMA), there was a 31% decrease in the incidence of structure fires throughout the United States between 1987–2001. A direct result of this reduction means less firefighting experience for our firefighters. As more-experienced firefighters retire, they are replaced by comparatively less-experienced personnel. This situation mandates optimal training techniques. Visualization and simulation technologies are maturing at a rapid pace and offer great potential to augment current training programs.

Using today's visualization technologies, a training simulator offers a realistic representation of real-world environments. Models of real-world facilities, buildings and areas can be rendered with great detail. In addition, the computational improvements made to PCs and laptops make them practical and inexpensive platforms to deploy. These training techniques also can be adopted easily by the current and next generations of firefighters who typically engage in video gaming for recreation. Due to the inherent dangers of training for fire emergencies, it is hoped that these emerging visualization technologies can augment current training techniques and better prepare firefighters for emergencies.

Models

The New York Fire Department (NYFD) recently built a multimillion-dollar extension to its training facility at Randall's Island (RI), near Manhattan. This facility is a one-block re-creation of typical architectures found in the five boroughs. Among the buildings are a brownstone, tenement, pizza shop and auto parts store. In September 2003, a team from the Naval Undersea Warfare

Center (NUWC) was invited to tour this facility and photograph it so they later could create textures for the models being developed for the NYFD.

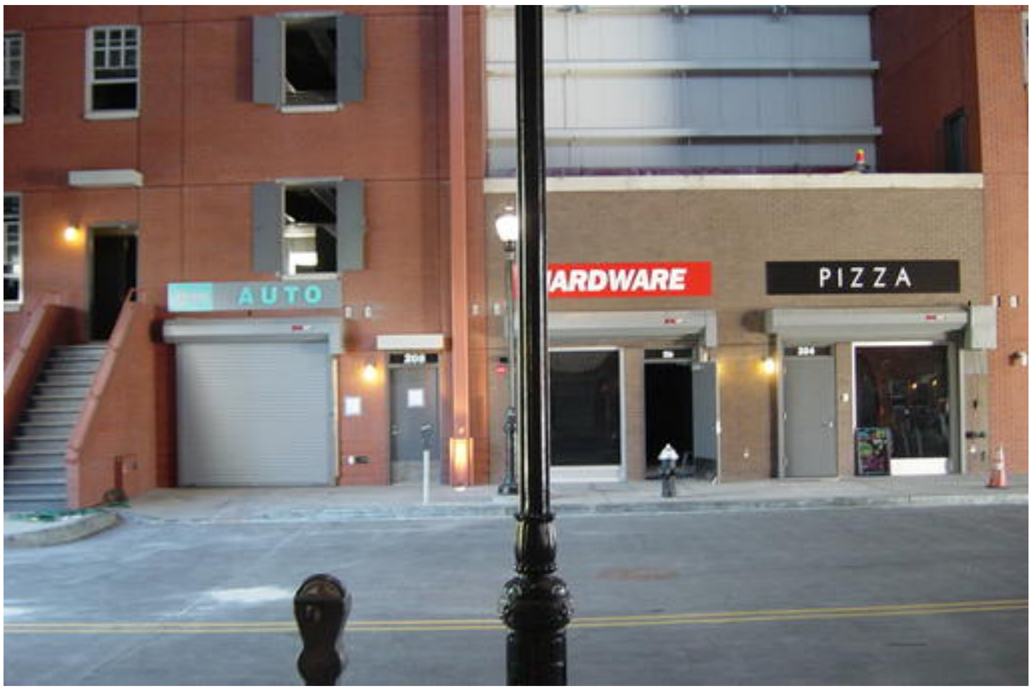


Figure 1. The NYFD Training Facility at Randall's Island

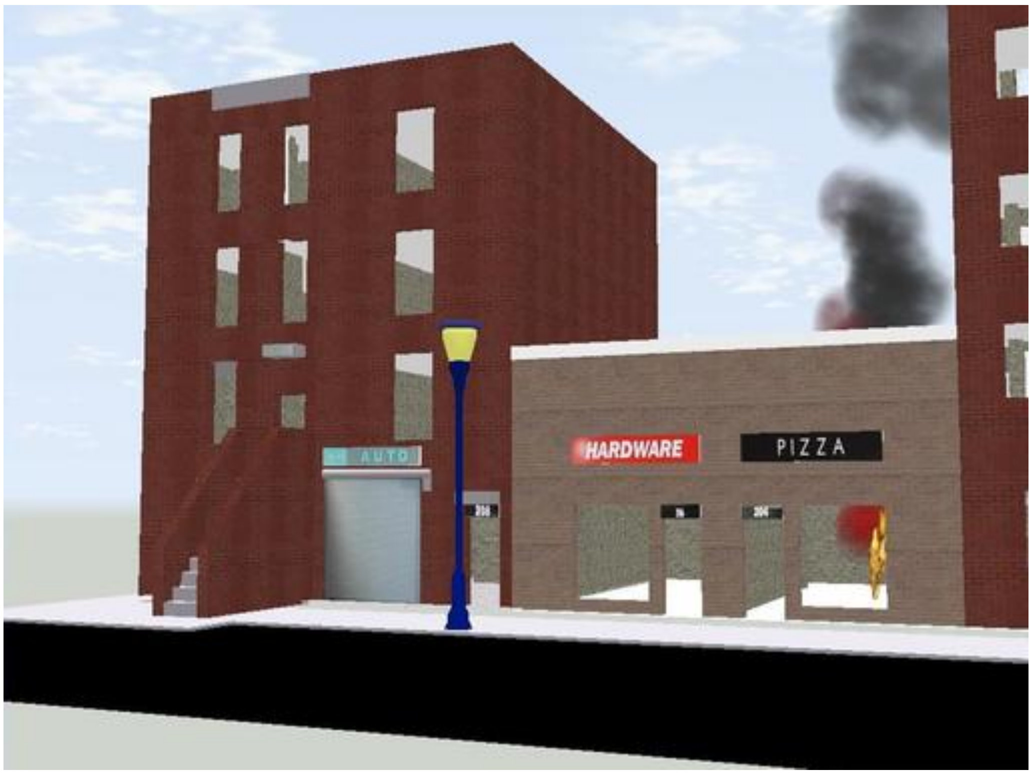


Figure 2. Model of the RI Facility, Kitchen Fire in a Pizza Shop

The NYFD's architectural contractor provided architectural plans for the buildings in the form of CAD drawings. The next step was to model the buildings from plans using Multigen Paradigm's Creator package. Lastly, the models were completed by texture mapping the models using the digital photos. This process took about four weeks.

This methodology can be applied to any architecture, from buildings to vehicles. Current plans for this technology include creating models of targets of strategic importance for use in advanced training of first responders. In addition, NUWC plans to apply this technology to the testing and evaluation of function of command and control centers for future naval architectures (Figure 3).

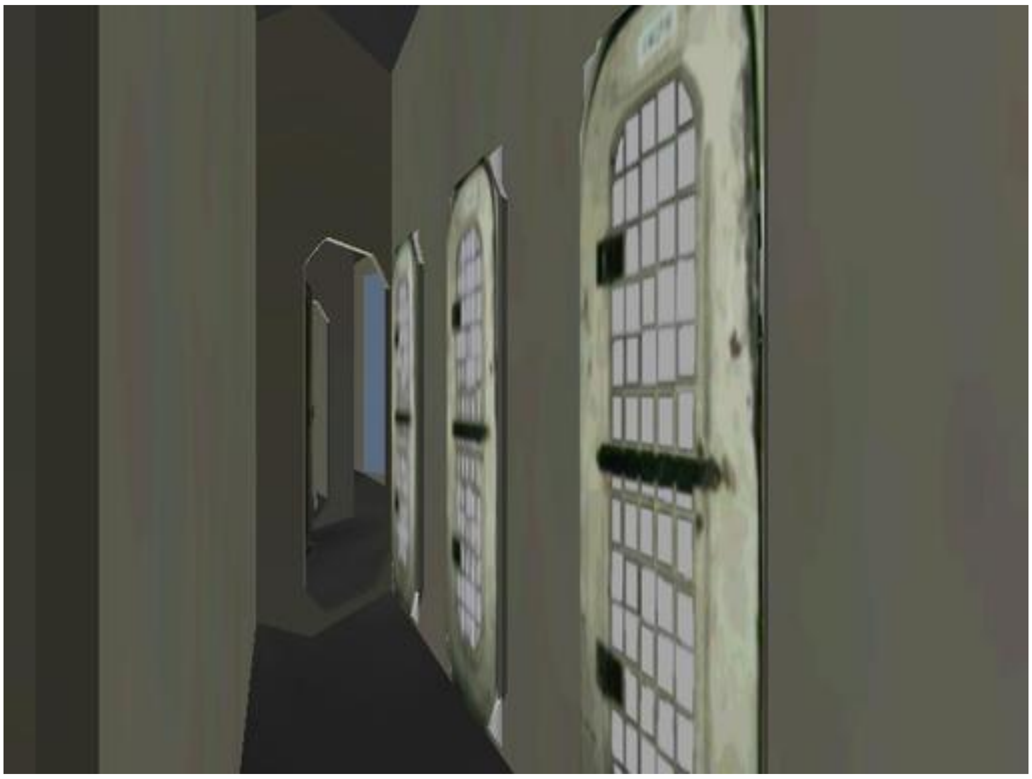


Figure 3. A corridor aboard ex-USS *Shadwell*, a decommissioned US Navy ship used as a damage control research facility.

Software

The underlying software used to control the environment was a modification of work done for my thesis research while working at the Naval Research Laboratory. With the help of Rob King, we created the gestural interface needed to navigate through a true 3-D synthetic environment. Using this methodology, the navigation is accomplished by pointing in the direction the user wants to travel and pressing a button to move. The navigational algorithm is accelerative, which means the longer the user depresses the forward button, the faster the user moves through the environment.

The graphics were handled by the SGI Performer 3.0.1 scene graph for Linux. Performer is a well-established scene graph based on the OpenGL graphics libraries, and it offers Linux users solid performance. It was important to retain a wide variety of display options, because we wanted to be able to deploy a prototype system that could be used in both stereoscopic and monoscopic modes, as well as on multiple display devices.

Hardware

The software is demonstrated using a modified Dell Precision 340. The system is an Intel Pentium 4 processor running at 1.8GHz, with 512MB of RAM and three video cards. The video subsystem includes one NVIDIA NV20 (GeForce 3/64Mb) AGP video card and two NVIDIA NV17 (GeForce 4 MX 440/64Mb) PCI video cards. The system is running a stock installation of Red Hat 9.

The software also is demonstrated on a Sony VAIO laptop powered by an Intel Pentium 4 processor running at 2.6GHz, with 512MB of RAM and an ATI RADEON IGP card. The purpose of this platform is to demonstrate the flexibility of the graphics options.

In addition to the graphics and displays, we used a Polhemus magnetic tracking system in the gestural interface as an alternative to a mouse interface. A Logitech Wingman joystick was re-engineered for use as a gestural input device. Figure 4 shows the inside of the joystick. A tracking sensor is embedded in the grip of the joystick. Communication with the computer is achieved by remapping standard serial mice and wiring them to the buttons within the joystick. The computer interprets the joystick button presses as mouse clicks.

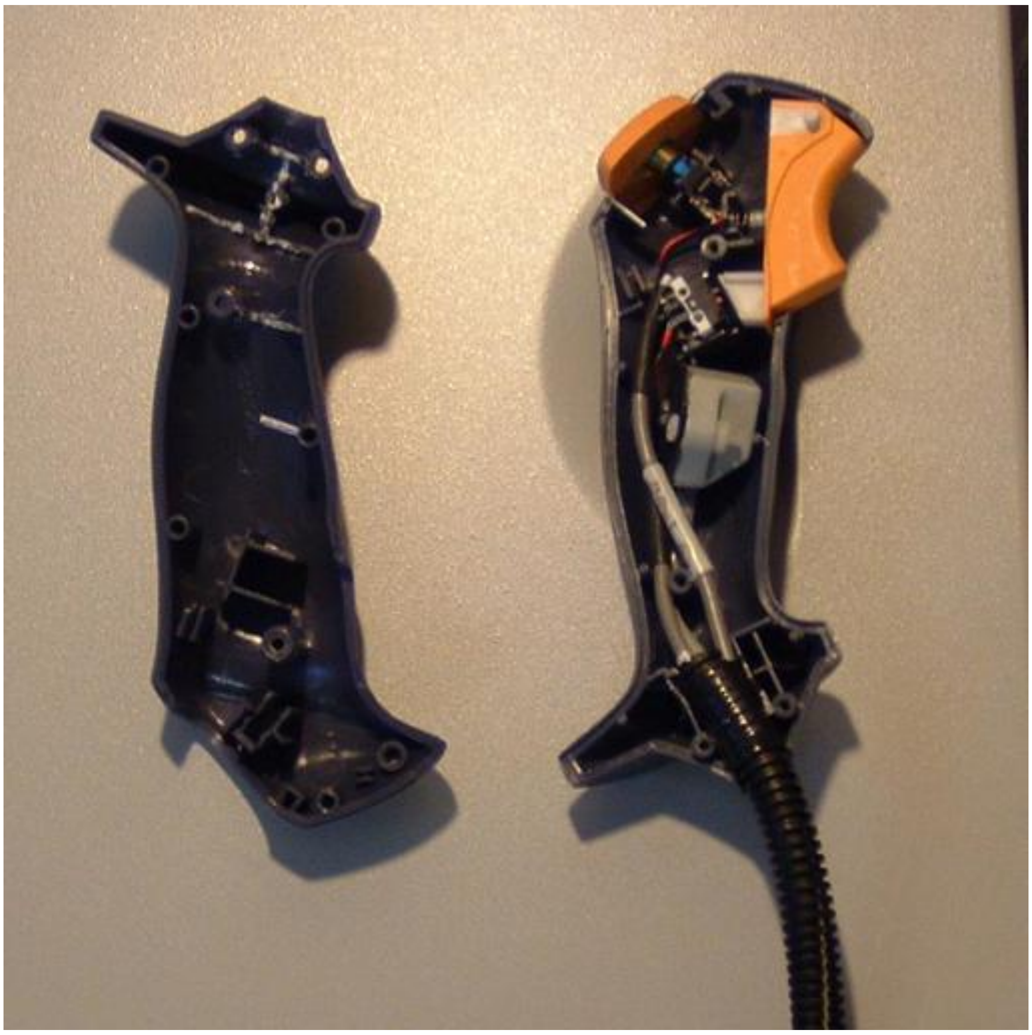


Figure 4. A Logitech Wingman Joystick with Tracking Sensor Added

Prototype

As testing progresses, we hope these visualization and simulation technologies can help meet ever-expanding training needs for both military and civilian emergency response teams. Applications for this limited prototype include pretraining practice runs for rescue workers and navigational training for workers unfamiliar with the environment.

Future plans and upgrades to this software are a bit more ambitious. One use for the system is as a scenario-driven classroom/firehouse trainer. It will include instructor-steered or preconfigured scenarios in which the trainer reacts to trainee/student input and adjusts the scenario accordingly. This will be supported by physics-based modeling of fire/smoke/heat and so on. As with the prototype, future iterations of this software will be available in various display configurations, from large/multiple-screen classrooms to PCs and laptops.

In addition to the scenario-driven training, this software also is planned to be a post-exercise debriefing tool. The location of trainees in a training environment, such as Randall's Island or a burn building, can be tracked at any time during the training exercise. Trainers could replay the training event, showing participants' locations at given times. Planned features include split-screen displays that could show participants' individual viewpoints during the exercise as well as the bird's-eye view.

Acknowledgements

The author would like to thank Mr Jim Pollock of the Naval Undersea Warfare Center for funding this project, Dr Stephan Hitman of the NYFD for providing logistical support and resources and Dr Larry Rosenblum of the Naval Research Laboratory for permission to use the ex-USS *Shadwell* model.

Resources for this article: /article/7499.

Douglas Maxwell is a mechanical engineer and research scientist at the Naval Undersea Warfare Center. His areas of expertise include design synthesis in virtual environments and synthetic training applications. He lives with his wife and dachshund in Newport, Rhode Island.

Archive Index Issue Table of Contents

Advanced search

Hacking Democracy

Doc Searls

Issue #122, June 2004

Two US presidential campaigns made a big stir by using the Internet as a two-way medium. The innovators got clobbered by Big Television this time, but the software is open source and ready for the next round.

In July 2002, Lawrence Lessig gave a speech that challenged technologists to become politically active, to take up the fight against forces determined to replace the Net's free and open commons with a plumbing system for content, valved at every juncture by mechanisms made to manage the digital rights of industrial producers. He didn't pull punches:

> Now, I've spent two years talking to you. To us. About this. And we've not done anything yet. A lot of energy building sites and blogs and Slashdot stories. [But] nothing yet to change that vision in Washington. Because we hate Washington, right? Who would waste his time in Washington?
>
> But if you don't do something now, this freedom that you built, that you spend your life coding, this freedom will be taken away. Either by those who see you as a threat, who then invoke the system of law we call patents, or by those who take advantage of the extraordinary expansion of control that the law of copyright now gives them over innovation. Either of these two changes through law will produce a world where your freedom has been taken away. And, if you can't fight for your freedom, you don't deserve it.
>
> But you've done nothing.

Larry was right. Against the RIAA, the MPAA, the big publishing and broadcasting lobbies and Congress itself, the good guys were being trounced, repeatedly.

So let's mark that point in time and fast-forward to December 2003, 17 months later. The presidential primary playoffs are about to begin, and already a Democratic frontrunner has not been chosen on the strength of his appeal but through the Internet and a variety of tools that run on the Net.

In an interview with Christopher Lydon, Larry said this:

> We're just at the moment when people realize that culture is not something that has to be fed to them, like the Soviet citizens at the end of the Soviet empire, where they realize that they can participate in the construction and sharing of culture. Technology has given us that opportunity. And the problem now is that the law takes that away. And so Creative Commons' objective is to find a way to get the law out of the way, so this extraordinary potential for human creativity can be realized in the context of this technology....What we want to do is make it easy for people to recognize the free culture that is out there for them to build upon, so that they'll build on that culture.

The “we” in this case isn't only Creative Commons. It's something new yet familiar: the free culture movement. “Just as Richard Stallman gave birth to the free software movement”, Larry said, “I think it's fair to say we're the free software movement for culture.” And, much like Richard, Larry is quick to make distinctions:

> There is an important difference between the free software movement and the open-source software movement, in that the free software movement's first goal is freedom. It's not promising better software. It's not promising a better business model. It's promising freedom. And I think that's what the free culture movement is about. It's about giving people the freedom to build and cultivate their culture.

As with free software, tools matter. “One of the most important examples” of free culture tools, Larry says, is the Weblog:

> Free culture is about the transformation between a broadcast culture and a procreative culture—from a broadcast culture where the few speak to the many to a procreative culture where the many speak to the many. That's what the Internet is supposed to have been about forever. But blogging is the first time that it happens in the context of political ideas that get translated and expanded upon as other people comment on them. In the context of political campaigns...they become better citizens. They become engaged citizens. There has been no new technology in the last 150 years that has produced more engaged citizens.

Larry said all this in late 2003, when Howard Dean had emerged as the leading Democratic candidate for president. During the 2003 calendar year, the Net-centric Dean campaign started from nowhere, raised record sums of money, involved record numbers of people and made its candidate a frontrunner in the polls as well as the purse. When it was over, and John Kerry ran away with the Democratic party nomination, the mainstream press predictably compared the Dean campaign to the dot-com bubble. Joe Trippi, Dean's campaign manager, voiced what everybody who truly watched the campaign or participated in it knew intimately:

> This was not a dot-com crash. The Howard Dean campaign was a dot-com miracle. Let's look at this thing. This guy starts...on January 31 of last year with seven people, $157,000 in the bank, 432 known supporters nationwide....He was an asterisk....How did it happen? It is a miracle that Howard Dean moved from there to $45 million, more money than any Democrat in history has raised....He didn't do it. I didn't do it. You did it.

Politics, Trippi said, was no longer something mediated by the media, no longer a horse race run and covered exclusively by professionals. It was, Trippi said, the end of an era that began with the televised Kennedy-Nixon debate in 1960:

> It took about five to ten years to realize that was the moment when television was going to change everything in America's politics. What no one could've predicted was that it would have become a race for money, a race to buy a one-way communications tool that would take the American people essentially out of the process. It was no longer about average Americans, it was about, “How do I find a rich guy to write me a $2,000 check and then how do I take that money and buy television with it?”

Joe Trippi said all this on February 11, 2004, in his keynote address to the Digital Democracy Teach-In, an event that opened O'Reilly's Emerging Technology Conference. That event was an idea I suggested to the O'Reilly folks at the end of the company's Open Source Convention in July 2003. It also was my idea to invite Joe Trippi to keynote the thing. That idea came to me while Britt Blaser, founder of xpertweb.com and an energetic Dean volunteer, was giving me a tour of the Dean Campaign headquarters in Burlington, Vermont. I was present in purely electronic form. My face was on Britt's laptop, my voice was on his laptop speakers and my eye was a camera mounted on the laptop lid. My body was in California. It was in this disembodied form that I met Joe. Walking around holding the laptop like an hors d'oeuvre tray at a party, Britt ran into Joe in the hallway. After saying hi, I asked Joe to keynote the February event. To my astonishment, he said yes.

A Long Strange Trip

My own involvement in this story, however, began more than a year earlier, on June 26, 2002, at the New York launch of eThePeople.org, a “public forum for a new democracy conversation”. I was the featured speaker, recruited for having coined the phrase “markets are conversations,” which was the opening thesis of *The Cluetrain Manifesto* and an inspiration for eThePeople's statement of purpose. Here's how that goes:

> Democracy is a conversation. It's a dialogue that includes politicians, interest groups, parties, journalists, lobbyists, pollsters, letter-writers, campaign contributors, protesters and voters. It's an ongoing negotiation about how our priorities and our values should translate into public policy.

eThePeople built their site on Linux. So did Scott Heiferman, founder of MeetUp, which was launched just 12 days earlier. Scott was there for the party and to let me know about MeetUp's Linux foundations. MeetUp went on to become the most important commercial accessory to democracy since the tavern. More than 165,000 people joined MeetUps for Howard Dean alone.

Britt called me so often from Dean Campaign headquarters in Vermont that I felt like I personally knew everybody there. Some of them knew me first, however, because the campaign consciously borrowed what *The Cluetrain Manifesto* said about networked markets and applied it to voters. *Cluetrain* said, “Markets are getting smarter—and getting smarter faster than most companies.” The Dean campaign said, “Networked citizens get smarter faster than most campaigns.”

More than 600,000 people joined the campaign, and an untold percentage of them were out in the world, stumping hard for the candidate, producing materials, holding meetings and rallies—with less direct guidance from the top than perhaps any campaign in the country's history. More than one volunteer told me the campaign was less a bandwagon than a runaway train.

I got a surprise ride on that train last summer when I wrote a piece called “Saving the Net” for the *Linux Journal* Web site. It went up on Monday, July 21. Two days later, on BlogForAmerica, the Dean Campaign Weblog, Matt Gross wrote a post titled “Saving the Net, and Politics, According to Doc Searls, or Bringing out the Bat”. The post was a challenge to raise more money than Vice President Dick Cheney was scheduled to raise at a $2,000/plate dinner the next Monday in South Carolina. The punch paragraph:

> We're thinking about bringing out the bat on Monday the 28th. We're thinking that you could demonstrate to

> Dick Cheney that there is another way to raise $250,000 in one day—through the people who own this democracy, rather than the special interests that buy it. What do you think? Should we bring out the bat? What should we call the event?

Comments, as always, ran in the hundreds. By Saturday morning they had $82,260.28, and one hour before Monday, they passed $250,000. The final tally (although I'm told it continued to go higher): $508,540.31 contributed by 9,601 people. That's an average of $52.97 apiece.

As Britt and others explained later, the people turning up the steam during this whole run-up weren't the campaign workers posting progress reports on the blog. It was the people down in the blog's comments section. Opening up comments on the campaign blog was like sinking an artesian well into the pockets of hundreds of thousands of supporters.

My visit to the campaign came right before the Iowa Caucuses in January 2004. The energy at headquarters was at maximum, and everything seemed to be on track for a series of wins in the primaries. The day after I returned to New York (for LinuxWorld Expo), Dean lost the Iowa Caucuses, coming in third. That night he uttered his famous “scream”, which ABC News admits playing a total of “700 times in a few days”. The rest is history.

Or maybe not. Oddly, DeanForAmerica.com comments kept rolling on. There were expressions of sympathy for the bombings in Madrid, suggestions for Kerry cabinet members and practical ideas to continue applying the energy that persisted in the campaign, even while few outside the campaign seemed aware of it. Some of the posts were by Howard Dean himself—something that rarely happened while the candidate's presidential hopes still were alive.

Adaptation Nation

One of my regrets, while working on this story, was not getting to Arkansas to see what Cameron Barrett, Tony Steidler-Dennison and other hackers were doing for the Wesley Clark campaign. Cam is one of the original bloggers, and Tony is a friend of ours at *Linux Journal* and a frequent contributor to the magazine.

Cam and Tony put the whole Clark campaign on an open-source footing, crafting a bunch of tools that now have returned to the ecosystem, along with their authors. After the Clark campaign folded, Cam went to work for the Kerry campaign. Tony went back to playing the role of interested observer. When I asked Tony what happened to the tools his team developed for the Clark campaign, he replied:

> The tools themselves provided a means for the Clark campaign, in particular, to get up to speed in a very short time. We wouldn't have been able to provide the interconnected set of supporter tools (Clark Community Network→eBlocks→on-line contributions→Clark Recruiters→Lawyers for Clark→various and sundry mailing lists, and so on) as quickly with proprietary tools. Or as cost-effectively.
>
> These were the first campaigns to use the Internet in a truly *two-way* fashion. Cam's Clark Community Network was, I believe, the most effective tool for supporter community-building of any of the campaigns. The 2000 campaign used the Internet to convey a message from the candidate to both actual and potential supporters. It was a one-way communication. The Dean and Clark campaigns were the first and most effective at providing a connected voice for supporters to talk back to the candidates and to talk with one another. In other words, the campaign communities made the campaigns more than merely vehicles for fund raising. I can tell you firsthand that the Clark campaign paid attention to the collective voice of the community. So, open-source tools allowed us to create a community that actually had a voice in the campaign.
>
> The most likely legacy of the Clark campaign is Clark TechCorps. The tools we created to organize the supporter communities are available under open-source licensing for anyone to use and improve upon. There's still a fair amount of activity on the TechCorps site as interested developers continue to work on those tools. We'll have to see where they go.

Britt Blaser provides a nice summary of the first well-hacked party presidential playoff season:

> Many spent 2003 hacking code because they thought it might transform politics, and they were more right than wrong. They are the open-source entrepreneurs of the governance tools space. Like all entrepreneurs, they are artists who create because they're incapable of not creating.

Joe Trippi called MeetUp the Dean Campaign's “killer app”. “We built a hammer”, says Scott Heiferman, “and they built a house with it”.

Zephyr Teachout, whose visibility as a Dean campaign worker was exceeded only by Joe Trippi, says MeetUp made such a good tool because what it built wasn't on-line. MeetUps happen in meat space—in physical, geographical reality. Off-Net. Scott Heiferman tells me MeetUp's most important corporate relationships are with the local restaurants and coffee shops where MeetUps meet.

So MeetUp succeeded in part because it violated at least one Web development maxim: it didn't over-provision its feature set or let its ambitions fall out of alignment with its core services. And it kept those services simple, in the tradition of great practical open-source applications.

Consider the Source

Joe Trippi's long career path included a tour of duty with Progeny, working with Ian Murdock, the co-originator of Debian. It's not coincidental that Joe called Dean's an “open-source campaign” and said “It's like Linux. The more people collaborate, the more likely we'll build a better thing.”

Phil Windley, former CIO of Utah, says we're coming to the end of an era in which governance is crippled by two waves of opposing ideologies and ideologues: anti-business lefties in the '60s and '70s and anti-government righties in the '80s and '90s. “Both groups failed to understand that most citizens don't subscribe to an ideology. In fact, all most people really want from government is to get the roads fixed.” What's exciting now, he says, is the potential for involvement and participation by ordinary citizens in the mundane machinery of governance. Already, he says, countless NGOs (non-governmental organizations) are bringing solutions to government rather than waiting for the reverse to happen.

I had a chance to witness this first-hand when a friend asked me to sit in on a meeting between a local group and county supervisors. The local group had some ideas for improving bandwidth in the county, and this meeting was the one where they were presenting concrete recommendations. To my surprise, I found myself participating in a productive conversation with elected officials about an important local issue—a new experience for me.

The friend who invited me to the meeting doesn't work in government, but he is an astute observer of government and how it works. What he told me after the meeting blew my mind, because it had never occurred to me before, was:

> Government isn't the problem. People need to bring solutions to government. Government is dying for answers. Bring some and you'll get somewhere.
>
> I don't have experience with the government stonewalling me at all. I experience interest and cooperation at every level, as long as I bring solutions and not just problems.
>
> A lot of helpless people want government to solve their problems or to carry their spear on one issue or another. That reflects an ignorance of how the whole ecosystem actually works. If you're constructive, you can participate in that ecosystem. Bureaucrats are

> crying for help on all kinds of issues. If we provide some, we can make stuff get done together.

“Democracy is the first open-source application”, Phil Windley says. More to the point, he considers open-source values such as code exposure, peer review, individual initiative and iterated inclusion of improved code in goods that are never quite finished essential to functional democracy in the long run. In fact, he believes open-source values and practices will help democracy finally deliver on ideals that have remained unfulfilled for thousands of years.

In other words, we may finally have the kind of democracy we've always idealized—governments not only representative of their citizens but open to participation by everybody with something to contribute.

Three interesting facts about that meeting with the county supervisors:

1. All the guests in the meeting were technologists. (I was the least technical of the bunch.)
2. All the public servants in the meeting (supervisors and various staffers) believed that improving Internet bandwidth was a no-brainer issue.
3. It was only natural for government to look to technologists for help.

Here at the local and regional levels of government, where the density of lobbyists and other pressure groups is lower than in Washington, pro-Internet technologists are starting to find themselves in a sellers' market for their expertise and solutions. At the very least, we're seeing some interfaces open up. Maybe we'll see quite a few more soon. According to Zephyr Teachout, over a hundred candidates now running for public office were inspired by the Dean campaign. “All the significant trends start with technologists”, Marc Andreessen says.

Those trends don't have to be technical. Clearly, they also can be political.

Perhaps now we won't get merely the government we deserve. Maybe now we'll get the government we hack.

Read more from Internet democracy innovators in “Voices from the New Hackers of Democracy” on the *Linux Journal* Web site: www.linuxjournal.com/article/7474.

Doc Searls is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

An Open-Source System for Electronic Court Filing

Jim Beard

Issue #122, June 2004

State and federal courts in the US are standardizing on XML for court documents and on open-source management systems to store and index them.

Over the past several years, court technology has gone through many changes. Most notable is the drive to facilitate the electronic filing of documents by using open-source solutions. Many people have led separate projects that attempted to provide a blueprint for proper electronic integration with court case management systems (CMSes). To date, large-scale success has not been demonstrated in the legal electronic filing industry, although many small projects have shown promise. All of this may be about to change, however, as the federal and state court systems start to embrace and support open-source software.

Each year, approximately 90 million cases are filed in the 17,500 courts in the United States. These cases generate more than 1.5 billion documents. Aside from the environmental impact, this creates a mountain of paperwork for court staffs. In addition, storage and indexing costs associated with archiving and retrieving documents are high. Many courts are starting to look toward integrated electronic systems to cut down on the quantity of paperwork and streamline day-to-day operations. Most courts now use some form of computer-based case management system or document management system (DMS).

The idea behind electronic court filing is to allow lawyers and the general public to submit documents to the court over the Internet. Law firms already generate an electronic version of any document they plan to file with a court. When filed electronically, the document automatically is placed in a DMS, and the pertinent case information is placed in a CMS.

The current accepted model of how filings should be generated and passed to a court involves three major components. The first is an electronic filing service provider (EFSP), the organization responsible for constructing an electronic legal filing. There could be many of these organizations in any given legal jurisdiction, all competing for a per-filing fee paid by lawyers and pro se litigants (people who represent themselves in court) who use their service. The second component is an electronic filing manager (EFM). The EFM has a one-to-one relationship with a court. All EFSPs that offer the ability to file documents with a specific court need to communicate with that court's EFM. Court personnel use the EFM to review filed documents to ensure that any court-specific rules have been followed. The final component is an adapter that communicates between the EFM and the court's CMS and DMS systems. This adapter relays the documents and case information associated with the filing to the court's information systems. Figure 1 shows the organization of this model.

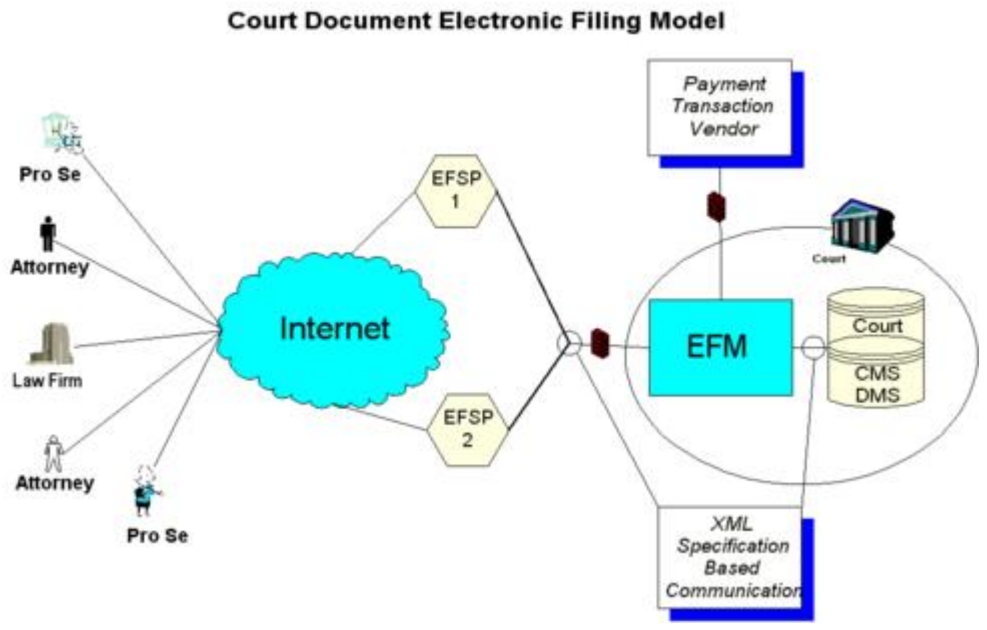

Court Document Electronic Filing Model
Pro Se
Attorney
Law Firm
Attorney
Pro Se
Internet
EFSP 1
EFSP 2
Payment Transaction Vendor
Court
EFM
Court
CMS DMS
XML Specification Based Communication

Figure 1. Transferring and Submitting Filings

A major hurdle to providing this type of functionality is that almost every CMS used by courts today has been customized on some level to fit specific needs of individual courts. Many courts even have developed their own custom solutions in-house. These differences require the EFM to CMS and DMS adapters to be rewritten for every court. In response to this problem, many court professionals have pushed for an industry-endorsed standard for transmitting data between legal systems. Winchel “Todd” Vincent III, founder of Legal XML, stated: “Having an agreed-upon specification to use can greatly reduce the amount of work required to provide complicated system integration. It allows for all involved parties to work on the problem together, instead of reinventing the wheel over and over again.” In addition to an industry standard, there also is a drive to provide an open-source EFM.

Legal XML Is Founded

In November 1998, Legal XML was established as a nonprofit organization. The goal was to provide open, nonproprietary technical specifications for exchanging legal documents and to organize the related information. It attracted participants from private industry, other nonprofit organizations, government and academia. Legal XML originally produced a DTD-based specification, which eventually was used to help courts exchange information by way of XML.

Georgia Interoperability Project

The completeness of the Legal XML 1.0 specification was put to the test when the Georgia Court Automation Committee (GCAC) used it during an interoperability pilot project. This project acted as a proof of concept for the electronic filing approach previously described. The first standards-based electronic filings to occur in the nation were a result of this project. Filings were generated and received through systems run by different software providers, proving the feasibility of interoperability based on open standards. Both federal agencies and state courts actively followed the project.

The interoperability phase of this project consisted of three EFM installations. Two of the EFMs facilitated filings with county superior courts; the other EFM interfaced with a county state court. Three providers managed the individual EFMs. Four companies provided EFSP interfaces that were used successfully to file with all three EFMs. Between January 2003 and January 2004, more than 1,000 filings were processed electronically. The project was considered a success. Since its completion, several law enforcement agencies in Georgia now file electronically on a daily basis.

After the Georgia Interoperability project, integration challenges led to talk of an open-source solution, and the topic started to spread among the electronic filing community. The National Center for State Courts (NCSC) led the way and was one of open source's bigger proponents, releasing an open-source EFM, inCounter, using the standard Linux, Apache, MySQL, Perl/Python/PHP (LAMP) approach. inCounter was designed to be a demonstrative application to showcase electronic filing capability and to promote an open-source solution. counterclaim, Inc., the company this author works with, concurrently released OpenEFM, a 100% Java application.

Why Open Source Is a Good Solution

Open-source software is a good technical solution to the electronic filing problem. In many open-source projects, forming an active development and support community is often difficult. However, a large number of developers currently are spread throughout courthouses across the United States. This loose-knit community can be organized within the US judicial system. These IT staffers share the same challenge of solving similar electronic filing problems. Because courts are noncompeting entities, the standard problems regarding intellectual property should evaporate in favor of collaborative community solutions.

Open source also provides a good economic solution to problems related to courthouse operational costs. Many courts are strapped for cash. Purchasing industry-specific proprietary solutions from large corporations often can be very costly. Although open-source software does not mean free software, it reduces up-front costs. Courts still incur costs associated with installing software, teaching people how to use it and customizing the open-source solution to fit their unique needs. Also, as with any software solution, ongoing maintenance is needed. Today, many large companies, in addition to selling software, also act as consultants to courts, charging large sums of money for services extolling proprietary solutions. Open-source solutions can cut these costs.

Open source is also a great benefit for court personnel, allowing them to evaluate the feasibility of a solution before making a commitment. Nothing prevents an IT technician in any court from downloading the open-source EFM code and playing with it to determine whether the software meets the court's needs. On the other hand, proprietary software often is not available for a complete evaluation until the court has committed to a contract. This can lead to problematic situations and can leave the court with little control over timelines and final costs.

NCSC Conference Exposure

The 2002 biennial eCourt Conference, hosted by NCSC, was a watershed moment, as open-source solutions were spotlighted. During the conference, counterclaim's OpenEFM and the NCSC's inCounter were demonstrated and exposed to the larger legal community. Hundreds of decision-makers from both federal and state courts all across the country, along with foreign representatives, witnessed the power of the open-source model. The information presented led to spirited discussions regarding the virtues of open source within the legal industry.

Legal XML Moves to OASIS

About the time of the beginning of the Georgia Interoperability pilot project, the Legal XML organization went through several changes. As with any large project, internal organizational issues existed. These issues resulted in the decision by the founder of Legal XML to leave the group. After this event, it was proposed that the Legal XML group move under the umbrella of the Organization for the Advancement of Structured Information Standards (OASIS), a not-for-profit global consortium dedicated to promoting development, implementation and acceptance of electronic business standards. The OASIS Legal XML Member Section is made up of representatives from both commercial software companies and governmental court organizations. The founder of Legal XML, Winchel “Todd” Vincent III, and Legal XML itself, both continue to provide valuable and significant contributions in the area of electronic court filing.

California Second-Generation Electronic Filing

In October 2002, the California Administrative Office of the Courts (AOC) decided it was time to define a standard for electronic filing and data exchange for California courts. This project is called the Second-Generation Electronic Filing Specification (2GEFS). Vincent spearheads this effort, together with a group of participating software companies and selected California courts. Over the course of several months, specifications were produced by a group of knowledgeable participants. Recently, the draft XML Schema-based specifications have been made available publicly for comment by the state of California.

A unique technical aspect that the 2GEFS specifications provides is the concept of a well-defined Schema framework. This notion came from the recognition that courts may need to modify data elements slightly to fit their specific needs. An example of this might be the way a person is defined by the specification. One court may need to include such information as a list of identifiable scars, tattoos and birthmarks, whereas another court may not care to capture this information. The Schema framework approach creates a decentralized framework of strictly versioned schemas. Each schema lives in its own namespace and can be published on multiple servers. This ensures that all past versions of Schemas are available, and XML instance documents representing filings can use older Schemas to validate against as needed. This allows courts to use slightly customized Schemas and put them in their own namespace under their own control.

These specifications also are out for public comment, and several companies are working on incorporating them into court filing software. The specifications are covered under a modified General Public License. Christopher Smith, the current overseer of the 2GEFS Project, had the following to say about its success: “We're very pleased with the specifications we have produced. Aside from being easy to understand and well documented, we feel that they can be a tremendous contribution to the electronic court filing industry. We are encouraging all interested parties to see what California has to offer.”

Proof-of-concept implementations of the 2GEFS specifications have been created, and an interoperability-testing phase is scheduled to last until the middle of 2004. Throughout this project, the state of California is working with a number of local courts to integrate their electronic filing systems.

The Open XML Court Interface

The Open XML Court Interface (OXCI) initiative was started in March 2000, but it finally got off the ground in mid-2003. Its goal is to develop an open-source version of an EFM. Eleven states have sponsored the initial development and actively take part in the OXCI consortium. The project is being led by the AOCs of Georgia and Washington State. The legal community across the country is anxiously watching the OXCI Project. In January 2004, OXCI was ready to fund development.

OpenEFM, the open-source EFM supported by counterclaim, was used as a point of reference many times by the OXCI team as they worked on an outline for the project requirements and architectural design. After submitting a response to the AOC of Georgia's Request for Proposals regarding the OXCI EFM Project, counterclaim was selected to complete the project development and to provide continuing support. The development of the OXCI EFM is slated to be complete in June 2004, after which three courts will be configured to use the new, open-source system. Once success has been demonstrated, it is likely that more courts across the US will adopt the OXCI EFM.

Vince Harris, leader of this project for the AOC of Georgia, had the following to say about OXCI: “It has taken a lot of work and effort by many different people to get us to this point, but we are excited to be here. With the amount of backing and support the OXCI Project has received, we have a great opportunity to help solve the electronic filing problem. Since the EFM will be available at no cost to the courts, we also feel that our solution has the potential to be very widespread.”

Current State

At some future point, all courts will offer some form of electronic filing. In fact, many jurisdictions have passed legislation mandating such services be made available. By educating the legal community about available open-source solutions, there is a great chance that the court community could pool its resources and come together to adopt a solution that addresses its common needs. OXCI is working to provide that open-source solution. The OASIS Legal XML Member Section is in a great position to pick and choose the best features from various specification attempts in order to provide an open industry-endorsed standard to go along with the open-source EFM. With any luck, and a whole lot of elbow grease, an open-source approach may become the standard solution endorsed by our judicial system as the best method to solve the electronic filing problem.

Resources for this article: /article/7492.

Jim Beard has worked in the court software industry for several years. He has participated in a number of different electronic filing projects and is currently involved in the OXCI and 2GEFS Projects. He can be reached at beard@counterclaim.com.

Archive Index Issue Table of Contents

Advanced search

GNU Radio: Tools for Exploring the Radio Frequency Spectrum

Eric Blossom

Issue #122, June 2004

Bringing the code as close to the antenna as possible is the goal of software radio. GNU Radio gives you the tools to join the communication revolution powered by today's fast processors.

Software radio is the technique of getting code as close to the antenna as possible. It turns radio hardware problems into software problems. The fundamental characteristic of software radio is that software defines the transmitted waveforms, and software demodulates the received waveforms. This is in contrast to most radios in which the processing is done with either analog circuitry or analog circuitry combined with digital chips. GNU Radio is a free software toolkit for building software radios.

Software radio is a revolution in radio design due to its ability to create radios that change on the fly, creating new choices for users. At the baseline, software radios can do pretty much anything a traditional radio can do. The exciting part is the flexibility that software provides you. Instead of a bunch of fixed function gadgets, in the next few years we'll see a move to universal communication devices. Imagine a device that can morph into a cell phone and get you connectivity using GPRS, 802.11 Wi-Fi, 802.16 WiMax, a satellite hookup or the emerging standard of the day. You could determine your location using GPS, GLONASS or both.

Perhaps most exciting of all is the potential to build decentralized communication systems. If you look at today's systems, the vast majority are infrastructure-based. Broadcast radio and TV provide a one-way channel, are tightly regulated and the content is controlled by a handful of organizations. Cell phones are a great convenience, but the features your phone supports are determined by the operator's interests, not yours.

A centralized system limits the rate of innovation. We could take some lessons from the Internet and push the smarts out to the edges. Instead of cell phones being second-class citizens, usable only if infrastructure is in place and limited to the capabilities determined worthwhile by the operator, we could build smarter devices. These user-owned devices would generate the network. They'd create a mesh among themselves, negotiate for backhaul and be free to evolve new solutions, features and applications.

The Block Diagram

Figure 1 shows a typical block diagram for a software radio. To understand the software part of the radio, we first need to understand a bit about the associated hardware. Examining the receive path in Figure 1, we see an antenna, a mysterious RF front end, an analog-to-digital converter (ADC) and a bunch of code. The analog-to-digital converter is the bridge between the physical world of continuous analog signals and the world of discrete digital samples manipulated by software.

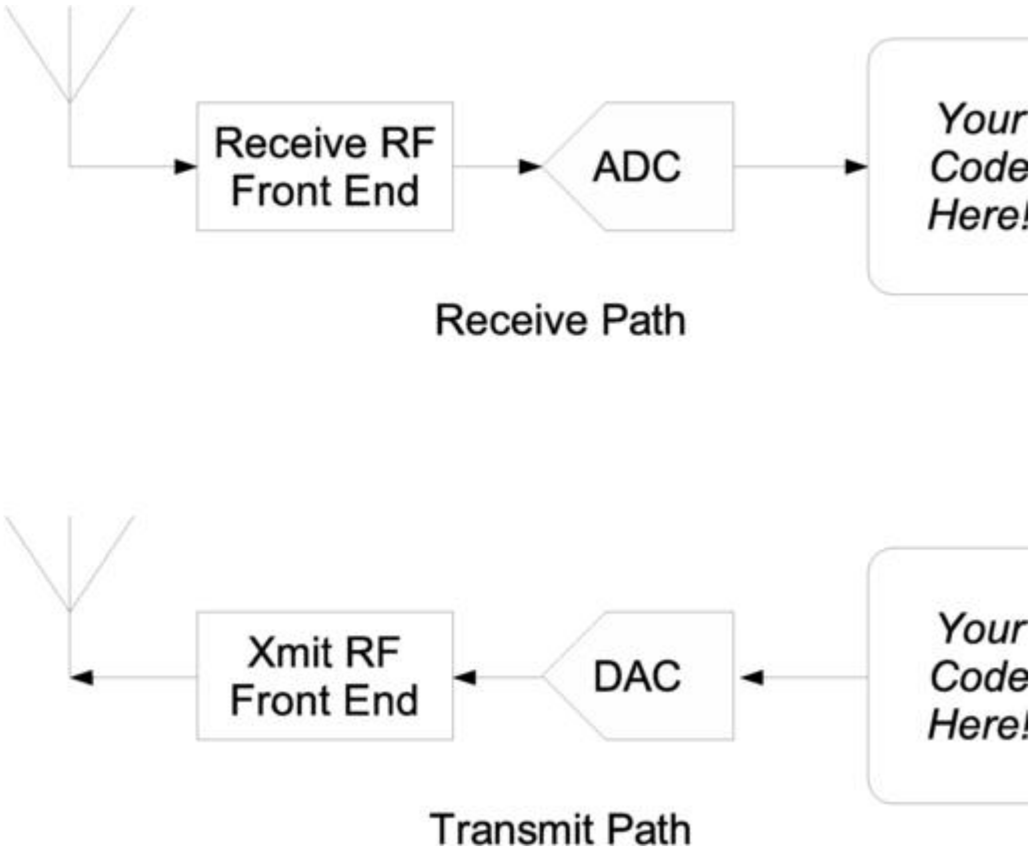


Figure 1. Typical Software Radio Block Diagram

ADCs have two primary characteristics, sampling rate and dynamic range. Sampling rate is the number of times per second that the ADC measures the analog signal. Dynamic range refers to the difference between the smallest and largest signal that can be distinguished; it's a function of the number of bits in the ADC's digital output and the design of the converter. For example, an 8-bit converter at most can represent 256 (2^8) signal levels, while a 16-bit converter represents up to 65,536 levels. Generally speaking, device physics and cost impose trade-offs between the sample rate and dynamic range.

Before we dive into the software, we need to talk about a bit of theory. In 1927, a Swedish-born physicist and electrical engineer named Harry Nyquist determined that to avoid aliasing when converting from analog to digital, the ADC sampling frequency must be at least twice the bandwidth of the signal of interest. Aliasing is what makes the wagon wheels look like they're going backward in the old westerns: the sampling rate of the movie camera is not fast enough to represent the position of the spokes unambiguously.

Assuming we're dealing with low pass signals—signals where the bandwidth of interest goes from 0 to f_{MAX}, the Nyquist criterion states that our sampling frequency needs to be at least 2 * f_{MAX}. But if our ADC runs at 20MHz, how can we listen to broadcast FM radio at 92.1MHz? The answer is the RF front end. The receive RF front end translates a range of frequencies appearing at its input to a lower range at its output. For example, we could imagine an RF front end that translated the signals occurring in the 90–100MHz range down to the 0–10MHz range.

Mostly, we can treat the RF front end as a black box with a single control, the center of the input range that's to be translated. As a concrete example, a cable modem tuner module that we've employed successfully has the following characteristics. It translates a 6MHz chunk of the spectrum centered between about 50MHz and 800MHz down to an output range centered at 5.75MHz. The center frequency of the output range is called the intermediate frequency, or IF.

In the simplest-thing-that-possibly-could-work category, the RF front end may be eliminated altogether. One GNU Radio experimenter has listened to AM and shortwave broadcasts by connecting a 100-foot piece of wire directly to his 20M sample/sec ADC.

On to the Software

GNU Radio provides a library of signal processing primitives and the glue to tie it all together. The programmer builds a radio by creating a graph (as in graph theory) where the nodes are signal processing primitives and the edges represent the data flow between them. The signal processing primitives are implemented in C++. Conceptually, primitives process infinite streams of data flowing from their input ports to their output ports. Primitives' attributes include the number of input and output ports they have as well as the type of data that flows through each. The most frequently used types are short, float and complex.

Some primitives have only output ports or input ports. These serve as data sources and sinks in the graph. There are sources that read from a file or ADC, and sinks that write to a file, digital-to-analog converter (DAC) or graphical display. About 100 primitives come with GNU Radio. Writing new primitives is not difficult.

Graphs can be constructed and run in C++, but the easy way to glue everything together is with Python. Listing 1 is the “Hello World” of GNU Radio. It generates two sine waves and outputs them to the sound card, one on the left channel, one on the right.

Listing 1. Hello World (Dial Tone Output)

```
#!/usr/bin/env python

from GnuRadio import *

def build_graph ():
    sampling_freq = 32000
    ampl = 8192

    fg = gr_FlowGraph ()

    src0 = GrSigSourceS (
        sampling_freq, GR_SIN_WAVE, 350, ampl)

    src1 = GrSigSourceS (
        sampling_freq, GR_SIN_WAVE, 440, ampl)

    sink = GrAudioSinkS ()

    fg.connect (src0, sink)
    fg.connect (src1, sink)

    return fg

if __name__ == '__main__':
    fg = build_graph ()
    fg.start ()        # fork thread(s) and return
    raw_input ('Press Enter to quit: ')
    fg.stop ()
```

We start by creating a flow graph to hold the primitives and connections between them. The two sine waves are generated by the GrSigSourceS calls. The S suffix indicates that the source produces shorts. One sine wave is at 350Hz, and the other is at 440Hz. Together, they sound like the US dial tone.

GrAudioSinkS is a sink that writes its input to the sound card. It takes one or two streams of shorts as its input. We connect the three primitives together using the connect method of the flow graph. Once the graph is built, we start it. Calling start forks one or more threads to run the computation described by the graph and returns control immediately to the caller. In this case, we simply wait for any keystroke.

A Complete FM Receiver

Listing 2 shows a somewhat simplified but complete broadcast FM receiver. It includes control of the RF front end and all required signal processing. This example uses an RF front end built from a cable modem tuner and a 20M sample/sec analog-to-digital converter.

Listing 2. Simple Broadcast FM Receiver

```
#!/usr/bin/env python

# simple broadcast FM receiver

from GnuRadio import *

#
# return a gr_FlowGraph
#
def build_graph (IF_freq):
    input_rate = 20e6

    CFIR_decimate = 125
    RFIR_decimate = 5
    fm_demod_gain = 2200

    quad_rate = input_rate / CFIR_decimate
    audio_rate = quad_rate / RFIR_decimate

    volume = 1.0

    src = GrHighSpeedADCSourceS (input_rate)

    # compute FIR filter taps for channel selection
    channel_coeffs = \
      gr_firdes.low_pass (
        1.0,           # gain
        input_rate,    # sampling rate
        250e3,         # low pass cutoff freq
        8*100e3,       # width of trans. band
        gr_firdes.WIN_HAMMING)

    # input: short; output: complex
    chan_filter = \
      GrFreqXlatingFIRfilterSCF (CFIR_decimate,
                                 channel_coeffs,
                                 IF_freq)
    # input: complex; output: float
    fm_demod = \
      GrQuadratureDemodCF (volume * fm_demod_gain)

    # compute FIR filter taps for audio filter
    width_of_transition_band = audio_rate / 32
    audio_coeffs = \
      gr_firdes.low_pass (
        1.0,            # gain
        quad_rate,      # sampling rate
        audio_rate/2 - width_of_transition_band,
        width_of_transition_band,
        gr_firdes.WIN_HAMMING)

    # input: float; output: short
    audio_filter = \
      GrFIRfilterFSF (RFIR_decimate, audio_coeffs)

    final_sink = GrAudioSinkS ()

    fg = gr_FlowGraph ()

    fg.connect (src, chan_filter)
    fg.connect (chan_filter, fm_demod)
    fg.connect (fm_demod, audio_filter)
```

```
        fg.connect (audio_filter, final_sink)

        return fg

    if __name__ == '__main__':

        # connect to RF front end
        rf_front_end = microtune_eval_board ()
        if not rf_front_end.board_present_p ():
            raise IOError, 'RF front end not found'

        # set gain and radio station frequency
        rf_front_end.set_AGC (300)
        rf_front_end.set_RF_freq (100.1e6)

        IF_freq = rf_front_end.get_output_freq ()
        fg = build_graph (IF_freq)
        fg.start ()        # fork thread(s) and return
        raw_input ('Press Enter to quit: ')
        fg.stop ()
```

Like the Hello World example, we build a graph, connect the primitives together and start it. In this case, our source is the high-speed ADC, GrHighSpeedADC. We follow it with GrFreqXlatingFIRfilterSCF, a finite impulse response (FIR) filter that selects the FM station we're looking for and translates it to baseband (0Hz, DC). With the 20M sample/sec converter and cable modem tuner, we're really grabbing something in the neighborhood of a 6MHz chunk of the spectrum. This single chunk may contain ten or more FM stations, and GrFreqXlatingFIRfilterSCF allows us to select the one we want. In this case, we select the one at the exact center of the IF of the RF front end (5.75MHz). The output of GrFreqXlatingFIRfilterSCF is a stream of complex samples at 160,000 samples/second. We feed the complex baseband signal into GrQuadratureDemodCF, the block that does the actual FM demodulation. GrQuadratureDemodCF works by subtracting the angle of each adjacent complex sample, effectively differentiating the frequency. The output of GrQuadratureDemodCF contains the left-plus-right FM mono audio signal, the stereo pilot tone at 19kHz, the left-minus-right stereo information centered at 38kHz and any other sub-carriers above that. For this simplified receiver, we finish off by low pass filtering and decimating the stream, keeping only the left-plus-right audio information, and send that to the sound card at 32,000 samples/sec. See the GNU Radio Wiki for discussions and tutorials on signal processing.

Graphical User Interfaces

Graphical interfaces for GNU Radio applications are built in Python. Interfaces may be built using any toolkit you can access from Python; we recommend wxPython to maximize cross-platform portability. GNU Radio provides primitives that use interprocess communication to transfer chunks of data from the real-time C++ flow graph to Python-land.

Hardware Requirements

GNU Radio is reasonably hardware-independent. Today's commodity multi-gigahertz, super-scalar CPUs with single-cycle floating-point units mean that serious digital signal processing is possible on the desktop. A 2GHz Pentium or Athlon can evaluate 2 billion floating-point FIR taps/s. We now can build, virtually all in software, communication systems unthinkable only a few years ago.

Your computational requirements depend on what you're trying to do, but generally speaking, a 1 or 2GHz machine with at least 256MB of RAM should suffice. You also need some way to connect the analog world to your computer. Low-cost options include built-in sound cards and audiophile quality 96kHz, 24-bit, add-in cards. With either of these options, you are limited to processing relatively narrow band signals and need to use some kind of narrow-band RF front end.

Another possible solution is an off-the-shelf, high-speed PCI analog-to-digital board. These are available in the 20M sample/sec range, but they are expensive, about the cost of a complete PC. For these high-speed boards, cable modem tuners make reasonable RF front ends.

The Universal Software Radio Peripheral

The preferred solution is the Universal Software Radio Peripheral (USRP). Figure 2 shows the block diagram of the USRP. The brainchild of Matt Ettus, the USRP is an extremely flexible USB device that connects your PC to the RF world. The USRP consists of a small motherboard containing up to four 12-bit 64M sample/sec ADCs, four 14-bit, 128M sample/sec DACs, a million gate-field programmable gate array (FPGA) and a programmable USB 2.0 controller. Each fully populated USRP motherboard supports four daughterboards, two for receive and two for transmit. RF front ends are implemented on the daughterboards. A variety of daughterboards is available to handle different frequency bands. For amateur radio use, low-power daughterboards are available that receive and transmit in the 440MHz band and the 1.24GHz band. A receive-only daughterboard based on a cable modem tuner is available that covers the range from 50MHz to 800MHz. Daughterboards are designed to be easy to prototype by hand in order to facilitate experimentation.

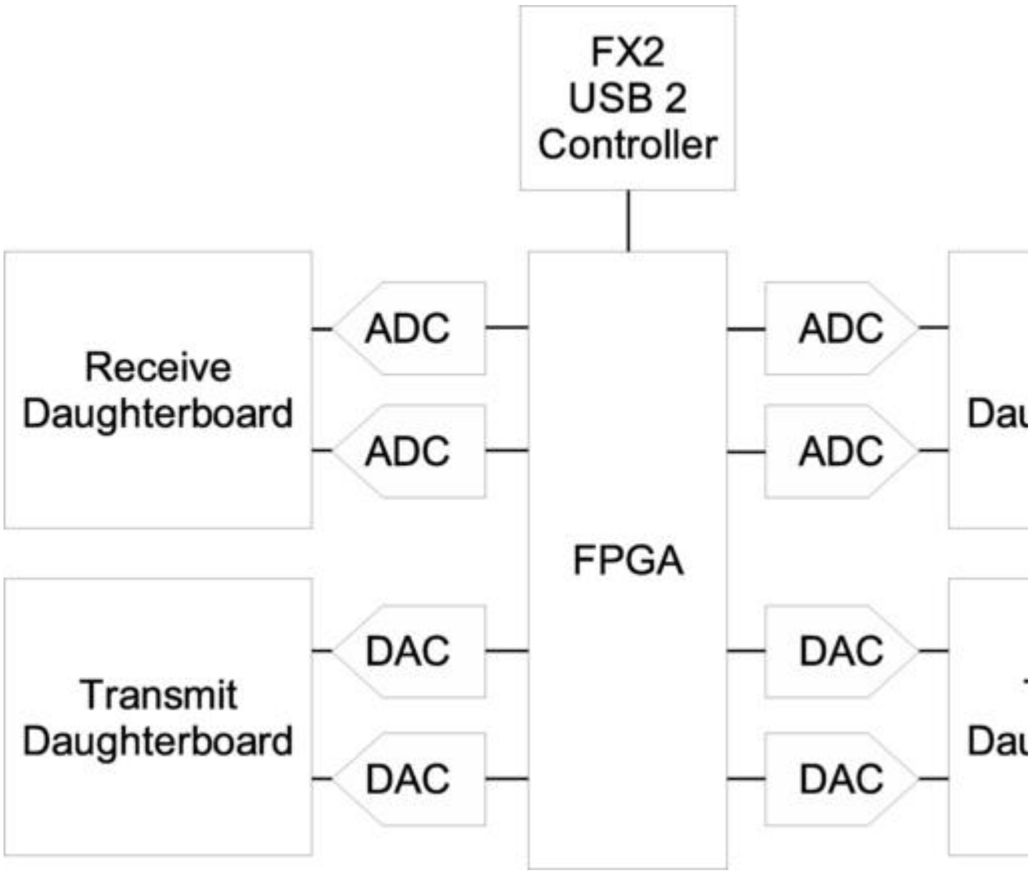


Figure 2. Universal Software Radio Peripheral (USRP)

The flexibility of the USRP comes from the two programmable components on the board and their interaction with the host-side library. To get a feel for the USRP, let's look at its boot sequence. The USRP itself contains no ROM-based firmware, merely a few bytes that specify the vendor ID (VID), product ID (PID) and revision. When the USRP is plugged in to the USB for the first time, the host-side library sees an unconfigured USRP. It can tell it's unconfigured by reading the VID, PID and revision. The first thing the library code does is download the 8051 code that defines the behavior of the USB peripheral controller. When this code boots, the USRP simulates a USB disconnect and reconnect. When it reconnects, the host sees a different device: the VID, PID and revision are different. The firmware now running defines the USB endpoints, interfaces and command handlers. One of the commands the USB controller now understands is `load the FPGA`. The library code, after seeing the USRP reconnect as the new device, goes to the next stage of the boot process and downloads the FPGA configuration bitstream.

FPGAs are generic hardware chips whose behavior is determined by the configuration bitstream that's loaded into them. You can think of the bitstream as object code. The bitstream is the output of compiling a high-level description of the design. In our case, the design is coded in the Verilog hardware description language. This is source code and, like the rest of the code in GNU Radio, is licensed under the GNU General Public License.

What Goes in the FPGA?

An FPGA is like a small, massively parallel computer that you design to do exactly what you want. Programming the FPGA takes a bit of skill, and mistakes can fry the board permanently. That said, we provide a standard configuration that is useful for a wide variety of applications.

Using a good USB host controller, the USRP can sustain 32MB/sec across the USB. The USB is half-duplex. Based on your needs, you partition the 32MB/sec between the transmit and the receive directions. In the receive direction, the standard configuration allows you to select the part or parts of the digitized spectrum you're interested in, translate them to baseband and decimate as required. This is exactly equivalent to what's happening in the RF front end, only now we're doing it on digitized samples. The block of code that performs this function is called a digital down converter (Figure 3). One advantage of performing this function in the digital domain is we can change the center frequency instantaneously, which is handy for frequency hopping spread spectrum systems.

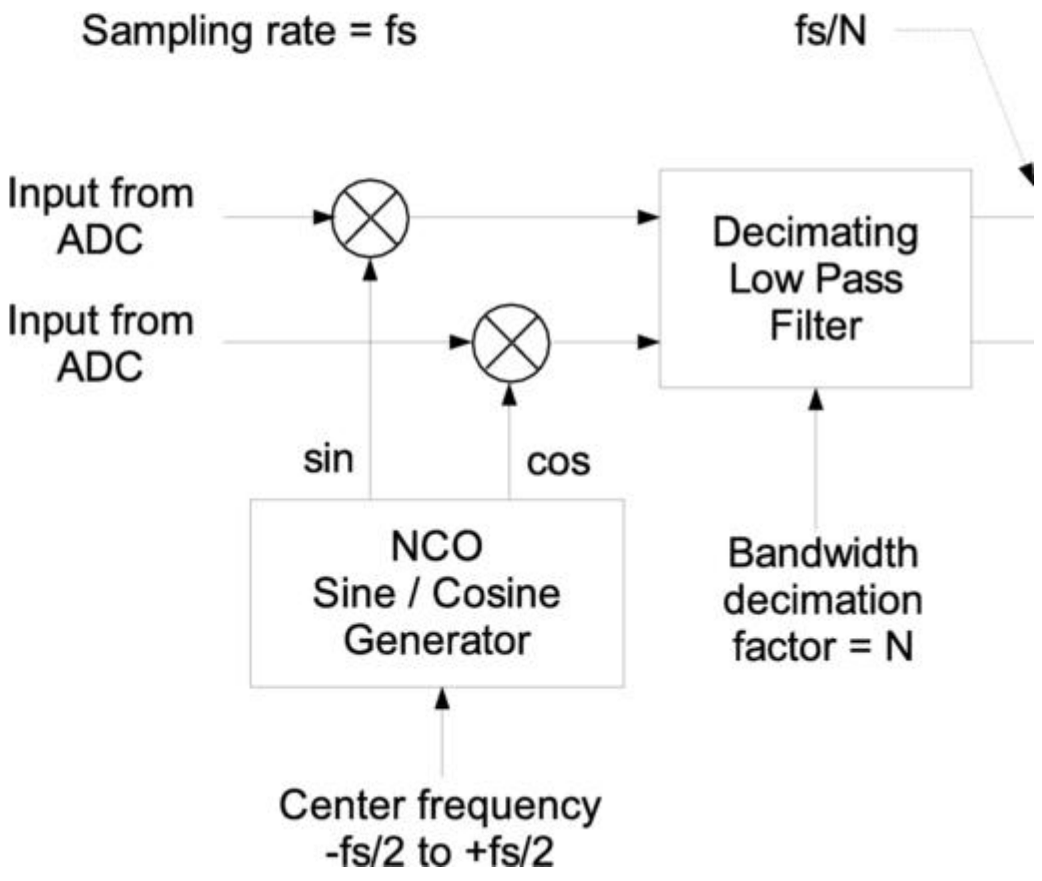

Sampling rate = fs
fs/N
Input from ADC
Input from ADC
Decimating Low Pass Filter
sin
cos
NCO Sine / Cosine Generator
Bandwidth decimation factor = N
Center frequency -fs/2 to +fs/2

Figure 3. Digital Down Converter Block Diagram

In the transmit direction, the exact inverse is performed. The FPGA contains multiple instances of the digital up and down converters. These instances can be connected to the same or different ADCs, depending on your needs. We don't have room here to cover all the theory behind them; see the GNU Radio Wiki for more information.

GNU Radio Applications

In addition to the examples discussed above, GNU Radio comes with a complete HDTV transmitter and receiver, a spectrum analyzer, an oscilloscope, concurrent multichannel receiver and an ever-growing collection of modulators and demodulators.

Projects under investigation or in progress include:

- A TiVo equivalent for radio, capable of recording multiple stations simultaneously.
- A passive radar system that takes advantage of broadcast TV for its signal source. For those of you with old TVs hooked to antennas, think about the flutter you see when airplanes fly over.
- Radio astronomy.
- TETRA transceiver.
- Digital Radio Mundial (DRM).
- Software GPS.
- Distributed sensor networks.
- Distributed measurement of spectrum utilization.
- Amateur radio transceivers.
- Ad hoc mesh networks.
- RFID detector/reader.
- Multiple input multiple output (MIMO) processing.

Politics

Every revolution has its political issues. Free software for building radios is troublesome to some people. In the US, we've run into opposition from the Motion Picture Association of America and its attempt with the Broadcast Flag to restrict the kinds of receivers that can be built for over-the-air digital TV.

The US Federal Communications Commission has issued a Notice of Proposed Rule Making (NPRM) concerning “Cognitive Radio Technologies and Software Defined Radios”. Several troublesome issues are raised in the NPRM, including restricting the sale of high-speed digital-to-analog converters, requirements for digital signatures or similar methods to keep unauthorized software out of software radio hardware and new restrictions on radios built for the amateur radio market.

Summary

Software radio is an exciting field, and GNU Radio provides the tools to start exploring. A deep understanding of software radio requires knowledge from many domains. We're doing our best to lower the barriers to entry.

Resources for this article: /article/7497.

Eric Blossom is the founder of the GNU Radio Project. Prior to his involvement with software radio, he spent many years in the secure phone business. When he's not hacking software radio, you're likely to find him practicing yoga or jujutsu. He can be reached at eb@comsec.com.

The Linux Soundfile Editor Roundup

Dave Phillips

Issue #122, June 2004

Whether you're making nifty sounds for desktop events or recording a whole album, you'll need one of these featureful sound apps.

A soundfile editor is the audio worker's essential utility for performing various editing operations to massage and finesse recorded sound. Some of these operations are analogous to those of a text editor, such as cut/copy/paste actions, while others are unique to editing audio data.

This article takes you on a whirlwind tour of soundfile editors for Linux. You don't need to know anything special about digital audio or DSP theory, and if you'd like to try any of these programs, all you need on your machine is a working sound system. But, before taking the tour, let's consider what a typical soundfile editor does and how it's typically used.

Common Features

Editing audio is best accomplished with a graphical interface. By representing audio data visually, it is simple to locate those parts of the sound that require attention, such as gaps and amplitude spikes. This ability to find quickly where an operation is needed speeds up the editing process. Regions can be marked or selected accurately, and zoom routines let users enlarge or diminish their view of a file at any given point, making it possible to perform large-scale and sample-accurate edits easily and quickly.

A well-designed soundfile editor should include at least these basic operations and capabilities:

- Cut/copy/paste.
- Splice/insert/replace.
- Move samples.
- Mix tracks.

- Synchronize files/tracks.
- Compress/expand time scale.
- Pitch shifting.
- Equalization/filtering.
- Sample rate conversion.
- Display in different time formats.
- Display multiple views of file.
- Display multiple files simultaneously.
- Independent X/Y axis control.
- Find maximum sample value.
- Display amplitude envelope in various representations (db, peak, RMS).
- Edit pitch and amplitude envelopes.
- Varispeed playback.
- Print out display of samples.
- Spectrum analysis.

As we shall see, the editors profiled in this article meet most of these baseline criteria, often adding unique functions and routines.

Table 1 presents a further set of features, most but not all Linux-specific, and outlines how the editors presented here accommodate them.

Table 1. Soundfile Editor Features

Editor	ALSA	JACK	LADSPA	GUI	Size limit	License
Snd	y	y	y	Motif, GTK [1]	disk	GPL
MiXViews	n [2]	n	n	InterViews	memory	[3]
DAP	n [2]	n	n	XForms	memory	GPL
Audacity	y	y	y	wxGTK	disk	GPL
ReZound	y	y	y	FOX	disk	GPL
Sweep	y	y	y	GTK	disk	GPL
GLAME	y	n	y	GTK	disk	GPL
LAoE	n [2]	n	n	Java	disk	GPL
Swami	y	y	y	GTK	disk	GPL

Editor	ALSA	JACK	LADSPA	GUI	Size limit	License
WaveSurfer	y	y	y	Tcl/Tk	disk	[4]
GNUsound	y	n	y	GTK	memory	GPL
KWave	n [2]	n	n	Qt	memory	GPL

[1] Can be compiled without graphics. [2] Works with ALSA's OSS/Free emulation. [3] Free for noncommercial use, freely distributable. [4] Free for unrestricted use, freely distributable.

Common Uses

Now, let's consider some possible uses for a soundfile editor. The following list is by no means a complete summary, it merely indicates how I typically employ an editor in my own work:

- Trimming excess silence from recordings.
- Cutting large files into smaller pieces.
- Normalizing.
- Adding effects such as reverb, chorus and flanging.
- Slowing playback speed without changing pitch.
- Removing cracks and pops from recordings.
- Converting sample rate.
- Converting file format.
- Equalizing or filtering the sound to make it brighter or give it more bass.
- Using the editor as a music and sound composition tool.

Dozens of other operations can be added to this list, and each user will find special uses for a soundfile editor.

Slowing the playback speed of a file without changing its pitch comes in handy when I am teaching. Students often bring in recordings that are not particularly easy to hear or transcribe when played at normal speed. I convert the original recording (CD or MP3) to a WAV file, load it into the Snd editor and slow the playback speed, still keeping the original pitch, until I can hear each note clearly. This makes it easier to write out an accurate transcription. Some editors let you do this in real time, and some allow a defined looping region to be arbitrarily redefined during playback, a very helpful feature.

Normalization raises the amplitudes of a file to their relative peaks, so all amplitude values are raised relative to the peak value. By normalizing my project files before burning them to CD, I balance the volume differences between pieces. Normalization is a common premastering operation in professionally produced recordings.

A soundfile editor also is useful for repairing badly formed compressed soundfiles. Some editors load MP3 and Ogg files directly, while others perform a format conversion and then load the converted file. I can remove excess silence and fix damaged spots in the recording by removing or redrawing them. The waveform display represents damaged spots as broken or incomplete curves. I also normalize and equalize the sound before converting it back to its original format. Converting a lossy-encoded file to a soundfile format and back again to a lossy format typically results in less than optimal audio quality, so I rebalance the sound's frequencies with the editor's equalization tools.

In the space allotted for this article, I can scratch only the surface of each profiled editor. We look at each program's salient aspects, but you should try each one yourself to measure the depths. Let's start this roundup with some of the older soundfile editors available for Linux.

The first wave of soundfile editors for Linux arrived at a time when OSS/Free was the system audio interface and Motif was the most attractive GUI toolkit. They were all designed to compile and run on a variety of UNIX systems, not only Linux.

Snd 7.0

Bill Schottstaedt has been crafting Snd in one form or another since the era of the PDP minicomputers. However, Snd as we know it dates from late 1996, with Linux support added in 1997. Snd is a remarkable program. It can be regarded as an exceptionally powerful soundfile editor, as an infinitely programmable audio editing toolkit or as a graphic component of the Common environment of music and sound applications. The Common family of audio applications include Bill Schottstaedt's Common Lisp Music (software sound synthesis) and Common Music Notation and Rick Taube's Common Music (metalanguage for music composition). These are all Lisp-based applications that can be configured for complex interactivity. The key to unlocking Snd's power lies in its Guile interface, based on the Lisp-like Scheme programming language. Snd's interface includes a window called the Listener in which the user can enter Guile commands for customizing and redefining every aspect of the program.

The screenshots illustrate how extensively Snd can be configured. Figure 1 shows the Motif version of Snd in its default appearance. In Figure 2, the user interface has been customized extensively. New menus have been added for Snd's internal DSP modules and LADSPA plugins, the background and colors are user-defined, and some complex widgets have been custom-built for some of Snd's effects processors. Figure 2 also shows off Snd's amplitude waveform view along with its OpenGL display of the sound's spectrum, its frequency content. Context-sensitive pop-up menus are available for the different displays as well as for selected and unselected regions of a file.

Snd is my favorite soundfile editor, but I admit that it may not be the editor for everyone. If you learn a little Lisp, you can discover more of Snd's power. Fortunately, Snd comes with exhaustive documentation to ease difficulties in the learning process. Predefined configuration files also are available for quicker and easier customization.

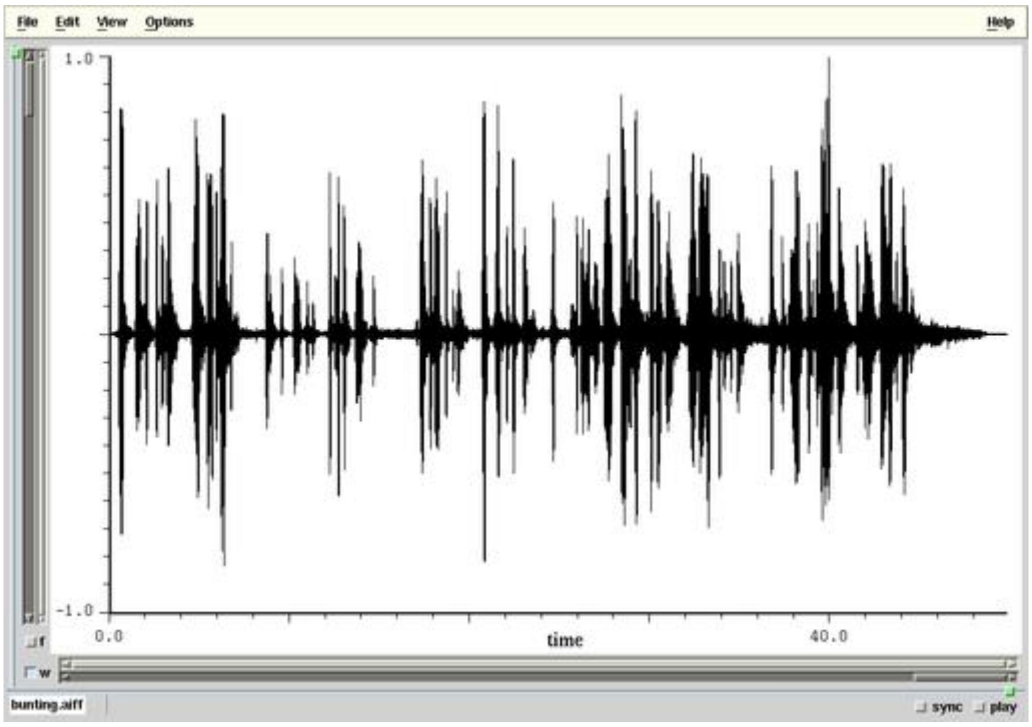


Figure 1. The Default Snd

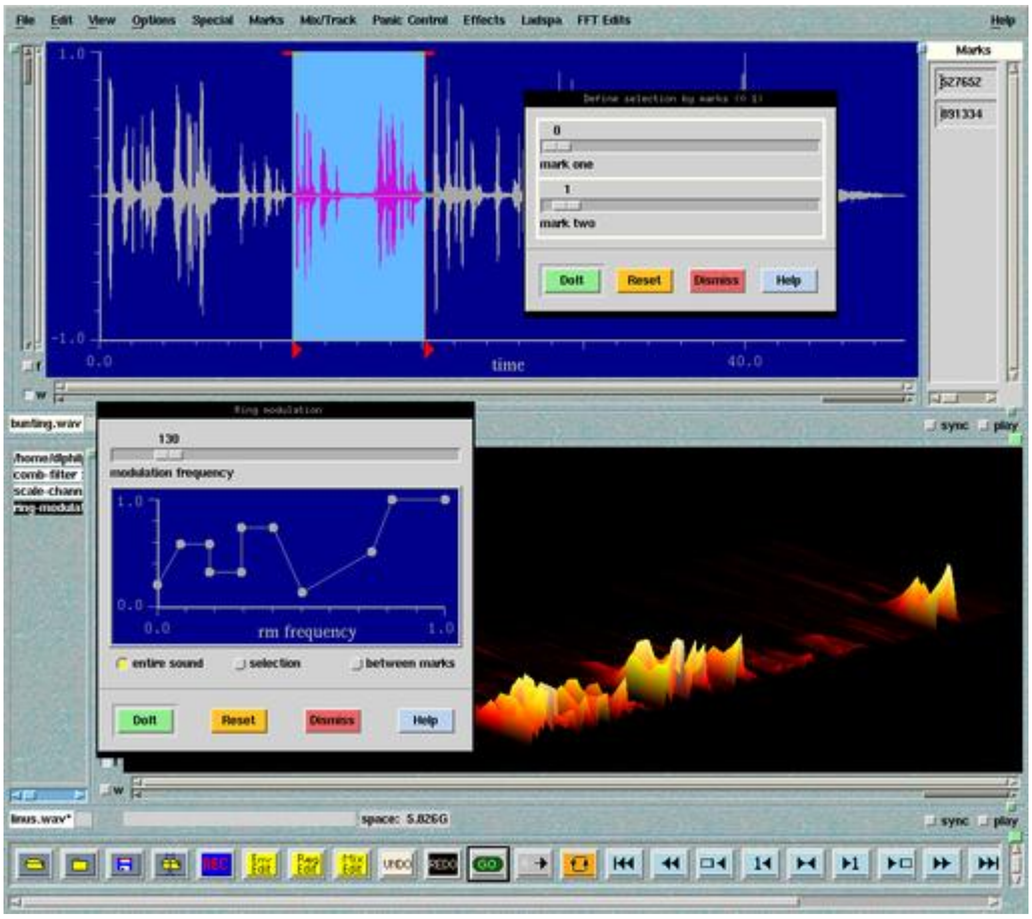


Figure 2. The Author's Snd

MiXViews 1.30

The 1.0 release of Doug Scott's MiXViews occurred in 1995, making it the longest-living Linux soundfile editor profiled here. MiXViews was and is a one-man effort to provide UNIX and Linux with a high-quality audio editor. The project continues to be a solo development effort, and it still provides a high-quality editor.

MiXViews provides a strong suite of the basic soundfile editing functions and adds some features found in no other Linux soundfile editor. Phase vocoding and linear predictive coding (LPC) are digital signal analysis/resynthesis tools more commonly associated with software sound synthesis programs such as Csound or Common Lisp Music. These tools analyze a sound for its frequency and amplitude values and store those values in special analysis file formats. An analysis file can be read by a program such as Csound that gives the user independent control over the frequency and amplitude components of the analysis data before its resynthesis to a soundfile. MiXViews provides a complete suite of its own LPC and phase vocoder utilities; it also can read and edit analysis files created by the Csound phase vocoder.

Figure 3 displays some of MiXViews' graphic tools for editing phase vocoder and LPC analysis data. Although the theory and mathematics behind these tools can be intimidating, the MiXViews interface invites experimentation, making the tools themselves easier and more interesting to use.

If you want to try MiXViews, I suggest using the prebuilt binary. Compiling MiXViews is somewhat tricky, and it requires an uncommon graphics toolkit (InterViews), so simply download the static binary and start using it.

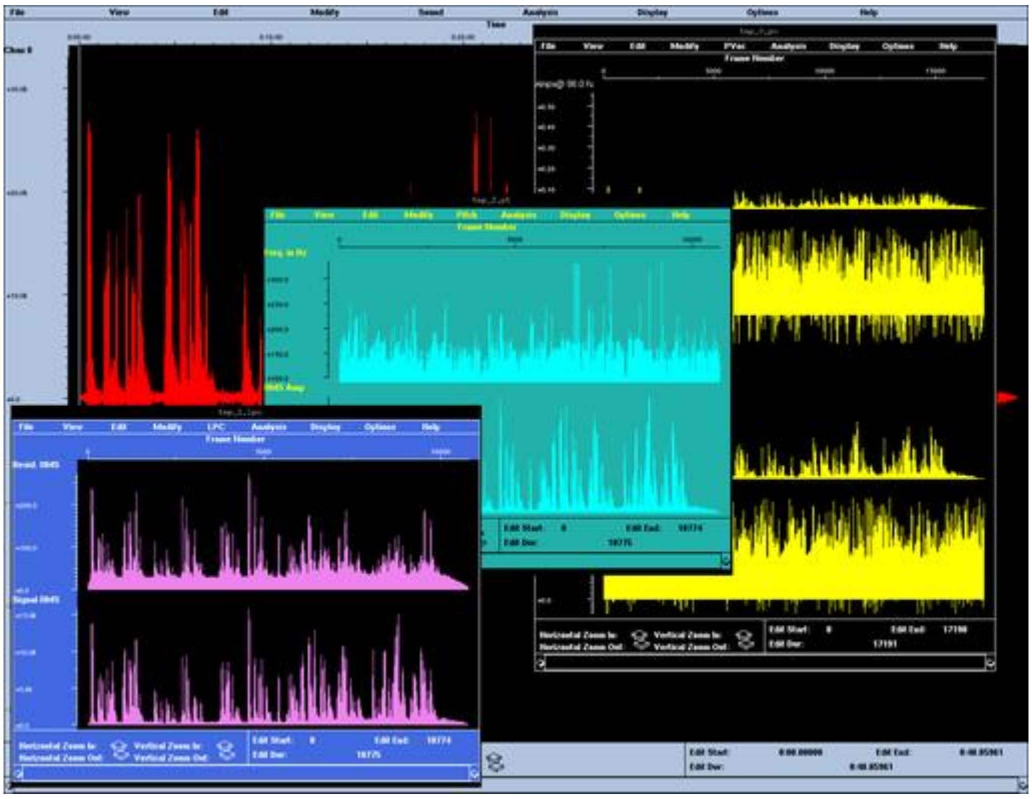


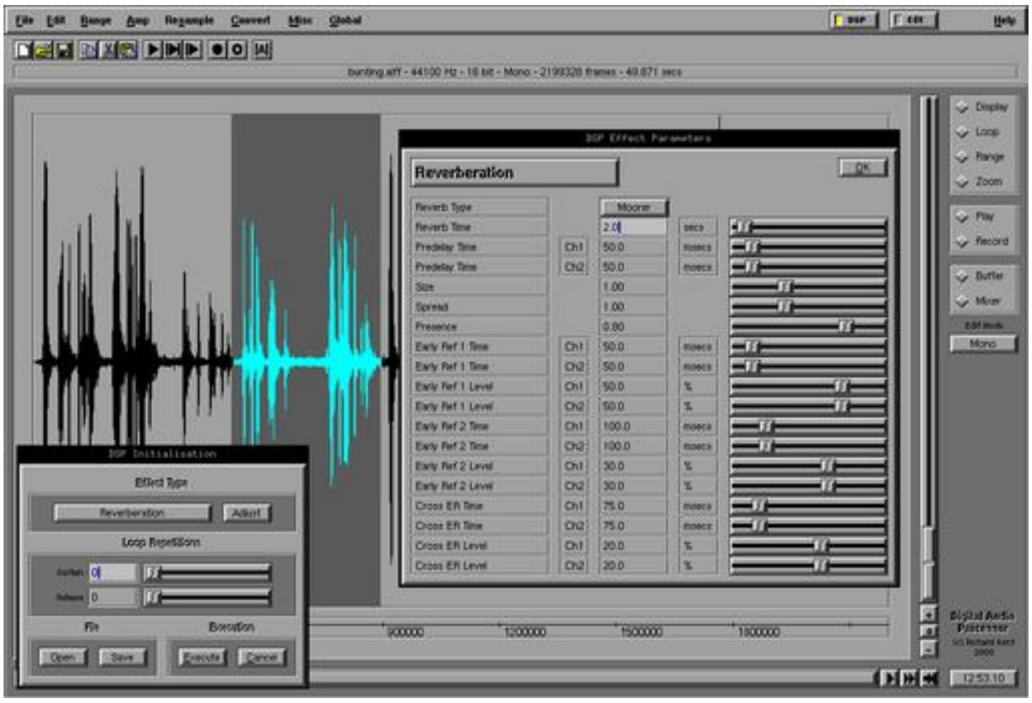
Figure 3. MiXViews

DAP 2.1.4

DAP (the Digital Audio Processor) is programmer Richard Kent's contribution to multiplatform soundfile editors. Like MiXViews, DAP's GUI is based on a not-so-new GUI toolkit, the XForms library. Also like MiXViews, DAP's workable soundfile size is limited by your system RAM. In addition, DAP includes some exceptionally well-implemented loop editing tools for AIFF soundfiles. DAP also includes a good selection of DSP modules (extended from Kai Lassfolk's SPKit code) and a handy mono-to-stereo and mono/stereo-to-quad converter.

Some of DAP's editing tools deserve special mention, particularly those found in the Resample and Edit/Mix dialogs. The Resample menu provides pitch and sample rate change with or without time stretching, while the Edit/Mix dialogs (Mix and Mix Range) provide a neat graphic control over the balance of the mixed file amplitudes. The influence of the AIFF file format and its loop support is found throughout the program. For example, when an effect is applied to a file the DSP dialog panel provides a control for the iterations of the sustain and release loops (Figure 4). Although DAP's design is biased toward the AIFF format, it also imports and exports files in RAW and WAV formats.

Alas, DAP is no longer consistently maintained. Its XForms GUI is showing its age, and its file size limitation is a serious drawback. The author is honest about DAP's limitations, but if you're working with AIFF files with embedded loop points, DAP is still a useful tool.



bunting.aiff - 44100 Hz - 16 bit - Mono - 2199328 frames - 49.871 secs

DSP Effect Parameters

Reverberation

Reverb Type		Moorer	
Reverb Time		2.0	secs
Predelay Time	Ch1	50.0	msecs
Predelay Time	Ch2	50.0	msecs
Size		1.00	
Spread		1.00	
Presence		0.80	
Early Ref 1 Time	Ch1	50.0	msecs
Early Ref 1 Time	Ch2	50.0	msecs
Early Ref 1 Level	Ch1	50.0	%
Early Ref 1 Level	Ch2	50.0	%
Early Ref 2 Time	Ch1	100.0	msecs
Early Ref 2 Time	Ch2	100.0	msecs
Early Ref 2 Level	Ch1	30.0	%
Early Ref 2 Level	Ch2	30.0	%
Cross ER Time	Ch1	75.0	msecs
Cross ER Time	Ch2	75.0	msecs
Cross ER Level	Ch1	20.0	%
Cross ER Level	Ch2	20.0	%

Figure 4. DAP

The next group of editors belong to the new wave of Linux audio development. Their natural environment includes the modern graphic interface toolkits and the newer Linux sound system components, such as ALSA, JACK and LADSPA. They also are conceptually more homogenous than their predecessors, offering resemblance to the popular editors familiar to Windows and Mac users.

Audacity 1.1.3

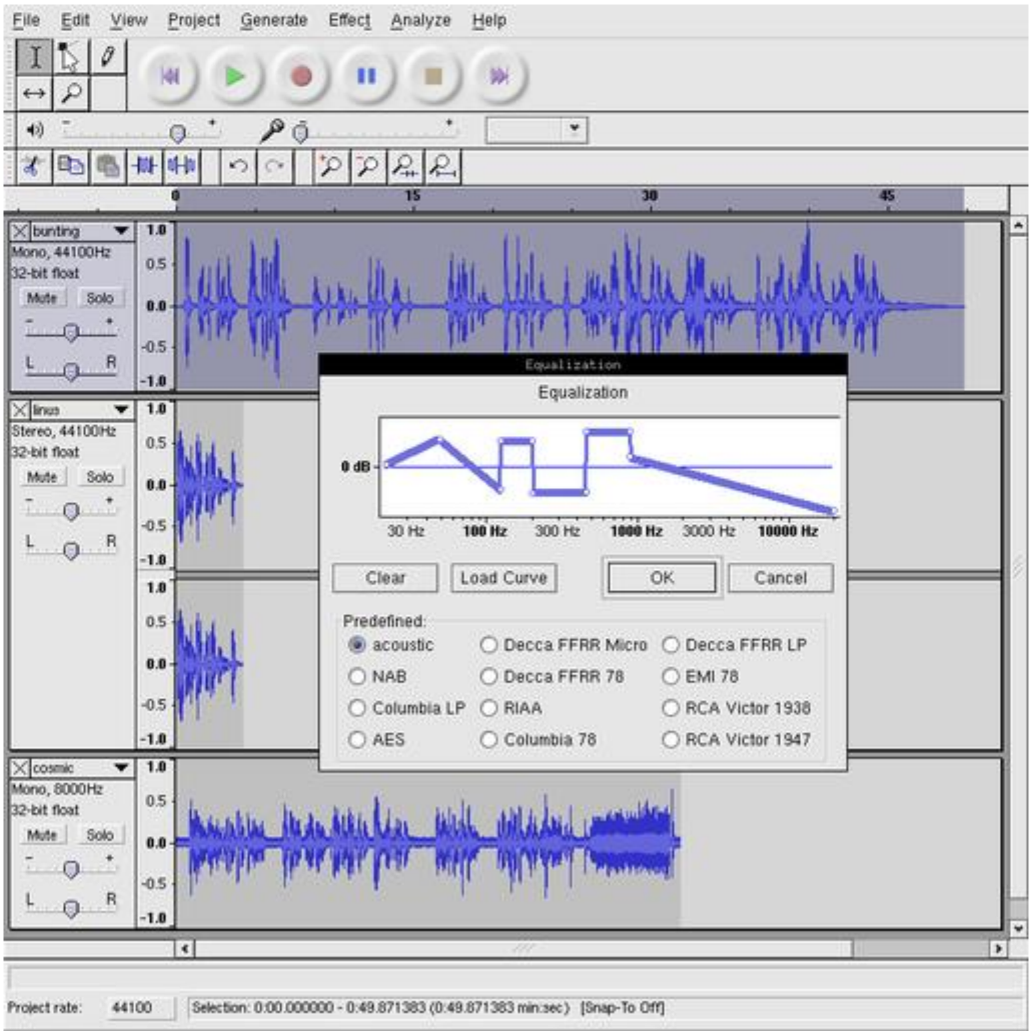
Audacity is a fitting first representative of this new wave of Linux soundfile editors. It is written in C/C++, uses the wxWindows cross-platform GUI toolkit and supports native and LADSPA signal processing plugins. Recent versions also are JACK-aware, giving Audacity the ability to route its I/O to or from other active JACK-aware programs.

Figure 5 illustrates Audacity with three soundfiles opened—a mono WAV file, a stereo AIFF file and a file in Sun's AU format. Audacity also imports MP3 and Ogg files. It exports Ogg files directly, thanks to the Ogg/Vorbis libraries, but MP3 export depends on a user-supplied encoder. Figure 5 also shows Audacity's native equalizer plugin at work.

Audacity's graphic editing tools are a pleasure to use. Figure 6 shows off the envelope tool's effect on the amplitude contour of one of the soundfiles seen in Figure 5. At the individual sample level, Audacity's drawing tool makes it easy to remove or repair amplitude spikes and other discontinuities.

Like Snd, Audacity features an interface to a Lisp-based programming language, Roger Dannenberg's Nyquist. Nyquist is a language designed for sound synthesis and signal processing, and Audacity's Effects menu provides a Nyquist Prompt that works essentially like Snd's Listener. You enter a Nyquist expression in the prompt dialog, click on the OK button and if the expression is valid, Audacity performs the intended process on the active soundfile.

There's far more to Audacity than I can possibly describe here. Fortunately, the program is easy to learn and use, so check it out for yourself.



Predefined:
[✓ acoustic] Decca FFRR Micro Decca FFRR LP
NAB Decca FFRR 78 EMI 78
Columbia LP RIAA RCA Victor 1938
AES Columbia 78 RCA Victor 1947

Project rate: 44100 Selection: 0:00.000000 - 0:49.871383 (0:49.871383 min:sec) [Snap-To Off]

Figure 5. Audacity

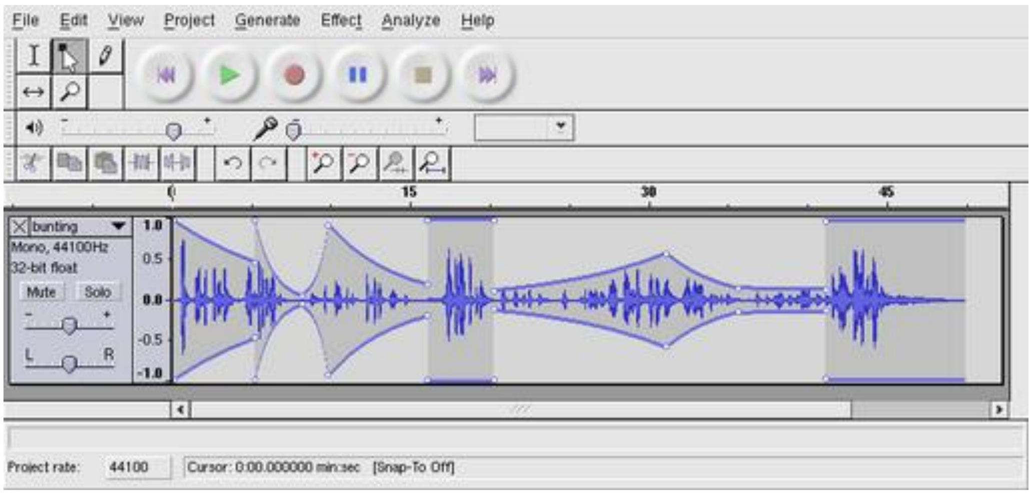


Figure 6. Audacity's Envelope Tool

ReZound 0.9.0beta

With its colorful interface and excellent organization, Davy Durham's ReZound is a pleasure to see as well to use. But eye candy is the least of ReZound's features; the program also provides a complete suite of editing tools, excellent transport controls, some impressive native filters, support for LADSPA plugins and a unique audio remastering/burn-to-CD facility.

Figure 7 illustrates ReZound with three soundfiles loaded and the Curved Balance tool at work on the active file. Curved Balance is one of ReZound's remastering utilities; others include a noise gate, a dynamics compressor and gain and normalization controls. These tools, along with ReZound's other editing amenities, let you massage your sounds to perfection before burning them to CD. ReZound even provides a simple dialog for burning the CD (by way of the cdrdao program) directly from ReZound's File menu.

ReZound's LADSPA support extends to good basic support of the LADSPA VST host plugin (vst.so) from Kjetil Matheussen. This plugin to support plugins provides a usable interface based on WINE for running VST/VSTi plugins under Linux. The vst.so plugin currently is in its early stages, and its degree of harmony with the hosting application varies considerably. Figure 8 shows ReZound employing a VST plugin to apply an effect to the active soundfile.

The most recent version of ReZound supports the JACK audio server, connecting ReZound to the JACK network of intercommunicating audio applications. More JACK enhancements are planned, along with effects previewing, noise removal tools, native time/pitch scaling and many other features and improvements.

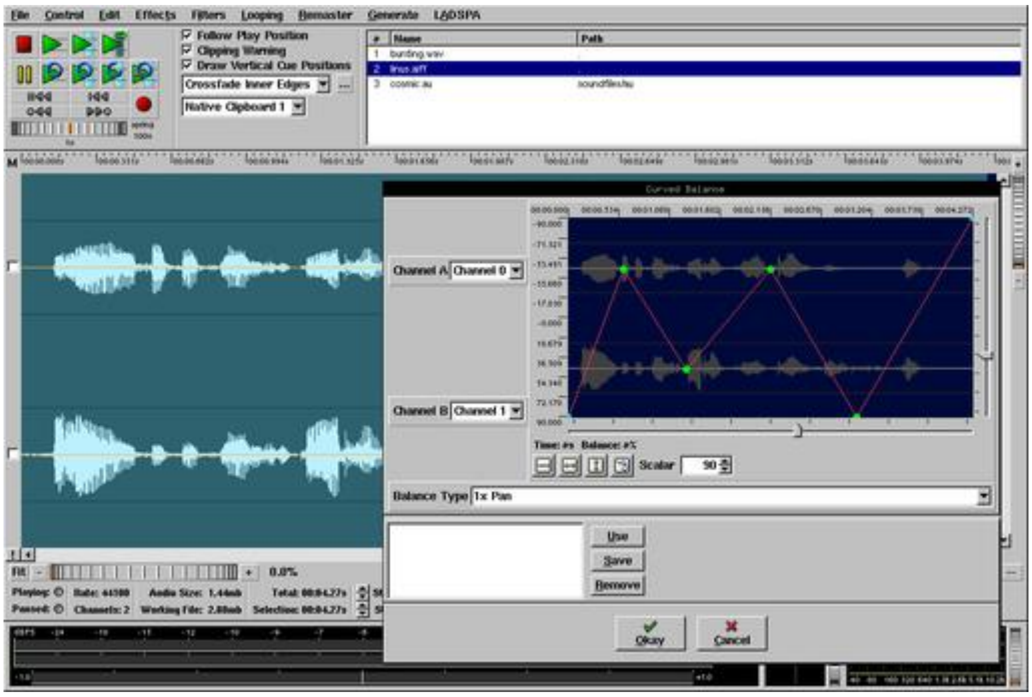


Figure 7. ReZound

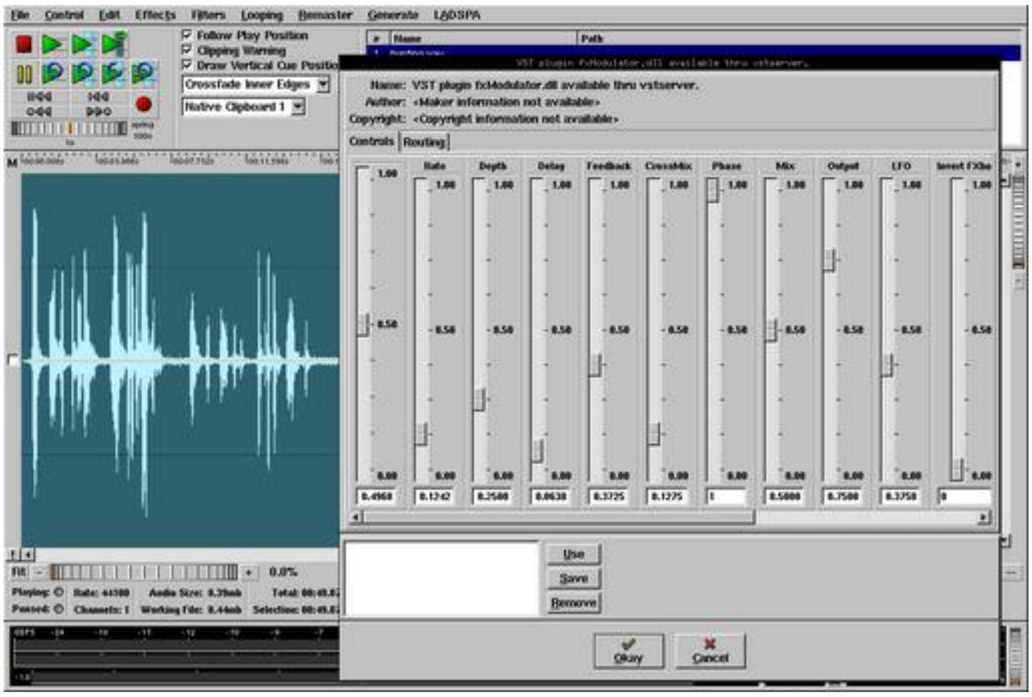

File Control Edit Effects Filters Looping Remaster Generate LADSPA
Name: VST plugin fxModulator.dll available thru vstserver.
Author: <Maker information not available>
Copyright: <Copyright information not available>
Controls
Routing
Rate
Depth
Delay
Feedback
CrossMix
Phase
Mix
Output
LFO
Invert FXbs
Use
Save
Remove
Okay
Cancel

Figure 8. ReZound with LADSPA/VST Support

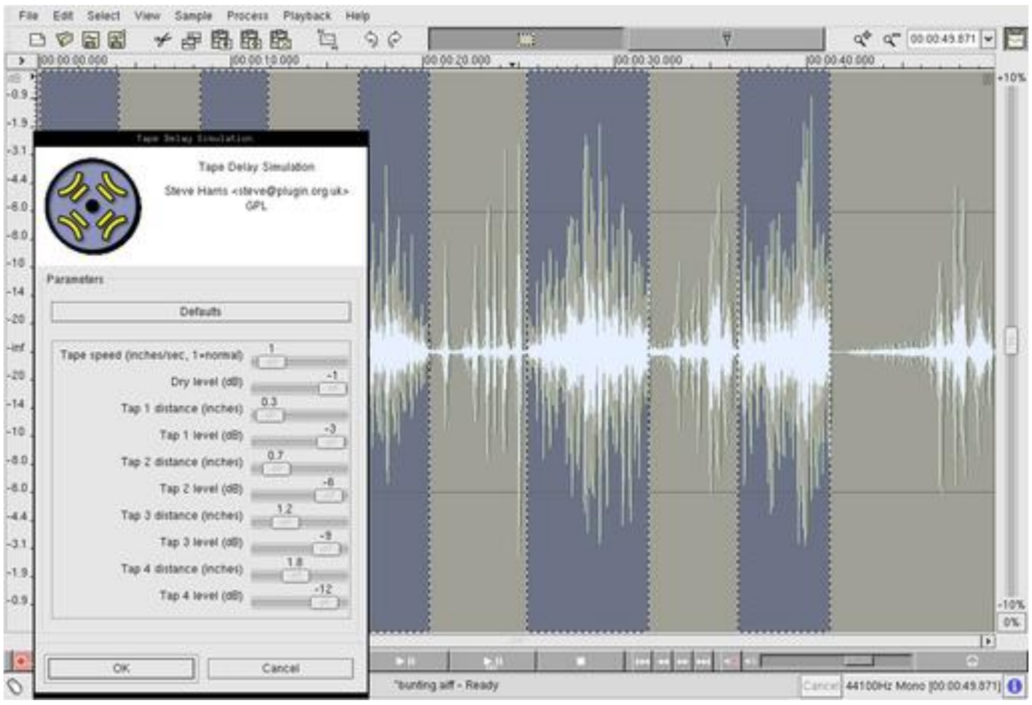
Sweep 0.8.2

At first look, Conrad Parker's Sweep seems much like the other newer editors profiled here. It's ALSA-aware, provides a good basic editing suite, supports LADSPA plugins and shows off a nice modern GTK interface. Sweep offers two unusual tools, though, that give it special value. One is the ability to define multiple regions for nonlinear editing; the other is an interesting little tool named Scrubby.

Defining a selection in a soundfile typically is achieved by placing the cursor at the selection start, then left-clicking and holding as you sweep the cursor to the selection end. This selection method is a common practice employed by all the editors reviewed here. That's how it's done in Sweep too, but Sweep also lets you define multiple selections. Hold down the Ctrl key while making your selections, and voilà, you have multiple-defined selections available for further processing.

An Invert Selection function provides a neat way to create a dialog of effects over a soundfile. Figure 9 illustrates the aftermath of this series of alterations: define multiple selections; apply reverse effect to those selections; invert selection areas (non-selected becomes selected, and vice versa); and apply LADSPA effect to new selections. This is fun stuff and a powerful and creative feature.

Scrubby is Sweep's virtual stylus, essentially functioning as a freely movable playback head with features more typically associated with a DJ's turntable system. Scrubby transforms Sweep into a performance tool, an unusual characteristic for a soundfile editor. A screenshot can't possibly do justice to Scrubby; you have to use the software to see and hear it in action.



Tape Delay Simulation

Steve Harris <steve@plugin.org.uk>
GPL

Parameters

Tape speed (inches/sec, 1=normal)	1
Dry level (dB)	-1
Tap 1 distance (inches)	0.3
Tap 1 level (dB)	-3
Tap 2 distance (inches)	0.7
Tap 2 level (dB)	-6
Tap 3 distance (inches)	1.2
Tap 3 level (dB)	-9
Tap 4 distance (inches)	1.8
Tap 4 level (dB)	-12

"bunting.aiff - Ready

44100Hz Mono [00:00:49.871]

Figure 9. Sweep

Swami 0.9.1a

The soundfont format (SF2) is a complex soundfile format that includes not only audio data but also data regarding various effects and performance controls. Soundfonts have become ubiquitous in the audio world, finding employment in applications such as Csound, jMax, Pd, Fluidsynth and many others.

If you want to work with soundfonts, you definitely want Josh Green's excellent Swami in your toolkit. Swami is a soundfont-only editor with a well-designed GUI and a wealth of useful features. You can edit existing soundfonts and create your own from the level of individual samples up to the composite instrument. You also can edit and design your own soundfont banks. Other tools are available for setting an instrument's velocity response curves, keyboard zoning (mapping) and modulator routing. Because of Swami's strict focus, I don't have much more to say about it; it is first-rate and is highly recommended Linux audio software.

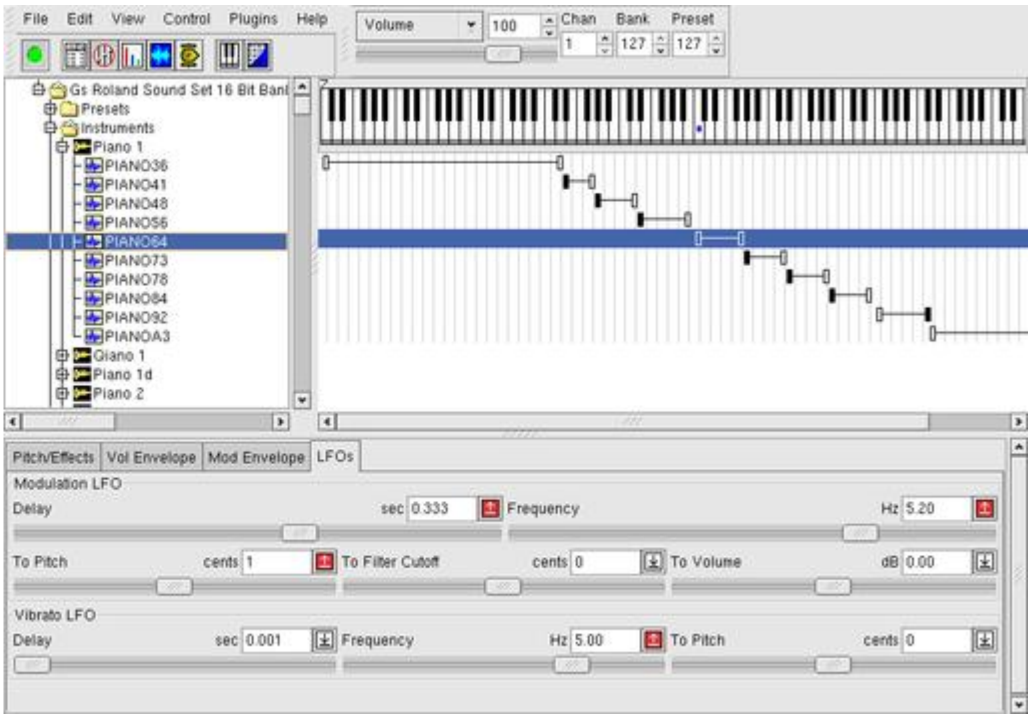


Figure 10. Swami

WaveSurfer 1.5.7

Kâre Sjölander and Jonas Beskow have developed WaveSurfer to function best in the context of speech research, a domain covering a variety of audio-related disciplines. WaveSurfer is a perfectly useful general-purpose soundfile editor, but its special strengths reside in its tools for analyzing, editing and visualizing the spoken word.

WaveSurfer is written in the popular Tcl/Tk scripting language and widget toolkit, providing the motivated user complete access to the program's internals. Sound processing in WaveSurfer is handled by the SNACK audio functions library, also written by Kâre Sjölander. SNACK itself may be extended by user-defined plugins written in C/C++.

Figure 11 illustrates a simple use of WaveSurfer in speech analysis and representation. The main panel displays the region highlighted in the complete waveform, and the label display indicates the sound's phonemes. The label track is only one example of WaveSurfer's speech-oriented amenities. Others include spectrographic displays, pitch curve extraction and support for a variety of soundfile and transcription formats.

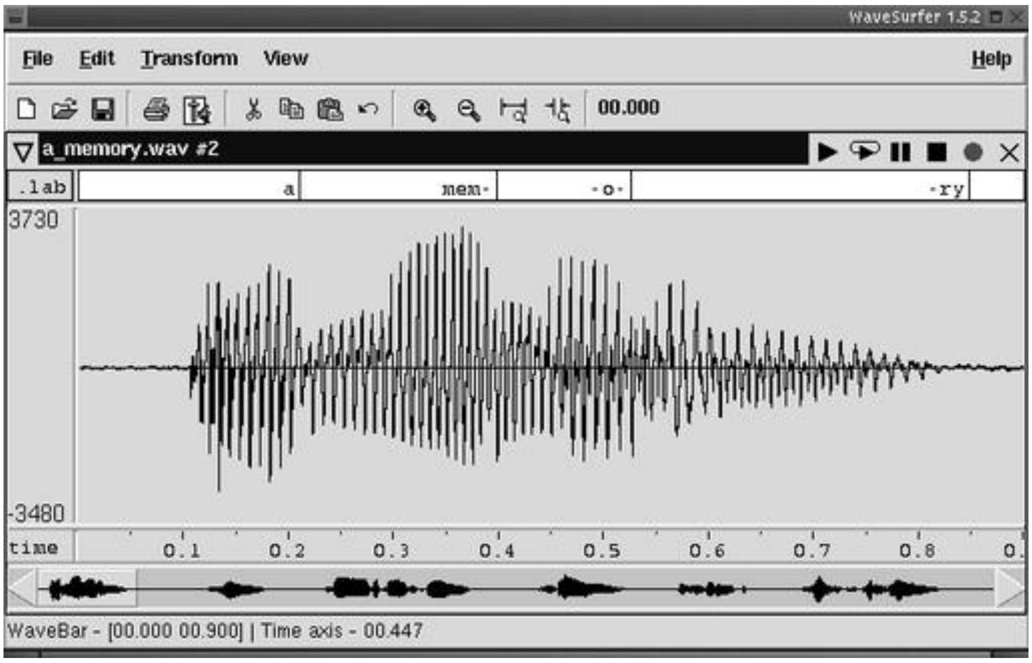
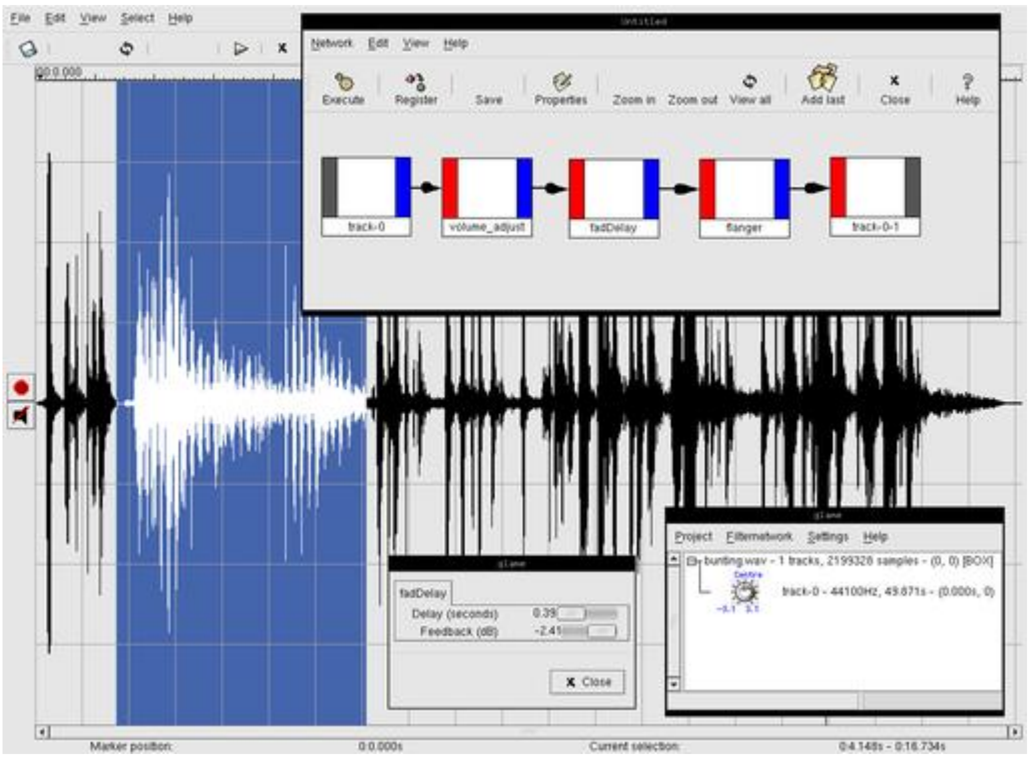


Figure 11. WaveSurfer

GLAME 1.0.1

GLAME's developers have implemented an unusual design philosophy in their editor. GLAME (GNU/Linux Audio MEchanics) supplies the expected palette of tools for audio editing, but it also includes a powerful synthesis and processing environment called the filternetwork. A filternetwork provides a canvas on which icons representing synthesis primitives are patched together to create a processing or synthesis chain. Current primitives include oscillators, envelope generators, filters, I/O modules and LADSPA plugins. Once a synthesis network has been designed, it can be run to produce real-time audio or output to a file for further processing (in GLAME, of course). Right-clicking on the waveform display pops up a menu that includes the Apply Custom item. By selecting this item, you can apply your filternetwork to the active soundfile, suggesting some interesting processing possibilities.

Figure 12 illustrates a simple example. The selection in the waveform display has been processed by a filternetwork composed of a gain control, a LADSPA delay plugin and a flanger. The track modules are included as the default I/O ports, representing the original input and the processed audio output.



bunting.wav - 1 tracks, 2199328 samples - (0, 0) [BOX]
track-0 - 44100Hz, 49.871s - (0.000s, 0)

Delay (seconds) 0.39
Feedback (dB) -2.41

0:0.000s
0:4.148s - 0:16.734s

Figure 12. GLAME

LAoE 0.6.03beta

Olivier Gäumann's Layer-based Audio Editor (LAoE) offers yet another unique design philosophy. An editing session in LAoE consists of building a stack of soundfiles and then opening the desired editing and processing tools for application upon one or more of the layers (soundfiles) in the stack. At first it felt like a rather strange way to work, but after comprehending the program's organization, I began to enjoy its layout and developed a fast work mode with it.

LAoE receives extra points for originality by providing direct editing in its spectral display. A user-defined brush is used to paint over areas for FFT filtering, and the filter itself can be adjusted for finer resolution. Most of this article's editors offer spectral displays, but only LAoE permits direct spectral editing.

LAoE also is the only Java-based editor reviewed here. I've installed Sun's JDK 1.4 on my 800MHz machine, not exactly a fast machine by today's standards, but LAoE's interface was quick and responsive throughout.

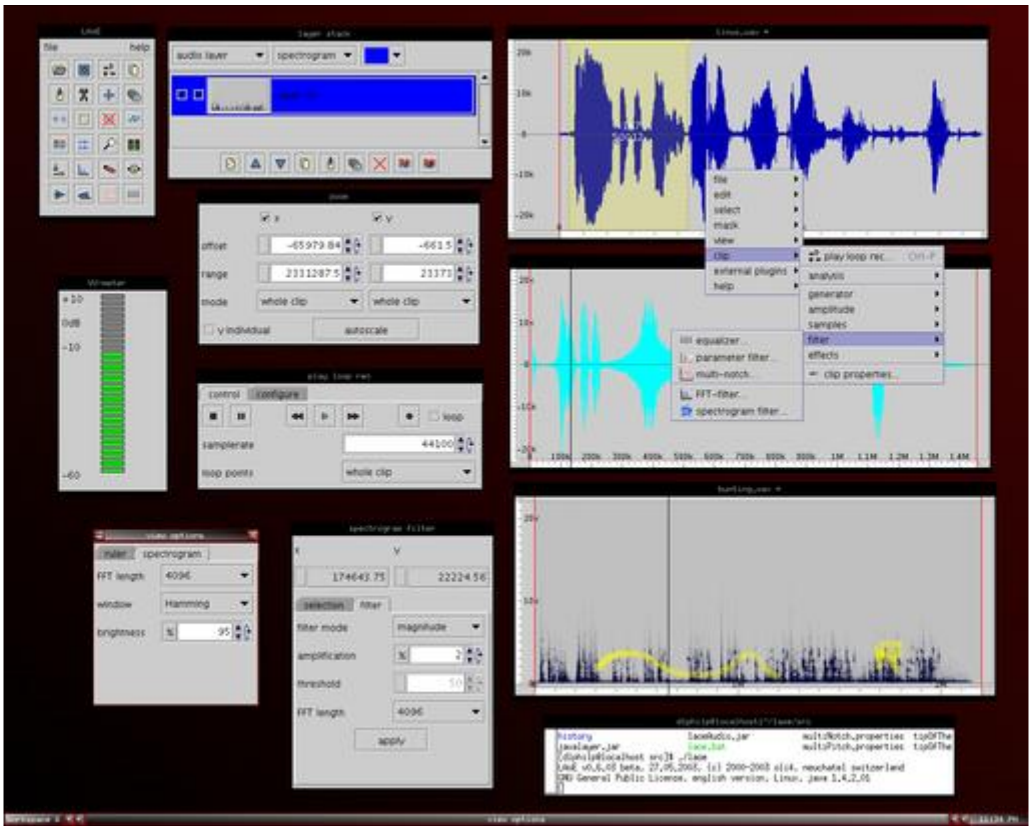


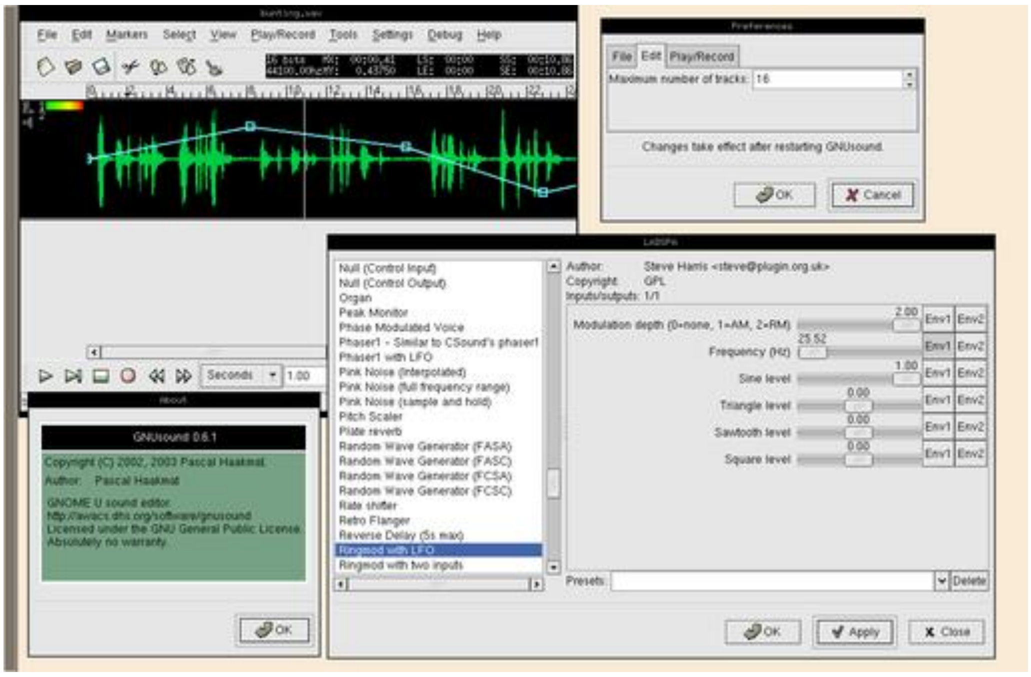
Figure 13. LAoE

GNUsound 0.6.1

Pascal Haakmat's GNUsound is modest in appearance but rich in content. Once again we have a full complement of the basic editing tools, LADSPA plugin support and some special tools for marking, selecting and viewing soundfiles. GNUsound also adopts the concept of tracks, that is, you can designate a number of files for mixdown in a process similar to the mixing process in a multitrack recorder.

Another neat aspect of GNUsound is its implementation of envelopes for effects processing. One of two user-defined envelopes may be selected as control curves for an associated processing parameter, giving a more dynamic contour to your effects processing.

Although GNUsound is intended for use in the GNOME environment, I had no trouble building it under a Planet CCRMA Red Hat 9.0 system and using it in the BlackBox window manager.



Maximum number of tracks: 16

Changes take effect after restarting GNUsound.

Null (Control Input)
Null (Control Output)
Organ
Peak Monitor
Phase Modulated Voice
Phaser1 - Similar to CSound's phaser1
Phaser1 with LFO
Pink Noise (interpolated)
Pink Noise (full frequency range)
Pink Noise (sample and hold)
Pitch Scaler
Plate reverb
Random Wave Generator (FASA)
Random Wave Generator (FASC)
Random Wave Generator (FCSA)
Random Wave Generator (FCSC)
Rate shifter
Retro Flanger
Reverse Delay (5s max)
Ringmod with LFO
Ringmod with two inputs

Author: Steve Harris <steve@plugin.org.uk>
Copyright: GPL
Inputs/outputs: 1/1

Modulation depth (0=none, 1=AM, 2=RM) 2.00
Frequency (Hz) 25.52
Sine level 1.00
Triangle level 0.00
Sawtooth level 0.00
Square level 0.00

GNUsound 0.6.1
Copyright (C) 2002, 2003 Pascal Haakmat
Author: Pascal Haakmat
GNOME U sound editor
http://awacs.dhs.org/software/gnusound
Licensed under the GNU General Public License
Absolutely no warranty

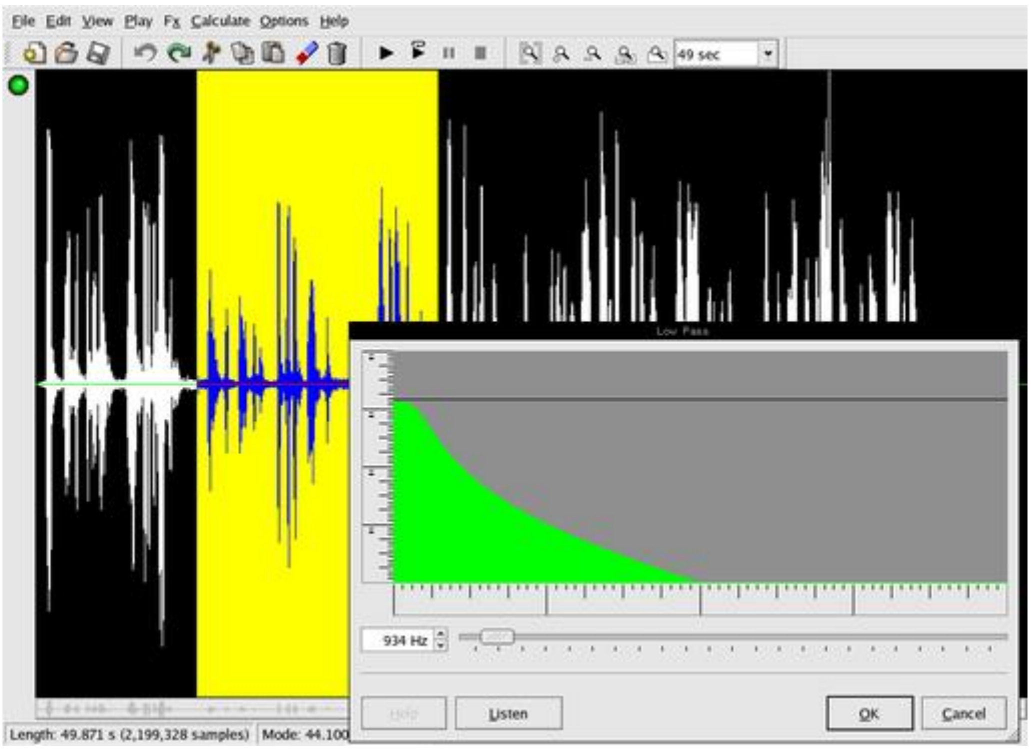
Figure 14. GNUsound

KWave 0.7.0-1

KWave has been in development since 1999, so perhaps I should have included it along with the venerable stalwarts. However, its development team has kept pace with its intended target environment, KDE, thus giving KWave a more modern look and feel along with some interesting improvements.

The new KWave has also retained the original's emphasis on graphical tools for processing your files. Figure 15 shows off KWave's low-pass filter editor, complete with a processing preview function. The Listen button loop plays your file or selection while you adjust the filter in real time, a handy feature for testing effect parameters.

Some of my favorite tools, such as the additive synthesis generator, from the original KWave have not yet been reimplemented. Those tools currently are grayed-out in the menus, but the developers plan to restore those functions and add new features. Like GNUsound, KWave's file size is limited to available memory, but it otherwise is a fine editor and is well suited for casual use on the KDE desktop.



Length: 49.871 s (2,199,328 samples) Mode: 44.100

Figure 15. KWave

Final Remarks

I hope this article has stimulated interest in checking out some of these applications. Believe it or not, other soundfile editors are available, though I have tried here to focus on the most popular ones. The Soundfile Editors section of the Linux Sound & MIDI Applications site has a full listing of the available Linux soundfile editors (see the on-line Resources section).

So which one is right for you? It's hard to say. I'm partial to Snd for its vast programmability and to ReZound for its GUI and organization, but you have to try some and see which fit your needs best. Above all, don't be intimidated by the apparent complexity of some of these editors. Approach them as you would The GIMP, testing their features at random until you have a sense of what they can do—don't be shy about clicking on the Undo button. Playing around with this kind of software can be fun and open some interesting creative avenues. And if you come up with some sounds to share, feel free to let me know about it. Now, go forth and edit those joyful noises.

Resources for this article: /article/7506.

Dave Phillips is a musician, teacher and writer living in Findlay, Ohio. He has been an active member of the Linux audio community since his first contact with Linux in 1995. He is the author of *The Book of Linux Music & Sound*, as well as numerous articles in *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Writing a Real Driver—In User Space

Greg Kroah-Hartman

Issue #122, June 2004

Now you can control USB hardware without touching the kernel and even make your driver run on BSD-based OSes with no code changes. Greg shows a working example for using the cross-platform USB library, libusb.

Last time we discussed how to create a simple USB driver that controlled a USB LED device manufactured by Delcom Engineering [*LJ*, April 2004]. I would like to thank all of the readers who have given me feedback on the column. It even enabled one reader to write a driver now in the main kernel tree. I also would like to thank everyone who has given me ideas about what kinds of devices to write about in future columns, but please remember, let's try to cover simple devices. We don't have the room here to go over how to reverse engineer a streaming video camera that has about 12 different modes of operation.

usbfs Overview

At the end of the last column, I said it is simple to talk to a USB device from user space, without needing a custom kernel driver. Way back during the development of the original Linux USB code, a few developers recognized that it would be wise to allow user programs to get at the raw USB data in all devices in addition to controlling the device. To that end, the usbfs filesystem was created. It originally was called usbdevfs, but many people confused it with the devfs filesystem and all of the nasty baggage that filesystem brought on, so it was renamed usbfs.

Traditionally, usbfs is mounted in the /proc/bus/usb directory on your machine. In that main directory exists a file called devices and a directory for every different USB bus connected to the machine. Those bus directories are named with a number that corresponds to the number the kernel has given that particular USB bus. In each bus directory is a file for every different USB device connected to the bus. For example, a box that has six different USB buses and a few USB devices connected might look like this:

```
$ tree /proc/bus/usb/
/proc/bus/usb/
|-- 001
|   `-- 001
|-- 002
|   `-- 001
|-- 003
|   `-- 001
|-- 004
|   |-- 001
|   |-- 002
|   `-- 003
|-- 005
|   `-- 001
|-- 006
|   `-- 001
`-- devices
```

If you do not have any USB host controller drivers loaded, the main /proc/bus/usb/ directory should be empty.

The /proc/bus/usb/devices file contains a list of all USB devices attached at that moment in time. It also shows how the devices are connected to one another and a lot of other USB-specific information about each device. For details on how the data in this file should be interpreted, see the documentation in the kernel tree at Documentation/usb/proc_usb_info.txt. Programs such as usbview or usbtree use this file to show information about USB devices.

usbfs Device Files

The files within the /proc/bus/usb/BBB/ directories, where BBB is the number of the USB bus, allow programs to talk directly to the different USB devices. The name of the files are the same number as the USB number assigned to the device: 001 for the first device, 002 for the second and so on. Do not rely on these numbers to be unique, as the USB subsystem reuses the numbers when devices are removed and added. If device 003 is removed and another, different device is added, it gets the 003 number.

If you read from the device file, the raw USB descriptor is returned—first the USB device descriptor and then the different configuration descriptors. For a detailed description of what format these descriptors are in and what all of the data means, see the USB specification, which is available for download at the USB Developers Web site (www.usb.org/developers).

The device file also supports a wide range of ioctl calls that allows programs to send and receive USB data from the device. These ioctls and the structures needed for the ioctls are described in the kernel file include/linux/usbdevice_fs.h.

Armed with these ioctls, the structures defined in this header file and a copy of the USB specification, we are set to write a user-space program to talk to our device. But do we really want to do this? Wouldn't it be great if someone wrote a library on top of this interface that would enable us to write sane programs? Luckily, a group of developers has created such a library, allowing programmers to ignore the ioctl mess that usbfs uses. This library is called libusb.

libusb

libusb is a library that works on a number of different operating systems: Linux, the various BSDs and Mac OS X. It allows programs to be written in a portable manner and yet still control USB devices on vastly different operating systems. Using this library lets us create a program to control the USB LED device. libusb can be downloaded from libusb.sf.net if it is not included in your Linux distribution.

The first thing any libusb program must do is initialize the library and have it scan all USB buses for all USB devices. This is done with the following three function calls:

```
usb_init(); usb_find_busses(); usb_find_devices();
```

After the call is complete, the program needs to find a USB device that matches the desired description. As all USB devices have unique vendor and product identification values, it usually is easiest to look for these values. As we remember from the kernel driver we created last time, the USB LED device has the following vendor and product values:

```
#define LED_VENDOR_ID   0x0fc5
#define LED_PRODUCT_ID  0x1223
```

With this information, the code to find this device using libusb is the following:

```
for (usb_bus = usb_busses; usb_bus;
     usb_bus = usb_bus->next) {
        for (dev = usb_bus->devices; dev;
             dev = dev->next) {
            if ((dev->descriptor.idVendor ==
                 LED_VENDOR_ID) &&
                (dev->descriptor.idProduct ==
                 LED_PRODUCT_ID))
                return dev;
        }
}
return NULL;
```

If the device is found in the system, a pointer to it is returned, otherwise NULL is returned. This pointer is of type struct usb_device. After this structure is found, the USB device must be opened and a handle must be created by libusb for the program to communicate with the device. This is done with the following simple code:

```
usb_handle = usb_open(usb_dev);
if (usb_handle == NULL) {
    fprintf(stderr,
            "Not able to claim the USB device\n");
    goto exit;
}
```

This usb_handle variable is of type struct usb_dev_handle, and it is what libusb uses to determine with which USB device it should communicate. This handle is all that is needed to set up our USB device to be ready to communicate with it.

When the program is finished with the USB device, a call to `usb_close(usb_handle);` is all that is necessary to clean up all of our structures and notify libusb that the device is no longer needed.

Changing Colors

Last time we set the color of the USB LED device from within our kernel driver with the following code:

```
usb_control_msg(led->udev,
                usb_sndctrlpipe(led->udev, 0),
                0x12,
                0xc8,
                (0x02 * 0x100) + 0x0a,
                (0x00 * 0x100) + color,
                buffer,
                8,
                2 * HZ);
```

libusb offers us an almost identical function call to send control messages to a USB device. It also is called usb_control_msg(), and to send the same type of color message as we did from within the kernel, our user-space program does it like this:

```
usb_control_msg(handle,
                0xc8,
                0x12,
                (0x02 * 0x100) + 0x0a,
                (0c00 * 0x100) + color,
                buffer,
                8,
                5000);
```

Other than the request type and request variables being reversed from the kernel function call, it looks identical.

Using libusb cuts down extremely the complexities of writing to a USB device, and it gives us a cross-platform program that is much better than a specific kernel driver for most devices.

Listing 1. Controlling a USB LED Device

```
/*
 * Set LED - program to control a USB LED device
 * from user space using libusb
 *
 * Copyright (C) 2004
 *     Greg Kroah-Hartman (greg@kroah.com)
 *
 * This program is free software; you can
 * redistribute it and/or modify it under the terms
 * of the GNU General Public License as published by
 * the Free Software Foundation, version 2 of the
 * License.
 *
 */
#include <stdio.h>
#include <string.h>
#include <usb.h>

#define NONE    0x00
#define BLUE    0x04
#define RED     0x02
#define GREEN   0x01


#define LED_VENDOR_ID   0x0fc5
#define LED_PRODUCT_ID  0x1223

static void change_color
        (struct usb_dev_handle *handle,
         unsigned char color)
{
    char *dummy;

    usb_control_msg(handle,
                    0x000000c8,
                    0x00000012,
                    (0x02 * 0x100) + 0x0a,
                    0xff & (~color),
                    dummy,
                    0x00000008,
                    5000);
}

static struct usb_device *device_init(void)
{
    struct usb_bus *usb_bus;
    struct usb_device *dev;

    usb_init();
    usb_find_busses();
    usb_find_devices();

    for (usb_bus = usb_busses;
         usb_bus;
         usb_bus = usb_bus->next) {
        for (dev = usb_bus->devices;
             dev;
             dev = dev->next) {
            if ((dev->descriptor.idVendor
                  == LED_VENDOR_ID) &&
                (dev->descriptor.idProduct
                  == LED_PRODUCT_ID))
                return dev;
        }
    }
    return NULL;
}

int main(int argc, char **argv)
{
    struct usb_device *usb_dev;
    struct usb_dev_handle *usb_handle;
    int retval = 1;
    int i;
    unsigned char color = NONE;

    usb_dev = device_init();
    if (usb_dev == NULL) {
        fprintf(stderr, "Device not foundn\n");
        goto exit;
    }

    usb_handle = usb_open(usb_dev);
    if (usb_handle == NULL) {
        fprintf(stderr,
        goto exit;
    }

    usb_handle = usb_open(usb_dev);
    if (usb_handle == NULL) {
        fprintf(stderr,
             "Not able to claim the USB device\n");
        goto exit;
    }

    if (argc == 1) {
        fprintf(stderr,
                "specify at least 1 color\n");
        goto exit;
    }

    for (i = 1; i < argc; ++i) {
        if (strcasecmp(argv[i], "red") == 0)
            color |= RED;
        if (strcasecmp(argv[i], "blue") == 0)
            color |= BLUE;
        if (strcasecmp(argv[i], "green") == 0)
            color |= GREEN;
        if (strcasecmp(argv[i], "none") == 0)
            color = NONE;
    }
    change_color(usb_handle, color);
    retval = 0;

exit:
    usb_close(usb_handle);
    return retval;
}
```

Listing 1 allows any mixture of the three possible colors this device offers to be set. Simply pass the colors as command-line arguments to make changes:

```
To set the red led:
        set_led red
To set the green and blue led:
        set_led green blue
To turn off all leds:
        set_led none
```

Conclusion

I hope that this example encourages you to experiment with libusb as a simple alternative to writing a kernel driver. USB devices almost always can be controlled properly with user-space programs instead of specialized kernel drivers. User-space programs using libusb are much easier to debug, do not require a special kernel version to be used and work across a wide range of operating systems.

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

At the Forge

XOOPS

Reuven M. Lerner

Issue #122, June 2004

Create a community site with an article queue, comments, user blogs and more.

Over the last few months, we've been looking at content management software (CMS) that makes it easy to create Weblogs, personal journals that continue to grow in popularity and influence. There are no hard-and-fast rules as to what constitutes a Weblog (or blog), but most people probably would agree that the contents typically are short articles, with the newest ones displayed first.

But there is a larger, slightly different, class of Weblogs out there as well, which probably can be called Slashdot-style, for lack of a better term. These Weblogs are edited by a small group of people but have a large number of contributors. In many ways, these are closer to on-line forums or bulletin boards than to Weblogs. However, such software also provides Weblog capabilities for its individual members. That is, a visitor to the site can read the main discussion, create his or her own Weblog and read through other users' Weblogs as well.

Although some packages—most notably Slash, which powers Slashdot, use.perl.org and some other high-profile sites—were designed around the idea of public and private Weblogs, other packages have grown to incorporate this idea. One such package is XOOPS, which supposedly stands for Extended Object Oriented Portal System. XOOPS is an outgrowth of the PHPNuke and PostNuke family of portal servers.

This month, we look at XOOPS, paying particular attention at WeBLog (pronounced Wee-Bee-Log), a module designed to facilitate the creation and administration of public and private Weblogs. I have reservations about XOOPS, but it can be an excellent tool for a certain type of audience. In particular, the ease with which it can be installed and administered, and with which each user can be granted his or her own Weblog, is hard to beat.

History

XOOPS testifies to the power of the GNU General Public License, to the Open Source community and to the fact that not all forks or splits in an open-source project are bad. The XOOPS story begins with PHPNuke, a suite of programs written in PHP and using MySQL for its database back end. PHPNuke apparently was built on top of ThatWare, a now-abandoned Slash clone written in PHP.

What happened next is controversial, to say the least, and demonstrates what can happen when an open-source project gets ugly: a large number of PHPNuke contributors decided to fork the project and named their version PostNuke. Today, PHPNuke and PostNuke both exist, and little love is lost between the user and development communities. From what I can tell, and without examining the code in any depth, PHPNuke mainly continues to be developed by Francisco Burzi, with the main criticisms having to do with his management style, the hobbyist orientation of PHPNuke modules and issues having to do with security. Although PostNuke is quite popular and seems to have the edge on security, administration, commercial-oriented modules and database-independence, maintainers are accused of changing APIs too frequently and of designing PostNuke to be too resource-intensive. Meanwhile, some of the central PostNuke developers have left that project, forking it into software that originally was known as LostNuke and now goes by the name of Xaraya.

Finally, XOOPS seems (from what I can tell) to be an object-oriented rewrite of PHPNuke. The XOOPS maintainers believe that by rewriting the system to use objects, it is more secure and easier to develop and maintain.

The good news is that all of these systems are available under the GNU General Public License. That said, forks generally are seen as a last resort in the Open Source community—and the fact that so many have happened on this software probably testifies to the lack of strong management and leadership of these projects. Reviewing the history of these projects renewed my admiration for people like Linus Torvalds, Larry Wall and Guido von Rossum, all of whom have managed to keep their communities together, despite differences of opinion among the core developers.

XOOPS seems to have a smaller community than either PHPNuke or PostNuke. However, the number of built-in and third-party modules is quite large, and its object-oriented facilities mean it is fairly easy to build your own module. Indeed, although the number of modules has remained fairly modest, I was impressed by how easy it was to install. Does this mean you should use XOOPS over another one of the Nuke-alikes? That seems to depend as much on politics and your planned use as on the technology itself, although the fact that XOOPS uses objects does give it an edge, at least in theory, from my experience.

Downloading and Installing

Now that we have reviewed the historical evolution of XOOPS and related pieces of software, let's set up our own XOOPS server. In order to run XOOPS, you need a MySQL server—I am running the admittedly old version of 3.23.58 on my workstation—and a copy of Apache (version 1 or 2) with PHP installed.

Installing XOOPS is one of the most impressive and painless software installs I have done in a long time. I downloaded the code from the XOOPS Web site (see the on-line Resources section) and opened up the .tar.gz file in a temporary directory. I then copied the entire HTML subdirectory into my Apache content root, as /usr/local/apache/html/xoops. Once I had done that, I was able to access the XOOPS installer at the URL /xoops on my system.

The installer is fully Web-based and walks you through the process slowly but surely. The most confusing part probably was the page on which you need to enter the user name and password of a MySQL user who is authorized to create a new database and tables. The installer also could be a bit more verbose in its error messages at this initial phase, given that many people installing XOOPS are unlikely to be experienced Web/database programmers.

That said, the installation was fairly simple and painless. I returned to the / xoops URL on my system and was greeted with a simple introductory screen that invited me to log in. I logged in as the site administrator, and although the main screen was relatively blank, I was able to move quickly to the Administration menu on the left side of the screen (Figure 1).

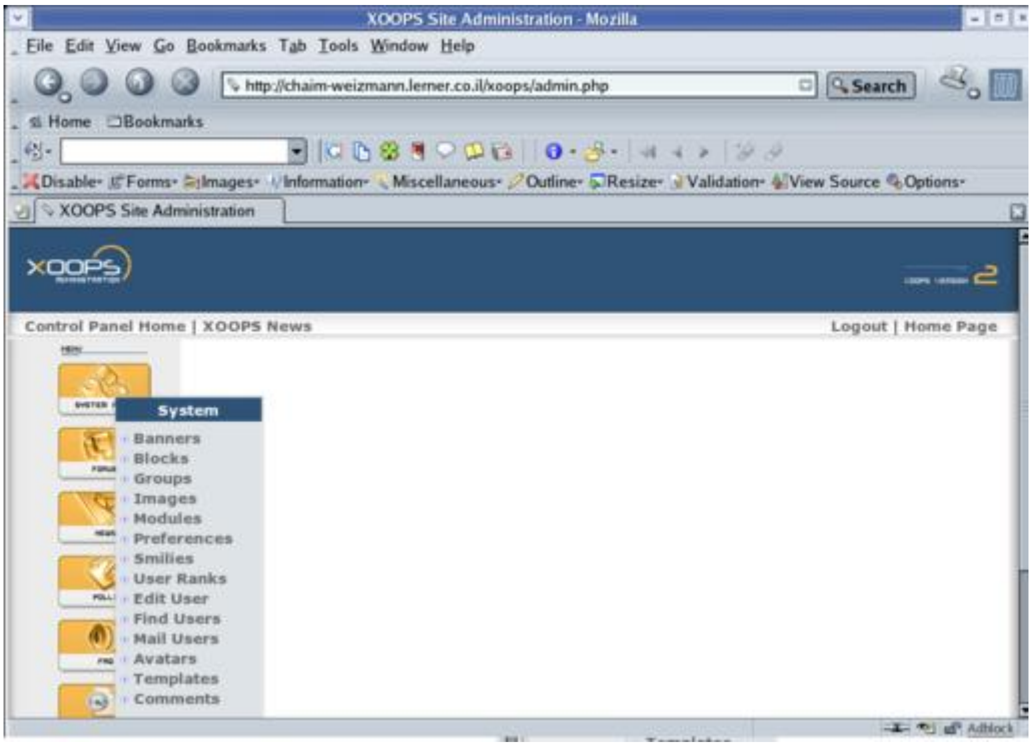
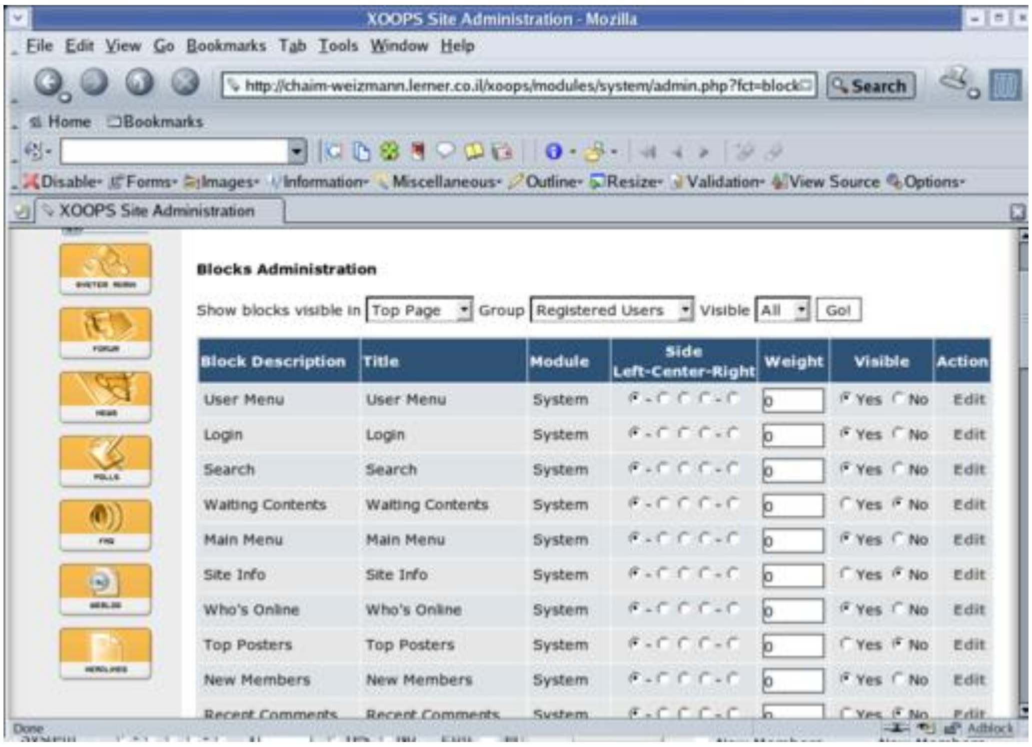

XOOPS Site Administration - Mozilla
File Edit View Go Bookmarks Tab Tools Window Help
http://chaim-weizmann.lerner.co.il/xoops/admin.php
Search
Home Bookmarks
Disable Forms Images Information Miscellaneous Outline Resize Validation View Source Options
XOOPS Site Administration
Control Panel Home | XOOPS News
Logout | Home Page
System
Banners
Blocks
Groups
Images
Modules
Preferences
Smilies
User Ranks
Edit User
Find Users
Mail Users
Avatars
Templates
Comments

Figure 1. The XOOPS site administration page is where you select the modules you want to appear on your site.

Administering XOOPS is fairly straightforward, once you understand that everything in XOOPS is a module. The first order of business for most XOOPS administrators then is to install additional modules. Go to the Administration menu and choose Modules. The top of the screen lists the modules already installed, and the bottom of the screen lists those available but uninstalled.

To install an available module, click on the appropriate icon next to the module's name. After you confirm your choice, you should see an indication of which tables are being created in the MySQL database, what access permissions have been added to the system and whether the module was installed correctly. Assuming that it was, you can return to the Module Administration page, where you should see the newly installed module at the top of the screen rather than at the bottom.

The default layout of a XOOPS site is rather empty. To include additional items on the home page, you must go to the Blocks administrative page, which lists the content features each module includes. The News module includes three blocks—bigstory, topics and top. The main Blocks Administration menu lets the site manager indicate which blocks are visible. But the real power happens in the Edit menu associated with each block, allowing you to indicate where on the screen the block should appear—on the side, in the center or somewhere else entirely.


XOOPS Site Administration - Mozilla

http://chaim-weizmann.lerner.co.il/xoops/modules/system/admin.php?fct=block

Blocks Administration

Show blocks visible in Top Page Group Registered Users Visible All Go!

Block Description	Title	Module	Side Left-Center-Right	Weight	Visible	Action
User Menu	User Menu	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Login	Login	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Search	Search	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Waiting Contents	Waiting Contents	System	● - ○ ○ ○ - ○	0	Yes [✓ No]	Edit
Main Menu	Main Menu	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Site Info	Site Info	System	● - ○ ○ ○ - ○	0	Yes [✓ No]	Edit
Who's Online	Who's Online	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Top Posters	Top Posters	System	● - ○ ○ ○ - ○	0	Yes [✓ No]	Edit
New Members	New Members	System	● - ○ ○ ○ - ○	0	[✓ Yes] No	Edit
Recent Comments	Recent Comments	System	● - ○ ○ ○ - ○	0	Yes [✓ No]	Edit

Figure 2. Create a unique site layout by positioning blocks from the Blocks Administration menu.

Modules

Each XOOPS module packages together HTML, PHP and SQL into a defined list of directories that is consistent across modules. For example, you can always expect to find class definitions in the class directory, database definitions in the SQL directory, blocks in the blocks directory and images in the images directory. Configuration and version information is always in a xoops_version.php file in the module's root directory.

Installing a new module is as easy as downloading it and placing its top-level directory in the modules directory. So, if you download module foo from the Internet, and if your XOOPS directory is /usr/local/apache/html/xoops, you simply can move the foo directory to /usr/local/apache/html/xoops/modules. The next time you visit the Module Administration page, the module is there, ready for you to install and configure it.

One important issue the XOOPS developers have kept in mind is that of database upgrades. It is easy to imagine how the PHP and HTML portions of a XOOPS module would be upgraded. Upgrading a module that combines PHP with SQL is a tougher nut to crack, and it seems as though the XOOPS people have done an admirable job. The SQL directory in a XOOPS module may contain not only a mysql.sql file with MySQL definitions but also update files with ALTER TABLE commands that change the table definitions from one version to the next. Because each module's tables are insulated from the rest of the system, you should be able to upgrade XOOPS modules independently of one another and in nearly any order.

WeBLog

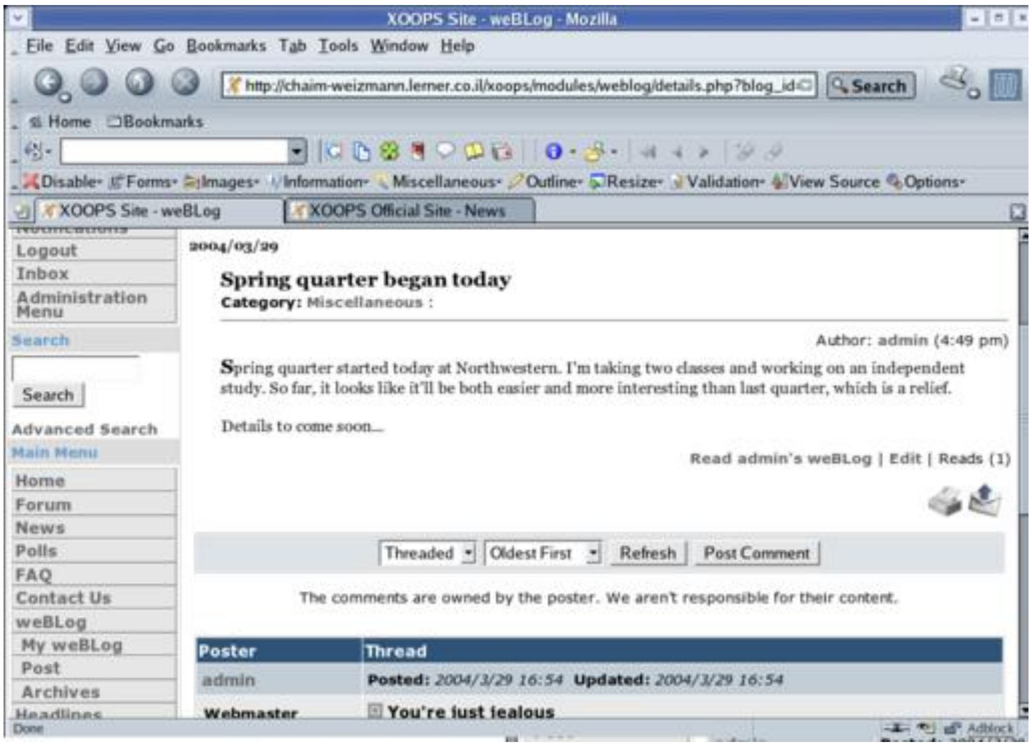
Now that we have gotten a taste of how to work with XOOPS modules, we are going to install the WeBLog module, written by Hiroyuki Sakai. This not only gives us a chance to download and install a third-party XOOPS module, but it also provides the functionality we originally were searching for, namely a Weblog system that is good for a community of users.

To install WeBLog, download the file from the main site (see Resources). As with most XOOPS modules, you should unpack it with `tar zxf`, and then copy or move the top-level weblog directory into your XOOPS modules directory. On my system, WeBLog was installed as /usr/local/apache/html/xoops/modules/weblog.

To install the module, simply go to the Module Administration page and click on install. You now can create and administer Weblogs of various sorts. The Module Administration page displays a new Weblog icon on the left side, allowing you to set permissions (which users are allowed to create and post to Weblogs), categories and even perform some basic database maintenance.

Now that you have made it possible to create a Weblog, return to your site's home page. You should see a WeBLog menu option on the left side of the screen with a collection of the most recent items from all of the Weblogs on the system. If this is your first time running the module, this page is mostly empty. You can create or modify your own Weblog by clicking on My WeBLog. Enter your posting in the appropriate parts of the HTML form, inserting at least 75 characters, unless you have changed this option in the Preferences Administration page. Once you have posted your new Weblog entry, you can view it by returning to the WeBLog menu item. That global viewing screen collects all of the individual Weblogs, but it also allows you to click on a user's name and view only his or her Weblog.

WeBLog provides a number of other useful features, ranging from RSS syndication to searchable and browseable archives, to the ability to comment on a posting. You also can ask to be notified when a new posting is added. Such notifications are placed in the XOOPS message system and are flagged on the main menu when you enter the site.


XOOPS Site - weBLog - Mozilla
File Edit View Go Bookmarks Tab Tools Window Help
http://chaim-weizmann.lerner.co.il/xoops/modules/weblog/details.php?blog_id=
Search
Home Bookmarks
Disable Forms Images Information Miscellaneous Outline Resize Validation View Source Options
XOOPS Site - weBLog
XOOPS Official Site - News
Logout
Inbox
Administration Menu
Search
Search
Advanced Search
Main Menu
Home
Forum
News
Polls
FAQ
Contact Us
weBLog
My weBLog
Post
Archives
Threaded
Oldest First
Refresh
Post Comment
Done

2004/03/29

Spring quarter began today

Category: Miscellaneous :

Author: admin (4:49 pm)

Spring quarter started today at Northwestern. I'm taking two classes and working on an independent study. So far, it looks like it'll be both easier and more interesting than last quarter, which is a relief.

Details to come soon...

Read admin's weBLog | Edit | Reads (1)

The comments are owned by the poster. We aren't responsible for their content.

Poster	Thread
admin	**Posted:** 2004/3/29 16:54 **Updated:** 2004/3/29 16:54
Webmaster	**You're just jealous**

Figure 3. Success! With the site configured, watch the user blog entries and comments flow in.

Should You Use XOOPS?

If you want to create an on-line community of people with Weblogs, you could do much worse than XOOPS. It is supported by an active development community, and the WeBLog module appears to be stable and well maintained. Furthermore, the technology is simple and straightforward enough that those with Web/database experience should be able to modify and maintain the system on their own.

As I explored XOOPS, however, I could not help but compare it with OpenACS, another community-oriented Web system that has a Weblogs module. Is there any reason why someone would prefer to work with XOOPS rather than OpenACS, which arguably is technically superior?

As much as I love OpenACS, the answer is a firm yes. OpenACS is a great system when hard-core Web/database hackers are around, when you have root access to your own server and when you can turn the OpenACS toolkit into a workable customized solution for your needs. But many people and projects don't have the resources or the budget for such high-powered expertise; they are looking to accomplish things quickly and easily on a shared server that provides access to PHP and MySQL. For such people, XOOPS may be just the ticket. It lacks a number of features I would like to see, but it can be installed anywhere, maintained by the large community of PHP/MySQL programmers and administered by non-programmers with a minimum of training. It doesn't have many features I would expect in a more sophisticated system, such as Unicode support, multiple instances of a single package, content versioning and hierarchical groups. But it does do a good job for what it was intended to do, and the community of XOOPS users seems ready to help anyone who has issues and problems.

I find it hard to classify XOOPS as a content management system in the traditional sense, despite what the package maintainers claim. Yes, it allows you to post news articles and to have those approved by a site administrator before they go live. But I think XOOPS is better thought of as an easy-to-use community system that makes it simple to create Weblogs and other standardized content. For many people, that is more than enough, and the ease with which they can create relatively sophisticated sites outweighs the other considerations.

Conclusion

XOOPS is definitely not for everyone—the politics, the object-oriented code and the lack of certain features I consider vital make me think twice about recommending it. But I also have seen the time and effort it takes to get up to speed on such technologies as Zope and OpenACS, and the trade-off has not always been worthwhile. If you are an expert in PHP and MySQL, and you are looking to create a community site with good Weblog facilities, you might do well to consider XOOPS.

Resources for this article: /article/7500.

Reuven M. Lerner, a longtime consultant in Web/database programming, now is a graduate student in Learning Sciences at Northwestern University in Evanston, Illinois. You can reach him at reuven@lerner.co.il.

Archive Index Issue Table of Contents

Advanced search

Kernel Korner

udev—Persistent Device Naming in User Space

Greg Kroah-Hartman

Issue #122, June 2004

Whether you're plugging a camera and scanner in to your laptop or adding another SCSI drive to your company server, it's time to end the current mess of major and minor numbers.

Starting with the 2.5 kernel, all physical and virtual devices in a system are visible to user space in a hierarchal fashion through sysfs. /sbin/hotplug provides a notification to user space when any device is added or removed from the system. Using these two features, a user-space implementation of a dynamic /dev now is possible that can provide a flexible device naming policy.

This article discusses udev, a program that replaces the functionality of devfs. It provides /dev entries for devices in the system at any moment in time. It also provides features previously unavailable through devfs alone, such as persistent naming for devices when they move around the device tree, a flexible device naming scheme, notification of external systems of device changes and moving all naming policy out of the kernel.

The /dev directory on a Linux machine is where all of the device files for the system should be located. A device file details how a user program can access a specific hardware device or function. For example, the device file /dev/hda traditionally is used to represent the first IDE drive in the system. The name hda corresponds to both a major and a minor number, which are used by the kernel to determine what hardware device to talk to. Currently, a wide range of names that match up to different major and minor numbers has been defined.

All major and minor numbers are assigned a name that matches up with a type of device. This allocation is done by the Linux assigned names and numbers authority (LANANA), and the current device list can be found on its Web site (see the on-line Resources section).

As Linux begins supporting new kinds of devices, these devices need to be assigned a major and minor number range in order for users to access them through the /dev directory. An alternative is to provide access through a filesystem; my 2002 linux.conf.au paper provides more details on how to do this (see the on-line Resources section). In kernel versions 2.4 and earlier, the valid range of major numbers was 1–255, and the valid range of minor numbers was 1–255. Because of this limited range, a freeze was placed on allocating new major and minor numbers during the 2.3 development cycle. This freeze since has been lifted, and the 2.6 kernel had the valid range of major numbers increased to 4,095. More than a million minor numbers are available per major number.

What /dev Entry Is Which Device

When the kernel finds a new piece of hardware, it typically assigns the next major/minor pair for that kind of hardware to the device. So, on boot, the first USB printer found would be assigned the major number 180 and minor number 0, which is referenced in /dev as /dev/usb/lp0. The second USB printer would be assigned major number 180 and minor number 1, which is referenced in /dev as /dev/usb/lp1. If the user rearranges the USB topology, perhaps adding a USB hub to support more USB devices in the system, the USB probing order of the printers might change the next time the computer is booted, reversing the assignment of the different minor numbers to the two printers.

This same situation holds true for almost any kind of device that can be removed or added while the computer is powered up. With the advent of PCI hot-plug-enabled systems and hot-pluggable buses, such as IEEE 1394, USB and CardBus, almost all devices have this problem.

With the advent of the sysfs filesystem in the 2.5 kernel, the problem of determining which device minor number is assigned to which physical device is much easier to determine. For a system having two different USB printers plugged in to it, the sysfs /sys/class/usb directory tree would look like this:

```
/sys/class/usb/
|-- lp0
|   |-- dev
|   |-- device -> ../../../devices/pci0/00:09.0/usb1/1-1/1-1:0
|   `-- driver -> ../../../bus/usb/drivers/usblp
`-- lp1
    |-- dev
    |-- device -> ../../../devices/pci0/00:0d.0/usb3/3-1/3-1:0
    `-- driver -> ../../../bus/usb/drivers/usblp

$ cat /sys/class/usb/lp0/device/serial
HXOLL0012202323480
$ cat /sys/class/usb/lp1/device/serial
W09090207101241330
```

Within the individual USB device directories pointed to by the lp0/device and lp1/device symbolic links, a lot of USB-specific information can be determined, such as the manufacturer of the device and the (hopefully unique) serial number.

As can be seen by the serial files in the above description, the /dev/usb/lp0 device file is associated with the USB printer with serial number HXOLL0012202323480, and the /dev/usb/lp1 device file is associated with the USB printer with serial number W09090207101241330. If these printers are moved around, say by placing them both behind a USB hub, they might be renamed, as they are probed in a different order on startup:

```
$ tree /sys/class/usb/
/sys/class/usb/
|-- lp0
|   |-- dev
|   |-- device -> ../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/1-1.1:0
|   `-- driver -> ../../../bus/usb/drivers/usblp
`-- lp1
    |-- dev
    |-- device -> ../../../devices/pci0/00:09.0/usb1/1-1/1-1.4/1-1.4:0
    `-- driver -> ../../../bus/usb/drivers/usblp

$ cat /sys/class/usb/lp0/device/serial
W09090207101241330
$ cat /sys/class/usb/lp1/device/serial
HXOLL0012202323480
```

As this description shows, the /dev/usb/lp0 device now is assigned to the USB printer with the serial number W09090207101241330 due to this different probing order.

sysfs enables a user to determine which device has been assigned by the kernel to which device file. This is a powerful association that previously had not been easily available. However, a user generally does not care that /dev/usb/lp0 and /dev/usb/lp1 are now reversed and should be changed in a configuration file somewhere. The user simply wants to be able to print to the proper printer, no matter where it is in the USB device tree.

/dev Is Too Big

Not all device files in the /dev directory of most distributions match up to a physical device that is currently connected to the computer. Instead, the /dev directory is created when the operating system is initialized on the machine, populating the /dev directory with all known possible names. On a machine running Red Hat's Fedora release 1, the /dev directory holds more than 18,000 different entries. This many entries soon become unwieldy for users trying to determine exactly what devices currently are present.

devfs

Because of the large numbers of device files in the /dev directory, a number of operating systems have moved to having the kernel itself manage the /dev directory, as the kernel always knows exactly what devices are present on the system. It does this by creating a RAM-based filesystem called devfs. Linux also has this option, and it has become popular over time in a number of different distributions, including Gentoo.

For a number of people, devfs solves their immediate needs. However, the Linux-based devfs implementation still has a number of unsolved problems. Most notably, it does not provide the ability to create device nodes with a persistent name.

udev's goals

In light of the previously mentioned problems, the udev Project was started. Its goals are to run in user space; create a dynamic /dev; provide consistent device naming, if wanted; and provide a user-space API to access information about current system devices. For more on how udev compares with devfs, see the on-line Resources section.

The first item, run in user space, is accomplished by harnessing the fact that /sbin/hotplug generates an event for every device added to or removed from the system with sysfs' ability to show all the needed information about all devices.

The second item, create a dynamic /dev, is handled by catching all /sbin/hotplug events, looking up the major and minor number in sysfs for the added device and creating a /dev file with the kernel name the device was assigned. If the device was removed from the system, it is easy to remove the /dev entry for that device.

udev achieved these first two goals back in April 2003 in an extremely tiny 6Kb of compiled code, proving that this scheme of catching hot-plug events and using sysfs was feasible and quite simple to implement. Since that humble beginning in early 2003, udev has achieved all of its goals. It provides users with the ability to name devices in a persistent manner using a flexible rule-based system.

udev's rules are contained in the /etc/udev/udev.rules file and describe any devices the user wants to name in a way that differs from the default kernel name. Here's an example of a udev.rules file:

```
# if /sbin/scsi_id returns "OEM 0815" device will
# be called disk1
BUS="scsi", PROGRAM="/sbin/scsi_id", \
RESULT="OEM 0815", NAME="disk1"

# USB printer to be called lp_color
BUS="usb", SYSFS_serial="W09090207101241330", \
NAME="lp_color"

# SCSI disk with a specific vendor and model number
# is to be called boot
BUS="scsi", SYSFS_vendor="IBM", \
SYSFS_model="ST336", NAME="boot"

# sound card with PCI bus id 00:0b.0 to be called dsp
BUS="pci", ID="00:0b.0", NAME="dsp"

# USB mouse at third port of the second hub to
# be called mouse1
BUS="usb", PLACE="2.3", NAME="mouse1"

# ttyUSB1 should always be called pda with two
# additional symlinks
KERNEL="ttyUSB1", NAME="pda", \
SYMLINK="palmtop handheld"

# multiple USB webcams with symlinks to be called
# webcam0, webcam1, ...
BUS="usb", SYSFS_model="XV3", NAME="video%n", \
SYMLINK="webcam%n"
```

A udev rule defines the mapping between a device's attributes and the desired device filename. To do this, a number of keys can be queried from the device to determine a match. If no match is found in the udev.rules file, the default kernel name is used. Below is a list of the different types of keys that udev understands:

- BUS: matches the bus type of the device; examples of this include PCI, USB or SCSI.
- KERNEL: matches the name the kernel gives the device.
- ID: matches the device number on the bus; for example, the PCI bus ID or the USB device ID.
- PLACE: matches the topological position on bus, such as the physical port a USB device is plugged in to.
- SYSFS_filename, SYSFS{filename}: allows udev to match any sysfs device attribute, such as label, vendor, USB serial number or SCSI UUID. Up to five different sysfs files can be checked in a single rule, with all of the values being required in order to match the rule.
- PROGRAM: allows udev to call an external program and check the result. This key is valid if the program returns successfully. The string returned by the program additionally may be matched with the RESULT key.
- RESULT: matches the returned string of the last PROGRAM call. This key may be used in any rule following a PROGRAM call.

After the different keys, a NAME and optional SYMLINK are specified. The NAME is what udev uses to call the device if the rule matches, and the SYMLINK specifies what, if any, symlinks also are generated. More than one symlink can be specified at once, with spaces between multiple symlinks. Both the NAME and SYMLINK files can contain directories to allow /dev to be simplified.

Example

So, back to our two-printer example. To name both of these devices in a consistent manner, the following two udev rules might be used:

```
BUS="usb", SYSFS_serial="W09090207101241330", \
NAME="lp_color"
BUS="usb", SYSFS_serial="HXOLL0012202323480", \
NAME="lp_plain"
```

These rules cause udev to look at the sysfs file serial for both printers and, depending on the value in the file, name the printer either lp_color or lp_plain. This ensures that no matter which device was plugged in first or detected first or whether another USB printer is added to the system, both of the printers have the same persistent name.

Advanced udev Rules

udev allows a number of printf-like string substitutions to be used in the NAME, SYMLINK and PROGRAM fields in the udev.rules file. These fields are:

- %n: kernel number of the device; for example, the device sda3 has a kernel number of 3.
- %k: kernel name for the device.
- %M: kernel major number for the device.
- %m: kernel minor number for the device.
- %b: bus ID for the device.
- %c: PROGRAM returned string. A number can be added to this modifier to pick up only a specific word within the string. This field does not work within the PROGRAM field for the obvious reason.
- %%: % character itself.

Also, a number of the different keys support a simple form of shell-style pattern matching. These patterns are:

- *: matches zero, one or more characters.
- ?: matches any single character but does not match zero characters.
- [ ]: matches any single character specified within the brackets; for example, the pattern string `tty[SR]` would match either `ttyS` or `ttyR`. Ranges also are supported within this match with the - character. For

example, to match on the range of all digits, the pattern [0-9] would be used. If the first character following the [is !, any character not enclosed is matched.

Because of the ability to do these simple string substitutions and string pattern matching, combined with the ability to have udev run any other program and use its result, udev has become an extremely flexible tool for naming devices. As an example of this power, take a look at the following rule:

```
KERNEL="[hs]d[a-z]", PROGRAM="name_cdrom.pl %M %m", \
NAME="%1c", SYMLINK="cdrom"
```

This rule matches any block device and calls the Perl script name_cdrom.pl with the major and minor number of the device. If this program is successful, udev uses the first word of the program's output to name the device and creates a symlink called cdrom. The name_cdrom.pl script can be found in the udev release.

What this script does is determine whether the device is a CD-ROM device. If it is, it queries the Free CDDB database to see if the CD-ROM present in the device is known in the database. If it is, it then names the device based on the CD. For example, my /dev looks like the following when using this rule:

```
$ ls -l /dev/S* /dev/cdrom
brw-------  1 root root 22, 64 Feb 15 08:26 /dev/Samiam-Astray
lrwxrwxrwx  1 root root      8 Feb 15 08:26 /dev/cdrom ->
/dev/Samiam-Astray
```

This shows how udev can query a database across the Internet to determine how to name a device. Yes, it's a crazy naming scheme to try to use, but it shows how powerful and flexible udev can be.

Acknowledgements

The author would like to thank Daniel Stekloff of IBM, who has helped shape the design of udev in many ways. Without his perseverance, udev would not be available at all. Also, Kay Sievers has been instrumental in implementing the majority of the advanced features in udev, most notably the ability to have string modifiers and pattern matching, both of which are essential for using udev in the real world. Without his help, udev would not be anywhere near as powerful and useful as it is.

Also, without Pat Mochel's sysfs and driver model core, udev would not have been possible to implement. The author is indebted to him for undertaking what most thought of as an impossible task and for allowing others to build

easily on his common framework, allowing all users to see the “Web woven by a spider on drugs” that the kernel keeps track of.

This article is based on the Ottawa Linux Symposium 2002 paper on udev (see Resources).

Resources for this article: /article/7496.

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

Cooking with Linux

When Democracy Becomes Crazy!

Marcel Gagné

Issue #122, June 2004

Reader polls are a popular Web site feature. Try one of these simple packages to add a poll to your site.

François, you cannot tell me you had nothing to do with this. The numbers here are skewed highly in your favor. Ordinarily, I would not question your integrity, *mon ami*, but the restaurant just opened and our guests have not arrived yet. How else would you explain the votes you have managed to acquire? *Quoi?* Yes, François, I can see that our guests are here now, but they were not here a few minutes ago. To the wine cellar, *immédiatement*. Bring back the 2001 Sonoma County Zinfandel. We shall discuss this later. *Vite!*

Welcome, everyone to *Chez Marcel*, home of tasty Linux fare, the world's finest wine cellar and, now, the home of electronic democracy in action—Linux-style, of course. While my faithful waiter is in the cellar getting your wine, let me tell you about tonight's menu. Electronic democracy is arguably still in its infancy, but it is making inroads nevertheless. With your Linux system and an Apache Web server, you can launch your own experiment with on-line polling and voting.

Ah, François, you have returned. Please, pour for our guests while I describe our first experiment in electronic democracy and a question that is near and dear to your heart. Democracy. *Mes amis*, you have seen Web sites with on-line polls of various sorts, asking questions ranging from serious to silly. With little effort, you can add a poll to your own Web site. The first item on the menu is Michael and Erin Spiceland's My Voting Script (see the on-line Resources section). To use this one, simply extract the tarball to your cgi-bin directory. On my system, it's /usr/local/apache/cgi-bin:

```
cd /usr/local/apache/cgi-bin
tar -xzvf vote-2.1.tar.gz
```

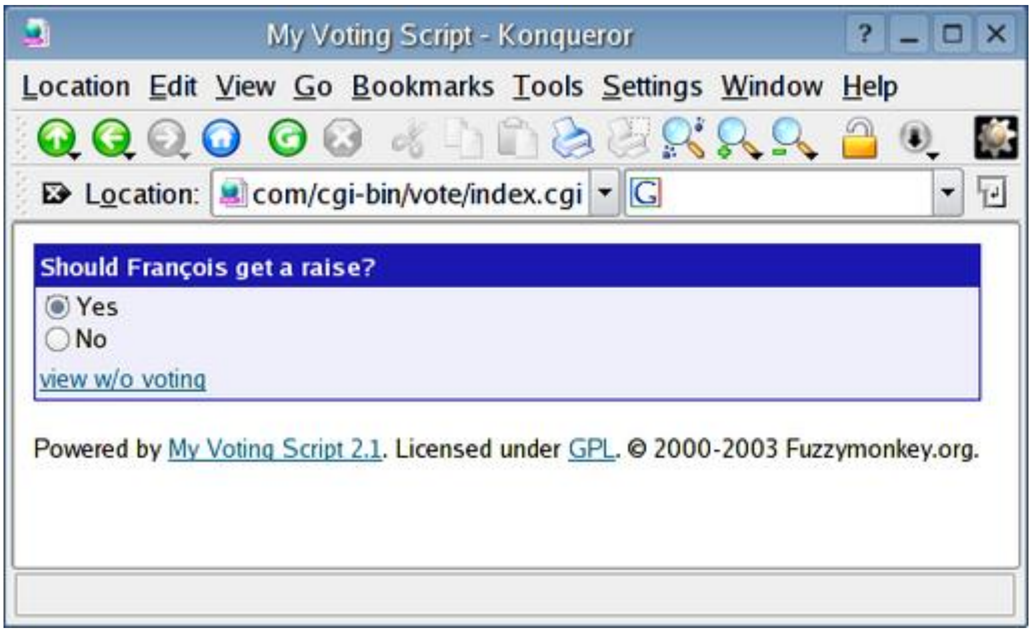
A directory called vote should have been created. Switch to that directory and edit the file called sitevariables.pl. Several parameters can be changed here, including the colors used when the poll is presented to the user. You should take a little time to familiarize yourself with the various tweaks. In the meantime, the things you probably are most interested in are the questions and possible answers:

```
$question = 'Should François get a raise?';
@answers = ('Yes','No');
```

The question doesn't have to be strictly yes or no. You could have something like this as well: `@answers = ('Yes','No','Maybe');`. Before we run this poll and decide François' salary, let's look at a couple of other important parameters:

```
$lockhours = 0;
$url = 'http://yourserver/cgi-bin/vote/index.cgi';
```

The $lockhours parameter defines how long an IP address should wait before another vote may be posted. Set to zero, as above, a user may vote any number of times from the same IP address. Finally, you should make sure that the address to your server and the path to the script are entered properly. To use your new poll, point your browser to the index.cgi file, http://yourserver/cgi-bin/vote/index.cgi. Your poll question should appear in the browser, as shown in Figure 1.


My Voting Script - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: com/cgi-bin/vote/index.cgi

Should François get a raise?

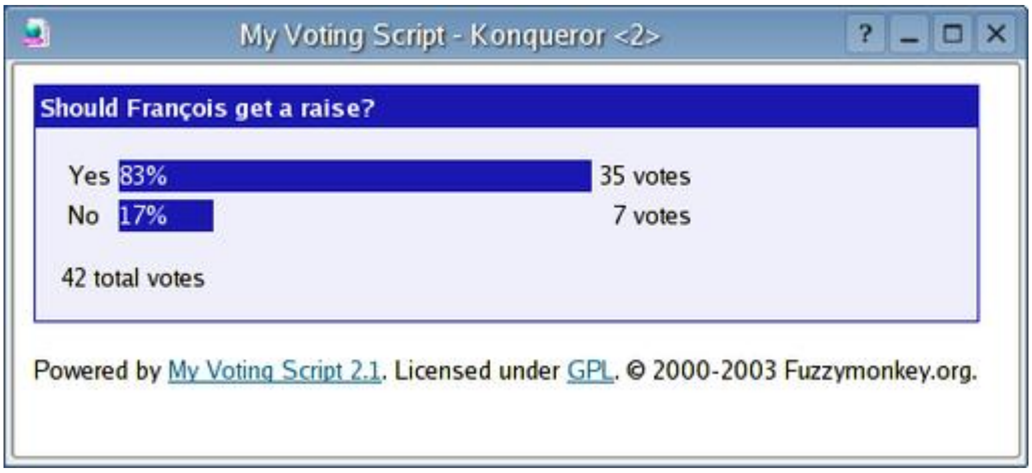
(●) Yes
( ) No

view w/o voting

Powered by My Voting Script 2.1. Licensed under GPL. © 2000-2003 Fuzzymonkey.org.

Figure 1. A Simple My Voting Script Poll

To register a vote, simply click on one of the answers, in this case, Yes or No. When the votes are entered, a pop-up window appears with instant results. A percentage of votes is tabulated along with a colorful bar offering a graphical representation of the results. As you can see in Figure 2, my faithful waiter has managed to garner amazing support for his raise.



Should François get a raise?

Yes	83%	35 votes
No	17%	7 votes

42 total votes

Powered by My Voting Script 2.1. Licensed under GPL. © 2000-2003 Fuzzymonkey.org.

Figure 2. Initially, a raise looks promising.

This script is simple and can handle only a single question at a time. For something a little more flexible, check out David S. Raeman's Sympoll. As the name indicates, this is a simple polling system, but it's simple more from a setup perspective than from functionality. The installation took me a few seconds and then everything simply worked. Nevertheless, this little voting package allows you to run multiple polls that allow for more than one choice per vote, thus adding a level of complexity to the available choices. The polls and related information are stored in a MySQL database file. For this one, you need an Apache server with PHP enabled and, of course, MySQL running on your system.

Extract the Sympoll source (see Resources) to an appropriate directory in your Web server's document hierarchy. For example, my document root is at /usr/local/apache/htdocs, so I extracted the source distribution in that location with `tar -xzvf sympoll-1.5.tar.gz`. Once the directory is in place, make sure that the config.php is writable by your Apache server, `chmod 660 config.php`. Before you head to your browser, I should tell you that the author recommends changing the name of the admin directory to something other than admin to make it a little harder to locate.

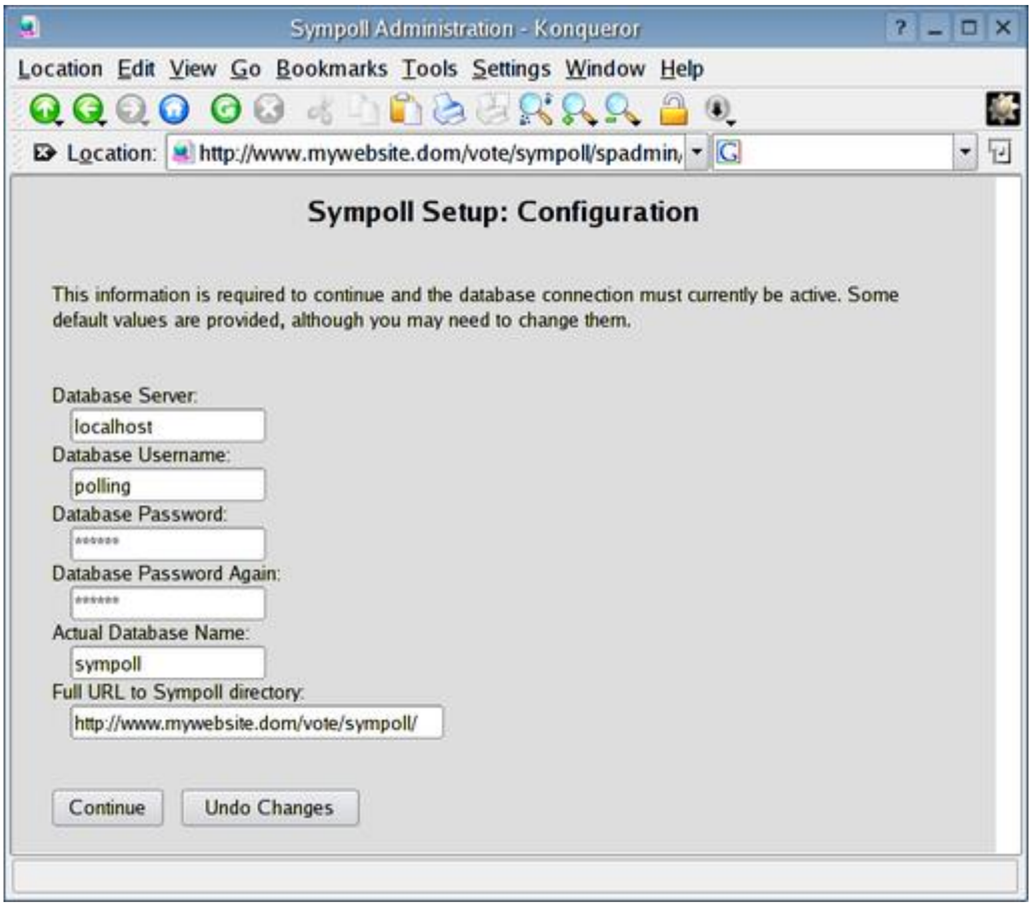
Next, I created a MySQL database called sympoll, the name could be anything you like, with the following command:

```
mysqladmin -u root -p create sympoll
```

I don't want root to be the database administrator for on-line Web access, so I created a new user called polling. I did this by connecting to MySQL with the command `mysql -u root -p`. At the `mysql>` prompt, I entered the following to create a user that would have control over my Sympoll tables:

```
mysql> grant all privileges on sympoll.*
  -> to polling@localhost identified by 'secr3t';
```

When you do this, make sure you use a different user name and password, *non*? We would not want every visitor to this restaurant to provide such easily guessed passwords. The next step is to create the tables in your new database, and this is done through the administrative interface. Point your browser to your server, using the directory where you loaded Sympoll, http://yourhost/sympoll/admin (Figure 3).


Sympoll Administration - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://www.mywebsite.dom/vote/sympoll/spadmin/
Continue
Undo Changes

Sympoll Setup: Configuration

This information is required to continue and the database connection must currently be active. Some default values are provided, although you may need to change them.

Database Server: localhost
Database Username: polling
Database Password: ******
Database Password Again: ******
Actual Database Name: sympoll
Full URL to Sympoll directory: http://www.mywebsite.dom/vote/sympoll/

Figure 3. Sympoll's setup is, well, simple.

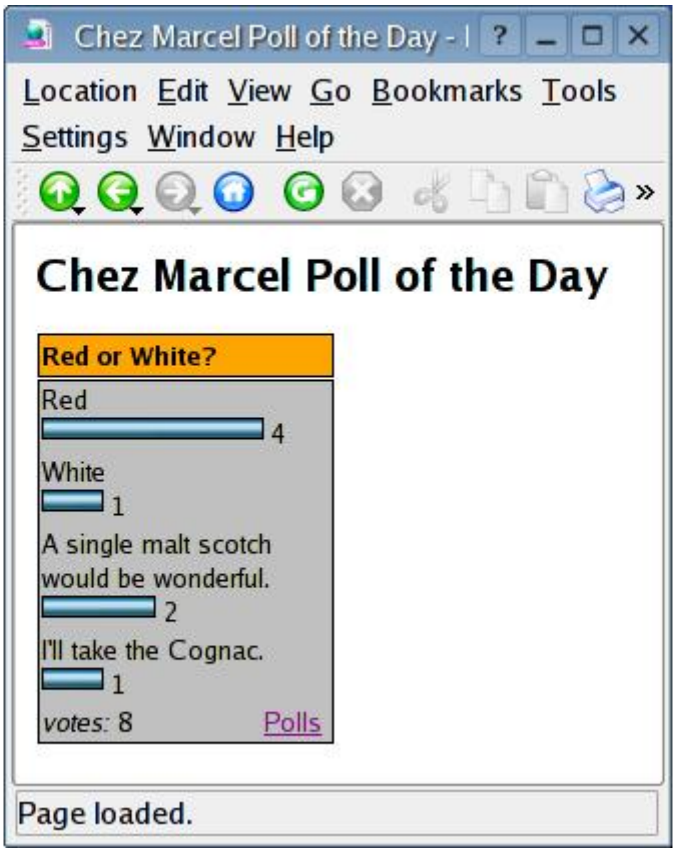
The admin setup is friendly and takes you through the configuration, step by step. It first requires that you enter the MySQL user name and password. Four tables get added to the system (sympoll_data, sympoll_list, sympoll_auth and sympoll_iplog). You are asked to confirm the creation of those tables. Click Continue until you are taken to the super administrator configuration screen. Here, this becomes the Sympoll administrator, the user that creates other admins and regulates polls. Enter a user name and password, confirm your choice of password, and you're done. You now are at the login screen, ready to launch your exercise in democracy.

The main menu is extremely simple to work with. You can add, delete or modify existing polls, create or remove admin users, change display attributes and log, plus a few other easy-to-modify options. Polls aren't static either. Imagine a poll with a choice of red or white wine. Along comes Roger at table 2 who wants Cognac added to the list, while Gar at table 16 absolutely must have single malt scotch as one of the choices. No problem. You can add more choices after the fact and continue with the voting.

Once you have entered a poll category, you are shown code similar to the following:

```
<?php require
 '/usr/local/apache/htdocs/sympoll/booth.php';
 display_booth(2); ?>
```

Enter that code on a PHP Web page, and you are on your way. Have a look at Figure 4 for an example of the results you can expect.


Chez Marcel Poll of the Day - I
Location Edit View Go Bookmarks Tools
Settings Window Help

Chez Marcel Poll of the Day

Red or White?

Red
4

White
1

A single malt scotch would be wonderful.
2

I'll take the Cognac.
1

votes: 8 Polls

Page loaded.

Figure 4. Red is looking good in early voting.

Take a moment to get familiar with the miscellaneous configuration options. The default colors and appearance of the poll display can be changed to something more to your taste.

The final item on tonight's menu comes from Benjamin D. Jones, and it certainly sounds political. It's called Recount. It was inspired, apparently, by the 2000 United States presidential election. Recount is a PHP application that allows you to create extremely flexible polls, both in terms of how they are presented and how the results are generated. It uses either MySQL or PostgreSQL as its database.

Download the latest Recount bundle from its Web site, and extract it into your Web server's hierarchy (usually right under document root). As I mentioned, Recount supports both PostgreSQL and MySQL, but I'm going to stick with MySQL for now. To start, we create a Recount MySQL database using the same technique we used for Sympoll above, `mysqladmin -u root -p create recount`. As in my last example, I'm going to use the polling MySQL user I created, which means I need to give polling the rights to work with this database. Again, I do this by connecting to MySQL with the command `mysql -u root -p`. From the `mysql>` prompt, I enter the following:

```
mysql> grant all privileges on recount.*
    -> to polling@localhost identified by 'secr3t';
```

Then, from the Recount source directory, I run the following command to create the necessary tables:

```
mysql -u polling -p recount < sql/mysql.recount.sql
```

Inside the Recount distribution directory, there's a file called config.recount.php-dist. Copy that file to config.recount.php and open it with your favorite editor. You should have no trouble with this file as it is well documented. However, the lines you probably are most interested in are these:

```
define('RECOUNT_HOME_PATH',
       '/usr/local/apache/htdocs/recount');
define('RECOUNT_DB_TYPE',
       'mysql');       /* "mysql" or "pgsql" */
define('RECOUNT_DB_HOST',
       'localhost');     /* db host */
define('RECOUNT_DB_USER', 'polling');
define('RECOUNT_DB_PASS', 'secr3t');
define('RECOUNT_DB_NAME', 'recount');
```

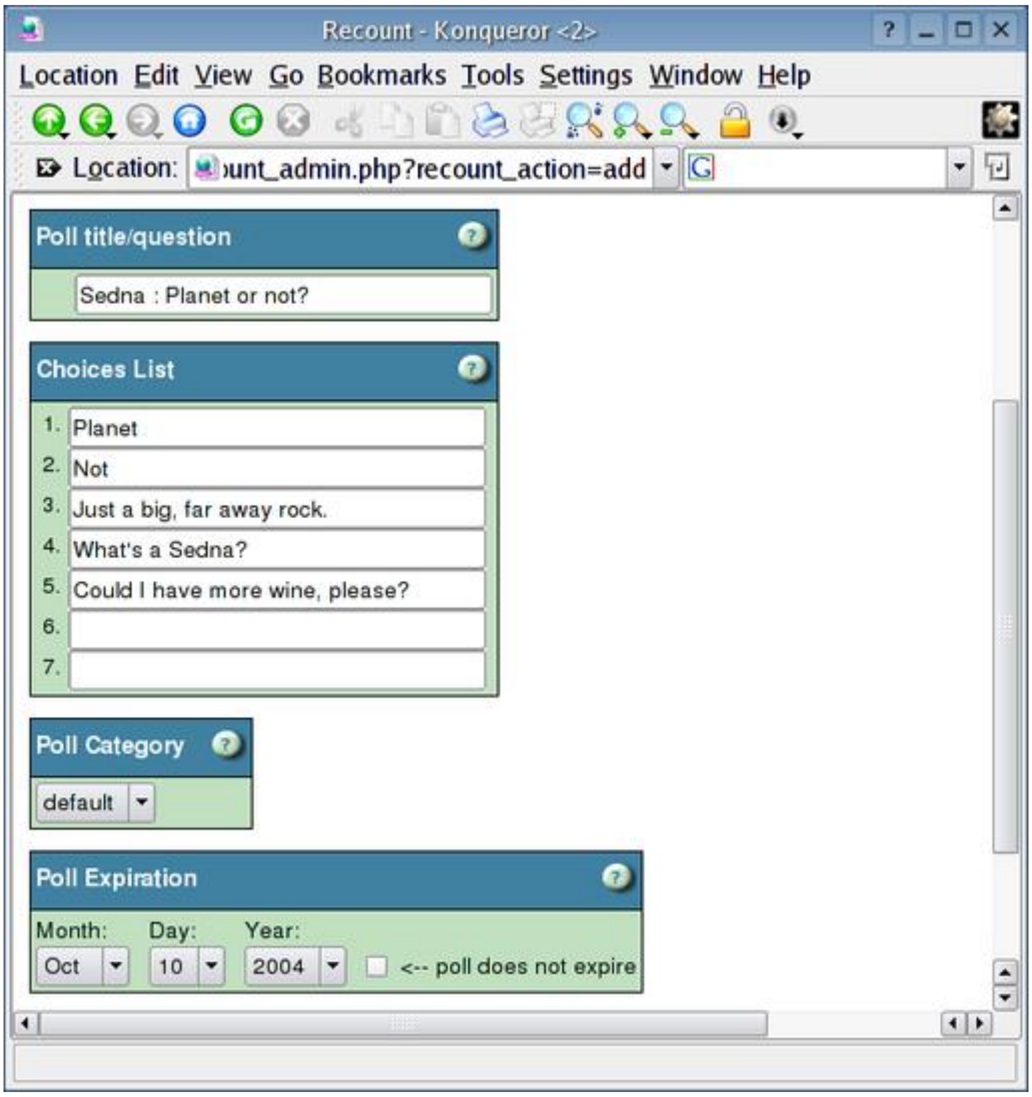
A little further down, enter a user name and password for the Recount administrator. This isn't your user name or even the MySQL user name but a name and password you use to administer the polls:

```
define('RECOUNT_ADMIN_USER', 'admin_name');
define('RECOUNT_ADMIN_PASS', 'admin_password');
```

When you are done with the configuration file, make sure you protect it by changing its permissions:

```
chmod 600 config.recount.php
```

That's the whole setup. To create a new poll, point your browser to http://yourserver/recount/recount_admin.php and log in using the admin user and password you created above. Keep in mind that this user is different from the MySQL user that updates the actual Recount database. When entering a new poll into the form (Figure 5), you can specify the usual question with multiple answers as well as an expiration date for the poll. If you already have polls defined, you can edit or delete them. When you have finished entering your information, your new poll is assigned an ID.


Recount - Konqueror <2>
Location Edit View Go Bookmarks Tools Settings Window Help
Location: ›unt_admin.php?recount_action=add

Poll title/question

Sedna : Planet or not?

Choices List

1. Planet
2. Not
3. Just a big, far away rock.
4. What's a Sedna?
5. Could I have more wine, please?
6.
7.

Poll Category

default

Poll Expiration

Month: Oct Day: 10 Year: 2004 ☐ <-- poll does not expire

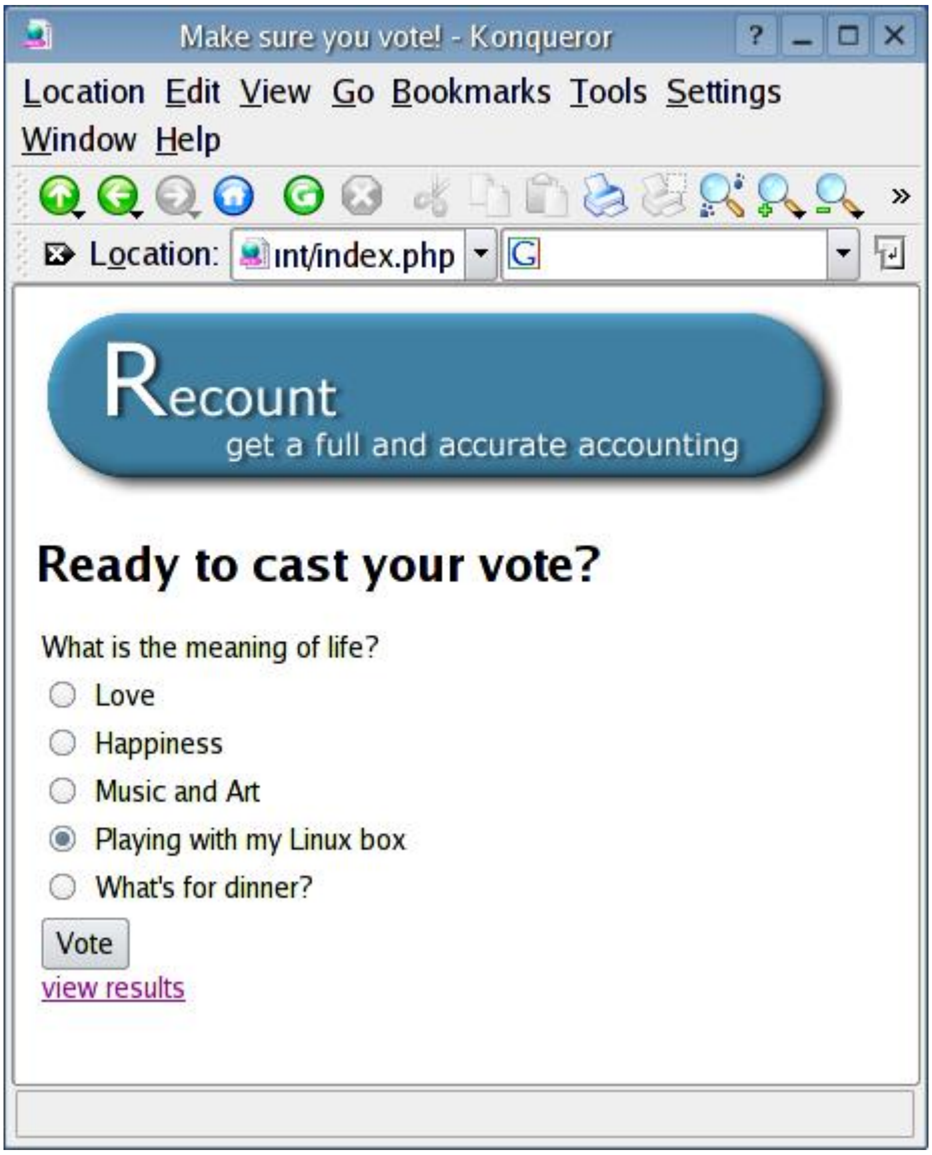
Figure 5. Creating a New Recount Poll

The package includes a ready-to-use page to test your new polls, called recount_testing.php. If you point your browser there, you can specify any poll ID. Both the poll and the results can be shown on the page. Creating a voting page for yourself specific to a given poll is easy. Have a look at this simple HTML/PHP page:

```
<html>
<head><title>Make sure you vote!</title></head>
<body>
<p><img src="images/logo.png">
<h1>Ready to cast your vote?</h1>
<?php
     require('config.recount.php');
     $poll = new RecountPoll(2);
     print($poll->basicForm());
?>
</body></html>
```

The poll ID is identified in the `$poll = new RecountPoll(2)` line above. In that example, the poll ID (in brackets) is 2. You can dress this up any way you want or include the PHP code in another Web page. To view the results requires another page with the following PHP code inside the default HTML tags.

```
     <?php
     require("config.recount.php");
     require("results.recount.php");
     ?>
```


Make sure you vote! - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: int/index.php
Recount
get a full and accurate accounting

Ready to cast your vote?

What is the meaning of life?

- ○ Love
- ○ Happiness
- ○ Music and Art
- ● Playing with my Linux box
- ○ What's for dinner?

Figure 6. What is the meaning of life?

That's all there is to it. Check out Figure 6 for a Recount poll in action. As I mentioned, this is a flexible package. It's worthwhile to look through the included documentation for extra tips.

Once again, it appears that closing time is fast approaching. Let's have a look at the results of François' raise poll. *Mon Dieu,* 17,972 votes for your raise and only 26 against? We don't have that many people in the restaurant, *mon ami*. I demand a recount. Perhaps while we restart the voting on this raise, you would be kind enough to refill our guests' glasses once more. I'm quite certain that the question of a refill is one issue where we can be assured of a yes vote. Until next time, *mes amis*, let us all drink to one another's health. *A votre santé Bon appétit!*

Resources for this article: /article/7504.

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of *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

Using Yum for RPM Updates

Mick Bauer

Issue #122, June 2004

You can eliminate most known security risks by removing unused software and keeping your system up to date. Here's how to do it with the new tool chosen by three of the most popular Linux distributions.

Updating your Linux system's software whenever security updates become available is one of the most important things you can do to reduce your exposure to bug-related vulnerabilities. It's so important that I've written at length about it both in *Linux Journal* (“Staying Current without Going Insane”, July 2002) and in Chapter 3 of my book *Building Secure Servers With Linux*.

Since I wrote those items (two years ago), the means by which you update Debian and SuSE systems haven't changed too much: apt-get and YaST, respectively, still are the preferred tools for this task. But the Red Hat world has a new and noteworthy utility called Yum (for Yellow Dog Updater, Modified). This month I explain where to get Yum, how to set it up and how to use it to simplify the task of keeping your Red Hat, Fedora or Mandrake system up to date.

Patching Is Only One Step in the Security Dance

Let me be clear about one thing, keeping up to date with software patches is something you must do *in addition to*, not instead of, other system-hardening tasks. These other tasks include but are by no means limited to:

- Removing unnecessary software packages.
- Minimizing the number of things that runs setuid root.
- Running dæmons as nonprivileged users whenever possible.
- Running dæmons in chroot jails whenever possible.

- Understanding and using whatever security features a given dæmon supports.
- Physically securing your system (for example, keeping it in a locked computer room).
- Making a habit of watching system and application logs for suspicious activity.

Like log watching, applying security updates must be an ongoing process. Subscribe to your distribution's security mailing list, and apply patches as soon as they're made available. The bad guys may not have the courtesy to wait before striking.

Yum Overview

As its name implies, the Yellow Dog Updater, Modified evolved from the Yellow Dog Updater, Yup, which is part of the Yellow Dog Linux distribution for Macintosh computers. Whereas Yup runs only on Yellow Dog (Macintosh) systems, Yum presently works on Red Hat, Fedora, Mandrake and Yellow Dog Linux, where it's replaced Yup. Yum is a project of the Linux@DUKE team at Duke University; Seth Vidal and Michael Stenner are credited with the better part of the development work.

In a nutshell, Yum does for RPM-based systems what apt-get does for Debian. It provides a simple command that can be used to install or update a software package automatically, after first installing and updating any other packages necessary to satisfy the desired package's dependencies.

Yum actually consists of two commands: `yum` is the client command, and `yum-arch` is a server-side command for creating the header files necessary to turn a Web or FTP server into a Yum repository. The yum-arch command is beyond the scope of this article, but you need to use it if you want to set up a public Yum repository, a private Yum repository for packages you maintain for local systems or even for a non-networked Yum repository on your hard drive. yum-arch is simple to use, and the yum-arch(8) man page tells you everything you need to know.

Unlike apt-rpm, a popular port of apt-get for RPM-based distributions, Yum is native to the RPM package format. And, says Michael Stenner, “Yum is designed to be simple and reliable, with more emphasis on keeping your machine safe and stable than on client-side customization.”

Getting Yum

The Yum download site (see the on-line Resources section) explains which version of Yum to download, depending on which version of Red Hat or Fedora

Linux you use. If you're a Fedora user, Yum is part of Fedora Core 1, and the package yum-2.0.4-2.noarch.rpm is on Disk 1 of your Fedora installation CD-ROMs. If you use Mandrake 9.2, the package yum-2.0.1-1mdk.noarch.rpm is included in the distribution's contrib/i586 directory.

Yum is written entirely in Python. Therefore, in order to install any Yum RPM, your system needs the Fedora/Red Hat packages python, gettext, rpm-python and libxml2-python or their Mandrake equivalents. Chances are, all of these packages already are on your system.

Finding Yum Repositories

From where can Yum pull its RPMs? Usually, this happens from a remote site over the Internet. This being a security column, my emphasis here is using Yum to grab security patches, so network updates are the focus of the rest of this column. In the interest of completeness, however, Yum can read RPMs from local filesystems or virtually local filesystems, such as NFS mounts.

Whether on a remote or local server, the RPM collection must be a Yum repository. It must include a directory called headers that contains the RPM header information with which Yum identifies and satisfies RPM dependencies. Therefore, you can't arbitrarily point Yum at any old Red Hat mirror or Mandrake CD-ROM.

If you use Fedora Core 1 or 1.90, you can use Yum with any Fedora mirror. Because Yum is an officially supported update mechanism for Fedora, Fedora mirrors are set up as Yum repositories. And, did you know about the Fedora Legacy Project? This branch of the Fedora effort provides security updates to legacy Red Hat distributions, currently Red Hat 7.3, 8.0 and 9.0. Thus, many Fedora mirrors also contain Red Hat updates in the form of Yum repositories.

If in doubt, a limited but handy list of Yum repositories for a variety of distributions is available (see Resources). Each of the links on this list yields a block of text you can copy and paste directly into your /etc/yum.conf file, which we explore in depth shortly. If all else fails, Googling for *mydistroname* Yum repository is another way to find repositories.

Configuring Yum

Configuring Yum is fairly simple; all you need to do is edit one file, which is named, predictably, /etc/yum.conf. Listing 1 shows the default /etc/yum.conf file that comes with Fedora Core 1's Yum RPM.

Listing 1. Fedora Core 1's /etc/yum.conf File

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
tolerant=1
exactarch=1

[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://fedora.redhat.com/releases/fedora-core-$releasever

[updates-released]
name=Fedora Core $releasever - $basearch - Released Updates
baseurl=http://fedora.redhat.com/updates/released/fedora-core-$releasever

#[updates-testing]
#name=Fedora Core $releasever - $basearch - Unreleased Updates
#baseurl=http://fedora.redhat.com/updates/testing/fedora-core-$releasever
```

As you can see, this file consists of a list of global variable settings, followed by one or more [server] blocks ([base] and [updates-released] in Listing 1), each of which specifies settings for a different type of RPM group. I'm not going to cover every possible global or server block setting; that's what the yum.conf(5) man page is for. But let's discuss a few key settings.

In the global section, debuglevel determines how verbose Yum's output is. This value may range from 0, for no output, to 10, for maximum debugging output. The default value of 2 is shown in Listing 1. As far as I can tell, this debuglevel affects only standard output, not Yum's log file, whose location is specified by logfile. Still, I like to change this value to 4, which I arrived at by playing with the `yum` command's -d value. For example, (`yum -d 4` *`yum-commands`*) is equivalent to and overrides debuglevel.

Also in the global section, pkgpolicy specifies how Yum should decide which version to use if a given package turns up across multiple [server] blocks. distroverpkg specifies the name of your local release file package. Your release file, for example, /etc/fedora-release or /etc/redhat-release, contains the name and version of your Linux distribution.

Each [server] block defines a set of RPMs. Personally, I wish these were called [package-type] blocks instead, because they don't distinguish by server but rather by RPM group. A single block may contain the URLs of many servers. In Listing 1, the [base] block contains a single URL pointing to the main Fedora repository at fedora.redhat.com.

Fedora mirrors that contain the same collection of RPMs can be listed with additional baseurl lines. Any line in a [server] block may use the variables $releasever, which resolves to the version number of your Linux distribution, and $basearch, which expands to the CPU family of your system. CPU families exist here in the most generic sense; Athlons are considered part of i386 in this context.

The /etc/yum.conf file installed by your Yum RPM probably works fine, but you should augment each default URL, http://fedora.redhat.com... in Listing 1, with at least one mirror site URL. Doing so minimizes the chance that your updates fail due to one server being unavailable. Be sure to use your favorite Web browser to test-drive any URL you add to yum.conf to make sure it successfully resolves to a directory containing a directory named headers. Also, make sure your URL ends with a trailing slash.

The other thing worth noting in Listing 1 is that one important [server] option is missing: gpgcheck. Listing 2 shows a corrected [base] block that uses this option.

Listing 2. Customized [base] Section

```
[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://mirror.eas.muohio.edu/fedora/
↪linux/core/$releasever/$basearch/os/baseurl=
↪http://fedora.redhat.com/releases/
↪fedora-core-$releasever
gpgcheck=1
failovermethod=priority
```

Setting `gpgcheck=1` causes Yum to check the GnuPG signature in each RPM it downloads. For this to work, you need the appropriate GnuPG keys incorporated into your RPM database. On Fedora Core 1 systems, these keys were installed on your system as part of the fedora-release package. To copy them to your RPM database, execute this command:

```
rpm --import /usr/share/doc/fedora-release-1/RPM-GPG*
```

The `rpm --import` command also can use a URL as its argument, so if the GnuPG key of your Yum source is on-line, you can use the form:

```
rpm --import http://your.distro.homepage/GPGsignature
```

where *http://your.distro.homepage/GPGsignature* should be replaced with a real URL.

This may seem like a hassle, but it's worth it. Several intrusions on Linux distributors' sites over the years have resulted in trojaned or otherwise compromised software packages being downloaded by unsuspecting users. Taking advantage of RPM's support for GnuPG signatures is the best defense against such skulduggery.

The other notable change in Listing 2 is that I've specified `failovermethod=priority`, which tells Yum to try the URLs on this list in order, starting with the one at the top. The default behavior, `failovermethod=roundrobin`, is for Yum to choose one of the listed URLs at random. Personally, I prefer the priority method because it lets me prioritize faster, closer mirrors over my distribution's primary site.

Running Yum Automatically

Now we come to the easy part, using the `yum` command. There are two ways to run Yum, manually from a command prompt or automatically by way of the /etc/init.d/yum startup script. If enabled, which you must do manually by issuing a `chkconfig --add yum` command, this script simply touches a runfile, /var/lock/subsys/yum, which the cron.daily job yum.cron checks for. If the script is enabled, that is, if the runfile exists, this cron job runs the Yum command first to check for and install an updated Yum package and then to check for and install updates for all other system packages. In so doing, Yum automatically and transparently resolves any relevant dependencies. If an updated package depends on another package, even if it didn't previously, Yum retrieves and installs the other package.

For most users, this script is powerful and useful stuff. If your environment demands meticulous change-control procedures, however, and you don't want any new software installed automatically, you should run Yum manually.

Running Yum Manually

To see a list of available updates without installing anything, use `yum check-update` (Listing 3).

Listing 3. Checking for updates (output slightly reformatted and truncated for readability).

```
[root@iwazaru-fedora etc]# yum check-update
Gathering header information file(s) from server(s)
Server: Fedora Core 1 - i386 - Base
Server: Fedora Core 1 - i386 - Released Updates
Finding updated packages
Downloading needed headers
getting /var/cache/yum/updates-released/headers/
↪coreutils-0-5.0-34.1.i386.hdr
coreutils-0-5.0-34.1.i386 100% |=================|
↪13 kB  00:01
Name                 Arch  Version   Repo
-----------------------------------------------------
XFree86              i386  4.3.0-55  updates-released
XFree86-100dpi-fonts i386  4.3.0-55  updates-released
XFree86-75dpi-fonts  i386  4.3.0-55  updates-released
XFree86-Mesa-libGL   i386  4.3.0-55  updates-released
```

To install a single update, plus any other updates necessary to resolve dependencies, use `yum update` *`packagename`*, for example: `yum update yum`.

This example actually updates Yum itself. If indeed an updated version of the package Yum is available, you are prompted to go ahead and install it. If you're invoking Yum from a script and you want all such prompts to be answered y automatically, use the -y option:

```
yum -y update yum
```

The check-update command, the Yum command that is, isn't mandatory before installing updates. If you prefer, you can use `yum update` directly—it performs the same checks as `yum check-update`.

In the last sample command, we specified a single package to update, `yum`. To initiate a complete update session for all installed packages on your system, you simply can omit the last argument, the package specification: `yum update`.

After Yum checks for all available updates and calculates dependencies, it presents you with a list of all updates it intends to download. Unless you used the -y option, it asks you whether to download and install each of them.

For the sake of completeness here's a bonus tip: you also can install new packages with Yum—you probably figured that out already. For any package contained in the sources you've defined in /etc/yum.conf, you can use the command `yum install` *`packagename`* to install the latest version of that package, plus anything it depends on. For example, to install the FTP server package vsftpd, you'd issue this command: `yum install vsftpd`.

Further Information

If you have any problems using Yum, ample help, including two excellent FAQs, is available on-line. See the on-line Resources section. If the on-line documentation doesn't help, a Yum mailing list is available too. Before posting a question, however, be sure to try a Google search or two. In the course of writing this article, I found a number of postings to the Yum mailing list addressing various questions and problems I had, all located with Google.

Resources for this article: /article/7501.

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

The Free Software Act

Maureen O'Sullivan

Issue #122, June 2004

An international standard law will help your free software license work the same in all your users' countries.

Free software licenses function in a number of different ways. First, they are a statement of rules about what free software programmers can and cannot do within their own communities. Part of the reason for complying with these conditions is the power of reputational capital: there is an incentive to play the game by the recognised rules, whether formal or informal, written or understood. The licenses also bind in a legal way because they are built on copyright and licensing or contract law. Copyright protects authors' original creations by allowing them to prevent access to their work; software licenses permit access to such works, under certain conditions. When the licenses are proprietary, what the user can do with the work is quite restricted, whereas free software licenses allow different degrees of freedom. Some, such as the BSD licenses, have practically no requirement to give anything back. Others, such as the GNU GPL, place strong obligations on authors when software is modified and published.

The law governing software licensing appears straightforward, but it is not. Contract law varies from jurisdiction to jurisdiction, as does copyright. A law drafted in one legal system may not bind in another. Therefore, ideally speaking, some international standardisation of law in this area should be in place to make sure rules that bind in one country do likewise in another.

This point was made clear to me when I began studying this area of law from a socio-legal perspective some years ago. I believed that free software legislation was something that programmers involved in this line of work should seek. I started work with the Free Software Consortium (FSC) during the summer of 2003 and was appointed as coordinator of their Legal Governing Body. In Costa

Rica last September, I commenced a project to draft free software legislation with a view to getting it passed by legislatures around the world. The project is in its infancy but is maturing rapidly.

My early take on the matter was that because the GNU GPL was the most commonly used license among free software developers, the law should be put in place to protect this scheme. Draft 1 of the Free Software Act attempted to enshrine its terms and added a few more, such as protection for any programmers who inadvertently infringed a copyright. This latter point would preempt any recurrence of vexatious lawsuits, such as the current SCO vs. IBM case. I quickly was put right, however, by colleagues in the FSC and by Richard Stallman himself, who advised me that such legislation should not seek to copyleft all free software but, rather, should endeavour to grant legislative force to all free software licenses. Draft 2 was an improvement on the first draft but still appeared to copyleft all free software. I began work on Draft 3 after Christmas 2003, and it recently was completed. I currently am working on Draft 4, which I hope will hit the mark more closely. Once completed, Draft 4 will be available on the FSC Web site, as are the earlier versions.

My idea about this project is to make it a venture of the Free Software community as a whole. The worst types of law are those that disrupt beneficial and functioning social contracts, such as the introduction of copyright law to cover software when previously it had been a shared resource. Lawyers have to respond to social movements and change. They should take part in such movements and share such drafting activities, so they become linguistic facilitators of communities' customs and wishes, rather than interlopers who enter the fray and disrupt it.

This project continues to be a learning experience for me. Initially, I saw value in protecting only the copyleft system, as it is the most common one, makes the most demands and potentially is most vulnerable in the eyes of the law. Other considerations need to be taken into account, however. Such a law would prove divisive if it protects only one, albeit the majority, view. It would be preferable, indeed, to have an international law that gives protection to all free software licenses. This would mean any clashes between the license and the contract or copyright law of the particular country in which the case is being tried would be irrelevant: the legislation would act to bind third parties and would ensure that rights recognised in one place would be recognised similarly in another.

At the time of this writing, Draft 3 is the most recent version of the Act, and it can be viewed on the FSC Web site at www.fsc.cc. Comments are welcome, and participants to the FSAct mailing list at Fsact@lists.fsc.cc are encouraged. Alternatively, you can e-mail me at maureen@fsc.cc.

Maureen O'Sullivan is a lecturer in law, based in the UK. Her research interests are in legal and socio-legal aspects of free software, including licensing.

Xandros Desktop 2.0 Deluxe

Dean Staff

Issue #122, June 2004

Xandros 2.0's list of built-in applications is extensive.

Product Information.

- Manufacturer: Xandros, Inc.
- URL: www.xandros.com
- Price: Xandros Desktop 2.0 Standard—$39.95 US; Xandros Desktop 2.0 Deluxe—$89.00 US; Xandros Desktop 2.0 Business—$129.95 US

The Good.

- Default set of applications.
- VNC and terminal services clients.
- Multimedia support.

The Bad.

- Latest CrossOver breaks older applications.
- No direct support for CrossOver Office.
- No backup application installed by default.

This review marks my second run at Xandros Desktop. I have been using version 1.0 for some time, but when I heard Xandros was releasing version 2.0, I had to take a look at it. Three versions of Xandros Desktop 2.0 are available, the Standard, Deluxe and Business editions. For this review I focus on the Deluxe edition.

Installation was quick and easy; a fresh install can take as few as four clicks if you accept the default settings. The install wizard found my existing copy of

Xandros Desktop 1.0 and gave me the option to upgrade it to 2.0. It kept my existing partitions and users intact. Individual user preferences were saved in the users' home directories, where they are easy to restore. It would have been better, however, if all user preferences had been detected and updated intact.

After the initial installation, a First Run wizard walks users through setting up their personal preferences. Users can change any of these preferences at any time by using the control centre or the First Run wizard icon found in the system section of the Launch menu. Right-clicking on the desktop opens the display properties, which makes changing the display settings easy.

Xandros 2.0's list of built-in applications is extensive. The Standard and Deluxe editions both come complete with OpenOffice.org 1.1.0. The Business edition includes StarOffice. Both the Deluxe and Business editions include CodeWeaver's CrossOver Office 2.1, an essential tool if you need to run Microsoft Windows applications. All versions include Mozilla, KDE's KMail, Adobe Acrobat Reader and a plethora of other applications. In addition, a second CD accompanies the Deluxe and Business editions that includes tons of additional applications, games and other utilities.

The big change for version 2.0 is the focus on multimedia. Xandros now supports drag-and-drop CD burning. You can create data or music CDs right from the file manager. There also is built-in support for digital cameras and scanners.

I tried all three of these new multimedia features, with great success. Although most CD-R/RW drives these days are IDE/ATAPI-compatible, my old SCSI CD-RW was detected and functional at first boot. The SCSI scanner was added to the system after installation, and the Hardware Detection wizard found my scanner and added it to the list of hardware with absolutely no problem. Even more amazing, absolutely no configuration was required. I was presented with a selection window where all scanning devices detected on the system are listed and chose which one I wanted to use. Having only one scanner, I really didn't have any options; I was given only one choice. When I clicked the Preview Scan button and my scanner started humming, I indeed was impressed.

I next tried connecting my digital camera to the USB port. I bought a new digital camera less than a month before doing this review, so it was not on the list of cameras Digikam supports. The Xandros user manual explains how to deal with this, though, and I was able to set up Digikam to access my camera in no time.

Two other applications I found extremely useful were VNC, which Xandros calls Remote Desktop Sharing (Figure 1), and the Terminal Services Client. The latter is not installed by default but is available from the Xandros Networks. I find both of these applications indispensable and use them on a daily basis in my work as a network administrator.

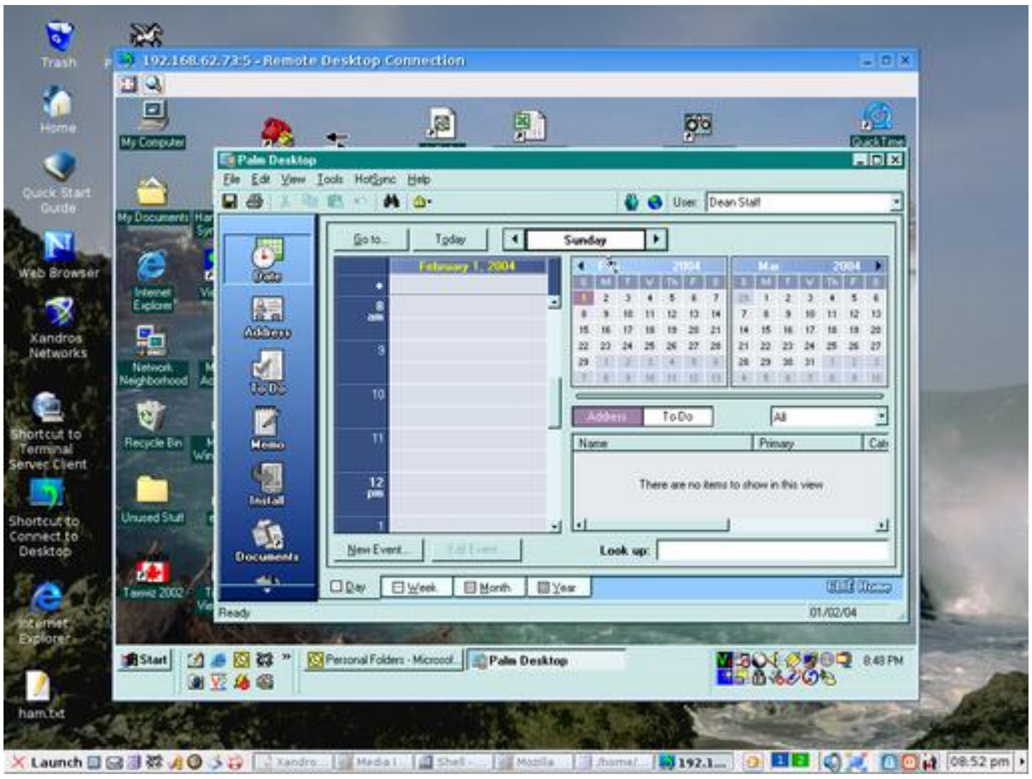
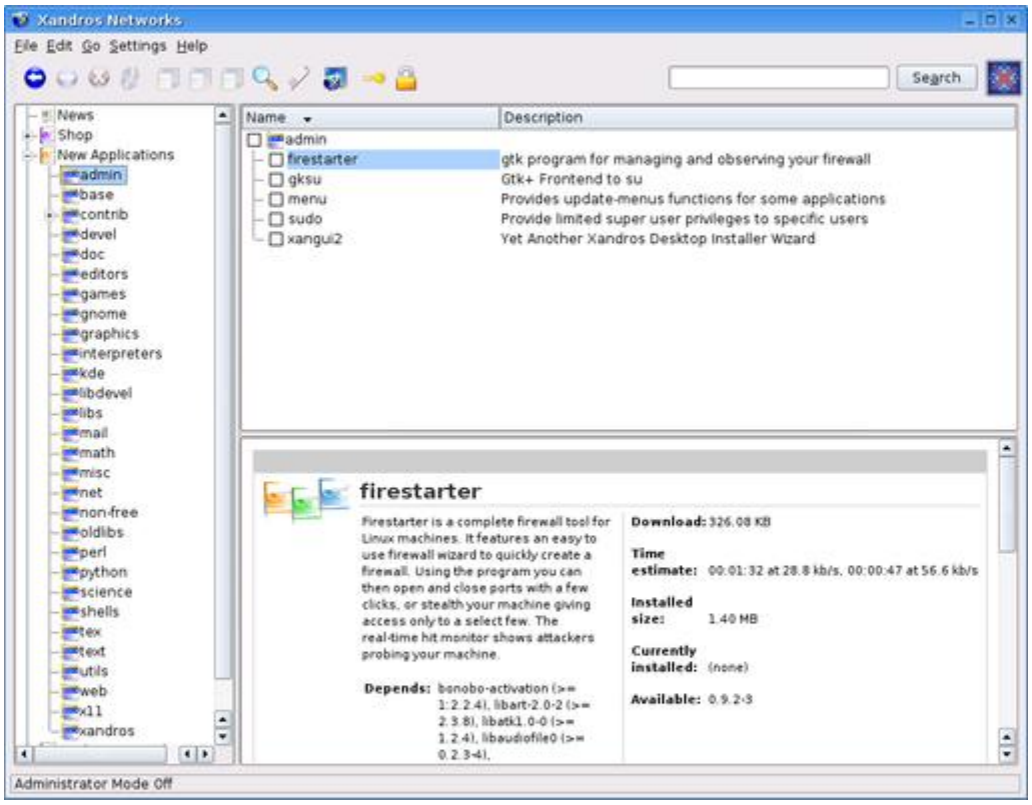


Figure 1. VNC makes it easy to use a Microsoft Windows desktop remotely.

Noticeably absent in the default installed application is a backup utility. One is available on Xandros Networks, but it seems to be difficult to configure. It certainly did not like my ARCHIVE DDS1 tape drive, and the only mention of backups in the user manual is a backup to CD.

One of the biggest changes I had to get used to was the absence of the previous version's update package manager. I was a little disappointed to see it not included, because I had become rather accustomed to it. Xandros now uses Xandros Networks exclusively. It works very well, but it takes a little getting used to. It's a bit like using a cross between Microsoft's Windows Updates and dselect on steroids. It's a browser-like GUI offering the options for a single-button update or individually selected package installation.

I spoke to Erich Forler of Xandros, and he told me that the Xandros Network subscription included with Xandros Desktop does not expire. The reason behind this, he said, is because Xandros wants to make sure that all users have access to all the latest security and bug fixes, as well as upgrades to any free software. There is a paid subscription service available, or will be shortly, that will allow a subscriber to download the latest enhancements for proprietary packages before they are released in the next version.



Name	Description
admin	
firestarter	gtk program for managing and observing your firewall
gksu	Gtk+ Frontend to su
menu	Provides update-menus functions for some applications
sudo	Provide limited super user privileges to specific users
xangui2	Yet Another Xandros Desktop Installer Wizard

firestarter

Firestarter is a complete firewall tool for Linux machines. It features an easy to use firewall wizard to quickly create a firewall. Using the program you can then open and close ports with a few clicks, or stealth your machine giving access only to a select few. The real-time hit monitor shows attackers probing your machine.

Depends: bonobo-activation (>= 1:2.2.4), libart-2.0-2 (>= 2.3.8), libatk1.0-0 (>= 1.2.4), libaudiofile0 (>= 0.2.3-4),

Download: 326.08 KB

Time estimate: 00:01:32 at 28.8 kb/s, 00:00:47 at 56.6 kb/s

Installed size: 1.40 MB

Currently installed: (none)

Available: 0.9.2-3

Figure 2. Xandros Networks is a powerful utility for adding and upgrading software.

As much as I liked the older Xandros Update, I have to admit I really like the one-button “apply all available updates from Xandros” option. It does not matter which package source you select, this button overrides it and checks a preconfigured URL for updates. One other difference between the the older Xandros Update and the newer Xandros Networks is it is much easier to add a custom package source. Simply type the URL the same way you would if you were editing /etc/apt/sources.list.

I was not as impressed by the newer version of CrossOver Office in Xandros 2.0. I was holding onto one Microsoft Windows application, Pegasus Mail for Windows, which I have been using for nearly a decade. I had great success running it previously under WINE and then CrossOver Office on Xandros 1.0. CrossOver Office 2.1, however, breaks Pegasus Mail. It still installs, and I still can send and receive e-mail, but the message window formating is messed up, and it is now unstable and crashes constantly. The fact that the version of Pegasus did not change, only the version of the OS and CrossOver, leads me to believe the issue is not Pegasus Mail. I tried going through Xandros support, but they offer support for only the officially supported applications.

I experienced one other disappointment. After all the hassles of not being able to run Pegasus any longer, I started using Ximian Evolution, which Xandros has available for free download from the Xandros Shop portion of Xandros Network. I discovered, however, that the GNOME conduits required to sync with a PalmOS device are not installed with Ximian, nor are they available on the Xandros Network. Xandros support was kind enough to point me to an outdated how-to for Xandros 1 and Ximian Evolution 1.2, but it did not work with Xandros Desktop 2 and Ximian Evolution 1.4.

One of Xandros Desktop's strongest features is its ability to integrate with an existing network. I personally run a small network that has some Microsoft Windows fileshares and a shared printer on it. Xandros Desktop can map the fileshare as folders under the user's home directory. It also has no problem using the shared printers from another workstation or server on the network. Xandros Desktop also uses SAMBA to share its resources with Microsoft Windows workstations without any problems.

Conclusion

Overall, I have to say this is the most user-friendly Linux distribution I have ever used. The Business edition promises the ability to authenticate users to a Windows Primary Domain Controller or Active Directory domain and offers SAP and Citrix clients. The Deluxe edition is ideal for home or small-business users in a workgroup environment or for people who don't need to authenticate to a Windows server. At $89 US, the Deluxe edition is a steal. I also have to say that even someone raised on a Windows platform would feel at home on a PC running Xandros Desktop 2.0. Better support for CrossOver Office and other third-party applications on the official update site is needed, but Xandros Desktop 2.0 still is my desktop environment and Linux distribution of choice.

Dean Staff is a network administrator for the IT Department, Inc., in Ottawa, Ontario Canada, where he lives with his wife and two children. He has been using Linux for about five years. His particular interest in Linux is as a desktop replacement for Windows and as an e-mail server.

From the Editor

Software as a Public Trust

Don Marti

Issue #122, June 2004

Jefferson said, “The government is best which governs least.” By committing itself to free software, government keeps citizens' options open.

As I write this, the Linux Users Group of Iraq is forming an organization called the Iraqi Open Source Organization (IOSO) to help get the new Iraqi government on the path of open-source software and open standards. They're joining a worldwide movement to put open source into government IT that's strong enough that proprietary software vendors have founded a lobbying group called the “Initiative for Software Choice” (ISC) to oppose it. The ISC doesn't merely oppose open-source code; it also tracks and opposes measures that would simply require specs to be openly published for interoperability.

Telling a government it should accept proprietary software licenses is like telling it that it needs to visit a shady rent-to-own store for furniture instead of using competitive bidding. Governments are big enough to get a better deal. And “software choice” for a government purchasing department could mean software lock-in for citizens. Until late last year, the US Federal Communications Commission's Web-based Universal Licensing System, mandatory for radio licensees, required a non-Linux OS.

On page 58, Jim Beard shows how open standards can make a government process work. Find out how courts, lawyers and organizations are keeping the new flood of paperless communication together with new standards anyone can use.

Our Senior Editor Doc Searls spent early 2004 in the storm of the presidential primaries, with special attention to Howard Dean frenzy. Our contributor Tony Steidler-Dennison went to work for the Wesley Clark campaign. Neither Dean nor Clark got very far, but Doc learned some valuable lessons that might help you in your next political campaign. Find out how you can join the political process on page 52.

Remember the little USB lamp on the cover of the April 2004 issue? Greg Kroah-Hartman told me that he got more responses to his article on a driver for it than to anything else he's written. Congratulations, new kernel hackers. Now, if you'd rather write user-space programs instead, Greg explains how to control a USB device that way on page 40. Now there's no excuse not to support whatever USB device follows you home from the electronics store. Be sure to let us know how it goes.

Dave Phillips is back with another software roundup—this time covering soundfile editors on page 84. And, if you're ready to have your mind expanded with a giant leap in communications design, flip to page 76 and get the facts on software radio from Eric Blossom.

Finally, as you might be able to tell from the cover, we're proud to bring you the latest from Douglas Maxwell on a new simulation for training firefighters (page 48). The project may be kind of esoteric, but it depends on the same Linux and free software we all use every day. If you're writing, documenting or testing free software, you're helping vital, even lifesaving projects without even knowing it. Stay free, and enjoy the issue.

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

Letters

Readers sound off.

Crypto Error? Quel Horreur!

Dear Mr Gagné as a fan of your column, as well as your other writing, when the paper copy of *LJ* was delivered yesterday, I set it aside to read this afternoon. However, I was disappointed to find a substantial error regarding digital signatures in your otherwise excellent April 2004 column “Francois, Can You Keep a Secret?” About 3/4 of the way through your column, you write: “Signing a message makes use of your key by attaching an electronic signature (your public key) to your message, but not encrypting it. The person receiving the message then has a means of verifying that the message did indeed come from you....”

The originator's public key has little to do with it other than providing a means to conduct the eventual verification. In the context of a signed e-mail, a digital signature is a hash (either MD5 or SHA1) of the signed object (a collection of bytes made up of the message text and any attachments) that is in turn encrypted using the originator's private key. Thus you have the original message, perhaps with an attachment, which has been hashed, and the resulting hash encrypted and then attached to the signed object. This process provides a mechanism for two services, establishing message integrity and message attribution. Message integrity is provided by the hash, and message attribution is provided by the encryption of the hash value. Signature verification is carried out as roughly outlined below.

The recipient of a signed message, when they choose to verify it, simply decrypts the attached signature using the public key of the purported sender, re-runs the hash on the received message and matches the results. If they match, two things are established: the sender had the private key associated with the public key used for the decryption, and the message had not been changed in transit. If they don't match, either the message was tampered with or the hash was encrypted using a private key that did not match the public key used. Most of the various public/private key schemes—GPG, PGP or X509 S/MIME—have individual mechanisms for determining whether it is a key mismatch or a change in the signed object.

—
Paul M. Livingston

Marcel replies: thanks for the sharp eye. On seeing your message, I was frankly shocked that I would have written something like this. I honestly do know better. I always take great care to make sure that what I write is both easy to understand and accurate. This one slipped by me.

/var/spool/fanmail

All I can say is wow. Thanks so much for the following four articles [*LJ*, April 2004]: “Writing a Simple USB Driver”, “The Hidden Treasures of iptables”, “Real-World PHP Security” and “SPF Overview”. I'm already using SPF, but I'm still glad it was there.

In case no one's mentioned it, Greg Kroah-Hartman is awesome. I've always wanted to baby-step my way into writing device drivers, and his article was just what I was looking for. I only wish the LED lamp wasn't $80 (I'm a dad and a small-business owner, so $80 is a bit out of my reach for a mood light).

—
john

Check out linux-usb.org for other USB driver projects in progress, and check out this issue for a double helping of articles from Greg. —Ed.

SPF for All

I would like to see the article “SPF Overview” from the April 2004 issue *Linux Journal* posted to your and pobox.com's Web sites. This was a very good article and a project that I hope makes it to the RFC state real soon.

—
Brent S. Smithline

LJ readers went for SPF in a big way. Watch our Web site and spf.pobox.com for more on SPF adoption. —Ed.

Ready to Take the Scribus Plunge?

I enjoyed reading the article “Introducing Scribus” in the November 2003 issue of *Linux Journal*. I was wondering whether *LJ* would consider eating your own dog food and convert to Scribus.

—
Martin van Nijnatten

Our other non-Linux internal system is going to move over to Linux first. Keep reading for details. —Ed.

New Game

I have written a new Rogue-like role-playing game. Its not completely finished yet and never will be, but it's already fun to play. Its home page is sourceforge.net/projects/lostlaby. I think there are still too few games like this for Linux.

—
Markus

Please Submit Your Drive Info to smartmontools

I didn't see the January 2004 article [“Monitoring Hard Disks with SMART” by Bruce Allen], but after reading Marshall Lake's comments on SMART [Letters, *LJ*, April 2004], I checked it out. Here are the results:

```
[root@gandalf tmp]# smartctl -i /dev/hdb
smartctl version 5.21 Copyright (C) 2002-3 Bruce Allen
Home page is http://smartmontools.sourceforge.net/

=== START OF INFORMATION SECTION ===
Device Model:     Maxtor 54610H6
Serial Number:    F6030KNC
Firmware Version: JAC61HU0
Device is:        Not in smartctl database [for details use: -P showall]
ATA Version is:   6
ATA Standard is:  ATA/ATAPI-6 T13 1410D revision 0
Local Time is:    Thu Mar 18 11:59:57 2004 MST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

I don't like that “Not in smartctl database” portion. If Marshall has a disk that is the same as mine, it MAY explain what his problems are. I am NOT going to run the smartctl command with the -t option given that kind of information. Because I never had a problem with disks, I am going to take my chances. I think a survey of other Linux users' experiences running smartctl is in order here.

—
HHH

Bruce Allen replies: if you run `smartctl -V`, you'll notice that smartmontools comes with ABSOLUTELY NO WARRANTY. And we mean it. You, as a user, can get open-source software like Fedora for free. But the flip side of this equation is that you, as a user, also have more responsibility: you have to read the documentation to understand what the software does and ultimately to contribute something back to the Open Source community. If you read the smartmontools Web page FAQ, you'll notice it states very clearly that not having a drive in the database is NOT an issue. Though, as an open-source user, it would sure be nice if you spent a few minutes (as described) to help us add it in to the database.

In fact, another user has already done this for your drive. Fedora is distributed with version 5.21 of smartmontools. If you upgrade to the current stable release (smartmontools 5.30), you'll find that your drive IS in the database.

A few minutes of research with Google (or browsing the smartmontools-support mailing list archive) will show that thousands of people have used these tools without problems. In fact, because they are distributed with virtually every major Linux distribution, I expect the number of users is several orders of magnitude larger.

You wrote, “Since I never had a problem with disks, I am going to just take my chances.” Without intending any lack of respect (I am sure you are an intelligent and thoughtful person) this is foolhardy. If you store any data you can't replace, you MUST back it up. Just “taking your chances” means you will, eventually, lose some or all of it.

Can smartctl provoke problems? As I said in my reply to Lake, the answer is yes, if a disk is already failing. An extended self test with `smartctl -t long` will read-scan the entire disk surface. If a disk already has problems, such a scan will probably reveal them. For a very marginal disk, it could be the “straw that breaks the camel's back” and provoke catastrophic disk failure. Note that this also applies to backups: the very act of backing up a disk (and thus reading all the files on it) can also cause a marginal disk to fail. Does this mean you shouldn't back up? No! It means you should keep multiple backups, and back up frequently, so that if a disk fails during a backup, not everything is lost.

Please keep in mind that `smartctl -t long` is simply sending a SMART command to the disk. The effects of that command depend upon the disk firmware, written by the engineers who designed the disk, not on smartmontools. I am aware of at least one disk family (an old IBM deskstar series) where some SMART commands did provoke such problems. smartmontools will recognize such disks (from the database) and issue a warning and a pointer to a site where you can get upgraded firmware from IBM. I am not aware of any such issues with Maxtor disks.

I advise you to re-think what you have said. Burying your head in the sand and taking your chances is not an intelligent and responsible approach.

SMART Saved My Data

Thank you for your wonderful article on the SMART tools [*LJ*, January 2004]. That one article was worth my entire year's subscription to *LJ*. Using the information in your article, I was able to run tests on all my machines and found two drives that were failing but had not yet died. As a consequence, I was able to preserve my data and avoid a rebuild. In both cases, the drives were still under warranty, so the manufacturer replaced them on the strength of the printout from smartctl. Excellent! This one has earned a place in my toolkit!

—
tim

Set-Top Box Docs, Please

I have started a grassroots campaign aimed at convincing IBM to open up and free the programming documentation for its STB (Set-Top Box) series chips (www-306.ibm.com/chips/products/digitalvideo/products/settopbox.html). You can find the campaign home page at www.users.bigpond.net.au/mysite/freestb.htm.

I'm not sure what the origin is of the practice of hiding chip documentation behind an NDA. Surely companies like IBM don't think that an NDA would in some way keep them one step ahead of the other chip manufacturers? I think such information hiding is a relic of the days of proprietary computing, and chips haven't felt the wind of open source, so to speak.

This is a sub-$100, TV-connected Linux machine that uses an STBx25xx chip (www.hauppauge.com/html/mediamvp_datasheet.htm). Here are the hardware specs of the Mediamvp machine: www.shspvr.com/forum/viewtopic.php?p=19411.

Projects like this would benefit from public and free access to the documentation and drivers (sourceforge.net/projects/mvpmc). I'm hoping to spread the word and build public pressure on IBM to address the issue. Any help that you might be able to give would be appreciated.

—
Andrew Stuart

Photo of the Month: Satisfied Home Users

As one of your subscribers, I can't help noticing that some members of readers' families are introduced to the benefits of Linux, and *LJ* of course, at a very early age. Quite rightly so of course, and so far we've already seen nice examples of pictures in the Letters section. However, I would like to demonstrate that the reverse is also true. I recently converted my parents, Mr and Mrs A. J. van der Kleij of Delfgauw, The Netherlands (ages 79 and 77), to Linux by upgrading their PC from Windows 95 to Red Hat 7.3. As can be seen in the photograph, they are very happy with Linux and found the conversion quite easy. They routinely use OpenOffice.org for their paper correspondence and Mozilla for e-mail and browsing. The only drawback is that I get less homecooked food now because they don't need my support as often.

—
Michel

Photo of the Month gets you a one-year extension to your subscription. Send photos to info@linuxjournal.com. —Ed.

Penguin Art Still in Demand

Please find attached a photo of my granddaughter the family artist. She's been admiring my small penguin collection and wanted to add one of her own. So as you see, I now have a Chloï¿½Original Tux for my collection, and I'm sure her artwork is much better than any I could have done myself. Plus, she's a future Linux user.

—
Robert Luzader

Bad register.com, Bad, Bad

After reading the article "SPF Overview" by Meng Weng Wong in the April 2004 issue, I tried adding a txt record for SPF to my DNS entry on register.com. After finding no way to do this I sent them an e-mail on the subject. I got the reply: "We would like to inform you that currently we do not support to add txt record to the DNS (name servers). We sincerely apologize for the inconvenience caused to you."

—
Stuart Scharf

A returnpath.biz study showed that 18.7% of legit marketing mail sent in the second half of 2003 was blocked by spam filters. If register.com wants to keep business customers who send newsletters or other marketing mail, they'll have to start supporting SPF. —Ed.

The GIMP Visits Scenic Portland, Oregon

Attached is my first attempt at using the Pandora plugin for The GIMP to create a panorama. The image was taken at the Columbia River Gorge outside of Portland, Oregon. It is actually three images stitched together.

—
karl

UpFront

diff -u: What's New in Kernel Development

Zack Brown

Issue #122, June 2004

The **BIOS** often is one of the least friendly parts of a computer. The source code typically is secret, and updates are almost nonexistent. Most decent OSes try to leave the BIOS behind as soon as possible, providing their own interfaces to similar functionality. Sometimes a BIOS bug forces OS developers to take steps to work around it. This happened recently with the **eMachines M6805**. The BIOS failed to recognize a CPU upgrade and reported the CPU speed and voltage values as those of the original CPU. **Tony Lindgren** has created a Linux kernel patch to perform some sanity checks on these machines to correct the values reported by the BIOS.

Benjamin Herrenschmidt is working on a **sysfs** interface to **Open Firmware** to provide PPC and PPC64 systems with a full Open Firmware path. Illustrating the Linux development style, as soon as Benjamin published his work, various design issues came under scrutiny. In particular, he had wanted to provide the Open Firmware data as part of the data for PCI devices, which had the potential to become a mess. Would all the different buses and all the different firmwares start having to store data about each other? And, what other parts of the system would want representation elsewhere as well? **Linus Torvalds** suggested keeping each aspect separate, so that Open Firmware data, even if it addressed PCI devices, would be found in the Open Firmware subdirectory.

Dipankar Sarma has produced an API function that should be able to clean up some **ReiserFS** (and other) code. The function, **rcu_barrier()**, waits for all the queued **RCU** callbacks to complete. RCU, or Read Copy Update, is a locking mechanism that allows the OS to access data shared between multiple CPUs at a low computational cost [see "Using RCU in the 2.5 Kernel" by Paul McKenney, *LJ*, October 2003]. Previously, ReiserFS had to implement its own logic to simulate this functionality. **Nikita Danilov** of the ReiserFS team has been hoping for this sort of API for quite a while. Once it's accepted into the main tree, the Reiser folks will be able to ditch a lot of ugly code.

With the release of Linux 2.0.40 (code-named The Moss-Covered Tortoise), **David Weinehall** has indicated his intention to continue maintaining the ancient 2.0 series. In 2.0.40, he has fixed a number of security holes and filesystem corruption issues, among other things. In the announcement, David affirmed that no new features would go into the 2.0 series. Users wishing to take advantage of features available in recent kernels, he says, must upgrade either to the 2.4 or 2.6 tree. Some systems may have a difficult time upgrading, such as active servers that have run only 2.0 kernels, which would need to be redesigned completely for any later kernel. In spite of this, it seems to be infeasible to provide a smoother upgrade path than what already exists, given that even-numbered kernels must strive for stability.

Max Asbock recently put together a device driver to support the **IBM xSeries RSA service processor**. The driver, called ibmasm, provides a user-space interface to send commands, wait for events and handle the remote video features. One snag in the current design, however, is that Max's interface is unique in Linux. Although the driver must provide an interface into the system, he doesn't see a way to provide the "one file, one value" mechanism of sysfs. Nor does the driver fit easily into the category of character device, so he has placed it in the /drivers/misc directory of the Linux kernel source. Given these anomalies, we might expect this driver to go through some changes before being adopted in the kernel proper. If it does get accepted quickly, the interface still may take some time to settle down, as it gets shoe-horned into the preferred Linux places.

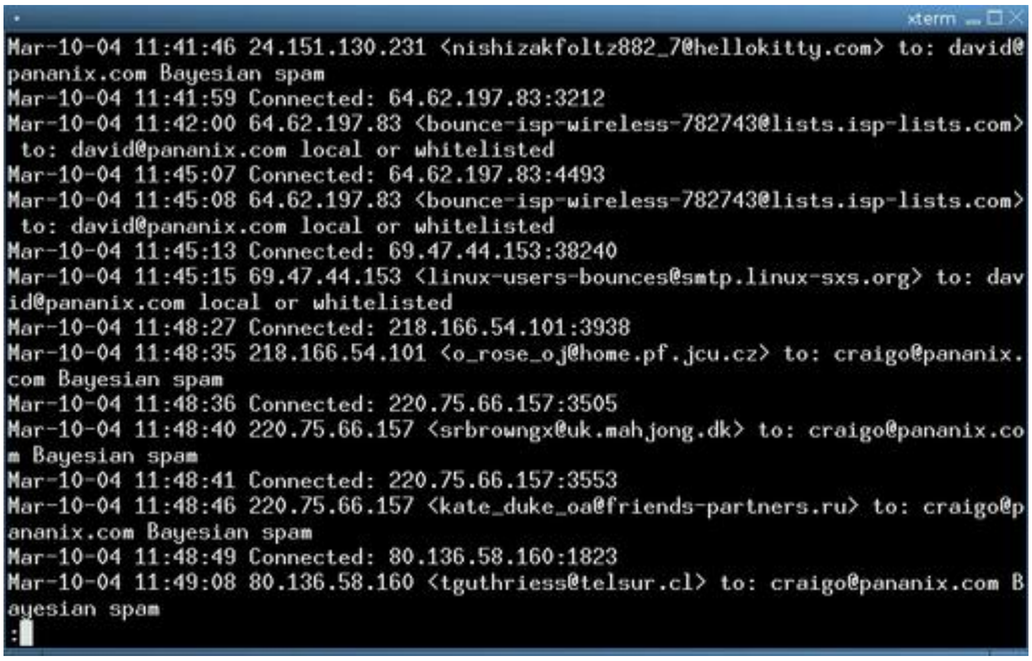
Gerd Knorr has written a driver that, although not actually providing any user-visible goodies on its own, will help other drivers deal with **infrared remote control** devices. Drivers like saa7134 and bttv now will be able to use the regular Linux input layer to access these devices in a clean fashion. One interesting twist is that the basic Linux module handling code has changed during the 2.5 time frame, and Gerd has kept to the old interface, even though his driver is for 2.6. The reason for this has been that he wants his code to compile under both 2.4 and 2.6, and 2.4 doesn't support some of the needed functions provided by 2.6. **Rusty Russell**, the developer most responsible for the module interface revamp during 2.5, seems to be quite active in responding to this type of situation, and he has said he will implement compatibility functions in 2.4 that will allow the preferred 2.6 constructions to compile properly on both kernel series.

Anti-Spam SMTP Project: assp.sourceforge.net

David A. Bandel

Issue #122, June 2004

ASSP is an SMTP proxy that performs Bayesian filtering, has whitelists and generally tries to identify and reject spam before it's passed on to your MTA for acceptance and processing. Among the good features is an easy way to reclassify non-spam as spam merely by forwarding it to assp-spam@your.domain. Reviewing spam is a little more difficult, because you actually need to take a quick look through the spam directory to ensure that no non-spam has been misclassified. Then, do a quick rebuild of the spam database and you're done. Setup is more intensive, because you have to let things run for a while, keep an eye on where they're going and move them as required. But, no mail ever is lost, even if it is misclassified. ASSP is probably not appropriate for an ISP, but businesses, especially smaller ones, will like it. Requires: Perl.



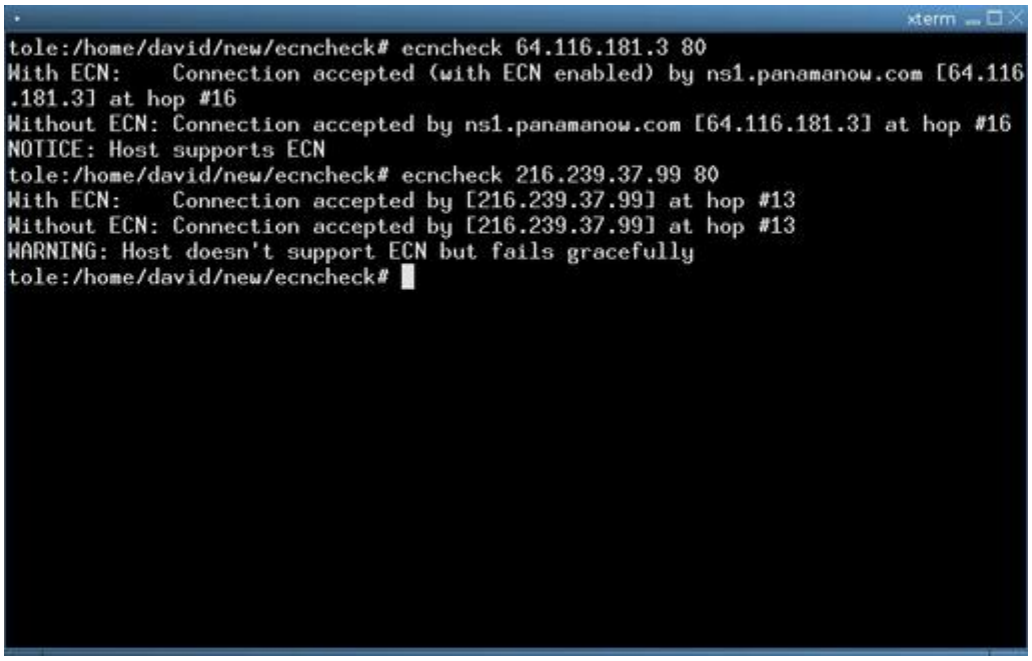
```
Mar-10-04 11:41:46 24.151.130.231 <nishizakfoltz882_7@hellokitty.com> to: david@
pananix.com Bayesian spam
Mar-10-04 11:41:59 Connected: 64.62.197.83:3212
Mar-10-04 11:42:00 64.62.197.83 <bounce-isp-wireless-782743@lists.isp-lists.com>
 to: david@pananix.com local or whitelisted
Mar-10-04 11:45:07 Connected: 64.62.197.83:4493
Mar-10-04 11:45:08 64.62.197.83 <bounce-isp-wireless-782743@lists.isp-lists.com>
 to: david@pananix.com local or whitelisted
Mar-10-04 11:45:13 Connected: 69.47.44.153:38240
Mar-10-04 11:45:15 69.47.44.153 <linux-users-bounces@smtp.linux-sxs.org> to: dav
id@pananix.com local or whitelisted
Mar-10-04 11:48:27 Connected: 218.166.54.101:3938
Mar-10-04 11:48:35 218.166.54.101 <o_rose_oj@home.pf.jcu.cz> to: craigo@pananix.
com Bayesian spam
Mar-10-04 11:48:36 Connected: 220.75.66.157:3505
Mar-10-04 11:48:40 220.75.66.157 <srbrowngx@uk.mahjong.dk> to: craigo@pananix.co
m Bayesian spam
Mar-10-04 11:48:41 Connected: 220.75.66.157:3553
Mar-10-04 11:48:46 220.75.66.157 <kate_duke_oa@friends-partners.ru> to: craigo@p
ananix.com Bayesian spam
Mar-10-04 11:48:49 Connected: 80.136.58.160:1823
Mar-10-04 11:49:08 80.136.58.160 <tguthriess@telsur.cl> to: craigo@pananix.com B
ayesian spam
:
```

ecncheck: firestuff.org

David A. Bandel

Issue #122, June 2004

Have you ever found that you can't get to a particular Web site from a Linux system, but other systems have no problem? Then, you later find out ECN (explicit congestion notification) was turned on in the Linux system and that's what caused the problem? Would you like to be able to check a site for compliance with the RFCs or simply find out if they support ECN? ecncheck can do it for you. Requires: libpthread, glibc and raw socket access.



```
tole:/home/david/new/ecncheck# ecncheck 64.116.181.3 80
With ECN:    Connection accepted (with ECN enabled) by ns1.panamanow.com [64.116
.181.3] at hop #16
Without ECN: Connection accepted by ns1.panamanow.com [64.116.181.3] at hop #16
NOTICE: Host supports ECN
tole:/home/david/new/ecncheck# ecncheck 216.239.37.99 80
With ECN:    Connection accepted by [216.239.37.99] at hop #13
Without ECN: Connection accepted by [216.239.37.99] at hop #13
WARNING: Host doesn't support ECN but fails gracefully
tole:/home/david/new/ecncheck#
```

LJ Index—June 2004

- 1. Linux percentage savings vs. Microsoft Windows: 70
- 2. Savings in US dollars to Van Shung Chong Holdings by switching to Linux and putting off server purchases: 100,000
- 3. Billions of hits to the NASA Web site in the first two months of 2004: 7.5
- 4. Peak traffic at NASA in January 2004 in Gb/second: 7
- 5. Number of Itanium 2 processors to which SGI's Altix platform can scale in a single system image: 256
- 6. Thousands of in-house beta testers for IBM's planned Linux desktops and laptops: 15
- 7. Year-over-year million-dollar increases in Linux server quarterly revenue: 960
- 8. Percentage year-over-year Linux server growth: 63.1

- 9. Percentage year-over-year Linux server shipment growth: 52.5
- 10. Percentage year-over-year Windows server shipment growth: 23.3
- 11. Percentage year-over-year Windows factory revenue growth: 16.1
- 12. Percentage Q4 factory revenue growth over one year earlier, for IBM servers: 17.7
- 13. Percentage Q4 factory revenue growth over one year earlier, for HP servers: 9.4
- 14. Percentage Q4 factory revenue growth over one year earlier, for Dell servers: 19
- 15. Percentage Q4 factory revenue growth over one year earlier, for Sun servers: –1.7
- 16. Percentage Q4 factory revenue growth over one year earlier, for Fujitsu-Siemens servers: 17.7
- 17. Number of AMD Opteron processors in the new Linux-powered Dawning 4000A supercomputer in China: 800
- 18. Trillions of operations per second the Dawning 4000A will perform: 10
- 19. Position this will put the Dawning 4000A on the current Top500 list: 4

- 1, 2: *BusinessWeek*
- 3, 4: vnunet.com
- 5: SGI
- 6: TheLinuxShow
- 7–16: IDC
- 17–19: AMD

Nameko: wiz.homelinux.net/php.php

David A. Bandel

Issue #122, June 2004

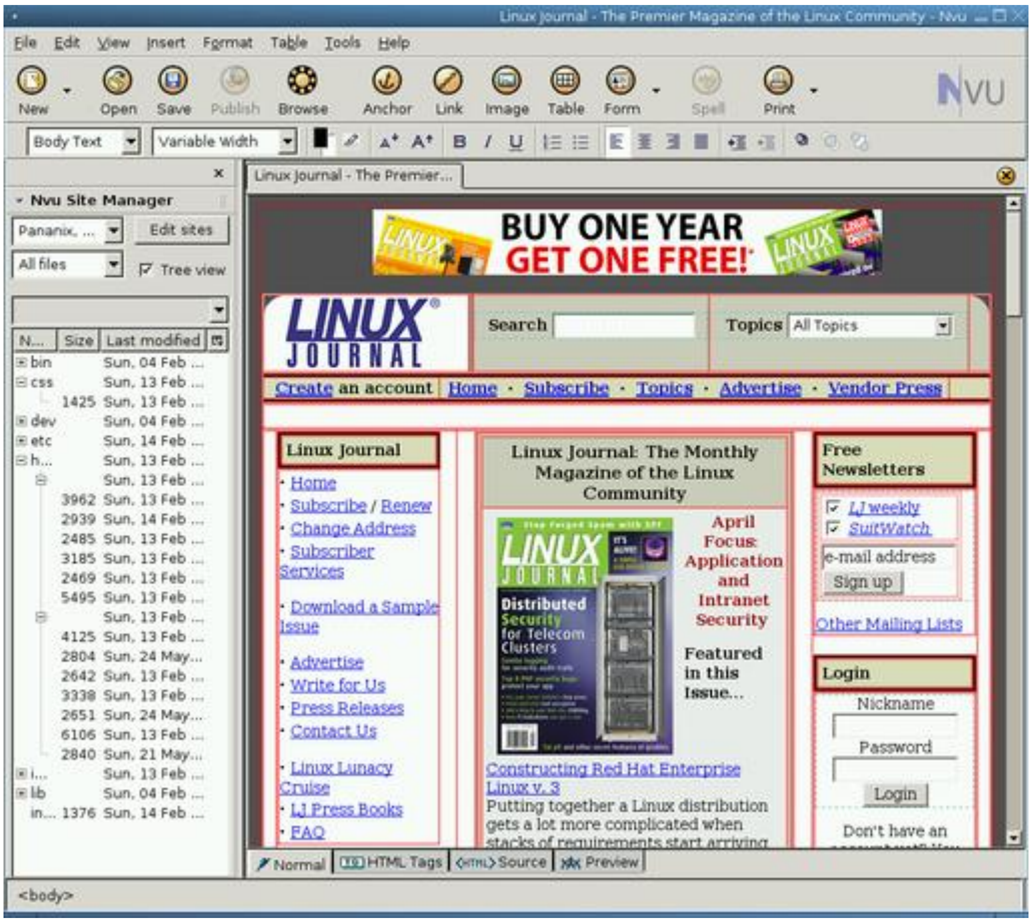
Nameko is about the easiest-to-install Web-mail application around. This spartan utility allows you to read and send e-mail on your mail/Web server. If all you need to do is read and send e-mail, and other e-mail Web applications are too difficult to install, perhaps this is worth a try. Simply copy it to your Web server, call up the URL and log in. Requires: mail and Web server with PHP, browser.

Nvu: www.nvu.com

David A. Bandel

Issue #122, June 2004

For those of you who have been looking for a Linux HTML editor to replace FrontPage or Dreamweaver, the newly released Nvu (N-view) aspires to be that replacement. With Nvu, the most difficult Web-coding jobs become a snap, including forms, CSS and more. Most of the features are available and appear to work quite well. Future releases should round out the missing pieces; the framework appears to be in place. Requires: libmozjs, libxpcom, libplds4, libplc4, libnspr4, libpthread, libdl, libgtk-x11-2.0, libgdk-x11-2.0, libatk-1.0, libgdk_pixbuf-2.0, libpangoxft-1.0, libpangox-1.0, libpango-1.0, libgobject-2.0, libgmodule-2.0, libglib-2.0, libm, glibc, libgcc_s, libstdc++, libX11, libXi, libXext, libXft, libXrender, libfontconfig, libfreetype, libz and libexpat.


Linux Journal - The Premier Magazine of the Linux Community - Nvu
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Anchor Link Image Table Form Spell Print
NVU
Body Text
Variable Width
Nvu Site Manager
Edit sites
All files
Tree view
Linux Journal - The Premier...
BUY ONE YEAR GET ONE FREE!
LINUX JOURNAL
Search
Topics All Topics
Create an account · Home · Subscribe · Topics · Advertise · Vendor Press
Linux Journal: The Monthly Magazine of the Linux Community
Free Newsletters
LJ weekly
SuitWatch
Sign up
Login
Nickname
Password
Normal
HTML Tags
Source
Preview
<body>

They Said It

A lot of our customers are Linux savvy. So we give them root access. Please, go hack away. We're already seeing a lot of community development. In fact, some of the features we're introducing...came from the community.

—Bob Fuhrman, President & CEO of Interact-TV, from an interview at CES

One more reason I think Linux may be getting close to serious contention: Wal-Mart, the retailing giant, is beginning to play the same hardball with desktop computing that it has played with everything else.

—Dan Gillmor, *San Jose Mercury News* technology columnist

I don't think Linux is questioned now...I used to have to write a paper as to why anyone would use the technology, but the systems we have developed have been rock-solid for the past five years.

—Ray Loyzaga, CommSecure (www.computerworld.com.au)

Linux server growth continued to accelerate, demonstrating that Linux servers are taking on important roles in IT customers' computing infrastructure. What began with edge and Web-centric workloads is branching out to include HPC and commercial workloads.

—Jean Bozeman, IDC (www.idc.com)

If Pacman had affected us as kids we'd be running around in dark rooms, munching pills and listening to repetitive music.

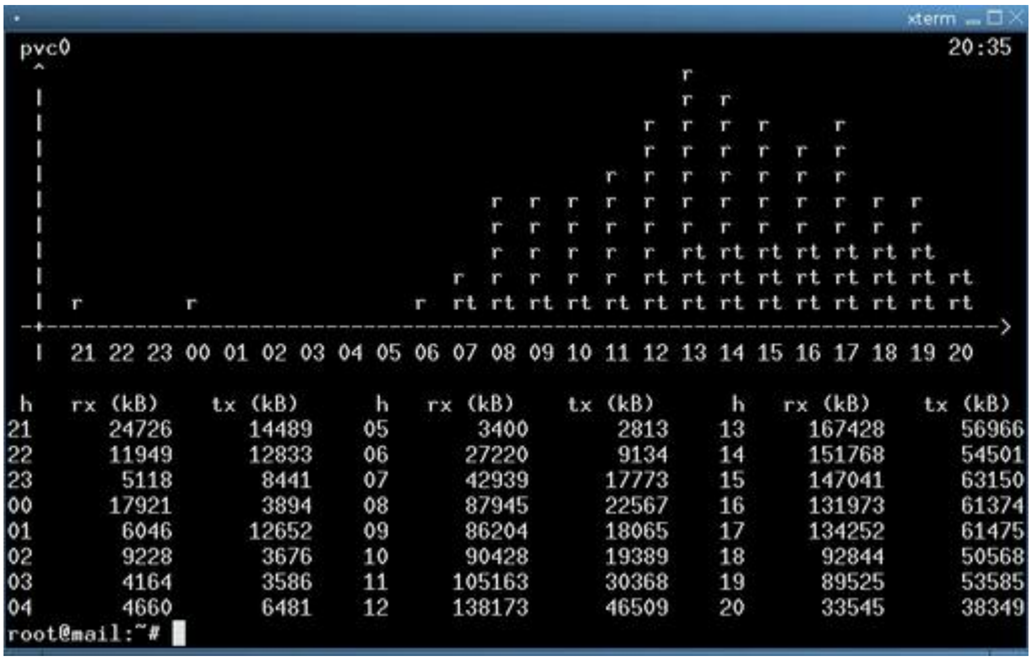
—Marcus Brigstocke (marcusbrigstocke.com), not Kristian Wilson from Nintendo, so fix your .sig, please—Ed.

vnStat: **humdi.net/vnstat**

David A. Bandel

Issue #122, June 2004

vnStat is a console-based traffic monitor. Unlike other monitors, it doesn't require root access or use many resources. Basically, vnStat updates its logs from the /proc filesystem. If you need a lightweight monitor that can be run by anyone, not only root, this might be a good choice. Requires: glibc.



```
pvc0                                                                     20:35
 ^                                                  r
 |                                                  r  r
 |                                               r  r  r  r     r
 |                                               r  r  r  r  r  r
 |                                            r  r  r  r  r  r  r
 |                                   r  r  r  r  r  r  r  r  r  r  r  r
 |                                   r  r  r  r  r  r  r  r  r  r  r  r
 |                                   r  r  r  r  r  rt rt rt rt rt rt rt
 |                                r  r  r  r  r  rt rt rt rt rt rt rt rt rt
 |  r        r                 r  rt rt rt rt rt rt rt rt rt rt rt rt rt rt
-+--------------------------------------------------------------------------->
 |  21 22 23 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20

 h    rx (kB)     tx (kB)      h    rx (kB)     tx (kB)      h    rx (kB)     tx (kB)
21       24726       14489   05        3400        2813   13      167428       56966
22       11949       12833   06       27220        9134   14      151768       54501
23        5118        8441   07       42939       17773   15      147041       63150
00       17921        3894   08       87945       22567   16      131973       61374
01        6046       12652   09       86204       18065   17      134252       61475
02        9228        3676   10       90428       19389   18       92844       50568
03        4164        3586   11      105163       30368   19       89525       53585
04        4660        6481   12      138173       46509   20       33545       38349
root@mail:~# 
```

Archive Index Issue Table of Contents

Advanced search

On the Web

Reaching the People

Heather Mead

Issue #122, June 2004

This month, the *LJ* Web site offers an interview with the developer of GNU Radio and more voices from the new hacker democracy.

In late 2003 and so far in 2004, much has been made of the effect the Internet is having on the 2004 US presidential election. From candidate blogs to open-source celebrities installed high in certain candidates' steering committees, it's true that the Internet and open-source mentalities have altered the national campaign playing field. If nothing else, the Dean campaign aptly demonstrated what can happen when people have the means to connect digitally with like-minded people across the country. Does it still qualify as grass roots if some of the grass is in Oregon and some is in Massachusetts? Or has grass roots been replaced by “Net roots”?

Being in the midst of one of the most contentious campaigns in the past 100 years, it's hard to have the distance necessary to understand the real effect the Internet has on these proceedings. What's more important, reading Kerry's blog from the road or determining whether on-line voting has the potential to make more people vote and to deliver those votes in an accurate manner? Perhaps the part of this changing electoral/political landscape that offers the most potential for serious good is the application of open-source philosophy. Doc's feature article in this issue, “Hacking Democracy”, explores some of these questions and the larger issue of what should be open source's role in government proceedings of all types. A companion piece, “Voices from the New Hackers of Democracy” (www.linuxjournal.com/article/7474) on the *Linux Journal* Web site, offers more commentary by some of those involved in this conversation.

As the Internet and Internet-based technologies grow and reach more people, what becomes of old technologies, such as radio? They morph into something else, in this case, software radio. In this issue, Eric Blossom writes about GNU Radio, his project that involves “getting code as close to the antenna as possible”. In a follow-up Web interview with our Editor in Chief, Eric talks about the prospects and politics of software-defined radio (www.linuxjournal.com/article/7502).

As we head into summer, we're at work on the design for the 2004 Ultimate Linux Box. Don Marti is back to author this year's article, and he offers a “ULB 2004 Preview” (www.linuxjournal.com/article/7503) on the *LJ* Web site. He writes, “Our first four-way SMP Ultimate Linux Box will run a 2.6 kernel. It's also the first Ultimate Linux Box with more memory than can be addressed with 32 bits. It has more memory than our 2000 Ultimate Linux Box had hard drive space. We built a 2.6 kernel with the default configuration, in a tmpfs filesystem in RAM, in 1 minute 41 seconds. Drivers permitting, it'll be a personal video recorder, too—for HDTV.” It's about time someone remembered ultimate is the key word in Ultimate Linux Box.

Our Web site also features regular columns by Dave Phillips, writing on Linux and audio; Bruce Byfield, writing on OpenOffice.org; and Chris DiBona, writing on, well, whatever he feels like. Other upcoming topics include a multipart article on the quest for fast boot times and an introduction to the Pygame library.

If there's a topic or application you'd like to see covered on the *Linux Journal* Web site, or if you have an open source in government project you would like to write about, drop me a line at info@linuxjournal.com.

Heather Mead is senior editor of *Linux Journal*.

Best of Technical Support

Our experts answer your technical questions.

Net Radio vs. Downloads

An engineer wants to listen to music over the Internet using a 1MB, two-way DSL connection. Another engineer complains during this session that his downloads are very slow. Now these two engineers want the technician to monitor the network and show the percentage of use of the music and downloads on this Ethernet connection. The technician has Ethereal installed and resolves most problems with this tool, but he finds it a poor performance monitoring tool. I, being the technician, want to know if I am not using Ethereal to its full capability in monitoring the network. Or, is there another tool I can use to perform this function?

—
George Van Tuyl

gvtlinux@xmission.com

A product called the Multi Router Traffic Grapher, www.mrtg.org, when configured properly and in conjunction with your iptables and other network settings, could provide the data analysis you need.

—
Felipe Barousse Boué

fbarousse@piensa.com

If you want to use traffic control on your Linux router to balance the need for responsive interactive connections with the demand for fast downloads, check out the “Ultimate Traffic Conditioner” section of lartc.org.

—
Don Marti

info@linuxjournal.com

Changing Desktop Environments

How do you change the default window manager on Red Hat? Let me clarify. I log in with text, and when I type `startx`, it starts GNOME, but I'd like to use KDE.

—
Weoh

errff@hotmail.com

You can change the content of /etc/sysconfig/desktop from `DESKTOP="GNOME"` to `DESKTOP="KDE"` or `DESKTOP="WINDOWMAKER"`, and X will start the corresponding window manager. The file that controls which window manager it starts is /etc/X11/xinit; take a look at it.

—
Felipe Barousse Boué

fbarousse@piensa.com

Hardware Specs for Laptop?

I am attempting to load Red Hat 7.3 on a Sony VAIO notebook, model no. PCG-GRT 250P. I need to know what horizontal and vertical sync values must be entered for the LCD screen.

—
Noel Haynes

atannah@hotmail.com

From what I have found, your VAIO should have a horizontal range of 27–115 and a vertical range of 50–160. My last suggestion would be to install, say, Fedora Core 1 instead of Red Hat 7.3, which has newer X versions. I do not own a VAIO, but in looking around I found this page that may be a good reference for you: www.prongs.org/linux/sony/sony.html. The page www.linux-on-laptops.com links to configurations for many laptop models.

—
Felipe Barousse Boué

fbarousse@piensa.com

Displaying an Image in an X Application

I'm working on digital image processing on Linux. I have read a .bmp image successfully. Now I have to display the image using X11. I have used XCreateImage() and XPutImage(), but the figure is not being displayed. I feel the problem is with the arguments I pass to these functions. How do I do it?

—
shilpa.s

s_shilpa_15@yahoo.com

Try using the sample code from this X tutorial: www.xmission.com/~georgeps/Xlib/Xlib_JPEG_tutorial.html.

—
Christopher Wingert

cwingert@qualcomm.com

Encrypted Cross-Platform Remote Desktop?

Does Linux come packaged with something similar to Windows 2000 Advanced Server's Terminal Services? I have a Windows XP workstation that needs to be configured to have remote access to a Linux server's applications. The connection would be over the Internet—a VPN, perhaps?

—
William

billhcmp405@hotmail.com

Yes, there is a cross-platform remote desktop solution. VNC for Linux and Microsoft Windows is available under the GNU GPL from RealVNC (realvnc.com), and there may also be VNC packages available for your Linux distribution. If your distribution supplies a VNC package, you should probably use it so that the software will be easier to update if necessary. See the

RealVNC FAQ for how to encrypt a VNC session using SSH.

—
Don Marti

info@linuxjournal.com

Mysterious Freezing Hardware

I recently purchased a new computer for the boss, an AMD 2500+ with a Biostar M7NCG motherboard, 1.5GB of RAM and two IDE hard drives. After a couple of months, it began freezing once or twice a day. No single application is to be blamed. I've tried replacing hardware and even went so far as to put the hard drives in a totally different computer, which did not resolve the issue. What are the best ways of troubleshooting such a problem?

—
Leroy Powell

lspow80@yahoo.com

First, this looks like a hardware problem, not necessarily Linux-related though. Without specifics, it is hard to guess what is going on. Let's suppose you have Linux in it, then when you say “freezing”, is it the whole machine that freezes or only the graphical user interface? Those are quite different situations. I'd first look at the system logs. If it is the whole machine locking, probably switch memory banks, take some memory out and try the system. If the GUI is what freezes, I would inspect the video card. Also, check for the latest drivers, or a newer Linux distribution may solve your problem.

—
Felipe Barousse Boué

fbarousse@piensa.com

I hate to say it, but in a business, a PC that's out of warranty and has an intermittent problem is worth less than the time required to diagnose it. Assuming your boss pays you a fair wage, he's better off buying a new box with a longer warranty. Put this time-sucker out on the sidewalk.

—
Don Marti

info@linuxjournal.com

Binding a Key to Send a Signal

What should I do to generate a SIGHUP signal when I press Ctrl-K?

—
Narayanan Iyer

narayanan.iyer@wipro.com

You can use the bash bind command to assign certain behavior to key sequences. The .inputrc file is related as well. Do a `help bind` from within bash or see the “shell builtin commands” section of the bash man page for further information. Take a look at en.tldp.org/HOWTO/Keyboard-and-Console-HOWTO.html for more information, or go to www.linuxgazette.com/issue55/henderson.html for a hot key creation tutorial.

—
Felipe Barousse Boué

fbarousse@piensa.com

New Products

HP Compaq Business Desktops, GForge Collaboration Station Servers, USB DUX and more.

HP Compaq Business Desktops

HP announced a new line of business desktop PCs, available in three categories: the Compaq 2000, 5000 and 7000 series. The dx2000 features four DIMM slots for single or dual-channel memory configurations, plus eight USB 2.0 ports. It can be preconfigured or customized with Celeron or HyperThread-enabled Pentium 4 processors with speeds up to 3.0GHz, up to 80GB hard drives, up to 1GB of DDR SDRAM and a choice of optical drives. The dc5000 is available in a small form factor or microtower design, both designs have tool-less access to internal components and drives. It can be customized with Celeron or Pentium 4 processors with speeds up to 3.0GHz, up to 160GB hard drives and up to 4GB of DDR SDRAM. The 7000 series offers the most advanced features for high-end environments and applications. All systems run Mandrake Linux.

Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, California 94304, 800-752-0900, www.hp.com/products/desktops.

GForge Collaboration Station Servers

The GForge Group now is offering fully configured GForge System Servers. GForge is a collaborative development environment that allows large software projects to be managed and tracked through a single interface. With Collaboration Station I, Collaboration Station II, GForge Rackmount I and GForge Rackmount II, GForge offers four plug-and-play, out-of-the-box server solutions. GForge collaborative development and management software, plus Debian, come pre-installed and configured on a server appliance built to the client's specifications. Sites are customizable and come with one-year hardware and software support.

The GForge Group, 512 129th Street, Urbandale, Iowa 50323, 408-907-2744, www.gforgegroup.com.

Embedded Planet RFID Reader Development Kits

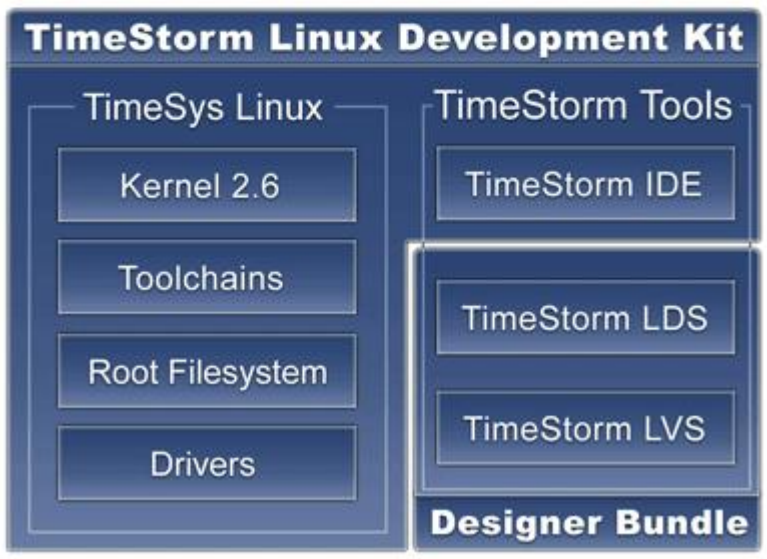
Embedded Planet has three new EPC Class 1-compliant RFID reader development kits available, all based on Alien Technology's ALR-9932-A reader/programmer OEM module. The development kits can be used in applications for industry, retail, military, aerospace and more. The kits enable RFID system integrators and hardware manufacturers to integrate read/write RFID capabilities into products. The EP RFID Reader Development Kit provides OEMs with 32-bit processor solutions for PowerPC, ARM and MIPS using the OEM's reader engine. The EP RFID Reader/Concentrator Development Kit includes a PowerPC single-board computer and up to ten ALR-9932-A RFID readers. The EP RFID Reader/802.11 Development Kit includes the ALR-9932-A engine and a Wi-Fi (802.11a/b/g) system with a PowerPC processor.

Embedded Planet, 749 Miner Road, Cleveland, Ohio 44143, 440-646-0077, www.embeddedplanet.com.

TimeStorm 2.6 LDKs

Linux Development Kits (LDKs) built on the 2.6 kernel now are available from TimeSys for the PowerPC architecture. TimeStorm LDKs include the Eclipse-powered TimeStorm IDE and a complete embedded Linux distribution based on the 2.6 kernel, cross-platform GNU toolchains and device drivers for supported hardware platforms. These tools and the 2.6 kernel combined offer developers better performance, improved device driver framework, expanded connectivity support, support for headless systems and real-time responsiveness. Initial LDKs are for the Motorola PowerPC 8260 processor and the Intel IA32 processors, with support for additional processors and architectures coming soon.

TimeSys, 925 Liberty Avenue, 6th Floor, Pittsburgh, Pennsylvania 15222, 888-432-8463, www.timesys.com.



USB DUX

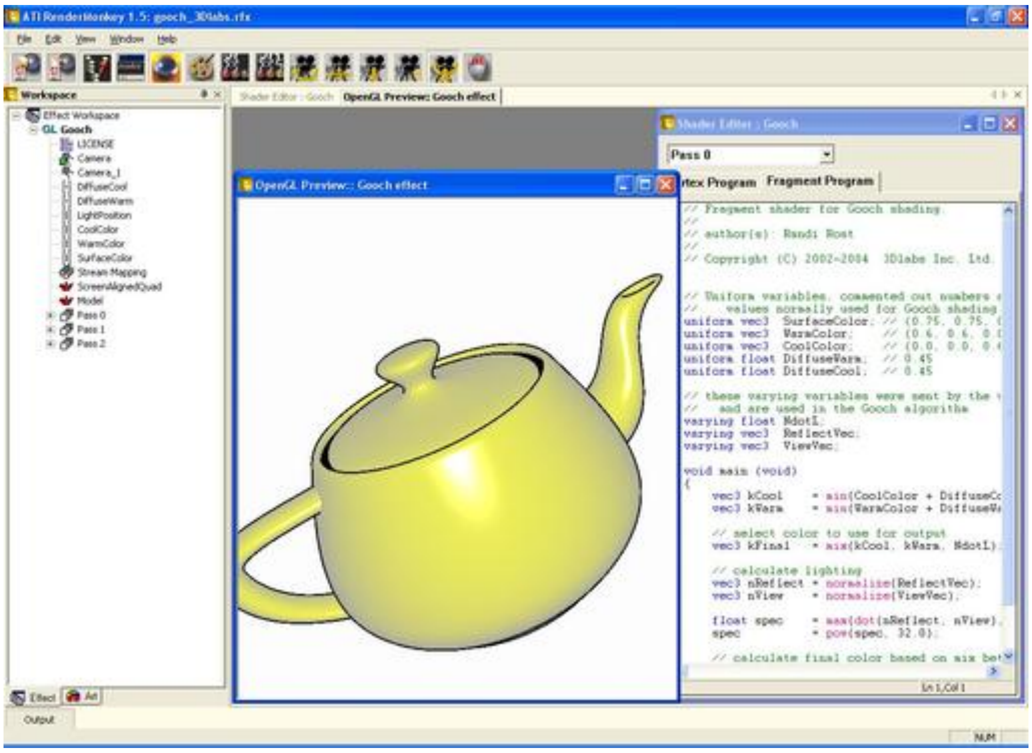
The USB DUX is a data acquisition unit for real-time monitoring and control. The board takes advantage of the real-time speed and power of the USB port and is designed to work under Linux in both embedded and desktop designs. The driver is part of the COMEDI framework, so it works directly with Labview. Two versions of the USB DUX are available, one with a standard D connector and one with 14 additional BNC connectors and two K-type thermocouples. Both the driver and the firmware are licensed under the GPL. USB 1.1 is supported under both 2.4 and 2.6 series kernels, and high-speed USB 2.0 is supported with kernel versions 2.6.4 and higher.

INCITE, University of Stirling, Stirling, UK, FK9 4LA, tech@linux-usb-daq.co.uk, www.linux-usb-daq.co.uk.

RenderMonkey 1.5

RenderMonkey 1.5 is an integrated, real-time OpenGL Shading Language development environment with a graphical user interface used to write and adjust shader attributes, such as the definition of color, textures, graphics states, scalars, vectors and matrices. It eliminates the need for handwritten OpenGL Shading Language programs for many visual effects. The shader development tool supports a plugin architecture that enables developers to create new, customized components as needed. RenderMonkey works with a number of shading languages, including OpenGL's, HLSL and assembly-level languages. Shaders and their attributes are stored in XML files for interchange with other applications.

3Dlabs US, 9668 Madison Boulevard, Madison, Alabama 35758, 877-286-1185, www.3dlabs.com.



```
// Fragment shader for Gooch shading
//
// author(s): Randi Rost
//
// Copyright (C) 2002-2004  3Dlabs Inc. Ltd.


// Uniform variables, commented out numbers a
//     values normally used for Gooch shading
uniform vec3  SurfaceColor; // (0.75, 0.75, (
uniform vec3  WarmColor;    // (0.6, 0.6, 0.0
uniform vec3  CoolColor;    // (0.0, 0.0, 0.6
uniform float DiffuseWarm;  // 0.45
uniform float DiffuseCool;  // 0.45

// these varying variables were sent by the v
//    and are used in the Gooch algorithm
varying float NdotL;
varying vec3  ReflectVec;
varying vec3  ViewVec;

void main (void)
{
    vec3 kCool    = min(CoolColor + DiffuseCo
    vec3 kWarm    = min(WarmColor + DiffuseWa

    // select color to use for output
    vec3 kFinal   = mix(kCool, kWarm, NdotL);

    // calculate lighting
    vec3 nReflect = normalize(ReflectVec);
    vec3 nView    = normalize(ViewVec);

    float spec    = max(dot(nReflect, nView),
    spec          = pow(spec, 32.0);

    // calculate final color based on mix bet
```