Advanced search

Linux Journal Issue #69/January 2000

Archive Index

Advanced search

Focus: Networks and Communication

Marjorie Richardson

Issue #69, January 2000

While the technical focus for this month's issue is Networking and Communications, we also wanted to feature “World Domination” for this first issue of the year 2000.

While the technical focus for this month's issue is Networking and Communications, we also wanted to feature “World Domination” for this first issue of the year 2000. To do that, we invited Eric Raymond to write an editorial for us and asked permission of Aardman Animations to use their penguin, Feathers McGraw, on our cover. Both graciously consented, and both are fun to have in our magazine.

As it has been said many times, this is the “age of communication”, and the growth of the Internet has proved it. More and more people are reaching out to get in touch with others all over the world, using their computers. With the right equipment and software, you can call internationally without paying long-distance fees to the phone company, and see the person you are calling on your computer screen. This can be done using Linux. In the business world, communication with others, as well as keeping track of what they are doing, must be fast and efficient. E-mail service and the Internet provide that service in both cases. If the network is down, we find ourselves wandering through the hallways until it is back up, and we can resume contact with the outside world. In enterprise today, we cannot afford to cut ourselves off from the rest of the world. Our networks must provide the secure communication channel we need to stay on top.

This month, our feature articles cover Internet Telephony, advanced packet testing and the features of the new release of BIND. We also have Part 2 of David Morgan's series on virtual private networking, and the real-world story of a company that provides audio and video streaming over the Internet. And don't forget “Strictly On-Line”, where we have an article describing how to hold company meetings in the virtual world.

LJ Archive CD

Our latest *Linux Journal* archive CD is now available. To celebrate the year 2000, we have our “Millennium Edition” containing issues 1 through 56 (1994 through 1998), as well as all issues of *Linux Gazette* through September 1999. Our most comprehensive CD yet, covering five years of publishing, it belongs in everyone's library. (Am I biased, or what?) Buy one for yourself and one for each of your friends!

—Marjorie Richardson

The OpenPhone Project—Internet Telephony for Everyone!

Greg Herlein

Issue #69, January 2000

Call your friends and family from your computer—a look at the future or the present? With Linux, the future is now.

The OpenPhone Project (http://www.openphone.org/) has a simple goal—to phone-enable every computer on the planet. If a computer can browse the Web and play audio from Internet radio stations, it should be able to place and receive phone calls, too. The basic technology is available today. The OpenPhone Project aims at fostering the development of the software that can make this a reality.

VOICE OVER IP Book Review

Internet telephony, or voice-over IP (VoIP) technology, has come a long way in the last few years, and the path is cluttered with a confusing array of ever-changing standards. I will aim to provide an introduction to the technology and standards on Internet telephony with the hope that more people will participate in the OpenPhone Project. Internet Telephony is one of the most exciting and fastest-growing areas in today's telecommunications world—and it's perfect for Linux!

The OpenPhone Project

Phone-enable every computer—what does that mean? It means every computer should to be able to act as a phone—hopefully, a very smart and programmable phone. There are several practical ways to accomplish this feat. The simple way is to mimic some of the capabilities of a phone with a computer's normal resources. For example, use the sound card and a microphone/speaker to communicate, and use either a screen-pop dialog or play a sound file to indicate *ringing*. Another much more sophisticated technique is to use a telephony interface board that allows you to plug a normal phone into your computer. Given today's inexpensive and powerful phones (especially cordless phones) and the availability of low-cost telephony interface cards, this is the approach favored by the OpenPhone Project.

Telephony interfaces come in a wide and confusing array of types and capabilities. They seem to fall into two basic categories: high-density multiline digital interface cards (T-1 or better) and low-density analog cards. Since most folks don't have a T-1 circuit in their home, the OpenPhone Project focuses on the low-density analog cards. These cards are no more expensive than a decent video card, and provide a whole host of critical features that make Internet telephony work. The simplest thing they do is let you plug a normal, inexpensive analog phone into your computer and provide full control over the ringing and audio. However, they also provide the hardware-based audio compression so critical to voice quality. This hardware technology will be discussed in more detail below.

The OpenPhone Project is not Linux-specific, although it certainly has a strong Linux leaning. Like fax machines, the usefulness of Internet phones depends upon how many other devices are out there that they can interoperate with. If a Linux-based OpenPhone can only call other Linux computers, it's of limited value. However, if an OpenPhone can call any other computer regardless of operating system, or any other phone anywhere in the world—that's powerful! This is the goal of the OpenPhone Project.

The Hardware

At the time of this writing, the OpenPhone Project is using the telephony interface boards made by Quicknet Technologies, Inc. The Internet PhoneJACK is available in PCI (peripheral connection interface) and ISA (industry standard architecture) bus versions and provides an RJ-11 interface into which any normal analog phone can be plugged. It also has headset/microphone and handset jacks. The Internet LineJACK has the RJ-11 POTS (plain old telephone system) port and an additional PSTN (public switch telephone network) port for use as a gateway to the normal phone system. The Internet LineJACK is presently available with only an ISA interface. These low-cost interface boards have Linux and Win32 drivers and provide the ability to use a single standard telephone with your computer. More information is available at Quicknet's web site (http://www.quicknet.net/) or in my article in last September's issue of *Linux Journal* (“Voice-Over IP for Linux”).

Talks are underway with several other hardware vendors to participate in the OpenPhone project. We encourage such vendors to make drivers available for as many operating systems as possible, and join us in making OpenPhone work across all platforms.

Voice-Over IP Technology

Telephony can be thought of as having two major parts: the audio channel used to communicate and the signaling channel(s) used to control the audio channel. In the traditional public switched telephone network (PSTN), the signaling happens on a separate private network owned and operated by the telephone companies. This separate signaling network uses a protocol called Signaling System 7 (SS7); it is used to control the setup and ending (teardown) of calls, using the switched circuits in the system.

The audio-channel portion of the PSTN is mostly composed of two parts: the local loop, and the central office (CO) equipment that links all the local loops together. The local loop is the pair of copper wires that comes into your house or business—the analog line. The CO equipment is made up of high-speed digital links; it is beyond the scope of this article. The local loop uses analog signals to carry your voice to the CO, where it is digitized and sent to the CO on the other end of the call. The other CO takes the digital signal, converts it back into an analog signal and sends it down the analog line to the called party.

Internet telephony works the same way, except that the digitization process happens at your computer, and the high-speed digital link between end stations is the Internet. Your telephony interface (or sound card) converts the analog signal to digital and sends it in IP packets to the destination. The destination computer converts it back to analog sound signals and plays it out your phone. Simple, right? Ah, but like most things that involve computers, the details are the tricky part.

Internet Telephony Audio

An Internet phone must carry the audio signal between the parties talking—this is the fundamental requirement. We all know that the Internet can carry high-fidelity audio, since many of us have listened to streaming audio music or radio programs using our Internet connection and PC. Internet telephony is a bit more complicated, though, since it has to do full duplex (two-way) audio in real time, and the human ear is quite sensitive to latency. Too much delay, and the call sounds like it's traveling across a satellite instead of across the Internet!

Bandwidth is also an issue, especially over dial-up lines to the Internet. In the normal PSTN world, the audio on the local loop is digitized into a 64Kbps digital data stream and presented to the phone company CO equipment, which then compresses the data for transport across the phone system backbone. Simple Internet telephony packages use a similar digitization, yielding a bandwidth requirement of 64Kbps in each direction for full-quality voice. That requires 128Kbps of bandwidth, plus extra for signaling and overhead, so this does not work well across a normal dial-up Internet connection that is perhaps 56Kbps.

Luckily, voice compression technology has come a long way and works quite well. It is commonplace to obtain 8:1 (or better) compression ratios using today's voice coders (codecs). Modern Internet telephony interface boards provide these codecs as standard features.

The normal phone system sets up virtual (or sometimes real) dedicated circuits for the voice packets to flow across. The Internet is much more chaotic. On the Internet, packets can take different routes from second to second, and may arrive at their destination out of order, late, not at all or staggered in time. Late packets might as well be lost, since if the packet is not there on time and ready to play, there will be a gap in the audio. Out-of-order packets are probably not useful, either—if it's out of order, it's probably late. Packets that arrive at the destination are unlikely to do so in a neat and orderly way. Sometimes they take a bit longer or shorter than average—the stream is not uniform. This staggering is called jitter. Several techniques are used to deal with these problems: the Real Time Protocol (RTP) and jitter buffers.

The Real Time Protocol

Audio packets need to arrive on time and in the correct order. RTP is a user-level protocol that provides a way to encapsulate data into packets time-stamped with enough information to allow the proper playback of audio. The protocol has a companion control protocol (RTCP) that provides a means for the end points to stay informed about the quality of services they are receiving. The complete protocol is described in RFC1889, which can be found at www.ietf.org/rfc/rfc1889.txt. Several implementations of RTP are available under various open-source licenses. See Resources for more information and pointers to those libraries.

Jitter Buffers

The Internet is not predictable, and packets sent at nice predictable rates do not always arrive at the same rate they were sent. They can arrive slightly sooner or later than the average latency. If a packet arrives slightly late, the audio device which is ready to play the next frame of audio has nothing to play. This causes a discontinuity that degrades the audio quality. In simple applications, this is a short silent period that makes the voice sound choppy. In more advanced applications, comfort noise or some form of audio blending is used to mask the gap. This can make the voice sound warbled or as if it's under water. These effects are observed while using cell phones or Internet phones, by the way—packet loss and latency is a general problem of all digital audio applications. RTP provides a means for the application software to know if packets are out of order, missing or running early or late. The application can then make the appropriate corrections for missing or misordered packets.

The best solution for dealing with jitter (short of a perfect transport path, which would eliminate it entirely) is to make sure the audio device never “runs dry”. This requires jitter buffers. These buffers store a small amount of audio at the beginning and then stay a bit ahead of the flow so that there is always an audio frame to play. Several one-way real-time audio/video programs will buffer up many seconds of data before playing any sound, thus ensuring they will always have plenty of data on hand to play. However, every frame buffered adds latency, which is especially relevant to voice calls. If you buffer 90 milliseconds (ms) of data, you add 90ms to the delay between the time the words are spoken and when they are heard. When added to the latency of the Internet itself, this can rapidly become unacceptable. Some people believe 200ms of latency is a good upper limit for what the human ear can tolerate. Given that many Internet locations are 100ms or more (one way) apart, adding 90ms of jitter buffer latency accounts for a significant fraction of the acceptable delay. A delicate balance lies between the need to jitter buffer and the need to reduce latency.

Jitter buffer techniques are one of the areas where I think the Open Source community can contribute significantly. There is much thought and experimentation going on now to find algorithms and techniques to use adaptive jitter buffers in two-way real-time audio streams. I suspect the cumulative efforts of the Open Source community will find some excellent solutions to this in the next year. I hope the OpenPhone Project can be a catalyst for making it happen sooner.

Audio Codecs

The standard PSTN digitization technique is described by the ITU G.711 specification. This document describes an 8KHz sampling rate using 8 bits per sample, for an effective 64Kbps data rate. There are two subsets of this technique: A-Law and Mu-Law. These are scaling factors that take into account the sensitivity of the human ear and allow for a more efficient utilization of the 8-bit encoding space for recording the human voice. Both are in wide use and are considered standard codecs.

However, 64Kbps is excessive for Internet telephony. Two-way audio encoded in this manner yields 128Kbps of audio alone (not counting signaling or overhead), making it impractical for use across a dial-up link, and wasteful of bandwidth even on a LAN or WAN link. Voice compression is the answer, and there are several options from which to choose.

Audio codecs can either be implemented in software or provided by the telephony interface card using an on-board DSP. On a lightly loaded PC, the extra processing load incurred by the encoding is not particularly burdensome, although it can be on more heavily utilized machines. Software-only solutions can add some latency, since the encoding usually happens in a user-space program and is thus at the whim of system load and the scheduler. Internet telephony cards use on-board DSPs to perform the encoding, virtually eliminating the load on the host CPU and the associated latencies.

Perhaps the most widely used codec in Internet telephony is G.723.1. This codec provides either a 6.3 or 5.3Kbps data stream of packets that hold 30ms of audio each. The encoding requires 7.5 milliseconds of lookahead, which, when combined with the 30-millisecond frame, totals 37.5ms of coding delay. This is the minimum theoretical latency using this codec. Of course, real-world latencies will include the time on the Internet and application-level processing at both ends (and the jitter buffer—mustn't forget that).

G.723.1 is patented technology. The patent holders charge royalties for its use, making it impossible to use in open-source software. However, most telephony boards (including the hardware used by the OpenPhone Project) include this codec as part of the board price. Aside from other technical advantages of telephony interface boards over sound cards, the inclusion of licensed compression codecs on the card is perhaps the single best reason we need these cards. With the codec in hardware, we can write open-source software and not violate any licenses. I'd like to point out that the G.723.1 codec is very different from the G.723 codec (a much higher bandwidth codec in which the source code is widely available, but not often used due to high bandwidth requirements).

Rapidly gaining in popularity are the G.729 and G.729A codecs. These codecs use an 8KHz sampling rate, a 10ms audio frame and a 5ms lookahead buffer, for a total of 15ms processing latency. Because the frames are only 10ms long, this codec suffers less from packet loss—the software has less to recover, given the loss of any one packet. G.729A is a version of the codec that uses fewer DSP resources, making it easier to implement on lower-performance (cheaper) DSP chips. The audio quality these codecs can deliver is astounding. I recently experienced a LAN-based call using the G.729A codec and stood next to the person calling me. I could detect no perceptible delay between the direct path (his mouth to my ear) and the link across the network. If I had not known otherwise, I would have sworn it was a normal PSTN call. This codec is the future of Internet Telephony.

In-Band Signaling

In-band signaling includes all the tones you normally hear in the course of a call, including ringing, busy-signal, fast-busy and the all-important dual-tone modulation frequency digits (DTMF). These tones are a normal expected part of using a telephone, but unfortunately, many do not compress well with the algorithms discussed above. One alternative, the one chosen by the

OpenPhone project, is to pass these signals as a separate data type within the real-time audio stream. This has the advantage of ensuring that the signals are reproduced accurately at the other end of the connection regardless of the audio codec needed, and it reduces the bandwidth needed to convey the data.

This technique is described in an Internet Draft (see Resources). It basically specifies a new RTP data payload type; the application layer simply sends the data in the stream along with the normal audio. One very tricky aspect needing more work is the code required to remove the actual audio signals from the audio data stream prior to compression. Should these tones and signals get compressed, the codecs will distort these signals at the far end, reducing the chances of properly detecting the signal or tone. It's far better to filter these signals out prior to compression and to pass the signal across the network directly, allowing the other end to reproduce the signal with perfect accuracy.

Internet Telephony Signaling

In the traditional PSTN world, the phone companies have a private dedicated network to do all the signaling. Every phone system in the world that wants to interoperate with the rest of the world must use the SS7 network and play by its rules. On the Internet, though, there is no separate dedicated network—control and data signals use the same network. How does signaling work, then? It's not very simple. There are several different ways to get the job done, and several different “standards” in place to provide guidance. This is a simplification of the issues and standards. I refer you to the Resources section for places to obtain more detailed information.

Several standards for Internet telephony are currently in use: H.323, SIP and MGCP are the predominant ones in use now. I cannot hope to provide more than a quick introduction to these protocols, but can certainly provide an overview and pointers to where to learn more.

H.323

H.323 is a family of protocols established by the International Telecommunications Union (ITU). It is highly complex, but covers most of what one would want to do with audio-visual conferencing or calling over networks. H.323 arose out of the telecommunications world, not from the Internet world. It is a binary protocol that uses ASN.1 notation/encoding for message passing. This protocol is in wide use and presently has the highest level of use. Many commercial packages use H.323. H.323 is so widely deployed that many feel new VoIP applications must support the protocol. However, H.323 is so complicated that interoperability between different implementations is not good. For better or worse, Microsoft's NetMeeting product seems to be a common benchmark for measuring interoperability. Since Microsoft makes

NetMeeting available for free, it's easily available, and if a product can interoperate with it, you are assured a wide user base. There are several excellent open-source projects working to provide this protocol to the community—most notably the OpenH323 Project. These folks have a working protocol stack now capable of making a call to a NetMeeting client on a Win32 machine. This is tremendous success, and I hope to see even more improvements as this code is used in more and more projects. The OpenPhone Project will use the OpenH323 libraries and the slimmer “Simple Endpoint Terminal” code that Vovida Networks derived from it. See Resources for more information and places where you can obtain that code.

SIP

The Session Initiation Protocol (SIP) is described in RFC-2543. This IETF protocol arose from the Internet community; it has a feel not too different from HTTP and similar protocols. It is a text-based protocol that uses fairly simple commands to get the job done. It is much more oriented towards telephony services, whereas H.323 provides details on full multimedia audio-visual services. SIP is growing in popularity because of its relative simplicity and ease of implementation. A few links to more information on SIP are in the Resources section. At the time of this writing, I am unaware of an open-source implementation of SIP, although there has been much talk about starting such a project. The OpenPhone Project would very much like to see this project, and perhaps we can facilitate getting it started. We will certainly provide a home for it if anyone is interested.

MGCP

The Media Gateway Control Protocol (MGCP) is a protocol and API for controlling voice gateways from a centralized server or “Call Agent”. The protocol is text-based and relatively straightforward. The complete protocol is described in an IETF draft (see Resources for link information).

This protocol is finding wider acceptance and is thought by many to be superior to H.323. However, it is interoperable with H.323, since the Call Agent can act as an H.323 Gatekeeper. This is probably where the future of VoIP call control is going, and the OpenPhone Project intends to use this protocol as the default wherever possible. An open-source implementation of MGCP is available from Vovida Networks (see Resources), and several commercial versions are available. However, since the specification is not ratified and is changing slightly (but frequently), there is no assurance at this time that two different MGCP implementations will work together. This is an area where open source can make a huge contribution to Internet telephony. Vovida's MGCP implementation is released under the LGPL license, which allows for commercial use without releasing the associated source code for the non-open portions of the application. With polish, such an open-source MGCP could be used in a wide variety of applications and systems, assuring interoperability by virtue of using the same base code for the protocol. The OpenPhone Project will be using the Vovida code for its MGCP control with the hope of extending and enhancing the protocol to stay current with the ratified MGCP protocol.

OpenPhone—A Basic Description

We envision a basic application that provides a framework for Internet telephony using plug-in modules to accomplish the various tasks to be performed. For example, one plug-in module would provide the signaling tasks, and another would provide the audio tasks (including use of RTP and jitter buffers). We envision an H.323 module, a SIP module and an MGCP module—the user could select which one to use based on the interoperability requirements. New modules could be plugged in as needed to evaluate different RTP/jitter buffer techniques. As improvements are made in signaling or audio-transport modules, all the user has to do is drop in the new module.

All that is needed to make this approach viable is a common API for applications to use to perform basic high-level functions. The modules would all provide those API functions; the appropriate module would be used to provide the actual functionality. Since many people refer to the signaling and audio code as a “stack”, we call this the Stack Adaption Layer (SAL). The SAL is a commonly defined and adopted API that will allow the application developer to focus on the functionality of the program, and the stack developers to focus on the detailed lower-level implementation.

To extend the concept down a layer and provide platform and hardware independence, we envision a Hardware Adaption Layer (HAL) that provides a set of common functions for controlling and using the hardware. This allows the signaling/audio stacks and the SAL to work seamlessly on top of the HAL code, regardless of which Internet telephony card is in use at the hardware level. This approach will allow us to realize the goal of true cross-platform multi-vendor interoperability.

Design specifications for the SAL and HAL layers are in active development as of this writing, and by the time of publication there should be several white papers and some reference code available on our web site. We encourage active participation and have started a mailing list devoted to the project. You can join this majordomo-hosted list by sending a “subscribe” message to developers@openphone.org.

Conclusion

Internet telephony has many intricate pieces that work together to make it function well. There are programs available now that work—but not at a level that is truly useful. Most of these implementations fall short because they don't use modern compressed-codecs, or because they don't use something like RTP and good jitter-buffer techniques to control the audio stream. Most also lack a standardized signaling protocol that provides interoperability with other programs. The OpenPhone Project aims to provide a new, highly flexible framework that uses plug-in modules to provide the components discussed above. It will be based on inexpensive, easily available hardware that can be used in normal, commonly available computers. OpenPhone will use modern techniques to provide near-toll-quality calls between any two phone-enabled computers, and hopefully will foster growth and acceptance of Internet Telephony to the point that all computers are phone-capable.

Resources

Greg Herlein (gherlein@quicknet.net) has been using Linux since the early 1.0 series kernels. He's built and run systems on remote mountains, on research ships on the high seas and in corporate high-reliability settings. He's now a member of the technical staff at Quicknet Technologies, Inc. in San Francisco and is leading a team of developers to create the next generation of Linux-based IP telephony software.

Archive Index Issue Table of Contents

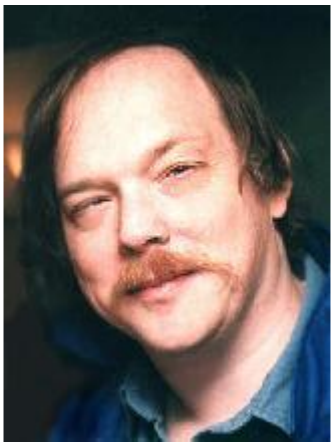
Advanced search

Guest Editorial: World Domination

Eric Raymond

Issue #69, January 2000

Eric takes a serious look at what the world will be like when Linux is the dominant operating system—or is he just kidding?



World domination. It's a powerful and faintly sinister phrase, one you might imagine some B-movie mad scientist muttering as he puts the finishing touches on some infernal machine amidst a clutter of bubbling retorts and crackling Tesla coils. It is not a phrase you would, offhand, associate with cuddly penguins.

When Linus says he aims at world domination, it seems as if he wants you to laugh at the absurd image of a gaggle of scruffy hackers overrunning the computing world like some invading barbarian horde—but he's also inviting you to join the horde. Beneath the deadpan self-mockery is dead seriousness—and beneath the dead seriousness, more deadpan self-mockery. And on and on, in a spiraling regress so many layers deep that even Linus probably doesn't know where it ends, if it ends at all.

The way Linux hackers use the phrase "world domination" is what science-fiction fans call a "ha-ha-only-serious". Some visions are so audacious, they can be expressed only as ironic jokes, lest the speaker be accused of pomposity or megalomania. It is in much the same spirit that I sometimes describe myself as Linus' Minister of Propaganda, as if I were some evil communist bureaucrat in a campy remake of Orwell's *1984*.

Such conscious self-subversion can serve many purposes. There's a touch of Zen in it—a reminder from ourselves to ourselves not to take our desires so seriously that we stop noticing reality. It's also tactically valuable. Like the cuddly penguin mascot, it invites our most dangerous adversaries not to take us seriously. They're serious-minded people themselves, too caught up in competitive testosterone games to have any room for Zen. When they hear “world domination”, they don't hear the irony, and thus tend to write us off as nutcases or flakes. That's good; it gives us time and room to blindside them.

Of course, articles like this are part of that game. We hackers are a playful bunch; we'll hack anything, including language, if it looks like fun (thus our tropism for puns). Deep down, we *like* confusing people who are stuffier and less mentally agile than we are, especially when they're bosses. There's a little bit of the mad scientist in all hackers, ready to discombobulate the world and flip authority the finger—especially if we can do it with snazzy special effects.

So, “World domination now!” we declaim, grinning at the hapless suits and wondering which ones are smart enough to be in on the joke. Because, of course, a world of mostly Linux machines wouldn't really be a domination at all. Linux has no stockades, no guard towers, no barbed wire, and no End User License Agreement saying you can't modify and can't look inside and don't have any kind of title to the bits on your own disks. When Linux “dominates”, there won't be any predatory monopolies bluffing, bribing and bullying competitors to protect the market for their own overpriced and shoddy software. When Linux “dominates”, software consumers and developers will win big—and only the kind of people for whom domination really does mean controlling other peoples' choices will lose.

That's a future worth working for. Linux world domination means no domination anywhere—a statement which is ironic in the strict rhetorical sense (“expression of something which is contrary to the intended meaning”) but dead serious all the same, and no regress need apply.

No domination anywhere, but rather a rich ecology of competing projects and distributions and customizations, each one of which constantly brings its own improvements to the globally shared code base. A world not of closed pseudo-standards that lock people in, but rather of true open standards that enable humans and programs to cooperate in ever-richer ways. A world not of lock-step uniformity and five-year road maps, but rather one of rapid evolution and variation and flexibility and almost instant market responsiveness to user needs. A world in which programming means solving new problems every day, instead of wastefully repeating work that somebody else locked up and hid away. A world in which play becomes the highest form of work, and vice versa.

The future of “no domination anywhere” will be a better world; not a perfect one, but perhaps the best that we, working as designers and programmers, can hope to create. So if you aren't yet part of our barbarian horde, join us! There's a lot of working and playing to be done in order to get from here to there. It won't be hard to find us; we'll be the mob yelling “World domination *now*!”--and *smiling*.

Eric Raymond is a Linux advocate and the author of The Cathedral & The Bazaar. He can be reached via e-mail at esr@thyrsus.com.

Archive Index Issue Table of Contents

Advanced search

Advanced Packet Data Testing with Linux

Wesley Erhart

Joseph Bell

Marc Hammons

Mark Mains

Issue #69, January 2000

At Nortel Networks, we have developed a Linux-based system for testing a second-generation packet radio service. During system development we explored the details of packet radio, the IP internals of the Linux operating system and device-driver development.

What is your 56K modem doing while you ponder the sophisticated on-line articles at *Linux Journal*'s web site? Nothing. The majority of the time you are connected to the Internet, your modem sits idle. However, your phone call ties up at least one dedicated circuit in a telephone network. The unused capacity and potential revenue of this circuit's bandwidth is lost forever.

Telephone company operators and manufacturers recognize that forming data paths by switching fixed-capacity circuits is inefficient. Interactive data communication is transmitted in bursts. You pull down a web page and whimsically move on to the next, unconcerned about the circuit bandwidth you tie up.

The same problem exists in cellular systems today. Most of the advanced second-generation digital cellular systems reserve a channel intended for voice transmission when making data connections. This scarce and wasted bandwidth, bought or leased from the governments of the world, is a very expensive commodity to lose while customers read web pages.

Glossary

If you only knew what was happening beneath that sleek, stylish mobile phone you carry in your pocket. You would be amazed at what it takes to deliver a simple dial tone—not to mention conferencing your uncle in on a forwarded call from your grandmother while crossing the border from Italy into France, all the while downloading the latest Linux kernel.

GSM

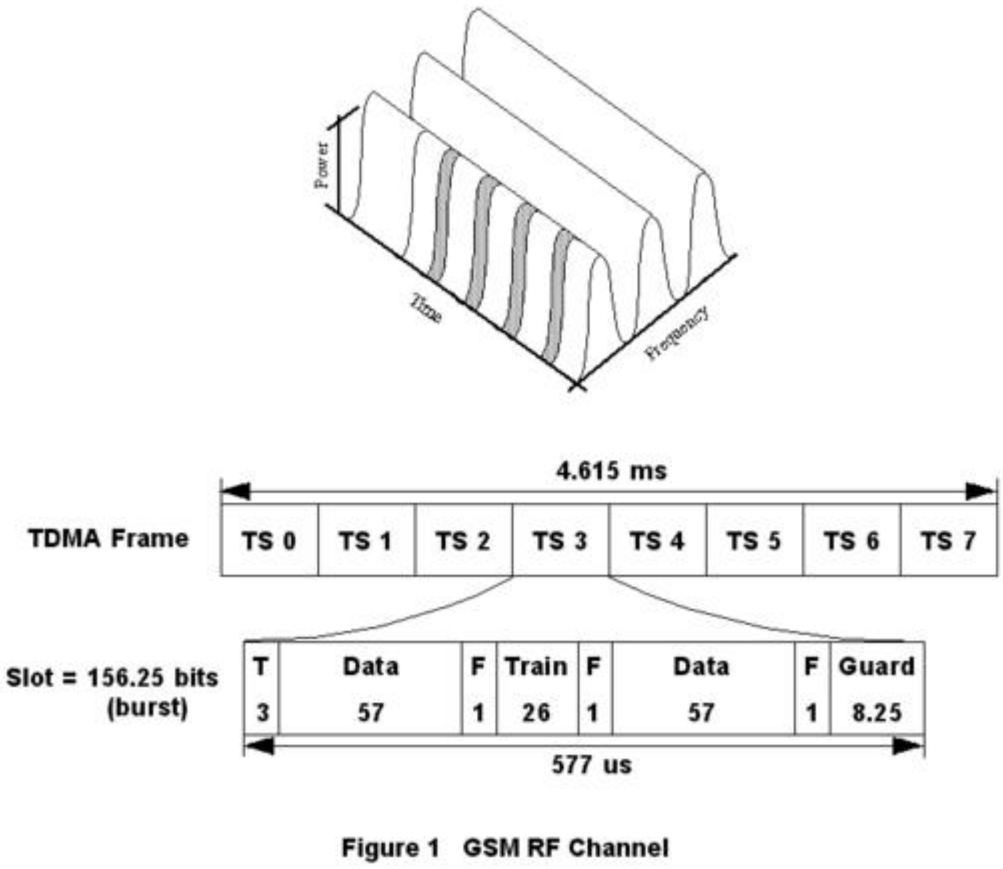


Figure 1. GSM RF Channel

If you are traveling from Italy to France, it is likely that your mobile phone is a Global System for Mobile communications (GSM) cellular telephone. GSM is a worldwide second-generation digital cellular standard. GSM offers many voice and short message services such as calling line ID, call barring, call forwarding and advice of charge. As shown in Figure 1, GSM's radio frequency (RF) interface is comprised of 200KHz bandwidth carriers divided into eight time slots. A time slot is one channel which can carry digitally compressed voice at 13Kbps. On a data call, this same time slot can carry up to 14.4Kbps of circuit-switched data.

The GSM standards body recognized the inefficiency inherent in circuit-switched data and added a service called General Packet Radio Service (GPRS) to the GSM standards. GPRS is a packet overlay of the GSM network. Packet-switched technology efficiency greatly exceeds circuit-switched technology efficiency for carrying and delivering data.

In a packet-switched cellular network, the mobile telephone shares the RF packet channels with the other mobiles in range of a cell site. Rather than reserving a specific time slot or channel on one of the base station's carriers for the entire session, the GPRS mobile uses one or more packet channels to transmit or receive data in bursts. Once the mobile finishes with that bandwidth, it releases the packet channels, making them available for other mobiles. A GPRS mobile reaches a peak data throughput of 120Kbps while a circuit-switched GSM mobile has a maximum throughput of 14.4Kbps for a data connection.

Nortel Networks is committed to providing integrated voice and data solutions with the highest level of quality. As we introduce new technologies such as GPRS, the challenges for a GPRS developer begin. How does a developer verify that the interface to the protocol layer he just completed works correctly? How does the developer complete an end-to-end call when there are no mobiles yet in existence that use the protocol? The authors of this article develop test systems that allow the GSM GPRS developers to verify that our GPRS products are performing at a high level of quality and dependability.

GPRS

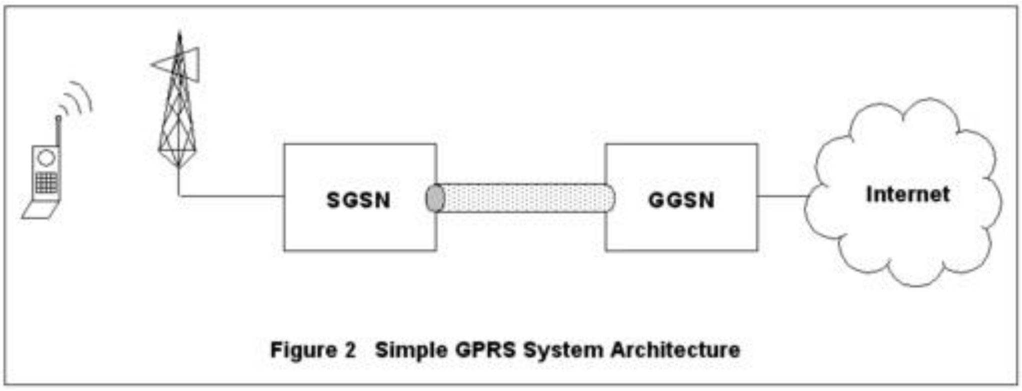


Figure 2. Simple GPRS Network System Architecture

A GPRS system consists of several network nodes including mobiles, base station system, serving GPRS service node (SGSN) and gateway GPRS service node (GGSN). These nodes are shown in Figure 2. When the mobile activates an Internet protocol (IP) packet data session, the base station notifies its SGSN. The SGSN activates the session, called a packet data protocol (PDP) context, with the GGSN. If necessary, the GGSN assigns a dynamic IP address to the mobile's context. When the GGSN acknowledges successful context creation, an IP tunnel using the GPRS tunneling protocol (GTP) is created between the SGSN and GGSN. The tunnel ID (TID) is tied to that mobile's context at both the GGSN and the SGSN.

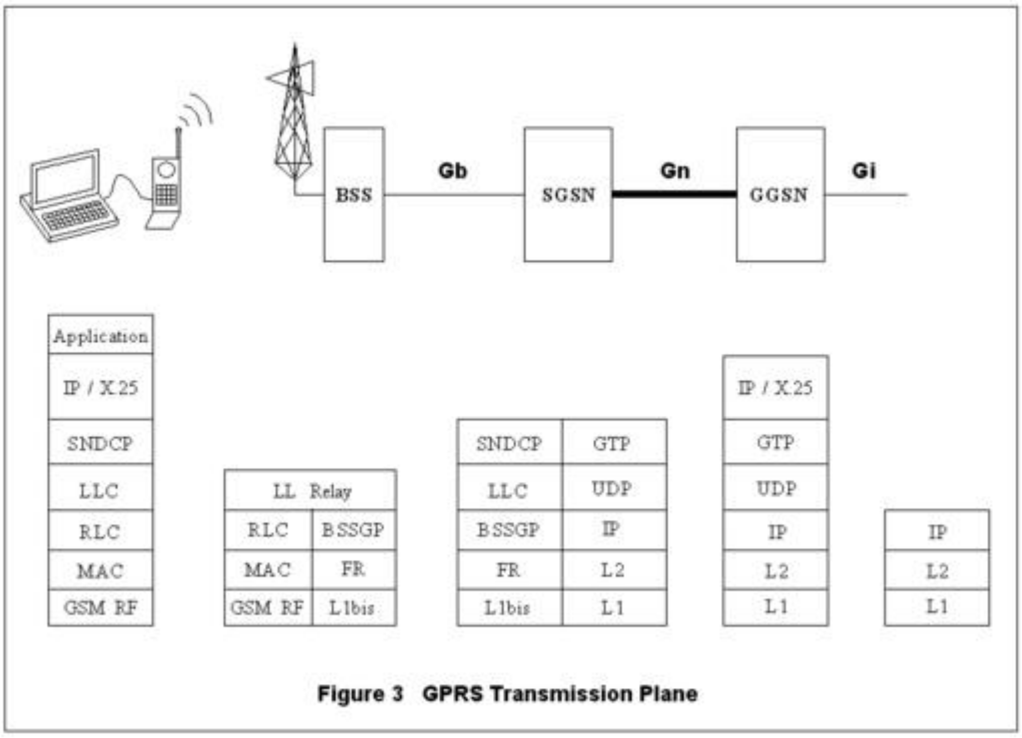


Figure 3. GPRS Transmission Plane

Tracing mobile-originated IP packets through the GPRS nodes (overlooking the RF access details) illuminates some of the testing challenges. When the mobile sends IP data, the packets are compressed and segmented by the subnetwork dependent convergence protocol (SNDCP) and sent down through the logical link control (LLC) layer traversing the stack shown in Figure 3. The mobile's LLC layer ensures the packets arrive in sequence and resends any packets not acknowledged by the SGSN. The LLC layer sends the packets through the radio-specific portion of the RF stack, over the RF link to the base station and the SGSN. At the SGSN, the peer LLC layer acknowledges the received packets, and the SNDCP compression and segmentation are undone.

As shown in Figure 3, the resulting IP packet is enveloped in a GTP UDP/IP message with a header containing the mobile's context TID. The SGSN sends the GTP packet to the GGSN's GTP port. The GGSN extracts the IP packet and places it on the Gi interface. An example of a Gi interface for IP is the Internet. For mobile-bound packets, a similar procedure is repeated in the opposite direction.

Automated Testing

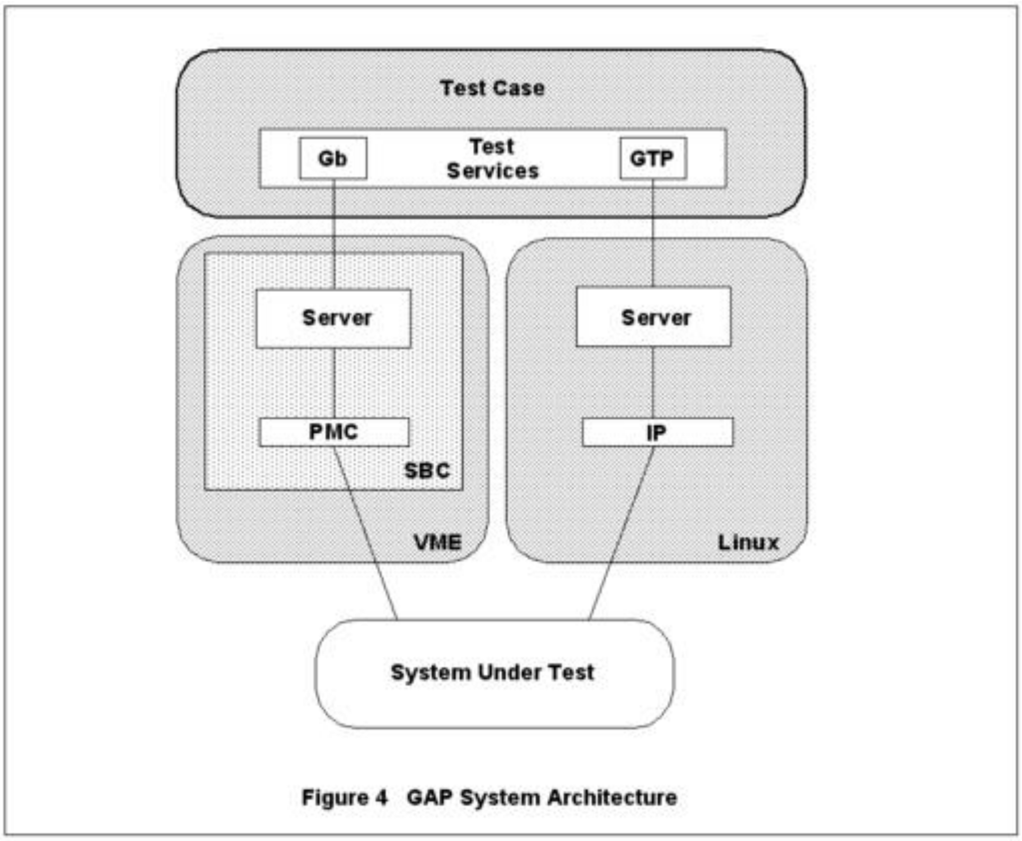


Figure 4. GAP System Architecture

Our typical approach to an automated test solution is to place a test server at each interface to be tested, as illustrated in Figure 4. The GSM developer then writes a test-case client using a “test services” application program interface (API) to send and receive messages via test servers. In response to these test-case requests, a server performs actions to the system under test (SUT). In message-based protocols, the server typically sends the test-case's message to the SUT. When receiving a message from the SUT, the server identifies the test-case client to which the message belongs, and routes and forwards the message to the test-case.

This testing paradigm was developed jointly by various groups at Nortel Networks and was named the test tools framework (TTF). Our automation team adopted the TTF paradigm and built the GSM automation platform (GAP) on top. Although originally written for HP-UX and VxWorks, TTF and GAP have been ported to Linux to allow greater flexibility.

This system performs well when the automation group writes the protocol stacks. However, the GPRS system requires the entire IP stack, including the applications such as TELNET, FTP and HTTP. We quickly realized that it would be impossible for us to generate real TCP/IP traffic without rewriting the IP, UDP, TCP and application stacks.

Linux's open and evolving IPv4 stack is very useful. When testing a protocol, we are required to step outside of the specification and test not only compliance, but error scenarios as well. For example, if the path maximum transmission unit is set to 1499, we will need to send a packet with 1500 octets to see what happens. Linux's open-source kernel and networking code makes this possible.

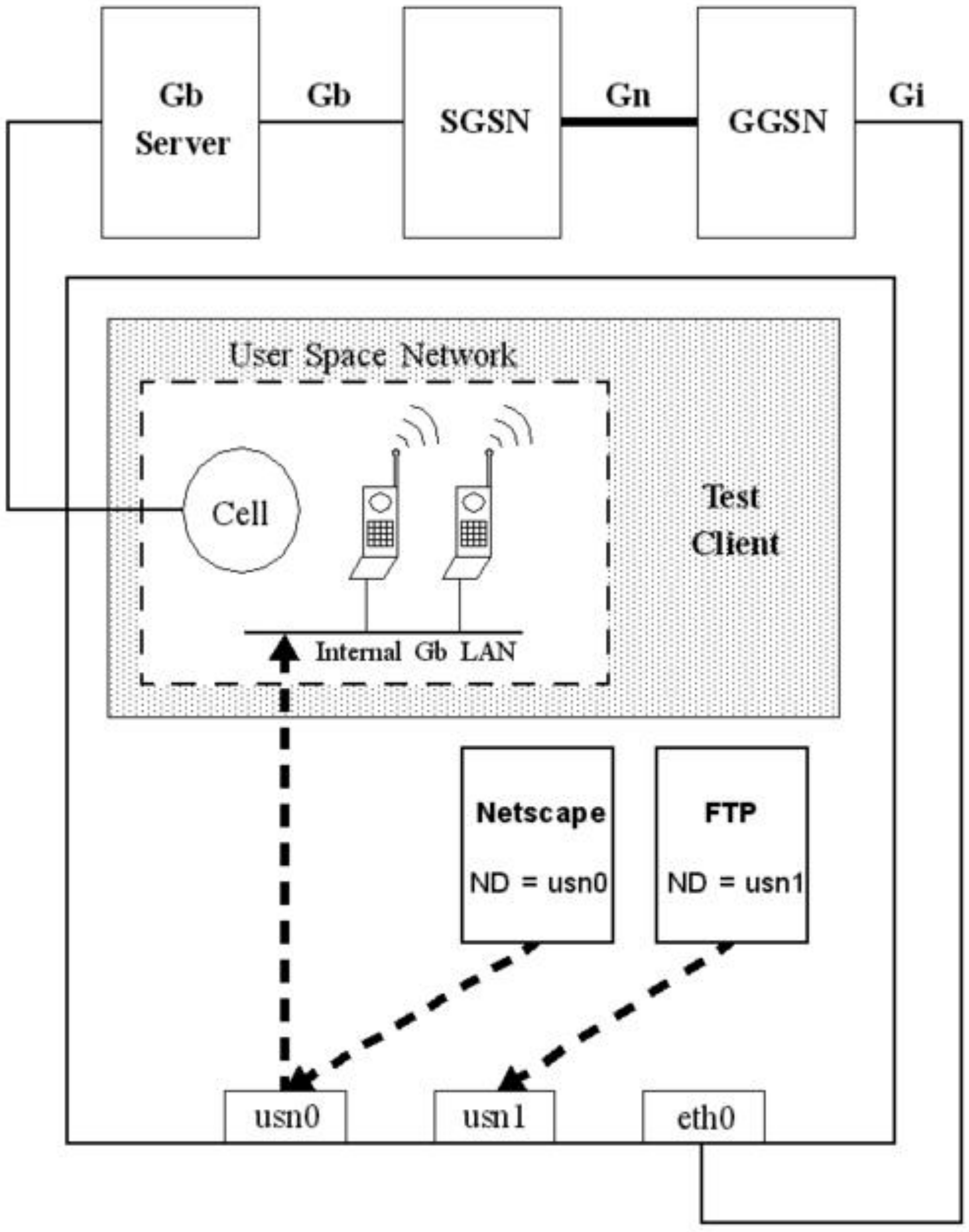


Figure 5 Mobile Application Data Path

Figure 5. Mobile Application Data Path

In the arrangement pictured in Figure 5, the test case running on the Linux test-case machine brings up a tunnel using GPRS signaling paths. Once the tunnel is established, the test-case machine creates an IP alias for the test case's Mobile IP. The test case machine is multi-homed on the user-space Gb LAN and the Gi LAN. The test case uses IP applications or sockets to generate real IP traffic through the Gb test server to the SGSN, through the Gn test server, on to the Gi LAN, and possibly to the test-case machine's Gi interface.

Testing the Gn Interface

The first question to be answered was how to reuse the Linux TCP/UDP/IP stacks. In our GPRS test system, the Gn test server receives the GTP tunneled packets on a UDP socket, pulls the packet out of the envelope and sends it out on another LAN (i.e., the Internet). In the other direction, the Gn test server acts as a gateway for the mobile receiving the packet from the Internet, enveloping it into a tunnel packet and sending it to the SGSN's GTP UDP port.

In the Linux system, the kernel transfers the packet from the Gi LAN Ethernet card to an enveloping, outgoing UDP message. At first, we brainstormed ways to use UDP sockets in kernel space; however, we eventually decided upon a different approach.

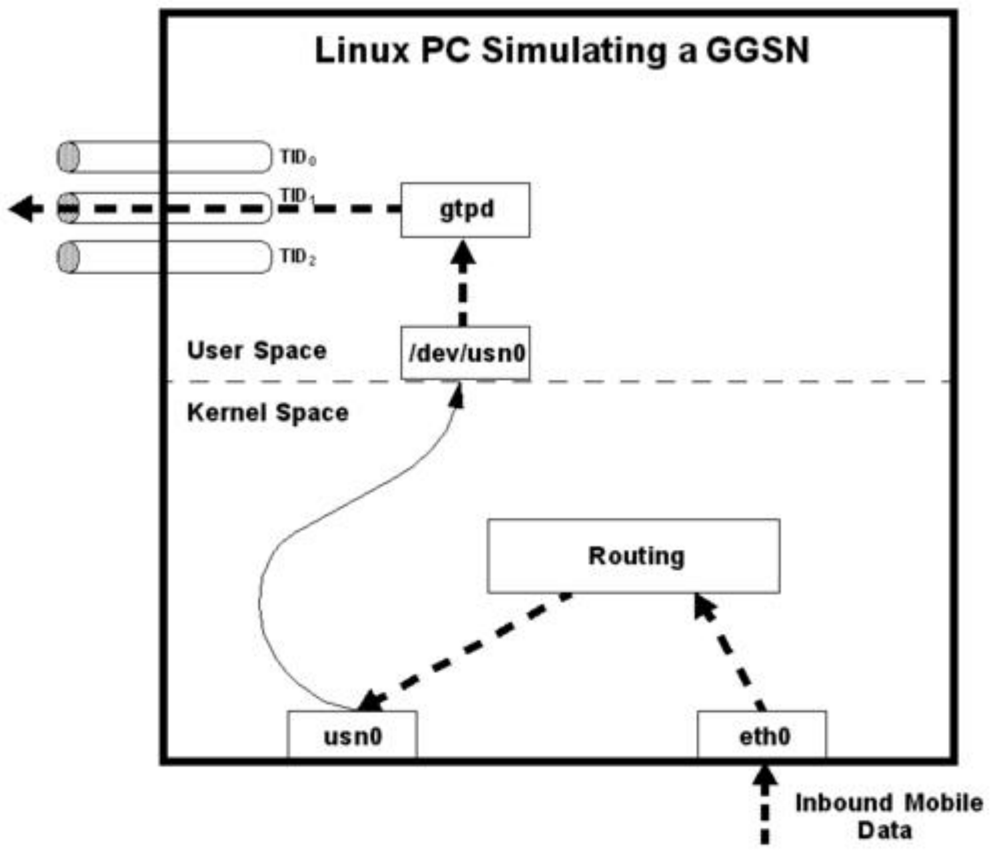


Figure 6   USN Device and Routing Configuration

Figure 6. USN Device and Routing Configuration

Inspired by Alessandro Rubini's book *Linux Device Drivers* (see Resources 3), we sketched out a dual-device driver. One side of the device driver would appear to be an Ethernet card to the Linux kernel, while the other side would be a character device as shown in Figure 6. Applications could read packets sent out the Ethernet side of this device by reading from the character-device side. Likewise, applications could send packets into the Ethernet side of the device by writing to the character file. We called this device the user-space network (USN) device. Effectively, the USN device allows the kernel to send and receive packets from any user-space application that opens the USN's character device file.

The Linux machine on which the Gn test server resides acts as a gateway for the mobile's IP subnet. The routing tables are configured so that the mobile-bound IP packets are routed to the USN device. When the kernel sends packets to this device, our user-space daemon (**gtpd**) triggers and reads the packets from the USN character-device file (e.g., /dev/usn0). Our daemon then places them in an enveloping GTP message and sends the message to the SGSN's GTP port. Before starting this development item, we spent a large amount of time admiring our brilliance.

However, as our driver development continued into what we thought was unknown territory, we began to notice Alan Cox's footprints when we discovered the 2.0 kernel's netlink device (see Resources 4). The netlink device is exactly half of our device driver. It provides a character device and shell for dealing with **sk_buff** structures. At this point, all we had to do was write the Ethernet device half of our driver. While joking that half our job was already done, we were wondering how truly original our idea was.

Around the time we finished the device driver, we obtained the Linux 2.2 kernel. We loaded it onto one of our development machines and discovered Alan Cox's ethertap device. The ethertap device is exactly our USN device. It has interesting potentials for tunneling and packet manipulation. Does your router allow only HTTP packets through? The ethertap device could be used to route Quake packets through your user-space application that envelopes them into benign-looking HTTP packets.

Due to the time it would take to certify TTF and GAP on a new kernel and, maybe, to prop up our diminished egos, we decided to stay with our USN device rather than move to the Linux 2.2 kernel and the ethertap device.

At this point, the work on the Gn test server was progressing nicely and we began to focus on the other pieces of the puzzle.

Testing the Gb Interface

Data transportation in the world of mobility is often treacherous. The Gb protocol stack (see Figure 3) is designed to support reliable data transportation for multiple network addressing schemes (e.g., IP or X.25), serving innumerable applications in the packet data network.

Another large piece of the puzzle was the development of a test server and test services to test the Gb protocol stack. This development began about halfway through the development of the GGSN test server. Naturally, Linux was our primary development environment, and we planned on reusing the USN device and the Linux UDP/TCP/IP stacks all over again. However, this time they would be used to test the Gb stack by simulating calls via virtual mobiles, cells and base stations.

The server portion of the Gb test interface is a distributed application consisting of two primary components. One component resides on a PowerPC single board computer (SBC) housed in a VME chassis running a commercial RTOS. This component drives a frame-relay interface running on PCI (peripheral component interconnect) Mezzanine cards that support multiple T1 or E1 carriers. The second component performs resource management, proxies the frame-relay services and implements the BSSGP (base station system GPRS protocol) and NS (network service) layers of the Gb protocol stack. Thanks to the diversity of TTF, this component runs on either a PC running Linux 2.0 or a PowerPC SBC target running an RTOS.

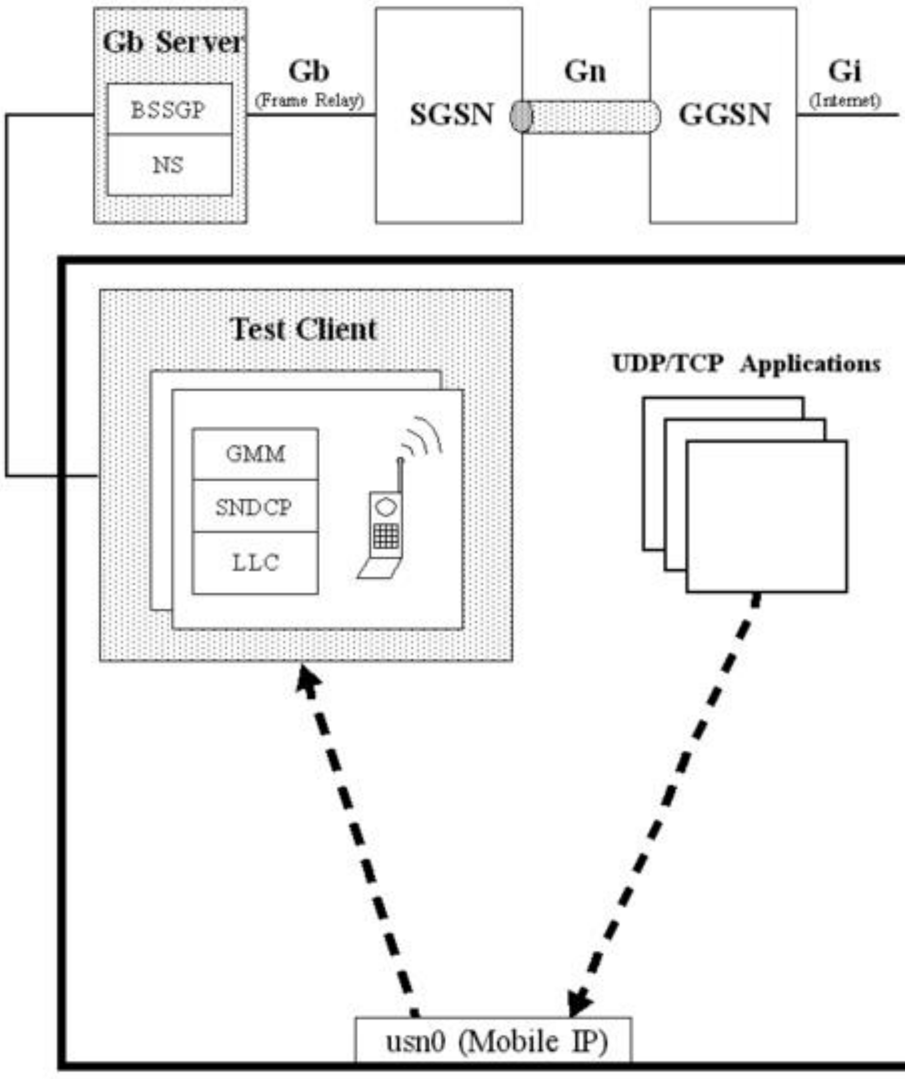


Figure 7. Gb Test System

Within the test case as shown in Figure 7, the developer creates instances of mobiles, cells and base stations using Gb test services, all of which act as clients to the Gb test server. These test services complete the Gb stack by implementing the LLC, SNDCP and GMM (GPRS mobility management) Gb protocol layers per virtual mobile.

Reserving mobiles, cells and base stations, the developer can bring up simulated calls within their test case. Once the calls are up and the PDP contexts are established, the developer uses basic UDP or TCP sockets to send and receive data. The data is encapsulated by the Linux UDP/TCP/IP stack and is routed to the USN device. From there, test services retrieves the data from the USN device, passes it through the Gb SNDCP, the LLC layers of the virtual mobile, and then the Gb test server. At the server, the data is passed through the BSSGP and NS protocol layers and then transported over frame relay to the SGSN. This process is reversed in the receive direction.

All of this was very nifty, but a larger problem loomed on the horizon. What if we wanted to do something “normal”? Perhaps something that could stress the SUT. What about bringing up several simulated calls and using Netscape over them to browse the Internet? That would mean binding an application's execution environment to a specific PDP context.

Kernel Modifications

The test-case machine sits on both the user-space internal Gb LAN and the Gi LAN. In an example where the test case has two mobiles, each with an active context, several routing problems become immediately apparent. In this case, the user might want to launch two instances of Netscape, “binding” each instance to a different mobile's context (i.e., IP address). If one of these Netscape sessions opens a TCP socket to a Gi LAN web server, the kernel has three choices for routing the TCP packets: the kernel could send the packets to either one of the two mobile's USN devices, or to the Gi LAN's real Ethernet device.

For UDP or TCP sockets opened by the test-case developer, the developer can use the **SO_BINDTODEVICE** option by calling **setsockopt**. Binding to a specific device forces all packets sent on that socket to use the bound device; however, applications such as FTP, TELNET and Netscape do not use this option.

After careful observation of a Linux Virtual Private Network software package, we noticed programs such as TELNET and Netscape routed all their IP traffic out the VPN tunnel. Using **nm** on the provided shared library, we found that the shared library contained a complete replacement for the socket API calls.

As web browsing is the major motivator for mobile data services, we had to provide a way to surf the Web through a mobile's context. After many hours looking through the Linux networking code to forego rewriting the entire socket API, we decided to hook into the IPv4 socket creation. Our hook searches the current task's environment table for an environment variable (**ND**), and if it exists, binds it to the device specified by the **ND** variable.

Hooking In

We wanted to do the hook as a loadable module. However, we found we had to modify the Linux 2.0.36 networking code. We modified the kernel to export the **inet_proto_ops** structure globally and tweaked the **sock_register** and **sock_unregister** routines to turn interrupts on and off correctly. The Linux 2.2 kernel appears to have made both these changes unnecessary and opens opportunities for loadable-module “over-binding” of the native IP stack operations.

When our module loads, we hook the socket creation process by calling sock_register with the **inet_proto_ops** structure's create function pointer pointing to our socket **create** function. From this point, whenever a socket is created, our routine is called.

While we were in the kernel, we applied kernel patches to provide a serial console and microsecond resolution time-stamping for received Ethernet packets. We also modified the netlink driver to increase its number of minor devices.

Searching the Environment

As mentioned before, when the hook is in place, our socket creation function searches the current task's environment table for the **ND** variable, and if found, binds the socket to that device.

The function first finds the environment table in physical memory (in which the kernel runs), but the environment table memory location is stored as user memory addresses under the task structure (current**-->mm-->env_start** to current**-->**mm**-->**env_end). We use **get_phys_addr** from fs/proc/array.c to convert from the user address to the physical address. Using this address translation function, our code searches the current task's environment for **ND=**.

If the **ND** variable is found, our code calls **sock_setsockopt** with the SO_BINDTODEVICE option and the specified device. However, sock_setsockopt copies the **ifreq** structure from user space to kernel space. Since our code lives in kernel space, we had to implement the device bind by setting **sk->bound_device** ourselves.

Test Case Execution Environment

Once we had the /dev entries for the USN device and the SO_BINDTODEVICE mechanism compiled into the kernel, we were ready to begin testing our concept. The initial test was very straightforward; we used **ifconfig** to configure a USN device and then set the new kernel-supported **ND** environment variable to that device. From that point, any application we ran in that environment had its outbound IP packets sent into our USN device. To actually watch the packets, we executed **tcpdump -i** on our USN device (in this case, usn0).

The **ping** program was the simplest way we knew how to start sending IP traffic, so we executed it first. Everything proved to work the way we envisioned. The ping packets began showing under **tcpdump**. There was no one reading the device file at the time, so no response was made to the ICMP requests. But for our purposes, everything worked as planned.

We then tried **ftp** and **telnet**, another pair of IP applications that generate IP traffic. We saw the ftp and telnet traffic appear on our USN device, and that was sufficient proof to us that our idea was sound—until we decided to open an xterm.

Naturally, as for any X application, to see the xterm we had to set the **DISPLAY** environment variable to the host name of our desktop machine. Setting the display is simple, but when we tried to launch the xterm, we didn't quite expect what happened next—nothing.

Unfortunately, our kernel modifications and USN device worked a bit too well. After launching the xterm, a glance at the tcpdump output made it apparent what was happening: all of our packets, including the X packets, were being rerouted to the USN device. While that had been a very good thing for ping, FTP and TELNET, it was now a very bad thing. X packets were being written to the USN device, never reaching the X server on the host machine.

We were, to say the least, disheartened. We had several options at that point. The first idea was to note the transport protocol in use (TCP) and the X port number as packets flowed through the kernel, and route those packets through the Ethernet device. This idea was quickly dismissed due to its complexity and lack of elegance.

The second solution was to create a virtual X-user environment the tester could open, as opposed to using telnet or **rlogin** to access the test-case machine. This concept was fostered by our recent introduction to some of the interesting possibilities the VNC (virtual network computing) software package created. VNC's abilities are impressive, due to the fact that it is multiplatform. Unfortunately, the VNC approach proved to be too slow.

The concept was sound, and ironically, preferable to having the tester telnet in, export his display, etc. For all intents and purposes, we could have testers log in from Windows PCs and open VNC sessions to the Linux boxes. However, since the test community was based in X running on HP-UX, why couldn't we turn to an X server that allowed nesting?

The Xnest X server is both an X client and an X server. It is a client to the X server on the host display (in our case, the HP), and it is a server to X applications you execute on the remote machine (the Linux box). The procedure to getting an Xnest session up and running is quite simple. First, you log in to the remote machine, export your display back to the host display, then run Xnest with a local display string. This display string is then used to run remote applications in the Xnest session.

Conclusion

Using our USN device, the **ND** environment variable and our Xnest-user environment, we provide Nortel Networks' GPRS developer with a desktop testing environment where they can exercise the latest GPRS nodes simulating mobile web surfers and the Internet.

Linux's open network code made all our work possible. Using Alessandro Rubini's *Linux Device Drivers* (see Resources 3) and the Linux source code as a guide, we developed a device driver in a very short amount of time, even though we had never before written a device driver. Rubini's book also helped us during our kernel modifications. It is hard to imagine being able to do something this complex in such a short period of time with a closed-source OS.

It is interesting to compare different versions of the Linux kernels. The kernel's networking code is becoming more modular and incorporates IPv6. With the latest kernel versions, all our device driver and kernel modifications would have been unnecessary. In addition to books such as Rubini's and being open source, Linux is a capable choice for any networking application because of the standards it supports and the constant improvements the Linux development community provides.

Resources

Wes Erhart (erhart@nortelnetworks.com) majored in electrical engineering at Texas A&M. He joined Nortel Networks in 1993 and has been managing the GSM automation development group for two years. Not a particularly good coder and harboring an irrational distrust of all computers, Wes did possess enough good sense to marry a goddess and now has a wonderful daughter godette filling his life.

Joseph Bell (jobell@nortelnetworks.com) is a Texas A&M University computer engineering graduate and has been with Nortel Networks since he was a sophomore in college. He enjoys all things Linux and anything that can be programmed. When not coding, he can be found ranching on his farm in north Texas.

Marc Hammons (hammons@nortelnetworks.com) is a University of Texas computer science graduate and has been with Nortel Networks since 1994. He has lost all faith in nihilism and is a born-again Linux enthusiast. In his spare time, he enjoys digging into source code, a good cup of coffee and the game of foosball.

Mark Mains (mmains@nortelnetworks.com) graduated from LSUS with three degrees, in Physics, Math and Computer Science. He started work for Nortel

Networks in 1997 and has been using Linux since 1996. Most of his spare time is spent in front of his computer, building circuits or working on his Z28.

BIND Version 8 Features

Eddie Harari

Issue #69, January 2000

Wondering about the latest version of BIND? Wonder no more. Mr. Harari is back this month to tell us all about it.

Many new features have been compiled into the new version of BIND (Berkeley Internet name daemon), including security bug fixes and several major changes that give the network administrator the needed tools and configuration options to achieve essential tasks unavailable in earlier versions of BIND.

Configuration Syntax

The first thing we need to know is that a few of the terms were changed. There are no more primary servers or secondary servers; these two terms have been replaced with master server and slave server.

The first major difference you will notice in BIND 8 is the configuration file syntax is completely different. This is confusing at the beginning. Until you learn the correct syntax and features, you may use a Perl script that converts from the old style /etc/named.boot file to the new style /etc/named.conf file. Please note that these file names are the default configuration file names for BIND 4 and BIND 8.

Listing 1

The BIND 8 configuration file syntax has a modular style. Each and every section is surrounded by "{ }" and each line is terminated with a ";". Each zone has its own section in the configuration file. Listing 1 is an example of a very simple configuration file.

Files and Utilities Location

With BIND 8, it is possible to change the location of some named files, such as named.pid, named.stats, named-xfer and a few others, with the following statement in the options section of the configuration file:

```
options {
   named-xfer "/
   };
```

This example changes the location of the command executed when the name server wants to make a zone transfer.

Name Checking

BIND 8 also checks the host names in its databases. It checks for conformance to RFC 952. If there is a host name in the zone database which does not conform to the RFC, BIND 8 will consider this zone to have a syntax error and will not load it. This can cause problems for people who are upgrading from earlier versions of BIND and have host names which do not conform, such as names with underscores. If you are upgrading from an earlier version of BIND, and don't want BIND to check the names of your hosts in the database, you can use the following option:

```
options {
             check-names master warn;
             check-names slave ignore;
             };
```

This will tell the server to send warning messages instead of errors when “bad names” occur in a zone for which the server is a master, and ignore “bad names” in zones for which the server is a slave. You can also ignore bad names in the server, but this is not a good thing to do since BIND 8 also checks names in response to queries. If you are a master for a zone which has bad host names in it, some Internet servers will not accept your server's replies. If you want your server to accept replies even if the host names do not conform with the RFC, use the following line in the options section:

```
check-names response ignore;
```

Logs Configuration

BIND 8, unlike older versions of BIND, gives the network administrator full control of the logging messages it produces. This logging does not have to go through the syslog daemon. Error, debug and other messages can be configured to go to the syslog, to a file or to standard error (STDERR). This is done with the help of two things: channels and categories. A channel is configured in the configuration file and tells the name daemon where the logged data should go. A category tells the name daemon what kind of data should go there. The message severity can be one or more of the following: dynamic, debug, info, notice, warning, error and critical.

Here is an example of a logging statement in the name daemon configuration file:

```
logging {
       channel my_new_logging_channel {
       file "named.messages";
       severity dynamic;
       };
};
```

This statement tells the name server to log all current debug-level messages in a file called “named.messages”. Some nice features are available for logging, including log-file versions, default operations, log-message format and more. You should refer to the README file and to the documentation that comes with BIND 8 in order to fully understand how the logging statement works.

Security and Privacy

System and network administrators can now decide which servers and hosts will query their server for each and every zone they serve. This is done via access lists. With access lists, we can tell the name daemon which hosts or networks can send queries to a particular zone. Access lists are defined by a name in the name daemon configuration file. An access list is basically a set of IP hosts or network addresses assigned to a zone in the zone section. An access list can be constructed from the names of other access lists defined in the configuration file. Here is an example of an access list:

```
acl "MySlaveServers" {
            {192.168.1.1/32;};
            {192.168.2.0/24;};
            };
```

This means whenever we use the access list “MySlaveServers” in the configuration file, we want to specify the host 192.168.1.1 and the network 192.168.2.0. The /32 and /24 are the number of bits in the bit mask of the IP address. The predefined access lists in BIND 8 are None, Any, Localhost and Localnets.

- Localhost: any of the IP addresses of this host.
- Localnets: any of the IP networks to which this host has an attached interface.

After we have defined the access list, we can use this access inside a zone section or in the options section:

```
zone "myzone.mydomain.name"  {
   type master;
   file "mydomain.db";
   allow-query { any; }; // This is default,
                         // But good as an example
   allow-transfer { "MySlaveServers";};
   // Others cant zone transfer ...
   };
```

This configuration example gives anyone the ability to query "myzone.mydomain.name", but zone transfers can be done only from 192.168.1.1 or 192.168.2.0. We can put the access lists in the global options section, and it will be applied to all zones specified in the configuration file. With the **allow-query** statement, we can handle inside domains even on our external DNS server; to do that, we simply need to set up our internal zone so that only our internal networks can query host names of that zone.

Daemons often run with root permissions. This was also the case with earlier versions of BIND. With BIND 8, **named** can run as a different user. It is a good idea not to run named with root permissions, for security reasons. If someone exploits the name daemon and tells it to run arbitrary code, then this arbitrary code should not be run with root permissions. The best thing to do is create an account and a group for the name daemon. The command that tells the name daemon to run with a different UID (user ID) is:

```
/usr/sbin/named -u named -g namedgrp
```

This command will tell the name daemon to run with the UID of the user "named" and the GID (group ID) of the group "namedgrp". We must first create this user and group. When running named with a UID that is not root, we must ensure that the name daemon database's permissions and ownership are set correctly. We can change the permissions with the **chmod** command, the group with **chgrp** and the ownership with **chown**.

Notification Messages

Older BIND versions used the following method to synchronize the database between the master server and the slave server. From time to time, the slave tries to invoke named-xfer with the correct zone parameters and the zone database serial number. If the server has a higher database serial number, it will give the slave its new database and update the slave. The database polling interval is set up by the master server inside the zone's database SOA (start of authority) record, and it is known as the refresh-time parameter. This can cause a big delay between updating our master server and the time the data is available all over the Net.

BIND 8 has one more method for synchronizing zone databases between the master and the slave servers. This method is known as the DNS NOTIFY message. The NOTIFY message is basically a message the master server sends to its known slaves whenever a zone's serial number is changed. A NOTIFY message is simply a DNS QUERY message with the opcode NOTIFY inside the message. The slave, after getting the NOTIFY message from the server, replies to the server with a NOTIFY response, telling the master server to stop sending the NOTIFY to this particular slave. Then this slave can continue to send named-xfer requests to the server in order to update the zone information. The configuration parameter, which tells the server whether it should or shouldn't send NOTIFY messages when the zone database changes, is **notify**, as seen in this example:

```
zone  "my.domain.sela.co.il" {
      type master;
      file "db.mydomain";
      notify yes;
      };
```

Sometimes there are DNS slave servers in our zone which are not registered as NS (name server) records inside our zone's database. NOTIFY messages are sent only to those slaves that are registered inside the zone database as NS records. If we know about an unregistered slave server and want our master server to send a NOTIFY message to this slave whenever the zone database changes, we can do that with the **also-notify** keyword in the zone section of the name daemon configuration file.

```
zone  "my.domain.sela.co.il" {
                    type master;
                    file "db.mydomain";
                    notify yes;
                    also-notify 10.1.1.1;
                    };
```

This configuration section tells the name daemon to notify the registered slave servers whenever my.domain.sela.co.il is changed, and also notify the 10.1.1.1 machine even though it is not registered as an NS record in the my.domain.sela.co.il zone database.

Dynamic Updates

BIND 8 allows authorized hosts to send the zone master server an UPDATE message. An UPDATE message can add, delete or change record information in the zone's database. The machine that sends the update message tries to find the master server for the zone from the zone's NS records. If the message is received by a slave server instead of the master server, the slave server should forward the message to the master server.

Dynamic updates are a good thing for hosts which get their IP addresses dynamically from a RARP (reverse ARP) or DHCP (dynamic host configuration protocol) server. These hosts can send an update message to the DNS server after they get their IP address from the DHCP/RARP server. The new resolver library routine, **ns_update**, is used by programs to send DNS update messages to the server. BIND 8 is shipped with a command-line utility, **nsupdate**, that gives the ability to send update messages to the zone's master server. By default, no one is allowed to dynamically update a zone; the configuration file keyword **allow-update** gives the system administrator the ability to configure the hosts which can send update messages for each zone. This update can help DNS and WINS to live together on the same network, since WINS also uses a dynamic-update method.

```
zone  "my.domain.sela.co.il" {
                    type master;
                    file "db.my.domain";
                    allow-update { 10.1.1.1;  };
                   };
```

This example lets the host 10.1.1.1 send update messages containing information on the my.domain.sela.co.il zone.

Conclusion

BIND 8 has many nice features, some of them needed by many network administrators. Since major security holes were discovered in earlier versions of BIND, upgrading to BIND 8 is almost a must. There are still stable older versions, such as 4.9 and later, but these do not support all the functions mentioned here.

Eddie Harari (eddie@sela.co.il) works for Sela Systems & Education in Israel as a senior manager and is involved in some security projects. He has been hacking computers for 13 years.

Using the Red Hat Package Manager

Kirk Rafferty

Issue #69, January 2000

This article will introduce you to RPM by showing you the most common features, namely how to query, install, upgrade and remove packages.

One of the mundane yet necessary duties a system administrator faces is software management. Applications and patches come and go. After months or years of adding, upgrading and removing software applications, it's often hard to tell just what is on a system: the version of a software package and its dependent applications. Often, old files wind up lying around, because no one's quite sure anymore who owns them. Worse, you may install a new software package, only to find it has overwritten a crucial file from a currently installed package.

The Red Hat Package Manager (RPM) was designed to eliminate these problems. With RPM, software is managed in discrete “packages”--a collection of the files that make up the software, and instructions for adding, removing and upgrading those files. RPM also makes sure you never lose configuration files by backing up existing files before overwriting. RPM also tracks which version of an application is currently installed on your system. While named after Red Hat, the RPM system is also a part of Caldera OpenLinux, SuSE and all distributions based on Red Hat, such as Mandrake.

What Is a Package?

In the generic sense, a package is a container. It includes the files needed to accomplish a certain task, such as the binaries, configuration and documentation files in a software application. It also includes instructions on how and where these files should be installed and how the installation should be accomplished. A package also includes instructions on how to uninstall itself.

RPM packages are often identified by file name, which usually consists of the package name, version, release and the architecture for which it was built. For example, the package penguin-3.26.i386.rpm indicates this is the (fictional) Penguin Utilities package, version 3, release 26. i386 indicates it has been compiled for the Intel architecture. Note that although this is the conventional method of naming RPM packages, the actual package name, version and architecture information are read from the contents of the file by RPM, not the file name. You could rename the file blag.rpm, but it would still install as penguin-3.26.i386.rpm.

What Is RPM?

At the heart of RPM is the RPM database. This tracks where each file in a package is located, its version and much more. The RPM also maintains an MD5 checksum of each file. Checksums are used to determine whether a file has been modified, which comes in handy if you need to verify the integrity of one or more packages. The RPM database makes adding, removing and upgrading packages easy, because RPM knows which files to handle and where to put them. RPM also takes care of conflicts between packages. For example, if package X, which has already been installed, has a configuration file called /etc/someconfig and you attempt to install a new package Y, which wants to install the same file, RPM will manage this conflict by backing up your previous configuration file before the new file is written.

The workhorse of the RPM system is the program **rpm**. rpm is the “driver” responsible for maintaining the RPM databases. Of rpm's 10 modes of operation, I will cover the four most common: query, install, upgrade and remove.

Query Mode

One of the strengths of RPM is that, ideally, it accounts for every system or application file on your system. Using RPM's query mode, you can determine which packages are installed on your system or which files belong to a particular package. This can be a big help if you want to locate a file that belongs to a certain package. Query mode can also be used to identify which files are in an RPM file before you install it. This lets you see the files that are going to be installed on your system before they are actually written.

The **-q** switch is used to query packages. By itself, **-q** will give you the version of a specified package. If you want to see which version of the **tin** newsreader you have on your system, you would issue the following command:

```
# rpm -q tin
tin-1.22-12
```

If you want to see which installed package owns a file, use the **-f** modifier. Here, we want to see which package owns /etc/passwd.

```
# rpm -q -f /etc/passwd
setup-1.9.2-1
```

Likewise, if you want to generate a list of files belonging to a certain package, use the **-l** modifier:

```
# rpm -q -l tin
/usr/bin/rtin
/usr/bin/tin
/usr/doc/tin-1.22
/usr/doc/tin-1.22/CHANGES
/usr/doc/tin-1.22/FTP
/usr/doc/tin-1.22/HACKERS
/usr/doc/tin-1.22/INSTALL
/usr/doc/tin-1.22/INSTALL.NNTP
/usr/doc/tin-1.22/MANIFEST
/usr/doc/tin-1.22/README
/usr/doc/tin-1.22/TODO
/usr/man/man1/tin.1
```

One of the most common modifiers to **-q** is **-a**, query all packages on your system. This system has 350 packages installed, but here's a truncated output:

```
# rpm -q -a
setup-1.9.2-1
filesystem-1.3.2-3
basesystem-4.9-3
ldconfig-1.9.5-8
...
code_crusader-1.1.0-1
lyx-0.11.53-1
xforms-0.86-1
wine-981211-1
```

Listing 1

For even more information about a package, use the **-i** (information) modifier:

```
# rpm -q -i passwd
```

Output is shown in Listing 1. Here's what some of the most important entries mean:

- **Name**: the name of the package
- **Version**: the version of the package
- **Release**: the number of times this package has been released using the same version of the software
- **Install date**: when this package was installed on your system
- **Group**: your RPM database is divided into groups, which describe the functionality of the software. Each time you install a package, it will be grouped accordingly.
- **Size**: the total size in bytes of all the files in the package
- **License**: the license of the original software

Typically, the file name will indicate what's inside the package, but not always. You may receive a package simply named glibc.rpm, which isn't very helpful.

You can use the **-p** modifier to find out which version and release this RPM contains, then perhaps rename it appropriately.

```
# rpm -q -p glibc.rpm
glibc-2.0.7-29
```

Install Mode

The install mode, as its name suggests, is used to install RPM packages onto your system. Installing a package is accomplished with the **-i** option:

```
# rpm -i penguin-3.26.i386.rpm
```

Before installing the package, RPM performs several checks. First, it makes sure the package you are trying to install isn't already installed. RPM won't let you install a package on top of itself. It also checks that you are not installing an older version of the package. Next, RPM does a dependency check. Some packages depend on other packages being installed first. In this example, you have just downloaded the latest RPM version of Penguin utilities and now want to install it.

```
# rpm -i penguin-3.26.i386.rpm
failed dependencies:
iceberg >= 7.1 is needed by penguin-3.26.i386.rpm
```

This error indicates the penguin package failed to install because it requires the iceberg package with a version equal to or greater than 7.1. You'll have to find and install the iceberg package, and any packages iceberg requires.

Finally, RPM checks to see if any configuration files would be overwritten by the installation of this package. RPM tries to make intelligent decisions about what to do with conflicts. If RPM replaces an existing configuration file with one from the new package, a warning will be printed to the screen.

```
# rpm -I penguin-3.26.i386.rpm
warning: /etc/someconfig saved as /etc/someconfig.rpmsave
```

It's up to you to look at both files and determine what modifications, if any, need to be made.

Upgrade Mode

The **-u** switch is used to upgrade existing packages. For example, if Penguin Utilities version 3.25 is already installed, issuing the command

```
# rpm -u penguin-3.26.i386.rpm
```

will replace the old version of the package with the new one. In fact, one of the quirks of RPM's upgrade mode is that the older package doesn't have to exist in the first place: **-u** works identically to **-i** in this case.

Remove Mode

The **rpm -e** command removes a package from your system. Like Install mode, RPM does some housekeeping before it will let you remove a package. First, it does a dependency check to make sure no other packages depend on the package you are removing. If you have modified any of the configuration files, RPM makes a copy of the file, appends .rpmsave onto the end of it, then erases the original. Finally, after removing all files from your system and the RPM database, it removes the package name from the database.

Be very careful about which packages you remove from your system. Like most Linux utilities, RPM assumes omniscience and will silently let you shoot yourself in the foot. Removing the passwd or kernel package would be devastating.

Summary

This has been a basic introduction to the idea of packages and basic package management. You should now have a fairly good idea of how to query, install, upgrade and remove packages from your Linux system.

Kirk Rafferty has been a UNIX System Administrator for twelve years and has been using and maintaining Linux systems for the last five. His hobbies include paintball, gaming and making the best home brew in Colorado. He can be reached for comment at kirk@rafferty.org.

Archive Index Issue Table of Contents

Advanced search

Advanced search

Workings of a Virtual Private Network in Linux—Part 2

David Morgan

Issue #69, January 2000

More about securing our communication with the Internet.

Part 1 of this article described the theory. Now let's pick up the VPN mini-HOWTO, study the script that creates the VPN, run it, and explain what we see. The HOWTO (ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/mini/VPN, text version, or metalab.unc.edu/mdw/HOWTO/mini/VPN.html) is required reading for this article. The script, by Miquel van Smoorenburg and Ian Murdock, is in section 4.10 of the HOWTO.

The script runs on the local VPN server. If you configure reciprocally, you can arrange to let it run from either side with functionally equivalent results. What I call local/remote, Arpad Magosanyi calls master/slave. For working purposes, **ssh** needs a remote user account under which to log in. So, a remote user “slave”, which doesn't correspond to any human user, is first created and configured (permissions, keys).

The heart of the script embodies running **pppd** and **route** as described in Part 1 (section entitled The Network). With simplifications, here it is. Line 33 is the centerpiece.

VPN HOWTO script:

```
line  1:  #! /bin/sh
line 19:  MYPPPIP=192.168.0.1
line 20:  TARGETIP=192.168.0.2
line 21:  TARGETNET=193.6.37.0
```

PPPD on both machines:

```
line 33:  /bin/pty-redir /usr/bin/ssh -o\
  'Batchmode yes' -t -l slave 206.170.217.204\
  /usr/local/bin/sudo /usr/sbin/pppd >/tmp/device
line 34:  TTYNAME='cat /tmp/device'
line 39:  /usr/sbin/pppd $TTYNAME\
${MYPPPIP}:${TARGETIP}
```

ROUTE on both machines:

```
line 45:  route add -net $TARGETNET gw $TARGETIP
line 46:  /usr/bin/ssh -o 'Batchmode\
yes' -l slave 206.170.217.204\
/usr/local/bin/sudo /home/slave/sshroute
```

I numbered the lines, then deleted extraneous ones. For readability, I also substituted shell variables and changed their values to reflect real file and path names on my Red Hat 5.1 servers. Ignore **sudo** in line 33—it is nothing more than a command filter/authenticator, a mechanism you set up to allow some commands to run and disallow others. It is here for additional security. It is set up to permit pppd (HOWTO section 4.9), so its absence from line 33 would not affect operations. When you read line 33, you should see

```
ssh -t -l slave 206.170.217.204 pppd
```

This says, “get logged into the remote machine under its user account 'slave' and run pppd; make the remote machine set up a pseudo-terminal (**-t**) as the destination for output of this pppd process.”

Doing It by Hand

Please note that in section 6 of the HOWTO, “Doing it by hand”, unlike the script, **-t** is absent from the command line. When I first did it by hand, I was unsuccessful at getting the desired “garbage right into [my] face” until I added **-t**. I got it going by using the following machine Internet addresses:

```
local machine's public IP:      206.170.218.30
remote machine's public IP:     206.170.217.204
```

The screen capture from the local machine was:

```
[root@localhost /root]# ssh -t  -l slave  206.170.217.204
/usr/sbin/pppd
~˜}#A!}!}!} }2}!}$}"(}%}&} } öŒ}'}"}(}""q~~˜}#A!}
!}!} }2}!}$}"(}%}&} } öŒ}'}"}(}""q~~˜}#A!}!}!}
}2}!}$}"(}%}&} } öŒ}'}"}(}""q~~˜}#A!}!}!} }2}!}$}
"(}%}&} } öŒ}'}"}(}""q~~˜}#A!}!}!} }2}!}$}"
(}%}&} } öŒ}'}"}(}""q~~˜}#A!}!}!} }2}!}$}"(}%}&}
} öŒ}'}"}(}""q~~˜}#A!}!}!} }2}!}$}"(}%}&} } öŒ}'}
"}(}""q~~˜}#A!}!}!} }2}!}$}"(}%}&} } öŒ}'}"}
(}""q~~˜}#A!}!}!} }2}!}$}"(}%}&} } öŒ}'}"}
(}""q~~˜}#A!}!}!} }2}!}$}"(}%}&} } öŒ}'}"}
(}""q~~.
Connection to 206.170.217.204 closed.
[root@localhost /root]#
```

and the simultaneous remote-machine log entries were:

```
Nov  7 20:17:19 localhost sshd[1403]:
log: Connection from 206.170.218.30 port 1023
Nov  7 20:17:21 localhost sshd[1403]:
log: RSA authentication for slave accepted.
Nov  7 20:17:22 localhost sshd[1405]:
log: executing remote command as user slave
Nov  7 20:17:22 localhost pppd[1405]:
pppd 2.3.3 started by slave, uid 507
Nov  7 20:17:22 localhost kernel:
registered device ppp1
```

```
Nov  7 20:17:22 localhost pppd[1405]:
Using interface ppp1
Nov  7 20:17:22 localhost pppd[1405]:
Connect: ppp1 <--> /dev/ttyp0
Nov  7 20:17:52 localhost pppd[1405]:
LCP: timeout sending Config-Requests
```

Augmenting the command with the author's **pty-redir** in the next step produced:

```
[root@localhost /root]# /bin/pty-redir
/usr/bin/ssh -t  -l slave  206.170.217.204  /usr/sbin/pppd
/dev/ttyp0[root@localhost /root]#
root@localhost /root]#
```

(where did the garbage go?) and simultaneous remote-machine log entries:

```
Nov  7 20:21:43 localhost sshd[1406]:
log: Connection from 206.170.218.30 port 1023
Nov  7 20:21:46 localhost sshd[1406]:
log: RSA authentication for slave accepted.
Nov  7 20:21:46 localhost sshd[1408]:
log: executing remote command as user slave
Nov  7 20:21:46 localhost pppd[1408]:
pppd 2.3.3 started by slave, uid 507
Nov  7 20:21:46 localhost pppd[1408]:
Using interface ppp1
Nov  7 20:21:46 localhost pppd[1408]:
Connect: ppp1 <--> /dev/ttyp0
Nov  7 20:22:16 localhost pppd[1408]:
LCP: timeout sending Config-Requests
```

So far, it's working. Tracing through the log, sshd hears ssh. sshd agrees to run the command requested by ssh (since I preconfigured keys across the machines). The requested pppd command is then run, which prepares to use an interface called ppp1 and associates it with pseudo-terminal /dev/ttyp0. The process stopped there only because pppd was never run from the other end; however, everything here on the remote side went right. We'll see the process consummated below when we run the full-blown script to completion.

What's the difference between these two invocations? On the remote side, nothing; the logs are the same. On the local side, in the second invocation, the entire **prior** command string was fed to pty-redir and executed under its control, resulting in the garbage going away, it seems. Actually, it just went elsewhere; pty-redir “redirects” it. pty-redir is a short C language program by Mr. Magosanyi. It identifies a pseudo-terminal device that is not in use and opens it. Then it forces standard output—normally directed to the console—to the pseudo-terminal instead. Anything the program (ssh) would send to the console gets diverted to the pseudo-terminal device.

Don't confuse this pseudo-terminal with the one created by the **-t** option—they're on different machines. **ssh -t** makes sshd on the remote side create a pseudo-terminal over there, whereas pty-redir operates on the local side. By number, both pseudo-terminals happen to be /dev/ttyp0 this time, but that won't always be so. The local pseudo-terminal is manifested above in the screen output, the remote one in the log.

While it's nice to see the “garbage in our face” for diagnostic purposes, it's better to keep it out of sight for production purposes. That's why pty-redir was written. The pseudo-terminal will prove convenient as the “receiving vessel” for the incoming pppd output stream. We can launch a local pppd into it in order to create the desired connection, instead of having to do it more intrusively at our real terminal.

PPPD—a Different Kind of Daemon

This is the place to contrast connections made by pppd versus ssh and other daemons. The VPN uses both symmetrical and asymmetrical connections to achieve “tunneling”. **ssh** is asymmetrical and uses the TCP/IP service-port connections; **pppd** is symmetrical and uses device-port connections.

The PPP HOWTO section 14, “Setting up the PPP connection manually”, illustrates pppd's requirements. It has to run simultaneously on both computers: the two output streams “meeting in the middle”, so they can negotiate the connection setup. **pppd**'s output stream, visible above, may look like garbage; however, it is pppd's signature and is very meaningful to another copy of pppd. (The PPP HOWTO section 9.5 introduces “ppp garbage”.)

“Meeting in the middle” means getting opposite pppd's to run over terminal or port devices (/dev/tty*X*) that connect, with the output of each coming in as input to the other. The idea is to get the incoming data stream fed into an identifiable local device, then launch a local pppd into that same device. The outgoing stream moves into the same “pipeline” from which the incoming one emerges. Any available terminal or serial port device will do. Since Linux provides /dev/ttypx pseudo-terminals that are serial port emulators, they will work just as well. That's essentially what the pty-redir program is doing; it arranges for opposing pppd data streams to meet in the middle. Once that happens, handshaking and negotiation ensue, ultimately manifesting as a pppx interface viewable with the **ifconfig** command.

This highlights one big difference between pppd and other daemons: symmetry. Last month, we said daemons belong to a matched pair of distinct programs: one the client, one the server or daemon. With pppd there's no distinction, i.e., no client. The program pppd talks to is always another copy of pppd *itself*, a conversation symmetrical with itself rather than asymmetrical with a different program.

The other difference is that other daemons such as sshd talk over TCP/IP connections involving TCP/IP ports. **pppd** talks over a connection and uses ports too, but it builds its own kind of connection with its own protocol, not TCP/IP. It uses port devices, not the numbered service ports (sockets) of TCP/IP.

Since computers can multitask, TCP/IP provides for simultaneous conversations between multiple pairs of processes on any given pair of computers. The conversations are broken into data packets. Each packet has a label (header) bearing the destination computer's IP address and port number. Of the (possibly) many conversations the destination computer might be conducting, the port number identifies the one to which this packet belongs. A conversation is uniquely defined by a set of four items in the header: IP address at each end and TCP port number at each end. The packets look like those in Figure 1 (simplified). See "Introduction to the Internet Protocols" at http://oac3.hsc.uth.tmc.edu/staff/snewton/tcp-tutorial/.

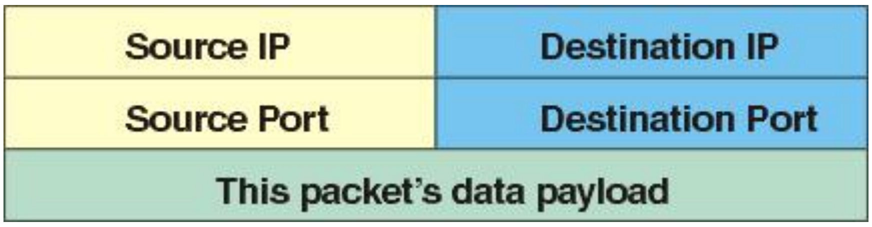


Figure 1

In the conversation logged above, we invoked ssh on the command line and directed it to remote computer 206.170.217.204. The first line in the remote log shows:

```
Nov  7 20:21:43 localhost sshd[1406]: log: Connection from 206.170.218.30 port 1023
```

ssh launched a packet to sshd on the remote computer. **sshd** is set up to "reside" at port 22, as ssh knows. The packet it launched looked like that in Figure 2.

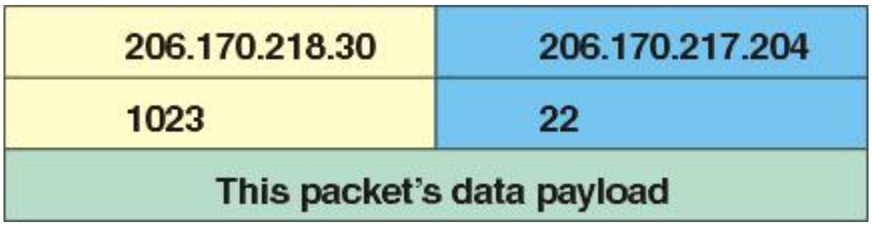


Figure 2

sshd on remote port 22 heard this and answered with packets of its own, swapping the above destination and port values left to right. A conversation was underway. The main order of business was encryption key negotiation and authentication, but before getting into that, it looks like sshd told ssh to change port numbers from 22 to 1406 (according to that log entry, which is "signed" by sshd). Thereafter, a flurry of packets like those shown in Figure 3 flew back and forth.

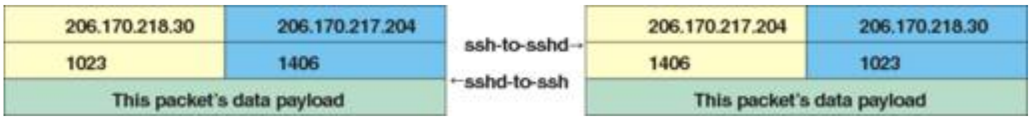


Figure 3

Certain server applications conventionally run on defined port numbers, known publicly so that clients have a place to contact them. These are called "well-known" or "privileged" ports and range from 0 to 1023. (See /etc/services. Port 80 is reserved for **httpd**, for instance, port 21 for **ftpd**.) Once contacted, servers often ask clients to switch port numbers to clear its well-known port for other incoming conversations. Port 22 is the main sshd switchboard. The operator there offloads you to another line as quickly as possible ("Please hold while I transfer you"), so she can handle other incoming calls.

It should now be clear that pppd could never use TCP/IP service ports, since no TCP/IP connection generally exists when pppd runs. The connection pppd has to work with is physical—a phone line or direct serial cable. TCP/IP is a transport-layer protocol. **pppd** (based on HDLC) runs at the lower data-link communications layer, and pppd packets are carriers or envelopes for TCP/IP (or other) packets. Because of these fundamental differences, the overlapping references to "daemons" and "ports" is unfortunate.

Running the Full-Blown Script

Listing 1

Now let's run the script to automate and complete what we did by hand. Here is a local screen capture:

```
[root@localhost /root]# /etc/rc.d/init.d/VPN start
setting up vpn
tty is /dev/ttyp1
[root@localhost /root]#
```

The simultaneous remote log is shown in Listing 1, which shows that sshd agrees to run the command requested by ssh; sshd puts the conversation on port 1022 this time. The requested command (see VPN HOWTO script line 33) is sudo, not pppd. Within the remote machine, sudo in turn runs pppd, which prepares to use an interface called ppp2 and associates it with pseudo-terminal /dev/ttyp1. Just about this time, somebody runs pppd at the local machine (see line 39). Two pppd's meet in the middle, and the local pppd (through ssh, then sshd) tells the remote pppd what IP address numbers it wants to use for each end of this connection. Remote pppd agrees, and records those addresses in the log. At that point, the secure link is in place, bilaterally. The script then proceeds to run route. It informs each VPN server by address which additional machines are available to be reached through the new connection beyond the other server (i.e., the address of the network the opposite server is on). In reaction to script line 46, the log shows sshd reinvoked under new process ID number 1439, this time to have sudo run **sshroute**, a script (see end of HOWTO section 4.10) that contains the appropriate route command.

The dust has settled. The secure link is now in place. How can we see and use it? It takes the form of a new PPP interface. See it by running ifconfig. Use it by referring certain addresses to it in the routing table (already done), then referring applications to those addresses.

Listing 2

ifconfig now shows two ppp interfaces on each machine, old and new. A screen capture on the local machine looks like that in Listing 2. A screen capture on the remote machine is shown in Listing 3. Local-machine's ppp1 and remote-machine's ppp2 are opposite ends of the same connection, or interfaces to one another. This connection is the one just constructed. Note interface number assignment is machine-specific, and the numbers need not be identical. Local and remote ppp0, the machines' respective ISP connections, aren't directly related.

Listing 3

To refer to the other, each VPN server now has a choice of IP addresses. Local-machine can still contact the remote machine at its public Internet address 206.170.217.204, but has the additional option of calling it 192.168.0.2. If the remote machine is running a web server, for example, a browser on the local machine will pull up exactly the same page by addressing itself to http://206.170.217.204 or http://192.168.0.2. The remote machine can be pinged using either address. To all appearances, our two ppp interfaces seem independent and equivalent. Logically, they are, and you can use them as if they were; physically, they are not.

Envelopes inside Envelopes—Tunneling

What's the difference? Primarily, that packets to 192.168.0.2 don't travel on the same footing as those to 206.170.217.204. Rather, the former are carried as data "freight" inside the latter. That's why I depicted the PPP connection as tributary to the public ISP connection in the network diagram in Part 1. The arrangement is called tunneling, because once freight packets arrive at their destination, they emerge from their enclosing packets as from a tunnel. They're released onto the destination network as functional packets, not just passive data.

You may have noticed that the 192.168.0.0 IP addresses chosen for the secure link belong to the "reserved" range prohibited for use on the Internet (PPP

HOWTO section 2), yet we are using these on the Internet. Tunneling is what allows us to get away with it. While on the Internet, packets bearing these addresses travel only as data. The Internet need not route them according to those addresses—they piggyback on packets addressed legitimately.

Tunneling enters the discussion of all VPN protocols. Some reflect it in their name, such as Microsoft's PPTP or Point-to-Point Tunneling Protocol. Tunneling is a characteristic of VPNs; if the data being tunneled also gets encrypted, you have a VPN.

That's the other difference here—encryption. Ping packets and web pages going to and from 192.168.0.2 get encrypted and decrypted. Those going to 206.170.217.204, even though it is the same place, do not. But you would not know about the encryption or tunneling, because they're transparent. Since both are present, this is a VPN.

Routing—Completing the Picture

I found the HOWTO's treatment of routing less than comprehensive, perhaps beyond its intended scope. It goes as far as informing each VPN server about the population of workstations on the opposite network, but it doesn't teach those workstations the reverse, and it doesn't provide for end-to-end awareness by workstations on one network about the workstations on the other.

The provided routing from the main script

```
route add -net 193.6.37.0 gw 192.168.0.2
```

puts an entry in the local VPN server's routing table that says “If you handle any packets whose address starts with 193.6.37, the clearinghouse for them is 192.168.0.2 (the remote VPN server), send them to him. He'll know what to do with them”, since that's the address of the network to which he belongs.

When I tried to ping a 193.6.37 from the local VPN server, it failed. While I had a route to those machines, the problem was their ignorance of a reverse route back to me. They got my ping packets all right, but they couldn't answer back with a reply packet because their routing table had never heard of me—dead-letter office. So, I never saw the evidence of my half-success.

There are two solutions. One, an entry in each of their routing tables that points the way, unfortunately requires separate adjustments on these multiple machines. The other requires a single change at the remote VPN server, using the **proxyarp** option of pppd.

ARP (address resolution protocol) is for use in an Ethernet environment. It's insufficient for workstations that talk over Ethernet to know one another's IP address. The only way they can trade TCP/IP packets is inside of Ethernet packets. Those go nowhere unless the destination Ethernet (not IP) address is known.

Machines maintain a directory for looking up a machine's Ethernet address, given its IP address. (The Linux **arp** command prints it.) If a desired target IP isn't in your list, you automatically emit an ARP broadcast to the network. If any other machine can come up with the matching Ethernet address, it will send it to you.

The **pppd proxyarp** command makes a white-lie entry in the remote VPN server's directory that equates the local VPN server's TCP/IP address with the remote VPN server's Ethernet address. This fixed my ping problem. If the remote workstations can get replies back to the remote VPN server, they can certainly get them the rest of the way back to me at the local VPN server. (See section 9 of “Introduction to the Internet Protocols” for more about ARP.)

The other routing problem is that local and remote workstations don't have routes to each other. The fix is an entry for each local workstation specifying that either the local or remote VPN server offers the route to the remote network. For each remote workstation, either the remote or the local server offers the route.

For local workstations, the command is

```
   route add -net 193.6.37.0 gw 192.168.0.1
   or
   route add -net 193.6.37.0 gw 192.168.0.2
```

For remote workstations,

```
   route add -net 193.6.35.0 gw 192.168.0.2
   or
   route add -net 193.6.35.0 gw 192.168.0.1
```

For local workstations,

```
   route add -net 193.6.37.0 gw 192.168.0.1
   route add -net 193.6.35.0 gw 192.168.0.2
   or
   route add -net 193.6.37.0 gw 192.168.0.2
   route add -net 193.6.35.0 gw 192.168.0.1
```

(The second one works only if you invoked proxyarp when you ran pppd.) For Microsoft machines on either net, give that net's VPN server as the Gateway under TCP/IP properties.

Now you are a VPN expert, so you also know the VPN is worthless unless the computer where it runs is secured in other ways. For instance, rsh and rlogin had better be disabled. ssh was designed to replace them. If they're still hanging around, you have one door double-bolted and another wide-open. Other services like telnet and ftp should be turned off, too. You can do that in the inetd.conf file. And, firewall rules should be deliberately applied. Though beyond the scope of this article, these important considerations require mention.

Resources

David Morgan is an independent consultant in Los Angeles and a computer science instructor at Santa Monica College. He got serious about Linux in 1998. While waiting for it to enter his life, he got degrees in physics and business, served in the U.S. Peace Corps as a teacher, held technical and product management positions at Rexon Business Machines, Nantucket Corporation, Computer Associates and Symantec Corporation. He bicycles, backpacks and cooks. Send him your recipes and VPN experiences. He can be reached at dmorgan1@pacbell.net and currently maintains web sites at http://www.geocities.com/Yosemite/Gorge/3645/ and http://skydesign.hypermart.net/.

Archive Index Issue Table of Contents

Advanced search

1999 Readers' Choice Awards

Jason Kroll

Issue #69, January 2000

You voted, we counted—here are the results.

Welcome to the 1999 *Linux Journal* Readers' Choice Awards, brought to you by the same people who brought you the Editors' Choice Awards last month (remarkable). The Linux scene has exploded since this time last year, with more software than we could keep up with. Hence, omissions were so frequent, the poll was almost a comedy of errors, but fortunately no spreadsheets showed up listed in the games section. Over 8,000 people voted, more than twice last year's turnout. It was fun, and the results were even different from last year! Worth noting is that Emacs received numerous write-ins in nearly all of the categories. Apparently, it does more than we thought. Included are some voter write-ins and comments (in italics) for your delectation.

Favorite Distribution: Debian GNU/Linux

> "Better than all the rest."

Debian GNU/Linux with 27.3% scores a 21-vote upset over last year's winner SuSE which got 27.0%, while Red Hat falls to number three by about 200 votes, with 23.6%. This victory for Debian GNU/Linux is sure to warm the hearts of free-source enthusiasts everywhere. In other news, Slackware is still alive at 5.7%, Mandrake and Caldera received 7.1% and 6.1% respectively, and the others have their followers, though not many (no other distribution scored 1%).

Favorite Window Manager: KDE

> “I prefer console mode.”

> “GNOME with Enlightenment in the future, KDE now.”

The K Desktop Environment (technically speaking, the K Window Manager) wins again with 36.6% of the vote, far ahead of its nearest competitors WindowMaker and Enlightenment, which gathered 17.8% and 16.1%, respectively. Other niche managers have devoted followings, but it appears as if the days of FVWM have wound down and a new era has begun—that of the desktop environment. Many window managers added and improved theme support, making it easier to change managers and aesthetics. The windowing diversity on Linux makes it easy to forget that other operating systems don't offer the freedom to choose in this area.

Favorite Audio Application: mpg123

> “mpg123 lets me play Pink Floyd; cthugha lets me see it…”

mpg123 garners 35% of the vote to prove the overwhelming popularity of the mpeg format which is so efficient, many lobbyists tried to have it outlawed. Fortunately it's still legal, and we're having a lot of fun with it. RealAudio is number two with 20%, while paranoia (to rip CDs into mpegs) with 10% takes third place. XMMS/X11Amp received several write-in votes.

Favorite Programming Language: C/C++

> “Something wants to make me vote for Logo… but I'll spare you. :)”

The old UNIX standard—the closest thing we have to a cross-platform assembler—wins nearly half the vote at 49.4%. We'll have to split C and C++ next year; we received countless “I hate C++” comments, a sentiment shared by nearly everyone who voted. Perl had an excellent showing, with 20.6% of the vote, compared to the up-and-coming Python which scored 4%. At 9.5%, Java appears to have become rather popular, and a concerted effort from PHP enthusiasts managed to score it 4.6%. Emacs (meaning ELisp, probably) received a large number of write-ins. Doesn't anyone use assembly code anymore?

Favorite Text Editor or Word Processor: vi & clones

> “vim vim vim vim vim vim vim vim vim vim vim vim vim vim.”

> “Emacs is the true path to Nirvana!”

Yet again, vi scores a large victory with 31.1% of the votes, over StarOffice and Emacs which had 15.2% and 14.9%. XEmacs, last year's runner-up, appears to be less popular this year, taking 10.4%. Popular with old UNIX hacks, vi is gaining a large following in the GNU generation. “Ack! Are you still using vi?” I once asked. “No, man. I *learned* vi.” Vintage software—the next cool thing.

Favorite Linux Web Page: Slashdot.org

Slashdot
News for Nerds. Stuff that matters.

> “Long live Cmdr Taco!”

> *“Next to Linux Journal? Slashdot.”*

> “Man are the ballots going to be stuffed on this one :-)”

Was there ever any doubt? It's not an exclusively Linux page, but Slashdot takes 41.3% of the vote. Freshmeat.net is second with 16.6%, while none of the others did very well at all. Discouraging news for Linux gamers: Happy Penguin (The Linux Game Tome) made only 0.8%.

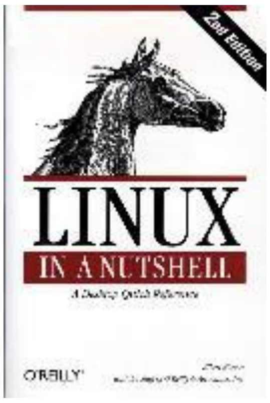
Favorite Shell: Bash

> “conch”

> “You know, bash could also stand for best of all shells. :^)”

The Bourne Again Shell gathered a monstrous 77.8% of the vote, by far on of the bigger victories in this year's poll. tcsh managed 10.5%, and ksh got 5.2%. There would probably be a different distribution between shells one likes to use and shells one likes to write scripts for, but it's clear that bash is where it's at. Shells have cool names like ash, bash, zsh (to keep in mind when naming your children).

Most Indispensable Linux Book: *Linux in a Nutshell*



> *"What a stupid name,* Linux Secrets*! What secrets can software whose source code is available on the Internet have?"*

> "/usr/doc is your friend. ;)"

Scoring a seven-vote victory, *Linux in a Nutshell* by Ellen Siever, et al. won over Matt Welsh's classic *Running Linux*, 19.0% to 18.8%. Last year's winner, *Network Administration Guide*, was third with 11.8% of the vote. There are many books for beginners, but these were selected as the best. Truly there is no shortage of Linux books, though I wonder how the Linux Documentation Project is coming along. It received many write-ins but not quite enough to make a showing, and technically speaking it's not a book. Still, the enthusiastic support for LDP shows where Linux users' loyalties lie.

Favorite Web Browser: Netscape/Mozilla

> *"Netscape, but it's a* cope *sort of thing."*

Five people "accidentally" wrote in Internet Explorer—they officially lose. Netscape, deemed the "lesser of the evils" by voters, wins with 86.7%. Netscape may sometimes leak memory like an NT box, but let us never forget what it did for the Open Source movement (that guy Raymond, Linus Pauling, et al.). Lynx came in second with 8.3%, and the K File Manager was third with 4.4% (good job, guys; this one is fast for a graphics browser). Mnemonic appears to be completely dead, and Arena is not so successful right now either. Hopefully, Mozilla will stand on its own by next year. Ah, who uses the Web anyway?

Favorite *LJ* Column: Kernel Korner

> “Kernel info is fun for the whole family.”

> “The one with Marjorie.”

> “upFRONT is my high-level Slashdot: what Slashdot won't tell, I can read from upFRONT. It has things to check out and news to talk about.”

“Kernel Korner” is the perennial favorite this year with 26.9% of the vote, followed by “Best of Technical Support” which had 21.2%. This probably says something about *LJ* readers. David Bandel's “Focus on Software”, each month a wild assortment of rarities from the software world, scored number three with 16.3%. Reuven Lerner's consistently popular “At the Forge” was a close fourth at 13.1%, and received many glowing comments. “Linux Means Business” had 11.1%, and the new “upFRONT” managed 4.1%.

Favorite Backup Utility: BRU

> “Backup? Who needs backup? Maybe it sounds crazy, that's okay, it is.”

BRU, with 35.5% of the vote, comes in ahead of its closest rivals Amanda which had 15.3% and Lonetar which had 10.8%. Worth noting were the number of write-ins advocating **tar**, a simple solution you can insert easily into **cron**. While BRU and Lonetar are proprietary, Amanda is freely available, in case you actually want to back something up—but Linux is so stable!

Favorite Communications Board: Cyclades

> "Slashdot!"

Cyclades, with 37.7%, seems to be more popular than its competitors Digi International with 15.8% and Boca with 10.0%. Cyclades has a cool name, but do you pronounce it like the island chain?

Favorite Database: MySQL

> "Who needs databases?"

MySQL, the free database from Finland and Sweden, scores big points with 42.4%, compared to PostgreSQL with 19.5% and Oracle with 15.2%. The Berkeley Database, which is embedded in practically everything, scored only 1.4%, which is odd. Still, it stands to reason that MySQL would be a favorite choice of the Linux community, considering its origins and relatively free license. Someone wrote in M$ SQL, as a joke... I think.

Favorite Graphics Tool: GIMP

> "I hope to earn my black-belt in script-fu some day."

In an amazing upset victory... oh, wait. The GIMP wins again this year, with 76.8% of the vote. **xv** and CorelDraw each got 8.1% of the votes (though xv was ahead by 4 votes). The GIMP blew away everything years ago; now it is polished and beautiful. You can do anything with it, even if you're not a graphics expert.

Then, you can use SatanPaint (which scored 0.1%) to convert your drawings into C include files. It is interesting that Linux has such a fantastic graphics package, while audio seems to have been largely neglected. Are Linux types visually inclined? Apparently so, and projects like the GIMP, GNOME, Enlightenment, KDE and the themes trend are all evidence, as well as the new wave of screen savers and demos.

Favorite Linux Demo: State of Mind by Skal of Bomb

> “Linux has a demo scene? I have to check this out!”

Even though we can't take over every processor cycle and sync our routines to raster positions, Linux finally has demos, lots of them. It is no surprise that State of Mind would win for this year, but what *is* a surprise is the third-place finisher, BB by AA (Big Brother, the aalib ASCII demo), which was not included but had many write-ins. Second place went to XDemo8 by Lame Over. Technically speaking, most write-in votes went for “What is a demo?” or some variant thereof. Someone wrote in linus.au; hmmm. Truly we have a cool scene, small though it may be.

Favorite Mailer: Pine

> “Pine is evil.”

Pine has displaced last year's winner Elm with a 41.4% vote, compared to the second-place finisher Netscape Communicator with 32.2% and the up-and-coming Mutt which had 14.7%. What happened to Elm? It fell to number four, with 6.8% of the vote. Several variants of Emacs mailers received numerous write-ins. A growing Pine/Pico recovery and support community will probably lead Mutt to greater popularity next year; in any event, it has colors and that's cool.

Favorite Development Tool: gcc

> “Emacs+gcc+gdb.”

What would development be without a compiler? The GNU C Compiler wins with 65.8% of the vote, while the Experimental GNU Compiler System took 20.9%. Now that the two are reconciled and together again, that would be an 86.7% collective. The recently delivered Metrowerks CodeWarrior scored 5.9%, a number which might be higher if it only ran on more distributions. The free Code Crusader was fourth with 2.7%, GNUPro Toolkit was fifth with 2.5%. The Linux standard, Emacs/gcc/make or some variant thereof, was the most common comment. Where would we be without GNU, or would we be at all without GNU?

Favorite Linux Game: Quake2

> “Old hackers never die, young ones do.”

Quake2, the sequel to Quake (did you know?), is the favorite Linux game with 23.1% of the vote. Civilization: Call to Power, the excellent port from Loki (a savior of sorts for Linux gamers) is second with 13.9%, while XBill is dangerously popular at 8.4% (beating out the original Quake by 0.1%). Gruesome violence is inexplicably popular, but in other news, the hacker classic NetHack managed 5.6%, ahead of FreeCiv which had 4.3%, a good showing for our free games considering the number of entrants in this category.

Favorite Platform: Intel x86

> *“Abaci even have a n33t plural form! The abacus is the* only *Y2K compliant computer known to humanity!”*

Compliant though it may be, the abacus lost to the Intel x86 which scored 82.4%, ahead of Alpha with 6.8% and PPC with 4.8%. There were hundreds of write-ins for AMD, and almost no one is under any delusions that Intel makes a better chip. However, the x86 is, technically speaking, for the most part Intel's fault; AMD is a clone, better than the original (as so many people wrote) but still a clone. The new IA-64 looks like it may actually be *well-designed*, and who knows what Transmeta (known more for hiring Linus than anything else) will come up with? Nowhere is the QWERTY syndrome stronger than in computers, but maybe we can at least get a better chip set sometime soon; otherwise, I'll see you at Abacus World Expo.

Penguin Playoffs Awards

Peter H. Salus

Marjorie Richardson

Jason Kroll

Issue #69, January 2000

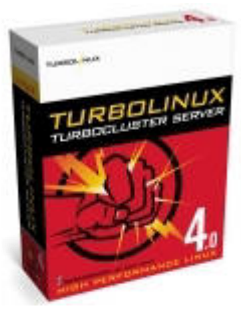
And the winners are....

This fall, *Linux Journal* and the Linux Business Expo teamed up to present the first annual “Penguin Playoffs”. These awards are designed to showcase outstanding enterprise products from those exhibited at the Linux Business Expo. The Expo was held in Las Vegas as part of COMDEX/Fall 1999 by ZD Events. The awards were given out at a ceremony on November 16, 1999, in conjunction with the presentation of *LJ*'s Editors' Choice awards announced in last month's issue. Linus Torvalds was there personally to hand out the awards to the winners.

This year's judges were Jason Kroll, *Linux Journal*'s Technical Editor; Marjorie Richardson, Editor in Chief of *LJ*; and Peter Salus, Editorial Director of SSC. There was a large number of submissions in the overall, office and web categories, but none in manufacturing. We agreed to change this category to “hardware”, so that we could reward a truly outstanding product.

Linux has been loved from the beginning by the hacker community, and for the last year has been making steady inroads into the business community. One reason for this is more companies have come up with software applications designed to solve the problems of the enterprise. To reach Linus' goal of “world domination”, Linux must be accepted in the corporate world, and our winners are helping to make that happen.

Best Web Solution: TurboCluster Server from TurboLinux



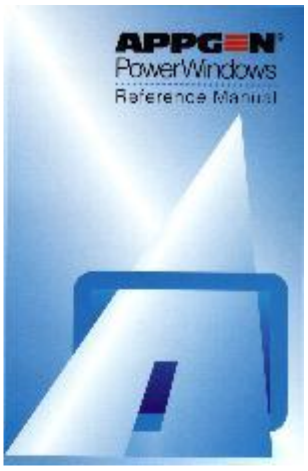
The TurboCluster Server (http://www.turbolinux.com/) aims at providing a low-cost, high-availability server for electronic business. It possesses automatic “failover”--should a machine in the cluster fail, the others pick up that load—dynamic load-balancing and redirection, support for Linux, Solaris and NT, notification support and recovery and maintenance support. In fact, if an application were to fail on a single machine, TurboCluster cleverly routes around it. Clusters of Linux machines provide a lot of power to the business community, and this excellent software is designed to support the company that chooses to exploit that power.

Targeted at corporate web administrators and ISPs, the TurboCluster Server was clearly this year's Best Web Application solution.

Best Hardware Solution: The Happy Hacking Keyboard

The Happy Hacking Keyboard from PFU America (http://www.pfuca.com/) was the clear winner in this category. Based on the Sun Type-3 keyboard with only 60 keys, it is smaller than a laptop keyboard. The **ESC** key is positioned next to **1** for devout vi users, and the **CTRL** key is next to the **A**, perfect for Emacs enthusiasts. The keys are full size, but function keys and the number pad of extended keyboards have been eliminated in order to maintain the small size. You never have to move your hands from the keys or stretch your fingers into uncomfortable positions; this means faster typing and no muscle strain. This is an ideal keyboard for UNIX and Linux users in the office. And it lists for only $69 US—a useful, pleasant bargain!

Best Office Solution: Appgen Business and Accounting Applications



We like Appgen's PowerWindows Business and Accounting Applications (http://www.appgen.com/) very much.

All businesses need accounting software—it's one of those facts of life. Appgen includes all the necessary modules to keep the books balanced and the office records in tip-top shape. This package ships with eleven applications, all with source code:

- accounts receivable
- accounts payable
- general ledger
- inventory control
- sales order processing
- purchase order processing
- payroll processing
- job cost tracking
- bill of materials
- billing
- bank reconciliation

Appgen provides a user-friendly system which features customization of all financial statements, the general ledger, inventory accounts and more. Customization features make this product work well for any size business, from small to large. Information from each module flows to the others effortlessly. A reminder system and security features are included.

This product is on the same level as accounting packages such as Platinum, Lawson and Eagle Systems. Don't let anyone tell you there aren't solid business software packages running natively on Linux. Appgen's solution is first-rate—one that will be welcome in any office.

Best Overall Solution: Global Media Corporation

We think that as time goes on, more and more transactions will be conducted electronically. For this basic reason, we have awarded Global Media Corp. (http://www.globalmediacorp.com/) our award for Best Overall Solution.

Global Media focuses on e-commerce tools for entertainment companies. They have built a network for the transmission of live audio and video over the Internet; their ready-made infrastructure is based on Linux.

The quality of Web content depends upon the quality of the stream for both audio and video, and we feel Global Media's private frame-relay network will result in a far better experience than what we have received in the past. The broadcast-encoding boxes all run a customized distribution of Linux. Global Media has teamed with RealNetworks where content is concerned.

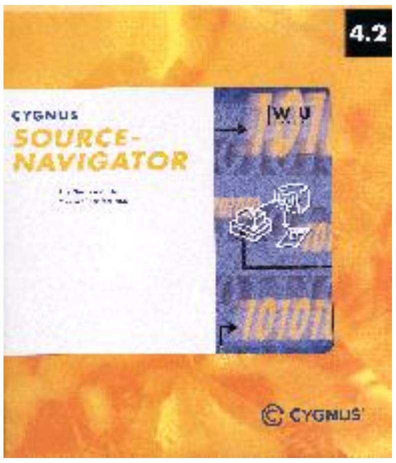
As the producers of a product with the potential of impinging on all our lives, Global Media has earned the Best Overall award.

Source-Navigator Version 4.2

Daniel Lazenby

Issue #69, January 2000

Source-Navigator (SN) is an extendable source code editor, organizer and analysis tool.

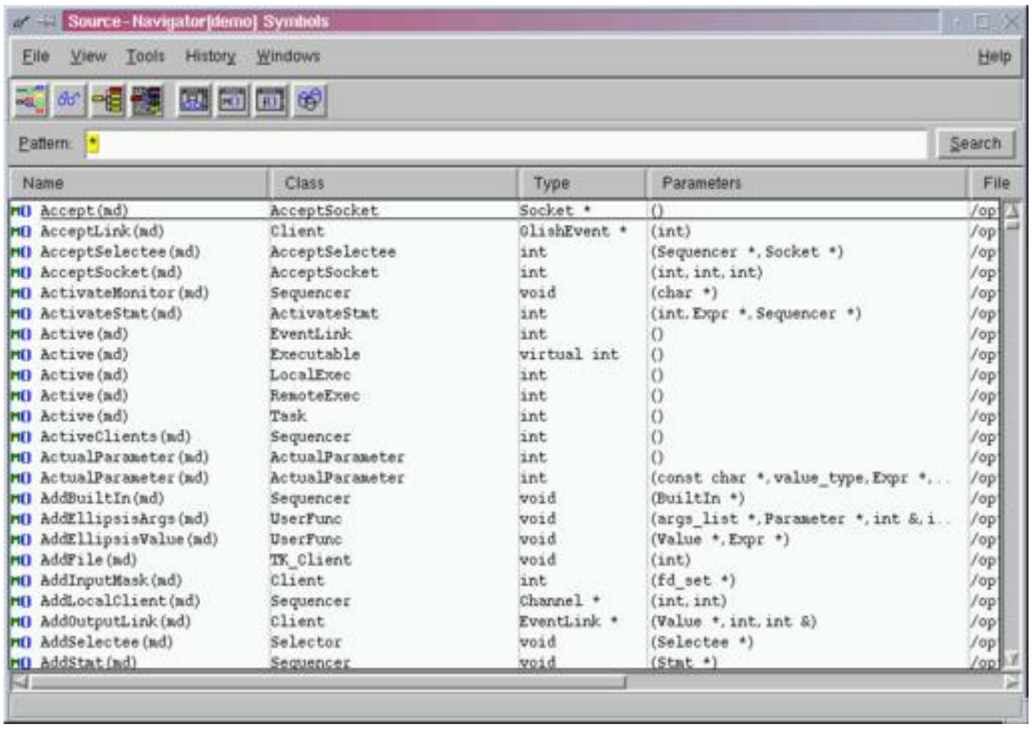


Source-Navigator Version 4.2

- Manufacturer: Cygnus Solutions
- E-Mail: info@cygnus.com
- URL: http://www.cygnus.com/
- Price: Developer Edition $149.00 US, Enterprise Edition $499.00 US
- Reviewer: Daniel Lazenby

Whether you are an individual developer creating that dearly needed utility or responsible for a team developing the next Linux killer application, Cygnus' Source-Navigator can add value to your software project. Source-Navigator (SN) is an extendable source code editor, organizer and analysis tool. SN can be used for creating, maintaining, reengineering or reusing code, understanding complex application code relationships or assisting with source-code configuration management.

There are at least two different ways in which this software tool can be used. Let's say you've just received maintenance responsibility for a legacy application with thousands of lines of code. The application's documentation is minimal, and source-code documentation is slim. How can one perform code archaeology to learn about the application's structure, functions, include relationships and other code interdependencies? Converting it into an SN project could go far toward building a foundation for understanding the code. Perhaps you've been assigned the task of managing development of the next Linux killer application. This assignment includes ensuring code reuse, tracking modifications, implementing source-code version control and keeping development teams focused on their portion of the development effort. SN can be used to achieve these objectives. In the next few paragraphs, I will highlight the various tools, browsers and editors that can be used to achieve these objectives.



Pattern: *

Name	Class	Type	Parameters	File
Accept(md)	AcceptSocket	Socket *	()	/op
AcceptLink(md)	Client	GlishEvent *	(int)	/op
AcceptSelectee(md)	AcceptSelectee	int	(Sequencer *, Socket *)	/op
AcceptSocket(md)	AcceptSocket	int	(int, int, int)	/op
ActivateMonitor(md)	Sequencer	void	(char *)	/op
ActivateStmt(md)	ActivateStmt	int	(int, Expr *, Sequencer *)	/op
Active(md)	EventLink	int	()	/op
Active(md)	Executable	virtual int	()	/op
Active(md)	LocalExec	int	()	/op
Active(md)	RemoteExec	int	()	/op
Active(md)	Task	int	()	/op
ActiveClients(md)	Sequencer	int	()	/op
ActualParameter(md)	ActualParameter	int	()	/op
ActualParameter(md)	ActualParameter	int	(const char *, value_type, Expr *, ..	/op
AddBuiltIn(md)	Sequencer	void	(BuiltIn *)	/op
AddEllipsisArgs(md)	UserFunc	void	(args_list *, Parameter *, int &, i..	/op
AddEllipsisValue(md)	UserFunc	void	(Value *, Expr *)	/op
AddFile(md)	TK_Client	void	(int)	/op
AddInputMask(md)	Client	int	(fd_set *)	/op
AddLocalClient(md)	Sequencer	Channel *	(int, int)	/op
AddOutputLink(md)	Client	EventLink *	(Value *, int, int &)	/op
AddSelectee(md)	Selector	void	(Selectee *)	/op
AddStmt(md)	Sequencer	void	(Stmt *)	/op

Figure 1. Symbol Browser

Source-Navigator's main work environments are divided into the Symbol Browser and the Editor Window. I think of the Symbol Browser as an overview tool (see Figure 1). Every project opens with the Symbol Browser. This provides a listing of the project's directories, files, methods, functions and classes. It is possible to display lists of virtually any symbol or source files with this browser. A symbol list can be set to display a single symbol or any combination of symbols. Generally, information on symbols is presented as a two- (or more) column list.

Notice the four right-most toolbar buttons. These are labeled from left to right: Class, Methods, Functions and Files. Choosing the File button lists all of the application's files, their file type and location. Selecting the Methods button will list all methods, their class, type and associated file location. The Class and Function buttons provide similar lists of class and function information. The four left-most buttons provide hyperlink access to the Hierarchy, Class, Cross-Reference and Include browsers. With a click of a button, these browsers provide more details about the selected symbol.

I created a software project with some C++ code I had never seen. After spending a few minutes with the Symbol Browser, I had a feel for what classes, methods, functions and files belonged to the application. After spending some more time with this tool, it became clear which classes, methods and functions were contained in which file(s). Having learned these relationships, I found myself wanting to print out a couple of the Symbol Browser's lists. Unfortunately, this release did not offer a print capability from the Symbol Browser.

Burrowing further into application details requires tools for dissecting and mapping specific code, file and symbol relationships. These tools are contained in the Editor Window. This window is the home of the Hierarchy, Class, Cross-Reference and Include browsers. Editor Window browsers are normally displayed as a single window. If needed, a browser window can be split to present two browsers within the same window. In addition to the four browsers, the Editor Window contains two text-pattern search tools and an editor. The Editor Window also includes the ability to look at something previously selected, relationships, files or symbols. These “views” are accessible through a project history list. The project history list is maintained on a per-tool basis. The Editor Window can also be used to set up projects, add and delete source files or directories, and organize and segment views of source files. The Editor window and its major parts are shown in Figure 2.

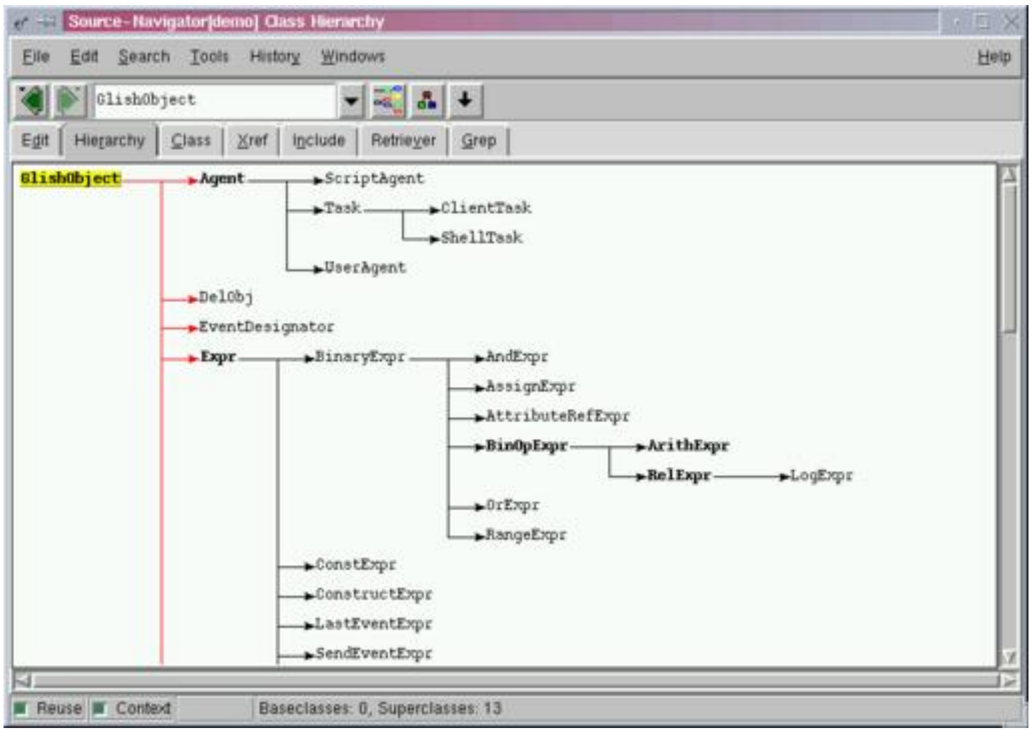


Figure 2. Editor Window

Understanding an application involves breaking the software into manageable chunks. These chunks can take the form of files, objects, code symbols, coding constructs or algorithms. Usually there is some kind of predetermined interrelationship between these application components. The Cross-Reference browser provides a means of graphically examining these relationships in a hierarchical tree format. For example, selecting a function and then the Cross-Reference browser (X-Ref) will display a tree of function calls. This tree can provide a point of view that shows calling functions higher in the tree and functions being called lower in the tree. Selecting the X-Ref browser and a method produces a similar tree, illustrating the variables, methods, classes and functions related to the method. Similar hierarchical trees are displayed when the X-Ref is used with classes. I believe one of SN's better attributes is its ability to diagram relationships.

Some languages allow other files to be included as part of the current file. Examining a file's **#include** statements provides a backward view of which files are being included in the selected file. Seldom does a file provide a list of files in which it is being included. The Include browser provides both a forward and a backward graphical picture of the various include file relationships. This browser shows which files are included in the selected file. It also shows which files include the selected file. Include relationships can be illustrated by this browser range from a simple one-to-one relationship to a complex many-to-many relationship.

Inheritance relationships between classes and objects are displayed in the Hierarchy browser. Like with the X-Ref browser, this information is displayed as a hierarchical tree. Three hierarchical views can be selected in this browser. One view displays the entire class hierarchy. There is a view that shows only the selected class and its related sub- and super-classes. Another view displays only the selected class and the related super-classes.

The Class browser can be used with either object-oriented or conventional languages (C, COBOL, etc.). This browser uses two window panes to display information. The Inheritance Tree pane displays relationships between the class selected for browsing and any related base or super-classes. A Member List pane lists members of the classes shown in the Inheritance Tree pane. Selecting or deselecting class names in the Tree pane will modify the members shown in the member's list. A member filter is included in this browser. The member list filter provides a way of displaying members meeting specific criteria. For example, the filter can be used to list only methods which have been overridden from a base class.

The included Editor provides typical editing capabilities, plus a couple of other services. The symbol-accelerator drop-down list contains either all symbols in the entire application or only those in the open files. Selecting a symbol from the list advances the editor to the appropriate file and highlights the symbol. SN's hyperlinking feature causes all other browsers to automatically prepare to display their information for the same symbol. If you wish, SN can be told to use your favorite editor instead of the default editor.

SN contains two more features. One is its ability to interface with debugging and programming tools. The other feature is version control support. The Tools Menu provides the Make and Debugger options. Existing projects' Makefiles can be run from within SN. The SN Make dialog box contains a **make** command field, a working directory field and an output window. Output from the make run is displayed in this window. Any compiler error messages displayed in this window can be used to open the SN editor. After the editor is opened, the cursor is placed on the line containing the error. Command-line compilers can be used with SN. The “Programmer's Reference Guide” provides information on how to integrate command-line compilers.

The GNU Debugger is used by SN to debug a program. Selecting the debugger menu option displays the “Program to Debug” dialog box. The debugger can be started once the program name and working directory have been entered. If you have the correct version of GDB debugger, SN will be able to open and display the source file when a problem occurs.

Managing software projects often involves revision and version control. With version control, one has the option of managing versions, version history, labels and related documents. SN's Edit Menu provides a graphical interface to four external version control systems. Each of these four products has already been integrated with SN. You must have one of them installed on your platform before SN can use it. On the non-commercial side, you can select either the Revision Control System (RCS), Concurrent Versions System (CVS) or Source Code Control System (SCCS). The fourth choice is a commercial version control product called ClearCase. All systems provide the basic checkout with lock, checkout without lock, check-in, check-in with lock, change description, file comparison and discard changes functionality. I'm not familiar with ClearCase. Since it is a commercial product, it may possess additional functionality.

Version control in a distributed environment implies all developers are ultimately checking code in and out of the same code repository. Several techniques for sharing file systems are available. In my current setup, I was unable to review how the product behaves in a shared version control environment.

Printing definitely requires a PostScript printer. Depending on the printer/ plotter capabilities, SN can print on several paper sizes. Supported paper sizes range from 8.5 by 11 inches to 42 by 59.5 inches. The print dialog box provides several additional options including image scaling, pagination, paper orientation and print color. Your printer's capabilities will determine which options you can use.

Installation

SN Developer and Enterprise Editions are supported on at least Sun Solaris 2.5.1, SuSE 6.0/6.1 or Red Hat 6.0. It is also supported on Wintel platforms running NT 4.0 and Windows 95/98. The Enterprise Edition is also supported on HP-UX 10.20. Literature indicated SN should run on the Caldera Linux distribution. The platform used in this review used the Caldera v1.3 and v2.2 Linux distributions.

By default, the product is installed under the installer's ID in the installer's home directory. An option to install the Source-Navigator product according to your product install conventions is provided.

Source-Navigator is a breeze to install. First, start the install script from within the X environment. Next, enter the desired target installation directory and identify how any problem reports will be sent to Cygnus. The option of installing a demonstration project is offered once the above choices have been made. Any or none of the six demonstration projects can be selected for installation. The tutorial is based on the C++ demonstration project. A dialog keeps you posted on the install progress.

The product installed without problems the first time. On the Caldera v2.2 install, some of the subdirectories and files had UID and GID numbers that were nonexistent on my system. To correct this, I assigned valid user and group IDs to the subdirectories and files.

SN uses a network floating-license scheme to control product access. I didn't find installing the license manager asset-key as seamless as the install guide indicated. I had to manually create the license file to include the asset-key information.

Running the license manager under Caldera v1.3 was straightforward. It wasn't quite as smooth under Caldera v2.2. The license manager wants to place a lock file in /usr/tmp. This directory did not exist in my v2.2 installation. I chose to create a soft link between /usr/tmp and /tmp. This link allowed the license manager to create its lock file. On a side note, the license-manager documentation doesn't recommend running the license manager under the root ID.

Documentation and Support

Source-Navigator comes with a *User's Guide* and a *Programmer's Reference Guide*. The *User's Guide* clearly presents installation instructions, contains a tutorial on creating a project, and factually describes each of the Source-Navigator browsers. The *Programmer's Guide* serves as a reference for customizing and extending Source-Navigator functionality. Descriptions and narrations in each of these manuals are supported with screen captures. Where appropriate, sample code is included in the manual.

The *Installation Guide* clearly presented UNIX system memory, disk, X11 and printer requirements. Comparable Wintel requirements are also given. I found the example of UNIX memory utilization for startup, post-project creation and per open browser informative. The Installation guidance also provided a rule of thumb for estimating disk storage requirements on a per-project basis. The default directory table leaves little doubt as to what directories are being created and the type of files being placed in the directories.

Cygnus uses GLOBEtrotter Software's FLEXlm to manage the SN license. The User's Guide contained virtually no information on starting, stopping or checking the current status of FLEXlm. The only reference I could find about starting this license manager was in the asset-key e-mail. A list of available commands and options is displayed if you enter **lmutil** at the command line without any options. More can be learned about the FLEXlm license manager by going to the http://www.globetrotter.com/ web site; there you will find a FLEXlm FAQ and a User Manual.

There is a 17-page tutorial introducing project creation and the basic use of each browser and editor. The User's Guide portion of the manual provides more detail on each of the items covered in the tutorial. I found the tutorial worth the time.

Software Development Kit

Out of the box, SN supports seven languages: C, C++, Tcl, [incr tcl], FORTRAN, COBOL and assembly. An SDK is included with the base product. Using the included SDK, one can modify the graphical user interface, write a new parser to incorporate additional languages, build applications to access the SN project database and implement communications between SN and external applications.

Two Editions

The developer edition is designed for the single developer. This edition can accommodate small- to medium-sized projects. According to Cygnus literature, small- to medium projects are less than 100,000 lines of source code. The Enterprise Edition of Source-Navigator supports work groups and large projects. Projects with more than 100,000 lines of code are considered large. The maximum project size seems to be limited only by disk space and system memory.

Conclusion

Whether doing code archaeology or building the next killer application, this is one of those products that should be in every tool kit. It can provide substantial support, structure and advance software development efforts.

By the way, a fully functional evaluation copy of SN is available from the Cygnus web site. Hopefully, I've piqued your interest enough to download a copy and use it for the evaluation period. It doesn't take long to see the possibilities.

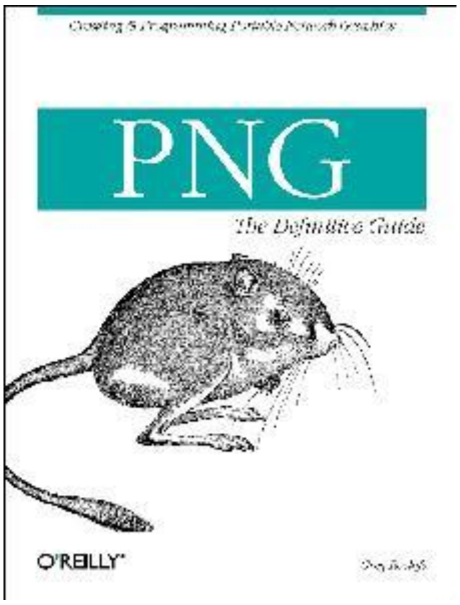
Daniel Lazenby (d.lazenby@worldnet.att.net) first encountered UNIX in 1983 and discovered Linux in 1994.

PNG: The Definitive Guide

Michael J. Hammel

Issue #69, January 2000

There are times when the right person, at the right time, with the right talent, produces a work which rightfully deserves the title of “definitive guide”. This is one of those times.



- Author: Greg Roelofs
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $26.36
- ISBN: 1565925424
- Reviewer: Michael J. Hammel

As a writer who prides himself on being truly good at many things but not great at anything, I often take a somewhat timid approach to anything calling itself “definitive”. What nerve it must take to think of yourself as such an expert! Yet there are times when the right person, at the right time, with the right talent, produces a work which rightfully deserves the title of “definitive guide”. This is one of those times.

PNG—The Definitive Guide is Greg Roelofs' description of the design and use of PNG. PNG is a specification for a raster image file format. Greg's text, published by O'Reilly, is a clear and thorough exploration of the topic. Ranging from the early history of the design to detailed programming information on chunks and gamma correction to application support for this new format, the text is a complete descriptive volume on the subject.

Pronounced “ping” according to Greg (although I have a hard time not vocalizing the three letters separately), PNG has evolved over the past four years in response to limitations in design and licensing of the older GIF format. Greg is uniquely qualified to describe the format and tell its tale of evolution. He was there when Thomas Boutell first announced the project, and has written numerous articles on the subject—including being one of the few contributing authors I've ever had to my “Graphics Muse” column in *Linux Gazette* (http://www.linuxgazette.com/).

The text is divided into three parts: Using PNG, The Design of PNG and Programming with PNG. The specification for PNG is platform-independent, and Greg keeps true to this philosophy with this text. It covers everything from Linux to Windows to BeOS to OS/2; however, the first section does finish with a focus on Macromedia's Fireworks 1.0, a decidedly non-Linux application.

Greg starts with a decent introduction to raster graphics in the first chapter. This leads into a long description of the things for which PNG is most useful. I found this section quite good. Although I knew there were advantages to using PNG, I never knew what the specific reasons were. One of the most important, in the near term if not immediately, will be its support of standardized color management.

Greg points out in part one that PNG does not compress as well as JPEG does, but that JPEG's compression algorithm can lead to some blockiness around sharp edges in an image. He explains that PNG perhaps makes a better format for intermediate editing than for web browsing. This makes sense. PNG has lossless compression, so repeated saves and reloads in your editing package will produce the same image each time. TIFF is an alternative, but TIFF's compression scheme doesn't offer the same speed vs. quality options that PNG does. PNG can also achieve extremely high levels of compression without data loss. Conversion to JPEG might be preferred when the image is prepared for the Web, but that could be done as a last step before publication. As Chapter 2 mentions, both MSIE and Netscape support PNG in their latest browsers, although to different levels, and neither fully supports transparency in their released versions.

Chapter 2 also has an interesting discussion on using content negotiation. This is where you specify a text file with a .var suffix containing text describing one of multiple choices for the image to be displayed. Apache supports this with the **mod_negotiation** module and determines which of the image files specified in the .var file to send to the browser. A fairly interesting topic if you are into supporting any browser on the planet.

Chapter 3 is a simple application reference guide for image viewers on multiple platforms which support the PNG format. Greg does a good job of researching many different platforms here.

Chapter 4 goes into several image-editing applications with support for PNG. Greg shows how an image can be created in each of four major applications—Photoshop 4 and 5, Paint Shop Pro and the GIMP—and then how that image can be saved with PNG.

Chapter 5 describes a number of command-line image processing tools that support conversion to or from the PNG format or provide information on PNG files. All of these tools run on Linux, and Greg provides download information at the end of each application-specific section of this chapter, as he does for every other application he discusses in the text. The chapter ends with a reference-style list of other conversion tools. Each entry includes OS and download information.

The first section of the book closes with a chapter on 3-D applications with support for PNG. Greg covers VRML browsers in the first part of this chapter and has short discussions on 3-D modelers and other 3-D applications in the latter half. Unfortunately, there isn't much support for VRML under Linux, and out of the few browsers and VRML applications available for Linux, even fewer still apparently support PNG. This seems strange, considering PNG has been adopted as one of two formats for VRML certification.

In producing this book, O'Reilly followed the same basic industry practice of stuffing a couple of color plates in the middle of the book. In this case, with only two pages of plates, they might as well have left them out. As with most graphics texts, the comparative images printed in black-and-white outside the color plates aren't very useful. The first one is a set of three images showing a child standing in what looks like one of those boxes your luggage must fit in to carry it onto an airplane. The three images are supposed to show the effects of quantization and dithering, but that's nearly impossible to see in this black-and-white image. There are full-color versions of these same images, which are also enlarged, in the two-page color plates in the middle of the book. These color images do show the differences in the images quite well. When will publishers realize that texts about graphics need to be in color? Yes, it increases the cost, but it also increases the informational value. I'm thankful SSC used four-color printing throughout when publishing my GIMP book.

One minor side note: O'Reilly placed one of those generally annoying advertisement cards at page 188. The card was physically connected to the binding. I tried to pull it out, and ripped the page. Fortunately, I didn't rip through any of the text on that page. They might want to think about adding perforations at the connecting point for the card.

Chapter 7 starts the second part of the book, covering the design of PNG. This chapter is an interesting history into the evolution of an Internet-supported standard like PNG. An interesting bit of trivia which I was not aware of comes at the end of the chapter, where Greg tells us that PNG has been adopted in response to the MHEG-5 UK Profile, by digital television makers like Sony, Phillips and Nokia for both televisions and set-top boxes. I hadn't thought much about bitmap formats for television—I guess there's a whole new area of graphics for me to explore.

Chapter 8 introduces the basic PNG specifications. Here you'll find a description of the chunk, the main building block of the PNG format. Chapter 9 moves into compression information. Although the second part of the book is technical in nature, end users will find useful hints sprinkled throughout. In Chapter 9, for example, there are tips that cover which image types to use (PNG vs. JPEG, for example) and what settings to look for, given the format chosen. Greg follows up with tips for programmers on using compression and filtering.

One of the reasons I jumped ship from programmer to writer was that I saw the world was full of people who could write code, but short on those who could explain that code. I still see many technical folks taking a shot at writing for the masses and falling short, often slipping back into the acronymal world of the techno-geek. Greg is one of a rare breed. His writing style is both clear and complete, detailed and comprehensible. They should give medals for this sort of writing or at least little gold sticky stars.

Chapter 10 moves into one of the more complex issues in computer graphics: gamma correction and precision color. Display devices differ from their hard copy brethren in many ways, and even in subtle ways from each other. Gamma correction helps to ensure an image displayed on one device looks the same on another. Some of the gamma-related features of PNG and how to use them are discussed here.

PNG is an extensible format, providing both a defined set of public chunks and a method for implementors to add their own private chunks. In Chapter 11, Greg covers many of the extensions described in the original specification or that have been officially registered. These include such topics as histogram information, suggested palettes for low-color displays, fractal-image support and pixel calibration. Each section of this chapter is short, providing a quick explanation for the purpose of the extension, and in some cases, detailed programming issues. Part two closes with a chapter on MNG, the multi-image format cousin of PNG. PNG doesn't handle animation formats; it wasn't designed to do so. Chapter 12 discusses MNG in some depth, but this format is both still evolving and fairly complex. The chapter does give a nice overview of the format, however.

Part three is a detailed discussion of the two reference libraries for the PNG specification, namely libpng and zlib. The latter is the standard compression library used with PNG images. Both libraries are freely available and not license-restricted. Chapters 13 through 15 present a thorough programming guide to both the libpng and zlib APIs, including numerous code examples. Chapter 16 closes the book with short notes on other libraries and specifications of interest to the PNG programmer.

Two additional sections follow Chapter 16. The first is a well-done, chapter by chapter bibliography (although not all chapters have references listed). The second is a glossary of terms. The definitions given here are quite good, most providing more than the typical one- or two-line descriptions found in other texts.

If it's not obvious by this point, I'll state it clearly: I really liked this text. Personally, I don't have a major need for information on the PNG format. Until it's more widely adopted by the people I work with (such as publishers when asking for screenshots or other artwork), I'll probably stick to using TIFF, but that's not related to why I like this text. I'm a writer, and writers perform a public function—they provide information (and/or entertainment). Not all writers do this well, especially technical writers. However, Greg has done a very good job with this text on PNG. He's covered the topic well, and he's done so with an understanding of his audience's needs. He's earned the right to call this book *The Definitive Guide*.

Michael J. Hammel (mjhammel@graphics-muse.org) is a graphic artist wanna-be, a writer and a software developer. He wanders the planet aimlessly in search of adventure, quiet beaches and an escape from the computers that dominate his life.

Archive Index Issue Table of Contents

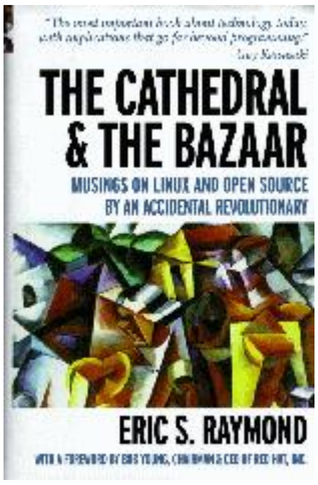
Advanced search

The Cathedral & the Bazaar

Peter Salus

Issue #69, January 2000

O'Reilly has done us all a good favor by collecting a number of Raymond's pieces and making them readily accessible at a price everyone can afford.



- Author: Eric S. Raymond
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: http://www.oreilly.com/
- Price: $19.95 US
- ISBN: 1-56592-724-9
- Reviewer: Peter H. Salus

Beginning in 1992, Eric S. Raymond has jotted notes and comments that were (and still are) net-accessible. Since 1996, several of his essays (most notably “The Cathedral & the Bazaar”) have become required reading. If anything, the obloquy heaped on Raymond by the PR folks in Redmond, WA (e.g., in the “Halloween documents”) has made him more important.

O'Reilly has done us all a good favor by collecting a number of Raymond's pieces and making them readily accessible at a price everyone can afford.

The volume contains “A Brief History of Hackerdom”, “The Cathedral & the Bazaar”, “Homesteading the Noosphere”, “The Magic Cauldron”, “The Revenge of the Hackers” and “Afterword”, plus two appendices.

These are the “commonsense” or the “Federalist papers” for the Open Source movement. They are the testimony of just why the BSDs and Linux, Perl and Python, Tcl and Java are successful: we have tens of thousands of programmers all over the world contributing to the excellence of programs and systems. We don't have an encapsulated proprietary system which no one can debug.

When I was writing *A Quarter Century of UNIX* (Addison-Wesley, 1994), I realized that essential to the “UNIX philosophy” was something alien to commercial programming: the changes to the kernel, the applications and the programs were all written by one or two or three hackers, not by teams of programmers. Eric Allman wrote Sendmail; Mike Lesk wrote the original **uucp** (even the mid-1980s version, HoneyDanBer, was by Peter Honeyman, Dan Nowitz and Brian Redman); Steve Johnson wrote **yacc**; Bill Joy wrote **vi**, etc. Brian Kernighan once told me AWK was the toughest project he ever worked on “because there were three of us” (Aho, Weinberger and Kernighan).

Of course, it's all the Internet's fault. Even with the semi-annual USENIX tape-swaps and uucp, stuff passed about more slowly. It's the Net that enabled a Finnish student to send his work to nearly every corner of the world and enabled thousands to contribute and feed stuff back to him.

In some ways, “The Magic Cauldron” is my favorite essay of Raymond's. Here, he shows that he understands the underlying economic reasons for the success of open software. This understanding is based on the anthropological study of gift exchanging and the concepts of what happens in a gift culture when “survival goods are abundant”, and therefore, the exchange is no longer interesting.

This is tied together with the notions inherent in the fact that software has two distinct values: use value and sales value. As Raymond says, use value is value as a tool; sales value is value as a salable good. One of Raymond's most interesting discussions is founded in this.

Food, equipment and books all retain value independent of the producer. If a farmer sells his farm, the food produced retains its value, etc. When a computer manufacturer (hardware or software) goes out of business or a line is discontinued, the price users are willing to pay plummets. The price users will pay is limited by “the expected future value of vendor service”.

Open-source software forces the vendor into a world of service-fee-domination and exposes “what a relatively weak prop the sale value of the secret bits in closed-source software was all along”.

The true advantage for all of us lies in the notion of high-quality software being built upon by the community, rather than being locked up in a vault or discontinued.

Raymond believes that in 2000/2001, Linux will be “in effective control of servers, data centers, ISPs and the Internet, while Microsoft maintains its grip on the desktop”. Most likely, that's correct. But with the advent of products like StarOffice and WordPerfect for Linux, there may well be inroads into the desktop market as well.

This is a fine, thought-provoking book that should be read by anyone interested in computing: open, academic or proprietary.

Peter H. Salus, the author of *A Quarter Century of UNIX* and *Casting the Net*, is Editorial Director of SSC. He can be reached at info@linuxjournal.com.

Simplified IP Addressing

Gene E. Hector

Issue #69, January 2000

A look at an easy way to figure out what those pesky IP addresses actually mean.

Some years back when I was teaching Novell courses, I noted that many students in my TCP/IP course were having problems understanding IP addressing. Other instructors were seeing the same results. Although I carefully walked the students through the material, many of them still had difficulty grasping the concepts. Furthermore, their problem was exacerbated by having to take the CNE tests, which were closed-book and timed. Hence, they needed a quick and effective way of working with IP addresses.

I fine-tuned an approach to minimize the chances for test errors and to help them memorize and apply the principles at almost any time, even years later. I used to give this one- to two-hour IP presentation toward the end of the course, and it worked great. Very few of my students failed the test.

In the last couple of years, I have been teaching at DeVry and found the need to “dust off” my IP presentation. It has worked equally well at DeVry. This article summarizes that presentation.

Basics

The first question concerns what constitutes a Class A, or a Class B, etc. network. Novices have trouble remembering where each class begins and ends. Table 3 shows a schema to help with this. First, let's discuss some basics of binary numbers.

A byte is a grouping of eight binary bits. Since a binary bit is either a 0 or a 1, a byte consists of eight 0s and/or 1s. No mystery here. So 10010101 is one byte and 11100000 is another. How do we convert these to decimal numbers? It turns out that the right-most bit has a weight of 1 (2<+>0<+>). The next bit to its left has a weight of 2 (2<+>1<+>), the next has a weight of 4 (2<+>2<+>), i.e., two raised to the second power and so on:

```
  2
```

or equivalently:

```
  128  64  32  16  8  4  2  1     : decimal weights
```

Thus, the binary number 10101001 has a decimal equivalent of

```
  1x1 + 1x8 + 1x32 + 1x128 = 169
```

If you assign contiguous 1s starting from the right, the above diagram can be used as a kind of calculator. Let's say you have 00001111 binary bits. To get the decimal equivalent, you could do the calculations the hard way, that is:

```
  1x1 + 1x2 + 1x4 + 1x8 = 15
```

or you could note the following (taking our number):

```
  128  64  32  16  8  4  2  1     :decimal weights
    0   0   0   0  1  1  1  1 :binary number
```

If you have all ones starting at the right side, you can simply take the weight of the first 0 bit (16 in this case), subtract 1, and you have 15—the decimal equivalent—without having to use a calculator. Thus, if all the bits on the right are 1s, you can determine the decimal value by using the above diagram as a kind of calculator.

Note that the bits go up in powers of 2, so the ninth bit has a decimal weight of 256. So if you have a byte with all ones, i.e., 11111111, then it has a decimal value of 255 (256 -1). 255 appears many times in IP addressing.

<u>Table 1</u>

Now we need to construct another calculator for handy reference (see Table 1). There is a thing called netmasking, which I will discuss later on. Standard procedure says to start the masking from the left and work down. So, if you make the eighth, or high-order bit, 1 and the rest equal to 0, the decimal equivalent is 128; if you made the first three bits 1 and the rest 0, the decimal equivalent is 224, etc.

<u>Table 2</u>

This table works fine, but is a bit unwieldy. Table 2 shows a short version. It says that if your byte is 11100000, then the decimal equivalent value is 224. If this bothers you, just use Table 1.

IP Addresses

We have set the groundwork for IP addressing, and I will now discuss the standard IPv4 addresses. The IP addresses are sometimes called “dotted quad” numbers. There are five classes of IP addresses, i.e., A, B, C, D and E. Classes D and E are reserved so you can work with classes A, B and C. However, I will show all five here. The class is determined from the first byte. Thus, an IP address of 205.140.187.31 is a class C address, since the first byte is 205. How do I know that? Well, let's look at Table 3.

Table 3

How did I get Table 3? I had to remember only a couple of pieces of information, then I constructed the rest. I know there are five classes of IP addresses, and the first byte of the IP address tells you to which class it belongs. I also know the schema for the binary starting value of the first byte, i.e., 0, 10, 110, etc. Because of the way it follows a schema, the second column is easy to construct. Now, using Table 2, it was easy to construct the third column.

Next, note that the fourth column (ending point) follows naturally by simply subtracting one from the beginning of the next class. A class C begins at 192, while a class D begins at 224. Hence, a class C must end at 223. Now you have no excuses about forgetting the beginning and ending points of each class; merely remember the binary schema, and take a minute to construct the table. On a side note, you don't have to worry about Classes D and E, except that the beginning of Class D tells you where Class C ends by subtracting 1.

Bitwise AND

We need to discuss netmasking, but first, let's digress for a moment. A Boolean AND is just like an “and” in English. You tell Johnny you will buy him an ice cream cone if he puts out the trash “and” makes his bed. If he does neither or only one of them, he doesn't get an ice cream cone. If he does both, he gets the cone.

Table 4

Bitwise ANDs work bit by bit. So, if you AND a 1 with a 1, you get a 1. If you AND two 0s, a 1 and a 0, or a 0 and a 1, however, you get a 0. Table 4 illustrates this operation.

Now let's take a whole byte and do a Logical AND with another byte. Suppose the first byte is 10110010 and the second byte is 01100111. Working from the right, note that the first byte has a decimal value of

```
0*1 + 1*2 + 0*4 + 0*8 + 1*16 + 1*32 + 0*64 + 1*128 = 178
```

while the second byte has a decimal value of

```
1*1 + 1*2 + 1*4 + 0*8 + 0*16 + 1*32 + 1*64 + 0*128 = 103.
```

Now, AND the two bytes:

```
1 0 1 1 0 0 1 0         178 decimal, ANDed with
0 1 1 0 0 1 1 1         103 decimal
---------------             gives
0 0 1 0 0 0 1 0         34 decimal
```

As a second example, let's AND 178 with 255.

```
1 0 1 1 0 0 1 0         178 decimal, ANDed with
1 1 1 1 1 1 1 1         255 decimal
---------------             gives
1 0 1 1 0 0 1 0         178 decimal
```

We know, then, that when you bit-wise AND any byte (number) with 255, you get the number dropping through, i.e., the result is merely the number again.

Netmasking

The default netmasks for the various classes are shown in Table 5 with some sample host IP addresses. Simply put, a host is anything that has an IP address. This includes servers, workstations, routers, etc.

Table 5

So, what does this mean and what do we do with it? Let's work through Table 5. If we take the sample Class A address, 10.0.1.23 and bit-wise AND it with its default netmask, we obtain 10.0.0.0. What is 10.0.0.0? It's the network address —look at the last column.

Notice that the first byte gives the network address when ANDing a Class A network with its default netmask, while the first two bytes give the network address when ANDing a Class B IP address with the default Class B netmask. Hence, we say that the first byte of a Class A IP address gives the network address, and the three remaining bytes give the host addresses, i.e., a Class A address has the form N.H.H.H where N stands for Network and H stands for Host. Likewise, the first two bytes of a Class B IP address pertain to the network, and the last two bytes pertain to the host address, i.e., N.N.H.H. Finally, the first three bytes of a Class C IP address pertain to the network, while the last byte pertains to the host, i.e., N.N.N.H.

Subnetting

Let's illustrate this with a Class B IP address such as 142.168.25.100. From Table 5, we know that the default netmask for a Class B network is 255.255.0.0.

Hence, ANDing the default mask with the IP address yields the address of the network that particular host is on, i.e., 142.168.0.0. So, a host with an IP address of 142.168.25.100 finds itself on a network with an IP address of 142.168.0.0 if a default Class B net-mask is used.

If you are granted a full Class B suite of addresses with a network address of 142.168.0.0, what do you do with them? Remember, a Class B network has the form of N.N.H.H, i.e., the last two bytes can be used for assigning host IP addresses. This yields a network with 2<+>16<+> - 2 host addresses. The -2 comes from the fact that 142.168.0.0 is the network address, so it can't be assigned to a host; the last address on the network, 142.168.255.255, is used for broadcasts, so it also can't be assigned to a host.

This would be a very big network (65,534 host addresses), far too big to be practical. A very simple approach is to "borrow" one byte's worth of host addresses and assign them as network addresses. That would yield 2<+>8<+> = 256 networks with 254 hosts on each. Even here, these are large networks. This process of borrowing host addresses and using them for networks is called subnetting. We accomplish this by using a sub-netmask (SNM). In this case, we would use a sub-netmask of 255.255.255.0, which is the default Class C netmask. Hence, we have taken one Class B network and turned it into 256 Class C networks.

If we AND 142.168.25.100 with 255.255.255.0, we get a network address of 142.168.25.0 with the first available host address of 142.168.25.1 and the last of 142.168.25.254, since 142.168.25.255 is reserved for broadcasts. Another way of doing this is to start with the network address (142.168.25.0 in this case), turn all host bits into 1s, and obtain the broadcast address. Here, the last byte is used for host addresses, so turning them to ones gives 142.168.25.255. This type of broadcast is called a *directed broadcast*, meaning that it jumps routers while a local broadcast (which doesn't jump routers) has the form 255.255.255.255 no matter which class of network is involved.

If you're not too stunned at this point, you may wonder if you can subnet only on byte boundaries or if you can subnet a Class C network. The answers are "no" and "yes", respectively; i.e., you can work in the middle of a byte.

Subnetting on Non-Byte Boundaries

Let's say you are granted a full Class C suite of addresses, e.g., 210.168.94.0 as your network address. You are allowed to assign the host addresses (the last byte) as you please. If you use the default Class C netmask of 255.255.255.0 (see Table 5), you can assign host addresses of 210.168.94.1 through 210.168.94.254 on a single network. That's feasible of course, but you may want to break this up into multiple networks of perhaps 25 hosts each.

Let's do some mathematics. If we have 4 bits for hosts, will it be enough? 2<+>4<+>-2 = 14 and is not enough. So, let's use 5 bits for hosts: 2<+>5<+>-2 = 30 which will work. However, we have 8 bits in the last byte for hosts, so let's borrow three bits for subnetworks; then we still have the requisite 5 bits for hosts. Great, but how many subnets do we have? How about 2<+>3<+> = 8? We have, then, eight subnetworks with 30 host addresses on each. If you are doing the math, you are probably saying, “but 8x30 is only 240 addresses; what happened to the others?” Valid question! Oops, don't get sore, but it's time to construct another table. Note that each address will have the form of 210.168.94.*last byte*, and the SNM (sub-netmask) will have the form 255.255.255.*last byte*. Let's just work with the last byte.

Table 6

From Table 2 (or Table 1), we see the SNM will be 255.255.255.224. The 224 comes from the last byte being 11100000. So what are the subnets? Table 6 shows them (last byte only).

Let's detail a few. First, take the smallest. The full subnetwork address of the smallest is 210.168.94.0. The next one up is 210.168.94.32, and so on. Remember that with three bits to work with, we get 2<+>3<+> = 8 subnets, and looking at Table 6, you see them.

Table 7

Back to the question of why we get only 240 host addresses. “(Gasp)—another table!” Looking at the last byte, we get Table 7.

Now let's answer the question of what happened to the other addresses. To do this, tally all the “invalid addresses”, i.e., those that can't be used for host addresses.

First, we have eight subnets, each with a subnetwork address and a broadcast address. So we lose 8*2 = 16 addresses here. Now if we subtract these 16 from 256, we get 240 available host addresses.

Doing it the other way is much easier. We have eight subnetworks, each with 30 valid IP addresses; this gives us 8*30=240 valid IP addresses total, the magic number.

For fun, let's do one more thing: analyze the sixth subnetwork in a little more detail. The last byte is 10100000 binary or 160 decimal. The full subnet address is 210.168.94.160 decimal, and we use an SNM of 255.255.255.224. Remember, I said to take the subnet address, set all the host bits to 1s and add them to get the broadcast address. If we do this correctly, it should give the same result as Table 7.

We use five bits for host addresses, so the decimal value of the sixth bit is 32. Subtracting 1 gives 31. Thus, setting the five host bits to 1s, i.e., 00011111, gives a value of 31 decimal. Adding this to the last byte of the subnet address (160) gives 191 for the broadcast address, agreeing with Table 7. Here is the "whole Megillah":

210.168.94.160 The Sub-Network address210.168.94.161-190 Valid host addresses210.168.94.191 Directed Broadcast address

One final point. Some authors use the term "sub-netmask" even when referring to the default netmasks—they are being just a tad loose with their terms. Happy IP addressing, and remember, Linux is inevitable.

Gene E. Hector is an Assistant Professor at DeVry, Pomona. He holds B.S.E.E and M.S.S.E. degrees, is a Registered Professional Engineer and a Novell CNE. In recent years, he has been consulting and teaching. Aside from playing with his grandchildren and going on a few weekend trips to local mountain resorts with his wife Barbara, his hobby is Open Source Software (OSS) including Linux. Professor Hector can be reached by email at geneh@netsnwebs.com.

Archive Index Issue Table of Contents

Advanced search

Audio and Video Streaming for the Masses

Gerald Crimp

Issue #69, January 2000

How one company used Linux to provide transmission of live audio and video over the Internet.

Linux has garnered much attention in the mainstream press recently. Such publicity has been both a boon to Linux enthusiasts and advocates, and a source of curiosity to those who may just be learning of this marvelous operating system. I am very encouraged to see Linux growing in popularity. It's always pleasing, however, to see this popular media attention complemented by concrete examples of Linux in everyday use. As one of those fortunate individuals who gets paid for his passion, I would like to provide one such concrete example by sharing details of how we use Linux at Global Media Corporation.

Global Media Corporation (http://www.globalmedia.com/) is a publicly traded company that has built a private network for the transmission of live audio and video over the Internet. For Network Associates who want to stream their content and/or offer merchandise through electronic commerce, we provide a ready-made infrastructure based on Linux.

Network Engineers

Figure 1. Network Engineering Team

I work on the Network Engineering team (see Figure 1) where Linux is the heart and soul of what we do. Credit for this commitment to Linux goes to Marco Belmonte, our Technical Lead. When the Powers That Be were planning the whole project, Marco told them of Linux. He showed them how Linux would provide a reliable, scalable, flexible network and be cost-effective. We haven't looked back.

Broadcasting over the Internet

To broadcast over the Internet, we send a Linux box to the Network Associate. This is most often a radio station, but can be anyone with an audio source. The box is connected to our Network Operations Centre (NOC) and to an Internet access point over our private frame-relay lines. From the NOC, we can monitor, control, configure and update all the remote stations.

The boxes consist of Intel Celeron-400 chips on Abit socket 370 ZM6 motherboards with 64MB of RAM. The motherboards are equipped with Winbond monitors to ensure we can detect intrusion and track temperatures, voltages and fan speeds. This is important so that we can ward off problems that might cause the box to crash—no broadcaster likes dead air. The audio feed enters the box through a SoundBlaster 128 card. The encoded signal leaves the box through a Sangoma S508/FT1 WAN card. A 14.4 Hayes-compatible modem allows us to carry out emergency maintenance should something happen to the frame-relay connection.

Figure 2. Sangoma WAN Card

This is a good moment to offer special recognition to Sangoma Technologies. As mentioned, Sangoma is the maker of the WAN cards (see Figure 2) we use to connect our boxes to the frame-relay network. Sangoma is also one of the few hardware manufacturers that has recognized and supported Linux from its earliest days. In the context of Global Media, this recognition is particularly appropriate, as they spent many hours adapting their software so we could configure their cards from a script rather than interactively.

Returning to our setup, the encoding boxes run a customized distribution of Linux. Since the machines would never have been used as workstations and only a few services were needed, the boxes were stripped down to core libraries and utilities. Added to this slimmed-down distribution were special configuration and monitoring scripts and the software for digitizing the audio signal.

To avoid the work of installing our distribution on every machine, the boxes are built from cloned disks. The cloning of disks and assembly of boxes is contracted out. The contractor ships a box to a new site after entering a unique identifier we specified. This identifier serves to configure the machine once it arrives at its destination.

Configuration is done over the wire by automated scripts. We chose this method for a number of reasons. First, as the boxes are shipped directly from the point of assembly, we have no opportunity to do the work ourselves. Second, security concerns dictated that we not provide elements of the configuration information to third parties. Next, we wanted the setup and operation of the box to be as simple as possible for our customer: they should only have to plug in the box and turn it on. Few clients, if any, would be comfortable running Linux commands on a text console. In addition, leaving this to the customer would increase the potential for typos and other related errors that would likely cost us many hours on the phone, trying to debug the problem. Finally, configuring machines by hand is drudgery best left to machines.

Automation ensures the boxes are configured and operational, both quickly and securely. Upon receipt of the box, the customer first attaches all the cables, then turns it on. From the boot scripts, a configuration client detects that the box has not yet been configured and contacts a configuration server in our NOC. Using the identifier sent by the remote box, the configuration server queries a PostgreSQL database to acquire all the appropriate information. Once the client has configured itself, it becomes part of our broadcast network. From that moment on, any audio coming into the box is available for listeners on the Net.

I mentioned we use private frame relay to pick up our audio signals. We could have simply used the Internet for this, but chose a private network in order to ensure the quality of the signal as far as possible along the path to the end listener. Using the Internet to gather our signals would have subjected us to congestion and competition for IP bandwidth. We would be forwarding to listeners an already degraded signal. We don't simply want to get listeners to their destination; we want to get them there in style.

RealNetworks—one of Global Media's partners—is our link to the Internet. RealNetworks relays our audio streams to their globally distributed network of servers. When a listener requests one of our URLs, RealNetworks determines the origin of the request and delivers the stream from the closest point. Thus, the portion of the total route that the requested stream has to travel over the Internet before reaching the user's player is reduced to a minimum. For the rest of the route, the stream travels over dedicated high-speed lines.

RealNetworks also provides us with the software that encodes the analog audio source (it runs on Linux) and the Global Media Player. RealPlayer and RealPlayer G2 can be used to listen to our streams, but the Global Media Player links the stream to the Network Associate's editorial content and storefront.

Electronic Commerce

E-commerce is all about web servers and databases. In addition to providing the page that the end user sees, web servers coordinate taking orders, verify credit card information and order fulfillment. The databases contain all information concerning the available merchandise and the static and dynamic content populating the served web page.

During the early stages of Global Media's growth, a third party handled this part of the business. Now that Global has a complete technical team, its web serving has been moved in-house. The web development firm was using a combination of Cold Fusion, SQL Server and Internet Information Server on NT. We found the whole situation less than ideal; it was ridiculously resource-intensive and slow. Scalability was also a concern.

The immediate concern when transferring the system, however, was to keep it running. Now that things are reasonably stable, the task of migrating the works to Linux has begun. Moving the static content to Apache on Linux is the first phase. Next, portions of the database will be moved from the SQL Server to Texpress for Linux. The final phase will be the transfer of financial transactions.

Although Texpress is a commercial database, it is worthy of a quick comment. It uses an interesting algorithm for text searches, building bit-pattern indexes. Lightning-fast, it is worth a look on some rainy day.

Network Operations Centre

Just as most of the broadcast network is located remotely—close to the audio sources—so most of the e-commerce network is far away, close to the big pipes. The Network Operations Centre is responsible for control and monitoring of all these remote machines.

For monitoring, we rely on SNMP (Simple network management protocal) and good old log files. We scan the logs in real time using the handy little utility **swatch**. When a message of interest is logged, swatch initiates the prescribed action. SNMP supplements and complements the log files by allowing an even finer-grained view of the inside of each box.

SNMP is the one place within the NOC where we were forced to find a solution outside of open source. On the broadcast network, the management end of the SNMP link uses HP OpenView on a Sun workstation. We did look at various Linux solutions for monitoring network and system information. Most of the functionality was available here and there, but not in a single tool, and it was not always stable. The state of the hardware in the audio-encoder boxes as well as the state of the encoding software and of the frame-relay link is fundamental to the success of our business. We needed something that could receive traps, discover network nodes, poll network nodes, and given the large number of nodes involved, graphically represent the whole. Commercial software seemed to be the only solution.

Nevertheless, the rest of the SNMP system runs on open systems. On the e-commerce side, where there are fewer nodes and polling is sufficient (i.e., no traps are used), NetSaint does the monitoring. Of course, the Linux boxes all use GNU SNMP agents. In fact, it is the open code that allows us to receive DMI (desktop managment interface) information from the audio boxes. The SNMP source code was modified so we could poll the encoder boxes for information concerning CPU temperatures, fan speeds and voltages.

Thanks to Linux, we did not need to make any expensive maintenance requests of vendors. Given these fine freely available tools, we can detect and fix trouble without the customer even knowing that anything was wrong.

One last Linux feature needs to be mentioned before closing. We shape the traffic on our pipe while the audio signal moves through our network toward the Internet. Sound quality depends on available bandwidth. One of the cornerstones of our business plan is high-quality sound; therefore, we have to protect the sound as far down the pipe as possible. The audio signal shares the pipe with monitoring and management traffic. Under normal circumstances, this should not present a problem. However, should a sudden surge in log activity occur or a new software package be sent back up the line, traffic shaping allows us to ensure it does not cut into the bandwidth required for our customer's broadcast.

Global Media Corporation produces the best audio signal possible for Internet listeners, while providing a cost-effective solution to broadcasters. Without Linux, we could not have achieved all we have as cheaply and effectively. Open source permits great independence. If the system had been built using a commercial operating system, we would have been subject to the cost and delay of vendor customizations, that is, if the vendor was even willing to consider making any modifications.

Many businesses still regard Linux and other open-source software with suspicion. It is still often perceived as a hacker toy unsuitable for real work. We hope our experience will help dispel such myths and encourage others to consider Linux as a viable work tool.

Gerald Crimp (gcrimp@globalmedia.com) is a software engineer at Global Media Corporation in Vancouver, BC, Canada. Debian is his distribution of choice. He looks forward to the day when Linux takes over the desktop.

Archive Index Issue Table of Contents

Advanced search

The Bullet Points: Linux 2.4

Joseph Pranevich

Issue #69, January 2000

A look at what's new in the next kernel release.

Every so often, something happens that is so breathtaking, so absolutely amazing, that it changes the world. The Linux 2.4 release probably won't be one of those momentous events, but it certainly is an important event in the evolution of Linux. Linux 2.4 marks the first kernel released under Linus' rapid-release plan, where more kernels are released with fewer new features so that those new features can get to the end users faster. Linux 2.4 also includes a lot of “loose ends” left over from Linux 2.2, including a number of features that make it more useful to the average desktop user.

I will attempt to outline some features of the new kernel which I think are most important, most revolutionary, or most newsworthy. Your opinions and mine are probably different, but please don't flame me too hard.

Universal Serial Bus

One of the features most requested in 2.2 but undelivered, Linux 2.4 will finally include support for the Universal Serial Bus (USB), an external device architecture that is becoming more and more common for devices such as sound cards, keyboards, modems and printers. This type of device is popular for both PCs (i386) and Macintoshes. The iMac uses USB almost exclusively.

At present, it isn't known how well USB will be supported for 2.4. It is generally expected that most hubs will work, as will keyboards, mice and sound cards (speakers). Support for many other devices will probably be added into the kernel before the release. As I am writing this a couple of months before you'll actually see it, things will probably have changed by the time you read this.

USB support is considered by many to be one of the larger obstacles to Linux's success as a desktop platform in the twenty-first century. While many USB ports in the world currently go unused, the amount of hardware available for this architecture is increasing daily. Sooner or later, it is likely that the “average” desktop will require USB support in an operating system.

ISA Plug and Play

Another feature long lacking in the Linux world is ISA PnP, the bus specification adopted by the Windows world to bring intelligence to an unintelligent bus architecture (ISA). Plug and Play has almost always been available under Linux through user-space configuration utilities and by using external modules for drivers. With PnP added into the kernel, it will be possible for the kernel to autodetect these devices explicitly and auto-configure their resources. No more fussing and mussing with unnecessary config files or module configurations. No more updating modules.conf by hand when you install a new card. Of course, your device order might still change.

How did the Linux developers do it? Linux 2.4 also includes a completely rewritten resource-allocation system that allows device resources to be managed more easily than before. It also allows Linux to manage resources for devices where the drivers have not yet been loaded, to prevent accidental misconfigurations of devices—just one of many “minor” changes that made major changes so much easier to make.

It's obvious that ISA PnP will benefit Linux most in the desktop arena. It will finally be possible to *easily* support a much larger percentage of the existing hardware of the world. Linux 2.4-based systems will simply be smarter and simpler in this respect.

PCMCIA Support

Wait! My Linux distribution has supported PCMCIA cards for quite some time! This isn't a new feature...

Actually, for several years, the various distributions have been shipping Linux variants with PCMCIA support provided by an external developer. For various reasons, he chose not to incorporate his code into the mainstream kernel, and left it up to the distributions to work out the issues. This added complexity to an already complex system, and more or less ensured that no distribution could really ship with a “stock” Linux kernel. Not to mention the headaches involved in actually getting the matches merged and compiled.

As of Linux 2.4, that will change. While you will still need an external package to handle card manipulation and other issues, support for the PCMCIA devices and buses themselves are now provided in the kernel. This will level the playing field for distributions in some ways, as well as make using PC cards that much easier for those of us who never use a distribution's pre-compiled kernel. I'm writing this on a laptop on which I've never managed to get a working PCMCIA driver for a non-Red Hat kernel; Linux 2.4 will be heaven to me—no more booting into Windows to mail out articles. (Of course, that could just be my stupidity.)

With this feature, Linux scores another point for the desktop user and also for the Linux user on the go. In short, mobile Linux is just more practical. Any feature that makes Linux more practical is a feature I like.

Wake One

These big changes for Linux have all been of great advantage to desktop users, but Linux 2.4 will include many advantages to those who are using more traditional Linux (server) boxes. The change which has garnered the most media attention is *wake one*.

At heart, wake one is a simple change meant to counteract the “stampede effect” seen under certain conditions on busy web servers. Previously, when a number of daemons or other processes were waiting for something to happen over the network, they would all sit idle and then all spring into action at the first sign of communication. Unfortunately, for every web connection, only one daemon can be *talking* and so most of these lose out and go back to waiting and hoping that they're *next*. Linux 2.4 has integrated changes which basically wake one of the processes instead of all of them. This results in less unnecessary thrashing of daemon processes and a faster system overall.

Simple? Maybe. In order to support this, Linux 2.4 required a complete rewrite of wait queues and other bits. In addition to solving this problem, Linux 2.4 queues will be more robust and scale better to multiple processors. Just to get an idea of how important Linus felt this change was to Linux, it should be noted that this was the *first* new feature in Linux 2.3 (the development version, which is to become Linux 2.4).

khttpd: the Kernel Web Server

Also on the server front, Linux 2.4 includes support for a rather revolutionary idea: a web server actually integrated into the kernel. The advantages of this include a faster response time from the server, because it can work directly at the caching layer and doesn't need to make any network calls to user space. However, this change is not meant as a general solution to web hosting; it can serve only files, not CGIs, and it has been designed to be as simple as humanly possible. Any requests it can't handle can be passed to user-space, where Apache or another waiting web daemon can snatch it up and serve it. This double-decker style of web hosting has already been observed to increase server performance in synthetic load environments.

Many Linux users have actually spoken out against this feature, citing the kernel as no place for a web server. My personal feeling is that serving files in this manner is no different than any other in-kernel file server, and it actually does make sense in cases where performance is the true “key” issue and simplicity can be assured. Currently, only **knfsd** actually does file serving from the kernel, but the precedent is there. I don't believe we should, for instance, serve FTP from the kernel or integrate a DNS server into the kernel, but if there is a good advantage to doing something this way, then it should be an option.

Raw Devices

Another feature for servers that Linux 2.4 will support is so-called “raw” devices. Raw devices are special device nodes which can be associated with normal block devices to provide direct and low-level access to devices. This can be used, for example, by databases that think they can handle their own caching better than the kernel. The jury is still out on how useful this feature will actually be, rather than how useful its proponents say it is, but this is nonetheless another big step toward providing “commercial grade” UNIX features under Linux.

This feature has long been requested by database developers and others who are used to the raw features of other UNIX systems for their server applications. I don't see much need for bypassing caching, except in the case where data integrity is paramount and loss of performance is not an issue. But maybe the feature just isn't for me, and makes perfect sense to a particular type of application.

NFSv3

Just as Windows and Macintoshes have their own protocols and methodologies for accessing shared drives, UNIX systems have NFS, the network file system. Linux has long supported NFS sharing and mounting in all of its flavors. Linux can even boot (without **initrd**) off such a shared drive. Linux 2.4 adds the oft-requested support for the latest version of the NFS protocol, NFSv3, which includes many improvements over the old system as well as ensuring Linux will be compatible with the commercial UNIX systems of the next millennium.

I believe both servers and desktop users will benefit from this change. Having a more robust NFS layer that supports the latest protocols and enhancements is always a win, and many desktop users who share files may be able to take advantage of the new features in the new NFS.

Parallel Ports

Although many users may not notice it, Linux 2.4 includes a complete rewrite of the parallel-port system (again). Like 2.2, Linux 2.4 includes modular support for parallel-port devices including printers and disk drives. Additionally, Linux 2.4 includes “generic” support which would allow for easier support of parallel-port scanners and other devices. Linux 2.4 will also support more parallel ports, including those which require DMA. Many existing systems have this disabled by default in the BIOS; however, it can be reenabled. Also, Linux 2.4 supports using parallel ports as the console. This feature allows some Linux users to use the printer as a debugging tool or keep a hard record of console output.

That's my “Top 8” bullet points of Linux 2.4. Opinions vary greatly as to what features or additions are the most important to each new version of the Linux kernel, and you may have a favorite feature not mentioned here. Linux 2.4 is another great milestone along the path to acceptance and usefulness in a wide variety of markets, and I am positive we will be sharing the “most important features” debate again soon for Linux 2.6. Until then—happy hacking.

Joseph Pranevich (jpranevich@lycos.com) is an avid Linux geek, and while not working for Lycos, enjoys all kinds of writing and working with a number of open-source projects.

The PPP Connection

Marcel Gagné

Issue #69, January 2000

Having trouble connecting to the Internet? Weep no more—the answer is here.

Ah! Bonjour! Nice to see your friendly faces back in my restaurant, “Chez Marcel”. Please, sit down. Your table is ready and François will bring you your wine immédiatement.

François! Wine! Vite! Vite!

As you already know, the topic for this issue is networks and communications. Take it from Marcel; a little networking can get you an awful lot of communication. A memo here and a memo there and all of a sudden you are drowning in e-mail.

Quoi? No e-mail? You have not yet set up your Internet connection? But this is terrible. How can you be overwhelmed with messages promising you easy ways to make money or offering you questionable services, such as finding love in all the wrong places, if you are not yet connected to the Internet?

Let me show you the basics of a PPP connection, and then I will show you ways to make the whole process even simpler. Sounds good, non? The very first thing you need to do is make sure you have PPP installed. PPP stands for Point-to-Point Protocol, and it is the means by which the vast majority of Internet service providers (ISPs) make it possible for their clients to connect.

PPP can be installed as a package. In Red Hat, Caldera, or any of the fine Linuxes that use the Red Hat Package Manager (RPM), you can use this command:

```
rpm -ivh ppp-
```

If you are running Debian, the command is similar:

```
dpkg -i ppp-
```

For brevity, I will not try to go into whether your kernel has PPP support compiled in and what to do if it does not. If you have a recent Linux distribution, I can guarantee it is there.

The next step is to get some information from your ISP. This is your user name (which will most likely be your e-mail address as well) and password, along with the DNS (domain name server) address. You should also have the mail server information for your e-mail package. The sort of information you need to get are POP (post-office protocol) and SMTP (simple mail transfer protocol) server addresses.

When your ISP gives you the DNS address, edit the file /etc/resolv.conf with your favorite editor (vi, anyone?) and add that information as below. The *XXX*s are, of course, replaced by the numbers your ISP provides.

```
nameserver
```

A PPP setup in its simplest form involves a chat script and a properly configured options file. These files are most likely in the /etc/ppp directory.

The first thing to do is edit the options file. Mine looks like this:

```
/dev/modem
     57600
     modem
     crtscts
     lock
     defaultroute
     noipdefault
     connect "/usr/sbin/chat -vf /etc/ppp/chat-script"
```

My modem lives on /dev/ttyS0, but I also have a symbolic link to /dev/modem for convenience (**ln -s /dev/ttyS0 /dev/modem** is the command to create that link, if it does not exist). What do the other items mean? **57600** is my baud rate, while **crtscts** means that my modem should use hardware flow control. The **lock** statement means that **pppd** (the PPP daemon) should create a lock file to ensure that some scoundrel does not try to use my modem while I am trying to connect to my ISP. The next parameter, **defaultroute**, means that a default route to my PPP interface should be established after the connection is up. That means all nonlocal traffic knows to go out through my point-to-point interface. The last parameter (before **connect**) is **noipdefault**, which means I did not specify a local address for the connection. The PPP daemon will try to figure this out on its own by getting the local address from the system.

The **connect** statement describes how we go about establishing this PPP connection and is our chat script. **chat** is a command whose purpose in life is to establish a connection with your ISP. This little program talks to your modem, watching for things like the **CONNECT** string or a login prompt. A sample chat script follows:

```
ABORT 'NO CARRIER'
     ABORT 'NO DIALTONE'
     ABORT 'ERROR'
     ABORT 'NO ANSWER'
     ABORT 'BUSY'
     "" ATZ
     OK ATDT5550000
     CONNECT ""
     ogin: myISPlogin
     ssword: myISPpassword
```

This is the classic “expect/send” script. Expect nothing, then send **ATZ** to reset the modem. When the modem responds with “OK” (you expected that, did you not?), send ATDT*yourISPphonenumber*, and so on.

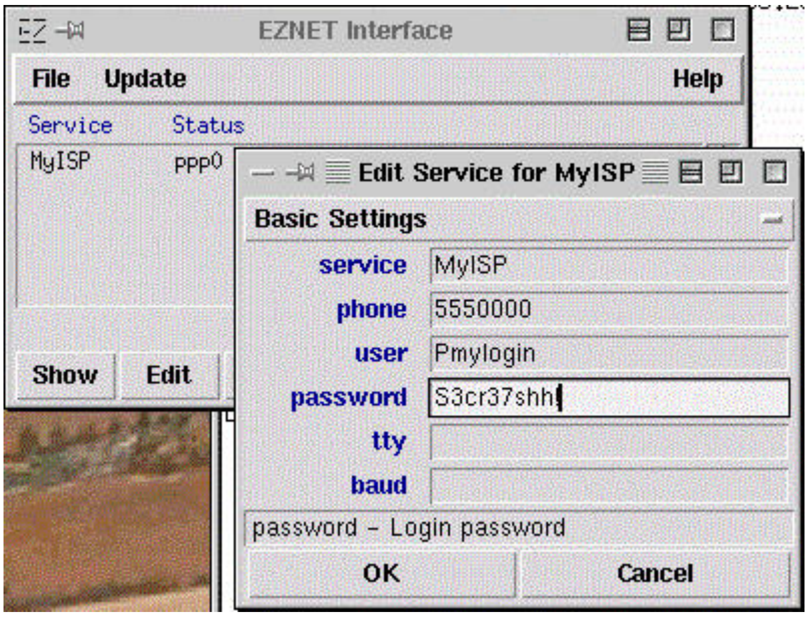
My example above is frighteningly simple. I kill this connection by finding pppd's process ID (use **ps**) and terminating it with a **kill** command. For the down and dirty of PPP, check out the latest PPP HOWTO on the Linux Documentation Project web site (see Resources).

Would you like an even *easier* way to do this?

The truth is that PPP setup is fairly simple; however, little ripples like PAP authentication or CHAP or even the vagueries of your ISP's setup can make the whole process somewhat less friendly. Instead, you can try some of these wonderful precooked programs for easy Internet access. As is the nature of Linux cooks, after working hard on making things work, they will often create recipes designed to simplify the process for others. The following PPP dialer entrées should have your mouth watering in no time. The only catch is that being on-line already makes it easier to get your hands on these, but if you are still fighting with your connection to your ISP, ask a friend to get these packages for you.

First on the menu is Hwaci's **eznet** software. This great little piece of software needs only your login name, password, and ISP's phone number, and you are ready to roll. While you are visiting the web site (see Resources), pick up Mark Hall's Tk front end to eznet, **xeznet**. This client makes it almost too friendly. A note of caution: the “eznet” binary must live in the /usr/bin directory.

Starting the xeznet interface is easy—simply type **xeznet**. For a screenshot of xeznet in action, have a look at Figure 1.



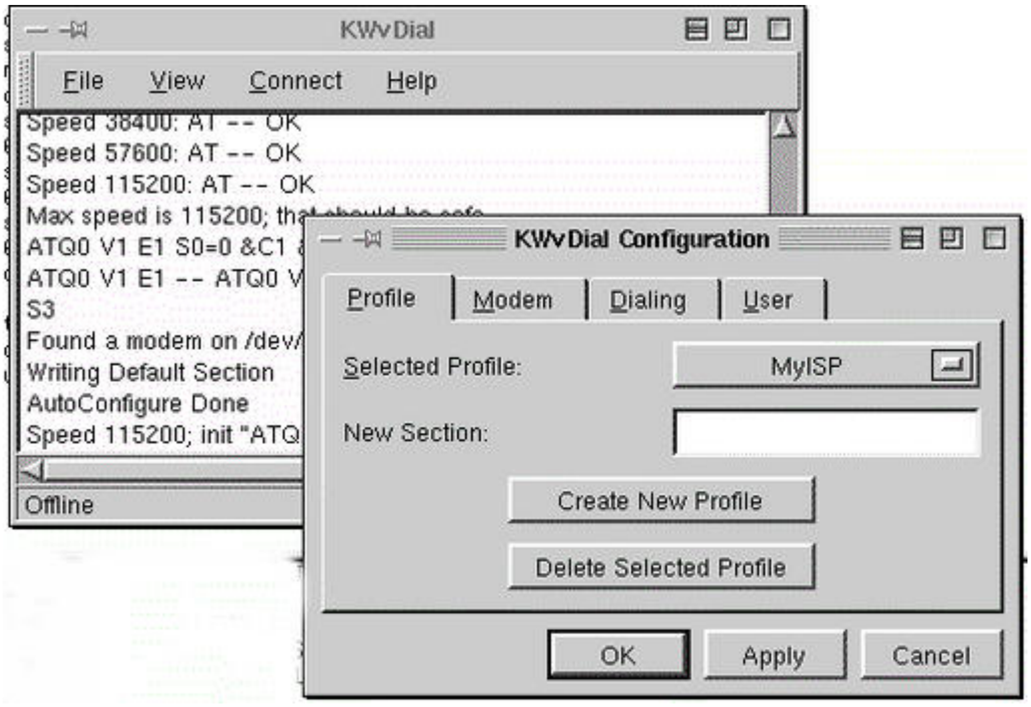
Service Status
MyISP ppp0

service MyISP
phone 5550000
user Pmylogin
password S3cr37shh
tty
baud
password - Login password

Figure 1. xeznet Screenshot

The next item on tonight's menu comes from Worldvisions Computer Technology Inc. Like our previous chef, they have created a wonderfully intelligent tool for connecting to your ISP. In running **WvDial**, I recommend you also pick up Patrick Patterson's KDE front end, a little something he calls **KWvDial**. After getting and building the package, I simply clicked on **Configure**. Kwvdial/Wvdial sought out and found my modem without a problem, setting baud rate on the fly. Armed with only my login name and password, Wvdial/Kwvdial had me connected to my ISP in seconds.

The command to start the interface is simply **kwvdial**.



```
Speed 38400: AT -- OK
Speed 57600: AT -- OK
Speed 115200: AT -- OK
Max speed is 115200; th
ATQ0 V1 E1 S0=0 &C1 
ATQ0 V1 E1 -- ATQ0 V
S3
Found a modem on /dev/
Writing Default Section
AutoConfigure Done
Speed 115200; init "ATQ
```

Offline

Selected Profile: MyISP
New Section:

Figure 2. kwvdial Screenshot

Both of these products are available in source distribution as well as various binary distributions. Check the web sites to decide what is easiest for you.

Well, mes amis, it is once again closing time. I do hope you'll see your way clear to paying us another visit at *Chez Marcel*.

Bon Appétit!

Resources

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and edits TransVersions, a science fiction, fantasy and horror magazine.

A Simple Search Engine

Reuven M. Lerner

Issue #69, January 2000

Searching your web site has never been easier—an introduction to search methods.

The CGI (“common gateway interface”) standard was originally designed to allow users to run programs via the Web, which would otherwise be available only on the server. Thus, the first CGI programs were simple interfaces to **grep** and **finger**, which received their inputs from an HTML form and sent the HTML-formatted output to the user's browser.

CGI programs, and server-side programs in general, have become more sophisticated since then. However, one application is as useful now as it was in the past: the ability to search through a web site for documents containing a particular word or string.

While search sites (now called “portals”) make it possible to browse through a large collection of pages spread out over a number of servers, the CGI programs handling the search have an easier job. They have to go through files only on the local server, producing a list of URLs matching the user's request.

This month, we will look at how to implement several different types of search programs. While these programs might not compete successfully with ht://Dig and Webglimpse, they do offer some insight into how these sorts of programs work, and the trade-offs programmers must make when writing such software.

Simple Command-Line Search

Perl has long been my favorite language for writing server-side programs. This is in no small part due to its strong text-handling capabilities. Perl comes with a rich regular-expression language that makes it easy to find one piece of text inside another.

For example, the following one-line program prints any line of test.txt containing the word “foo”:

```
perl -ne 'print if m/foo/' test.txt
```

The **-n** switch tells Perl not to print lines by default, and the **-e** switch allows us to insert a program between the single quotes (**'**). We instruct Perl to print any line in which the **m//** (match) operator finds the search string. We can accomplish the same thing inside of a program, as shown in Listing 1.

Listing 1

Of course, the above program searches for a single pattern (the string “foo”) inside of a single file (test.txt). We can generalize the program more by using an empty **<>**, rather than iterating over **<FILE>**. An empty **<>** iterates through each element of **@ARGV** (the array containing command-line arguments), assigning each one in turn to the scalar **$ARGV**. If there are no command-line arguments, then **<>** expects to receive input from the user. Listing 2 is a revised version of the above program, which searches through multiple files for the string “foo”. Notice how this version of the program prints the file name as well as the matching line. Since **$_** already contains a newline character, we need not put one at the end of the **print** statement. Listing 2 could be rewritten in a single line of Perl with the following:

```
perl -ne 'print "$ARGV: $_" if m/foo/;' *
```

Listing 2

Finally, we can make our simple search a bit more sophisticated by allowing the user to name the pattern, as well as the files. Listing 3 takes the first command-line argument, removing it from **@ARGV** and putting it in **$pattern**. To tell Perl that **$pattern** will not change, and that it should compile the search pattern only a single time, we use **m//** with the **/o** option.

Listing 3

Thus, to search for the pattern f.[aeiou] in all of the files with a “txt” extension, we would use:

```
 ./simple-search-3.pl "f.[aeiou]" *.txt
```

Sure enough, every line containing an **f**, followed by any character, followed by a vowel is printed on the screen, preceded by a file name.

File::Find

The above would be a good skeleton for our web-based search if all documents on a web site were stored in a single directory. However, the opposite is normally the case: most web sites put files in a number of different directories. A good search program must traverse the entire web hierarchy, searching through each file in each directory.

While we could certainly accomplish this ourselves, someone has already done it for us. **File::Find**, a package which comes with Perl, makes it possible to create a **find**-like program using Perl. File::Find exports the find subroutine, which takes a list of arguments. The first argument is a subroutine reference invoked once for each file encountered. The remaining arguments should be directory and file names, which File::Find will read in sequence until it gets to the end.

For example, Listing 4 is a short program that uses File::Find to print all of the file names in a particular directory. As you can see, File::Find exports the variable **$File::Find::name** which contains the current file name as well as the find subroutine. The current directory is stored in **$File::Find::dir**.

Listing 4

Listing 5

Listing 5 contains a version of simple-find-2.pl, which uses File::Find to search through all of the files under a given directory tree. As with many programs that use File::Find, the bulk of simple-find-2.pl is spent inside of **find_matches**, a subroutine called once for every file encountered under the directories passed in **@ARGV**. To find all files containing the pattern “f.[aeiou]” in directories under /home and /development, type:

```
    ./simple-find-2.pl "f.[aeiou]" /home /development
```

Line 11 of simple-find-2.pl is particularly important, in that it undefines **$/**, the variable that determines the end-of-line character. Normally, Perl's **<>** operator iterates through a file line by line, returning **undef** when the end is reached. However, we want to search across an entire file, since a pattern might have to extend across lines. By undefining **$/**, the line

```
   my $contents = (<FILE>);
```

puts the entire contents of the file handle **FILE** inside of **$contents**, rather than just one line.

Searching via the Web

Now that we can search for a pattern through all files under a particular directory, let's connect this functionality to the Web, searching through all of the files under the HTTP server's document hierarchy. Such a program will need to receive only a pattern from the user, since the web hierarchy does not change very often.

Listing 6

Listing 6 is an HTML form that could be used to provide such input. This HTML form will submit its contents to simple-cgi-find.pl, the CGI program in Listing 7. Its parameter, *pattern*, contains a Perl pattern to be compared with the contents of each file in the web hierarchy, simple-cgi-find.pl will return a list of documents matching the user's pattern.

Listing 7

Unfortunately, the version of File::Find that comes with Perl does not work with the **-T** flag, which turns on Perl's secure *tainting* mode. CGI programs should always be run with **-T**, which ensures data from outside sources is not used in potentially compromising ways. In this case, however, we cannot run our program with **-T**. File::Find relies on the **fastcwd** routine in the **Cwd** module, which cannot be run successfully with **-T**. For the time being, I suggest using these programs without **-T**, but when the next version of Perl is released, I strongly recommend upgrading in order to run CGI programs with full tainting enabled.

Our search subroutine, find_matches, has been modified slightly, so that its results will be more relevant for web users. The first thing it does is to make sure the file has an extension indicating it contains HTML-formatted text or plain text. This ensures that the search will not try to view graphics files, which can contain any characters:

```
  return unless (m/\.html?$/i or m/\.te?xt$/i);
```

Some web sites mark HTML files with extensions of .htm (or .HTM), and their text files with .txt or .TXT rather than .text. The above pattern allows for all of these variations, ignoring case with the **/i** switch and ensuring the suffix comes at the end of the pattern with the **$** metacharacter.

After retrieving the contents of the current file, find_matches checks to see if **$pattern** can be found inside of **$contents**, which contains the document's contents. We surround **$pattern** with \b characters, to look for **$pattern** on word boundaries. This ensures that searching for “foo” will not match the word “food”, even though the former is a subset of the latter.

If a match is found, find_matches creates a URL by substituting **$search_root** with **$url_root**, which hides the HTML document hierarchy from outside users. It then prints the file name inside a hyperlink to that URL:

```
if ($contents =~ m|\b$pattern\b|ios)
{
my $url = "$File::Find::dir/$filename";
$url =~ s/$search_root/$url_origin/;
print qq{<li><a href="$url">$filename</a>\n}
}
```

Improving on our Web Search

While simple-cgi-find.pl works, it does have a few problems. For starters, it fails to differentiate between HTML tags and actual content. Searching for “IMG” should not match any document containing an <IMG> tag, but rather any content outside of HTML tags that contains that string. For this reason, we will modify our program to remove HTML tags from the input file.

Beginning Perl programmers often think that the best way to remove HTML tags is to remove anything between < and >, as in:

```
$contents =~ s|<.+>||g;
```

Since “.” tells Perl to match any character and “+” tells Perl to match one or more of the preceding character, the statement above looks like it tells Perl to remove all of the HTML tags. Unfortunately, this is not the case—the statement will remove everything between the first < and the final > appearing in the file. This is because Perl's patterns are “greedy”, and try to maximize the number of characters they match.

We can make “+” non-greedy and try to match only the minimum number of characters by placing a ? after it. For example:

```
$contents =~ s|<.+?>||g;
```

There is also the sticky issue of what to do if **$pattern** contains white space. Should it be considered as a search phrase containing one or more white-space characters? Or should it be considered several different words with an “or” or “and” search?

Listing 8

In this particular case, we can have our cake and eat it, too. By adding a set of radio buttons to the HTML form, we can allow the user to choose whether a search should be literal, require all search terms be found or require any one of the search terms be found.

Now we can modify our program to handle “phrase” searches (as we have been doing until now), “and” searches (in which all of the words must appear) and “or” searches (in which one or more of the words must appear).

To implement an “and” search, we break the elements of phrase apart by using Perl's “split” operator. We then count the number of words we must find, iterating over each of them and checking to see if they all exist in **$contents**. If **$counter** reaches **0**, we can be sure all words appear:

```
elsif ($search_type eq "and")
  {
    my @words = split /\s+/, $pattern;
    my $count = scalar @words;
    foreach my $word (@words)
    {
    $count- if ($contents =~ m|\b$word\b|is);
    }
    unless ($count)
    {
    print qq{<li><a href="$url">$filename</a>\n};
    $total_matches++;
    }
  }
```

An “or” search is even easier to implement: once again, we break apart **$phrase** across white space. If even one of the constituent words matches, we can immediately print the file name and hyperlink, and return from find_matches:

```
elsif ($search_type eq "or")
  {
    my @words = split /\s+/, $pattern;
    foreach my $word (@words)
    {
    if ($contents =~ m|\b$word\b|is)
    {
    print qq{<li><a href="$url">$filename</a>\n};
    $total_matches++;
    return;
    }
    }
  }
```

Finally, we should have some way of telling the user how many documents matched. We do this by creating a new variable, **$total_matches**, which is incremented each time a document matches (as seen in the above code fragments for “and” and “or” searches).

These improvements are incorporated into the search program called better-cgi-search.pl, in Listing 9, not printed here but contained in the archive file (see Resources).

Excluding Directories and Files

We now have a fairly full-functioned search program which can handle most types of searches people want to do. The problem is that we have created a program which might be *too* good to be useful. Many clients of mine put information on their web sites before it is meant to be released. Without any links leading to these directories and documents, it is unlikely someone will be able to find them. However, our search program does not depend on hyperlinks in order to find documents.

One common solution is for a search program to ignore any directory containing a file named .nosearch. This file does not need to contain any data, since its mere existence means a directory's contents will be skipped.

The easiest implementation would check for the existence of a .nosearch file in the directory currently being probed. However, checking for a file with each invocation of find_matches would reduce our program's already slow performance even more. It would be better if the program looked for a .nosearch file, then stored that information in a hash to be retrieved when future files in that directory are examined.

The Other Problem

We can solve these problems with two lines of code. The first, placed at the beginning of find_matches, returns immediately if a .nosearch file has already been found in the current directory:

```
   return if ($ignore_directory{$File::Find::dir});
```

If we reach the second line, it means that no .nosearch file has been found for this directory. However, there are several circumstances under which a .nosearch file wasn't found, yet should still be in force: when we are examining the .nosearch file itself, when a .nosearch file is in the directory or when a .nosearch file is in the parent directory. After all, if the parent directory should not be searched, then neither should the child directory. Here is the code fragment that accomplishes this:

```
   # Mark the directory as ignorable ...
       $ignore_directory{$File::Find::dir} = 1
           if (($_ eq ".nosearch") ||
               (-e ".nosearch") ||
               (-e "../.nosearch"));
```

Listing 10 contains a version of better-cgi-search.pl with these additions and can be found in the archive file (see Resources).

Is This Any Way to Run a Search?

If you have already run these programs, you most likely found the main problem with the system outlined above: it is very slow. If your web site contains 100 files, this system works just fine. However, if your site expands to 1000 or 10,000 files, users will stop the search in the middle because it will take too long.

For this reason, most serious search engines employ a different strategy, one which separates the searching into two different stages. In the first stage, an indexing program takes the files apart, keeping track of where they might be. A second program is then run as a search client, looking through the pregenerated index for matches.

Next month, we will examine some ways of creating such indices, as well as how to look through them. Perhaps our simple search programs will not be able to complete with Glimpse and ht://Dig, but at least we will understand roughly how they work and what trade-offs are involved when writing search programs.

Reuven M. Lerner is an Internet and Web consultant living in Haifa, Israel, who has been using the Web since early 1993. His book Core Perl will be published by Prentice-Hall in the spring. Reuven can be reached at reuven@lerner.co.il. The ATF home page, including archives and discussion forums, is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #69, January 2000

X Ship Wars, GTK Puyo Puyo, CPAN and more.

Last month, I didn't include any games, although I usually do. I am occasionally reminded that computers, or at least PCs, are as ubiquitous as they are because people like to relax and play games. Most of us have heard of, if not played, **freeciv**, the Civilization clone. But the real Civilization (Call to Power), one of several titles released for Linux by Loki Games, is well beyond freeciv. Does that mean great graphics or game play is restricted to commercial ventures? As I think you'll see with the first two titles below, this is simply not the case. Good games are needed. Educational ones would be even better—let's get the kids involved with Linux, too.

X Ship Wars: fox.mit.edu/xsw/index.html

This is a fun game similar to, but more complex than, the old “Star Trek” game. It has very nice visuals. The basic instructions would have you believe you can just install the client. In fact, while that is true, you also need to connect to a server. There are several available, but you'll want to build and install a server just so you can invite your friends over to be knocked off by you, while you still have the edge. Play takes place over any TCP/IP network. It requires libm, libX11, libXpm, libXext and glibc.

GTK Puyo Puyo: http://chaos2.org/xpuyopuyo/

This amusing game is a multiplayer game in the Tetris genre. You can play against another human or an AI. The AI has many levels, from extremely unskilled to extremely competent. Instead of filling rows completely with various shaped blocks, the blocks are always in twos, but will vary in color. Your objective is to get groups of four or more colors to match in any combination of left to right or up and down. As you or your opponent get matches, the other will get grey blocks to interfere with play. It requires libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, libXpm and glibc.

CPAN: www.perl.com/CPAN/modules/by-authors/Andreas_Koenig/CPAN-1.50.tar.gz

For Perl aficionados, this is a great way to keep up on the latest Perl modules from CPAN, or even grab a new module. It is not the easiest install around, as your initial contact will require loading an index; but once that's done, you're good to go. First bundle I had to get was libnet, because CPAN complained I didn't have it, and it wanted to use it. Once you get the hang of this, you won't want to be without it. It requires Perl.

irm: http://www.redshift.com/~yramin/atp/irm/

The IT Resource Manager is a good tool for tracking hardware and trouble tickets. The baseline for the database is a unique *irm* number given to each system. Users can submit trouble tickets via the Web if they know the system's *irm* number (although most would rather pick up a phone, in my experience). This has good support for a number of areas and is a worthy competitor to MOT. It requires Apache (or another PHP-capable web server), PHP3 and MySQL.

ssh buddy: http://www.sundilla.net/sshbuddy/

I use **ssh** habitually. In fact, anyone who uses **telnet** or the **r** commands must not be connected to the Internet, or else they don't care about security. I'll not belabor my disappointment over the license change on ssh 2.*x*. I will tell you that this graphical client makes connections easier than ever. This will save any administrator time in connecting to servers, and the more servers you have to connect to via ssh, the more time this will save you. If you haven't had time to implement **ssh-agent**, this is the next best thing. It requires expectk and ssh client.

aide: www.cs.tut.fi/~rammer/aide.html

For those who are familiar with Tripwire, this package is meant to replace it with an equivalent under the GPL. Tripwire has been removed from the public domain, and while the older version 1.3 is still good (version 2.*x* is the current, proprietary version), it's getting more difficult to build with changes in the system libraries. Aide is already doing as much as Tripwire, and plans to do even more. As a replacement for Tripwire, this shows much promise. It's already running for me as a cron job on one system where Tripwire 1.3 would not compile. It requires glibc.

Web Downloader for X: http://www.krasu.ru/soft/chechelo/

Got a lot of files you'd like to grab from the Internet? Doesn't matter if they're web pages or FTP files; you can start multiple transfers from the same site. This will probably be of more interest to those with cable modems or fast Internet connections. If you have twenty or thirty items to download, this utility will make multiple connections. It will also allow you to specify a maximum number of connections. It's all packaged in a nice-looking GTK GUI. This GUI will allow you to schedule downloads as well. It requires libpthread, libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libstdc++, libm and libc.

brass monkey: http://personal.bellsouth.net/lig/z/z/zztzed/brassmonkey/

This particular piece of software functions like the comment page on Slashdot. If you have the password, you can start a new thread and others can add comments—many possibilities here. Perhaps I should put each of the blurbs from this column on a web site and invite comments. It requires Apache (or another PHP-capable web server), PHP3 and MySQL.

David A. Bandel (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is co-author of Que Special Edition: Using Caldera OpenLinux, and he plans to spend more time writing about Linux while relaxing and enjoying life in the “Crossroads of the World”.

Letters

Various

Issue #69, January 2000

Readers sound off.

Watch out for the Snakes

Multiple people have asked me about Python and my review of *Learning Python*. To answer their questions and those from people who didn't write, I drafted this response.

First, about the book. It is clearly put together by someone who has been teaching Python to people. I would say the audience has been experienced programmers, which at least for me, is good.

Now, why Python? Is it the “language of the future”? I think so. It makes a lot of sense in a lot of areas. Let me try to address them.

Ease of learning: Python is a small language. That is, the syntax is simple and the number of commands is fairly low.

Powerful: the power in Python (much like C) comes from its library. While the language and its syntax aren't complicated, calling the right function to do a task is what makes it powerful. There are a lot of functions available, but you can acquire knowledge of them as they are needed. For example, if you want to do CGI programming, the functions available are powerful and grouped so you can find them.

Being interpreted, you can quickly try things.

Python is extensible: if you have something you need to add or something that must execute quickly, you can easily add code to do it. Using C for things that need to be fast makes sense.

Python won't teach you any bad habits. There are virtually no “gotchas” in the syntax, and I have yet to find a case where you need to write a cute workaround to deal with a language limitation.

Python is object-oriented, but in a nice way. I have dabbled with C++ but have never actually done object-oriented programming. Python encourages modular and object-oriented programming, but doesn't require it. Thus, you can write a quick-and-dirty solution to a one-time problem, but you can also write solutions to big problems in an easy-to-read, easy-to-debut, easy-to-maintain way. I contrast this with Perl, where you are encouraged to use the quick-and-dirty solution and must go out of your way to create an object-oriented solution.

Perl is a logical outgrowth of an assortment of UNIX tools including **tr**, **sed** and **awk**, as well as shell scripting. While Perl seems familiar to anyone who has used these tools, it makes no sense for someone not familiar with these tools to consider Perl. You can't even define the syntax of Perl, because it doesn't really have any. Python, on the other hand, is well-designed from a formal language sense as well as a programming sense.

I hope this helps explain where Python fits or could fit in your life.

—Phil Hughes info@linuxjournal.com

Linus, A Regular Guy

I have been reading your interview with Linus whilst waiting for our Internet connection to the outside world to be restored—very, very fun read. I have no idea whether the world of Linuxers is interested in this kind of “human interest” stuff or not, but I know that it's great to hear this regular person talk about himself, how he grew up, his ordinary life, etc. and not just the usual Linux advocacy stuff, which Linus seems to avoid, thankfully. This may be because he's less awed by it than everyone else, since he put it together—wonderful modesty on his part. Truly enjoyable journalism on yours.

—David Penn dvdpenn@yahoo.com

Trick or Treat

Thanks very much for the great short article, “Stupid Programming Tricks” by Jason Kroll in October's *Linux Journal*. It introduced me to the world of SVGAlib. I enjoyed his shapes.c program and even had fun changing it on my own.

For the last month, I've been muddling through trying other stuff, but had been eagerly awaiting the more impressive-looking stuff you alluded to in last month's article. Alas, as I'm drooling through my hot-off-the-presses edition of this month's *LJ*, I can't find any mention of console graphics.

Do you plan on continuing the article in the future?

—Patrick A. Kirchner kirchner@csd.uwm.edu

Tricks was inadvertently dropped from layout in the November issue. It resumed in December and will continue to be a regular feature in upFRONT —Editor

MS Breakthrough

Upon reading the interview with Linus in the November 1999 issue, we've noticed that Microsoft has produced a new class of computational analysis problems.

- NP Completeness: all non-deterministic polynomial problems are equivalent, we know we can solve them all, but not how long it will take.
- NT Completeness: all Windows NT boxes are essentially alike, we know they will all crash eventually; the question is, how soon and what data will they take with them when they go?

Thanks for a world-class magazine about a world-class phenomenon.

—Phil Salkie, pvs@howman.com Jennifer Hamilton, jhamilto@umdnj.edu

Ahhh, typos can sometimes be fun. Sorry for the error but not for the laugh. Thanks for writing —Editor

Lie Low

In issue 67's “Best of Technical Support”, Sam Hart said he was concerned about security because the average user could read the lilo.conf file and get the LILO passwords from it. Two suggestions were to put the lilo.conf on floppy (reasonable) or modify the source code of LILO (okay if you want to do much more work than necessary).

The simplest solution is to just use **chmod** on lilo.conf and its directory. This is, in fact, what the LILO documentation says to do.

—Michael James Obrien mobrien@unm.edu

IBM Correction

In the November article “Using Java Servlets with Database Connectivity”, Bruce McDonald talks about IBM's Servlet Express. The product is now called WebSphere and has been for quite some time. WebSphere supports the newest versions of Apache and IBM HTTP Server. If you are on the Windows NT platform, it supports even more web servers. It seems your information on the IBM products is not up to date. Please see http://www.software.ibm.com/ for more information.

—Jakob Carstensen jakobc@us.ibm.com

Archive Index Issue Table of Contents

Advanced search

More Letters

Red Hat and Cygnus unify

Today marks a significant day in the history of Cygnus, Red Hat, Free Software, and open source software. Both Cygnus and Red Hat have long admired each other's organization and innovative leadership in engineering, promoting, and maintaining Linux software. We are both proud of the fact that the software developed and maintained by our companies has become fundamental to the free software and open source communities, and is becoming fundamental to the larger commercial markets as well. We are also mindful of the fact that we did not get to this point alone.

Having spent a lot of time over the past few months contemplating and discussing a possible merger, we have amicably and willingly signed a definitive agreement to merge. To say that we are excited about the possibilities is the understatement of the decade. We believe that by enabling developers from both companies to work together more closely, with a common and larger purpose, we can drive the open source revolution faster and further than otherwise would be possible.

We hope you will continue your support of both organizations as one and help us move open source even further. Please don't hesitate to email myself or anyone else you know at either company if you have any questions, concerns, or suggestions about how we can make this merger a Good Thing for everybody.

—Donnie Barnes and Michael Tiemann djb@redhat.com

Re: Red Hat and Cygnus Unify

I have two concerns that I would like publically addressed.

The first is that Cygnus traditionally has addressed groups well beyond Linux. Other versions of Unix, other types of embedded systems, Windows. Many of these areas directly compete with Linux. I think that people who rely on Cygnus in these areas will want to hear a guarantee that the new combined company will continue to support them

The second is quality control. Red Hat is gaining an unfortunate reputation for having poor quality control. Cygnus has traditionally been trusted and depended upon to deliver quality. The switch to the EGCS code base (no matter how good the reasons for it) has already caused issues with people who want to know that they can trust their compiler to act predictably, this move will raise more question marks.

I have more private concerns (eg a personal distaste for this land grab) but I would not expect those issues to be publically addressed.

—Ben_Tilly@trepp.com

Re: Red Hat and Cygnus unify

It's a sad day when the employees of a company have to read a public forum to learn about what's going on in their own companies. This is the first I'd heard about this (aside from the “rumor” on Slashdot), and since it's being announced in this forum, I'm replying in this forum (trying to keep my focus on the relevence to this forum, of course).

> we have amicably and willingly signed a definitive agreement to
> merge.

Sigh. What happened to the “good old days” when companies had pride in their success, and fought to retain their independence? Is this a sign that neither Cygnus nor Red Hat feel that they were being successful enough? That, in the long run, money is more important than corporate morals, goals, and identity? Is it the goal of all companies, fsbs or otherwise, to merge into the biggest conglomerates they can?

> To say that we are excited about the possibilities is the
> understatement of the decade.

I would replace “excited” with “concerned” myself. I think smaller companies can maintain focus better, will retain their links with the free software community better, and in general be more in tune with their corporate health. I've worked for large companies, and I felt they couldn't even keep track of their own factions, let alone the outside community.

> We believe that by enabling developers from both companies to work
> together more closely, with a common and larger purpose, we can
> drive the open source revolution faster and further than otherwise
> would be possible.

Is that a good thing? I didn't realize we were in such a hurry to overturn the world order. The free software movement has been about people doing the job right, not doing it quickly. We see what happens when companies (MS) push for speedy delivery instead of quality work. Is this the fate of free software also?

—DJ Delorie, dj@delorie.com

Is Gnome the Answer?

I just finished reading Phil Hughes editorial, “Is KDE the Answer?”, where he suggests that GNOME developers should jump on the KDE bandwagon in order to speed up development. I disagree. In my opinion, the GPL is the best of the Open Source licences, in that it ensures that all future development efforts must also remain free software. In the long run, this helps the growth of free software. Consequently, when choosing between software projects, I favour those with a GPL licence (as I'm sure do many others).

—Darryl Plank, dplank@wanadoo.fr

---

spelling

In the latest (Nov.) issue, I noticed the SI prefix “tera-” misspelled as “terra-” in three places: p.9 (item 22); in the Arkeia ad on p.21; and in the second column on p.88.

Please note that the prefix is spelled with only one “r”; it is derived from the Greek “teras” (monster), the same stem as in “teratology”. (See any good modern dictionary, or http://www.cacr.caltech.edu/~roy/dataquan/ety.html—or any of many other reliable sources.)

Editors are supposed to know stuff like that. Please wise up.

—Andrew T. Young, aty@mintaka.sdsu.edu

> *Very embarassing, I admit. I have no idea how we let that one get by us. Sorry.*
> *—Editor*

Interesting info about so-called “SEC IPO quiet period”

Kind Greetings to All,

I sent this email for no particular reason other than I thought you might find it both interesting and enlightening, especially since the alleged “SEC IPO quiet period” has received frequent mention in the press in recent months.

I also sent this information along to a few other people I thought would also find it interesting.

About two months ago I viewed a financial program on CNN which among other subjects, briefly discussed the infamous IPO “quiet period.” A senior Bloomberg financial analyst pointed out that the alleged SEC mandated IPO “quiet period” is entirely myth.

According to popular belief, the SEC quiet period is touted as an SEC requirement that is mandated by SEC regulations which require the parties involved in the IPO to remain silent for a specified period of time with respect to almost anything pertaining to business being conducted by the involved parties, their respective business entities, and the IPO itself.

The Bloomberg analyst explained that there are no SEC regulations which mandate and regulate this so-called IPO quiet period. The analyst pointed out the SEC does require all parties involved in an IPO to fully disclose all relevant business information in a responsive fashion at all times, including during an IPO, but to do so in a responsible way that fully and accurately represents all the related business facts without hype, speculation, manipulation, etc. That is, “just the facts, ma'am.”

Being curious I directly queried the SEC myself. Below is the question I posed followed by the email response from the SEC:

[My question, submitted by email:] What is the truth of this alleged SEC mandated IPO quiet period? Is it a myth, a law, rule, regulation or recommendation? What are the factual details? Please!

—Steven M. Ward, sward@cfa.harvard.edu

[The SEC email reply to my query:]
Re: Does the so-called “IPO Quiet Period” SEC rule really exist?

Mr. Ward

Thank you for contacting the SEC. Our Division of Corporation Finance provides the information below in order to explain what is a “quiet period.”

Al Lapins

In Securities Act Of 1933 release no. 5180 (Guidelines for the Release of Information by Issuers Whose Securities are in Registration - August 16, 1971), the Commission has emphasized that there is no basis in the securities acts or in any policy of the SEC which would justify the practice of non-disclosure of factual information by a publicly held company on the grounds that it has securities in registration under the Securities Act of 1933. Disclosure of factual information in response to inquires or obligations under the antifraud provisions of the securities acts at a time when a registered offering of securities is contemplated or in process, can and should be done in a manner that will not unduly influence or facilitate the sale of securities in the proposed offering.

It is incumbent on issuers to establish internal procedures designed to avoid problems relating to the release of corporate information when in registration.

[end of SEC email reply]

SCO FoxBase

I would like to suggest that we pester SCO into releasing SCO FoxBase+ to the Open Source Community. Or, port it to Linux, Please!

As an old xBase hound [What is the “PIP” Command], I made a living coding apps in SCO FoxBase+ running on SCO Xenix/Unix. I was a SCO beta tester and had FoxBase beta #2. In about a week I had six users and two printers hanging off an Arnet I/O box attached to an IBM AT running SCO Xenix-86. We hacked SBT modules and made them multi-user. Guess what - that client still runs SBT today.[Yeah, Foxpro 6.0 on Netware] And, the code evolved right along.

I still have a few 10yr+ old apps running on SCO Unix V on Fox. Linux would be an ideal OS for FoxBase. Fox is slim, trim, and really fast. It would make a perfect “back end” for a web transactor too. If anyone has some info, or knows how to get some traction with SCO, let me know.

—Andy Thornburg, athorn@midwest.net

So Many Pictures of Linus

I always look forward to my monthly *Linux Journal*. With great articles, lots of helpful hints, I even appreciate the ads specificly for Linux hardware. I however am wondering about the infatuation with Linus. Yeah, I realize how important the fella is to Linux and he seems to be a smart human being but I counted 9 pictures of him in this last issue. Some that bordered on a if I dare say it a “Teen Beat” kind of look. (Notice the almost center fold like picture of him sitting -all you needed was the bear skin rug and fire place).

Anyway keep up the great magazine,but please find other people to highlight fifty billion times a year besides Linus. Better yet maybe a person of color. That would be a really cool, novelle and trend setting kind of thing to do. There are people of color in the high tech industry . Many that I am sure who would or who have contributited to linux.

—root, wsg@seanet.com

Comment about a 'best of technical support' response...

In the nov 99 issue, the best of technical support covered a 'multiple authorized users' q&a.

The gist of it goes (or my interpretation of it):

system uses lilo, don't want regular users to access lilo.conf which contains authorization passwords, don't want students to gain priviledged access to the machines, lab crew to access but not modify it..

My solution to this problem is simple....

```
chown root:root /etc/lilo.conf
chmod 400 /etc/lilo.conf
```

As root you have access to it when you reinstall the boot loader. The permissions are set up to read only by root and no one else which is the best you can really do. If you have access to the root user, you can

change the permissions to writable and mess with the file. Option 1 of using a floppy is just as insecure as an unscrupulous person could copy the file to the h.d. and mess with it before reinstalling lilo. Option 2 creates a static password which makes it more difficult to change the password if it has been compromized.
My solution may not be much better, but it at least makes it more convenient and addresses the original problem of users other than root gaining access to the file. If a users gains root access, then you have other security issues to deal with. If the lilo/bios password has been compromized, it is easier to modify:

```
  chmod u+w /etc/lilo.conf
  vi /etc/lilo.conf
  chmod 400 /etc/lilo.conf
  lilo
```

—Paul-Emile Gaudet, paulgaudet@home.com

Re: Synchronizing Clocks

With respect to the November issue of *Linux Journal*, John Morley asked about how to synchronize his clocks. I thought I'd provide my $.02. If this should be sent to the letters section, maybe you could forward it for me.

I've got two Linux boxes and an SGI O2. One Linux box is connected to the internet via a cable modem. The other two boxes are on my private LAN.

I installed xntp3-5.93-2 from the RedHat CD-ROM (RH 5.1). It was quite simple to set up. I found three sites who were level 2 ntp servers on the internet, spread out around the country. Each of these sites are supposed to be synced up with level 1 ntp servers which, in turn, are supposed to be synced up with an atomic clock site.

I sent e-mail to the administrators of these sites and told them I was going to be using their ntp services (not always necessary). Then I added these servers to the /etc/ntp.conf file.

I used chkconfig to enable xntp on boot up and the gateway system was now synced up.

On the gateway machine, I modified /etc/rc.d/rc.local to have the following lines:

```
     echo "Starting timed:"
     echo "  /usr/sbin/timed -n local-net -M -F gateway-host"
     /usr/sbin/timed -n local-net -M -F gateway-host
```

Where: *local-net* is a network I defined in /etc/networks which is the netmask of my local network. *gateway-host* is the hostname of the gateway host.

This starts the timed daemon as the master (since the gateway machine is locked to the ntp sites, it should have a reliable clock so I want it to be the master). The -n option keeps timed on the local network so it doesn't try to respond to time requests on the internet. The -F option tells the timed daemon to trust only itself.

On both of my other machines, I started timed the same way, by adding the same lines to the /etc/rc.d/rc.local file except with the following options:

```
     /usr/sbin/timed -n local-net
```

This makes the other machines slaves to the master. Now, all the machines are synced to the gateway machine which is synced to ntp servers which are synced to an atomic clock.
I don't know if others are interested in this, but you're welcome to print it if you want.

—Chris Carlson, carlson@CX31480-A.msnv1.occa.home.com

Feedback on November issue

This issue was generally quite good, but I was quite disappointed in some of the articles:

- the Linus interview was interesting in covering his background, but had too little technical content - I'm more interested in what Linus does and how he thinks than how he grew up!

- the MySQL article was excellent, giving a good overview of why it is how it is - I would like more detail on why they omitted transactions, which are IMO one of the key reasons to use a DBMS!

- the EMU article was excellent, more like this please!

- the client/server and diald article was also very good - I hadn't realised some of the limitations of diald before.

- the VA workstation review was OK, but please put basic material such as the intro to L1 and L2 cache and SDRAM in a call-out box! Most PC users know what these users are, never mind Linux users! These are probably useful for some people, but best done in a separate box.

- the DB2 review was so short it should not have been published - it hardly reviews any of the features of the product, and reads as if the writer ran out of time - it should have been held over until completed.

- the RAID controller review was great, exactly what's needed.

Overall, I think you could benefit from more stringent editing - stories that relate real world experience or investigate products in depth are preferable. The ones I find really fascinating are where someone has applied Linux in a real world environment, learning about Linux and project problems on the way.

I am now a subscriber BTW, hope you go from strength to strength!

—Richard Donkin, rdonkin@Orchestream.com

Interview with Linus

I find it odd that I would write to a computer journal about religion, but then I find it odd that a computer journal would express opinions on religion.

Marjorie Richardson and Linus Torvolds seem to me to completely misunderstand what separation of state and religion are about. It is NOT, as they imply, that religion, or religious views, are not to play a role in politics. Rather it is that the US government was not meant to establish any official religion or denomination.

Many of the founding fathers of the USA were deeply religious men, and their religious values played a central role in the writing of the constitution. The idea that religion should be kept out of politics would have been strange to them. Many people came to the new land to escape religious persecution - persecution from denominations that had become part of the government of the countries they came from. They wished to found a land were people could be free to express and practice their religious beliefs, without interference from the government. But today that notion has been reinterpreted to say that the government should proceed without interference from religion.

As a native of England, I find America's deep connection to religion endearing. Linus is living in the USA because the economy and opportunities have created an environment in which interesting companies like Transmeta can be formed. I believe that this strength of America comes from its religious background, and that if America begins to ignore its religious roots, that it will cease to be the power that it is today.

I would also like to take the opportunity to compare the founding of America with another event of that time, the French Revolution. The French Revolution was more atheistically inclined than the American Revolution. While the American Revolution has lead to a powerful nation that has in many ways championed the rights of men and women over the whole world, the French Revolution was taken over by the despotic Napoleon, and followed by many bloody revolutions, lasting over a century.

—Stephen Montgomery-Smith, stephen@math.missouri.edu

Wasted Opportunity

The interview with Linus Torvalds (November 99) was a waste of time, ink and space. Who really cares about his politics, religion, family or anything else that has absolutely nothing do with Linux? This was an opportunity for probing his mind about anything related to Linux instead we are treated to soft questions and New Age psycho-babble. This 'interview' passes for real journalism? I thought this magazine was a serious forum for the Linux community. I was mistaken.

—Gopi Shah, gshat@aqmd.gov

---

lilo/silo passwords (in BTS)

Hi,

I just noticed your question about LILO/SILO passwords in *Linux Journal*.

This is actually trivial to do; the ease follows from the fact that only root runs LILO. (I assume the it would be similar for SILO.)

Based on the above assumption, the following commands would make lilo.conf unreadable by normal users:

```
     chown root /etc/lilo.conf  # probably a no-op
     chmod 600 /etc/lilo.conf
```

Assuming that both the hardware is secured (BIOS passwords, locked covers etc.) and the “restricted” and “password” LILO options are set, this is IMHO better than “hard-coding the password in /sbin/lilo and making it mode 0700”.
—Ambrose Li, ai337@freenet.toronto.on.ca

upFRONT

Various

Issue #69, January 2000

Stop the Presses, *LJ* Index and more.

STRICTLY ON-LINE

The Collaborative Virtual Workspace by Stephen Jones gives us a look at office meetings of the future: held over the network in a virtual meeting room. Mr. Jones' current project involves developing this virtual space so that it gives more meaning to social interactions and allows participants to pass around documents and pictures. The software for this project is being made freely available.

AppSwitch 2000: Network Switching with Ada by Ann S. Brandon relates how this product was developed using Linux, then converted to GNAT. All programming was done in Ada rather than C, and Ms. Brandon tells how and why management made these controversial decisions. This article is as much a product review as it is a salute to ADA and Linux. The AppSwitch analyzes network application traffic and enforces quality of service policies automatically.

Writing an Alphanumeric Pager Server for Linux by Eric Max Francis gives us all the information we need to do just that. Mr. Francis covers protocols, profiles, devices, filters, server and clients, then tells us how to put all the pieces together whether we support a single or multiple devices.

Linux and Banking by Josip Almasi is a “Linux Means Business” column about a small bank in the country of Croatia. Mr. Almasi tells us how they came to choose Linux as their operating system, and the benefits, such as GNU tools, which Linux has brought with it.

Perl Annotated Archives by Paul Dunne is a book review and delivers just what book reviews should: a look at what's in the book, and information to help you decide if it is worth reading.

Core PHP Programming: Using PHP to Build Dynamic Web Sites by Allen Riddell reviews this book on the popular web scripting language PHP. Everything you need to know, or a waste of time? Read the review to see.

LJ Index - January, 2000

1. Place of Swedish among the languages spoken in Linus' home: **#1**
2. Percentage of Finns for whom Swedish is the mother tongue: **5**
3. Length in hours of Linus' honorary doctoral award ceremony at the University of Stockholm: **3**
4. Number of times the two mountain bikes in Linus' garage have been used in two years: **2**
5. Elevation of Santa Clara, where Linus and his family live: **94 feet**
6. World to which Linus says he sometimes thinks Silicon Valley belongs, due to its absence of good electronic banking: **3rd**
7. Hosts per 1,000 people in Finland: **52**
8. Position of Finland in hosts per thousand among the 20 most industrialized countries: **#1**
9. Number of U.S. cities that have achieved greater than 50% Internet penetration: **5**
10. Internet penetration of Washington, D.C.: **59.9%**
11. Washington, D.C.'s position among most-penetrated Internet cities: **#1**
12. Position of Pittsburgh, PA: **#64**
13. Internet penetration of Pittsburgh: **30.8%**
14. Household debt as a % of income in 1999: **98**
15. Household debt as a % of income in 1998: **80**
16. Amount borrowed by U.S. households to fund stock purchases in 1999: **$179 billion US**
17. Increase in borrowing for stocks over previous five years: **3 times**
18. Room capacity for the Linux for Suits panel discussion sponsored by *Linux Journal* at Fall Internet World featuring Linus Torvalds: **350**
19. Approximate number of attendees at the event: **400**
20. Number of pages in the January 1999 issue of *LJ*: **100**
21. Number of pages in the December 1999 issue of *LJ*: **132**
22. Number of complaints concerning the discussion of religion in the November *LJ* interview with Linus Torvalds: **7**

23. The latest expected release date of Windows 2000: **2/17/00**
24. Approximate number of employees at *Windows Magazine*: **35**
25. Number of employees working on *Linux Journal*: **14**
26. Average number of days the male emperor penguin fasts during courtship and incubation: **90-120**
27. Percentage of body weight lost during fasting: **41%**
28. Percentage of time spent asleep during fasting: **70**

Sources

- #1-3 and 6: Linus Torvalds
- #4: *San Jose Mercury News*
- #5: Career.com
- #7-9: Matrix Information and Directory Services
- #10-13: Scarborough Research
- #14-17: *Business Week*
- #18-22 and 25: Jason Schumaker, *Linux Journal*
- #23: ZDNet, October 26, 1999, Mary Jo Foley
- #24: *Windows Magazine*
- #26-28: *The Penguins* by Tony D. Williams

WHO'S SURPRISED?

Gains by Linux are old (if not Red) hat by now. Still, it's gratifying to see the numbers. Here are the latest (April 1999) from The Internet Operating System Counter (http://leb.net/hzo/) in graphical form.

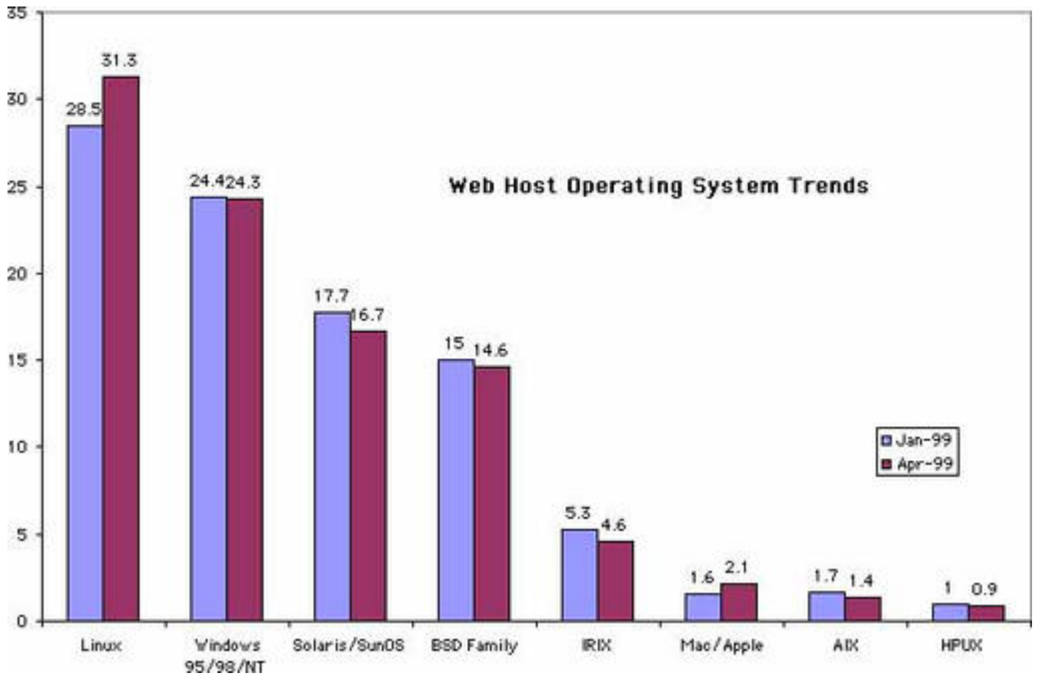
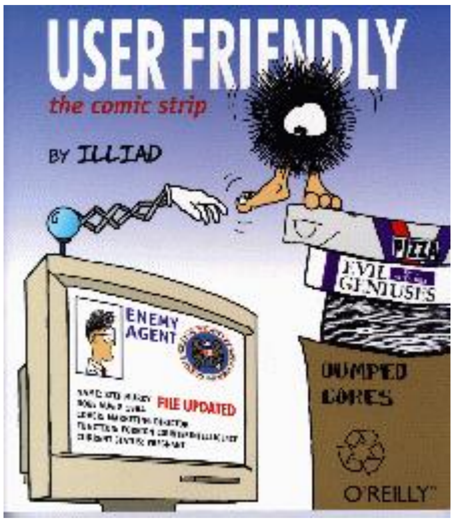


Figure. Linux Numbers

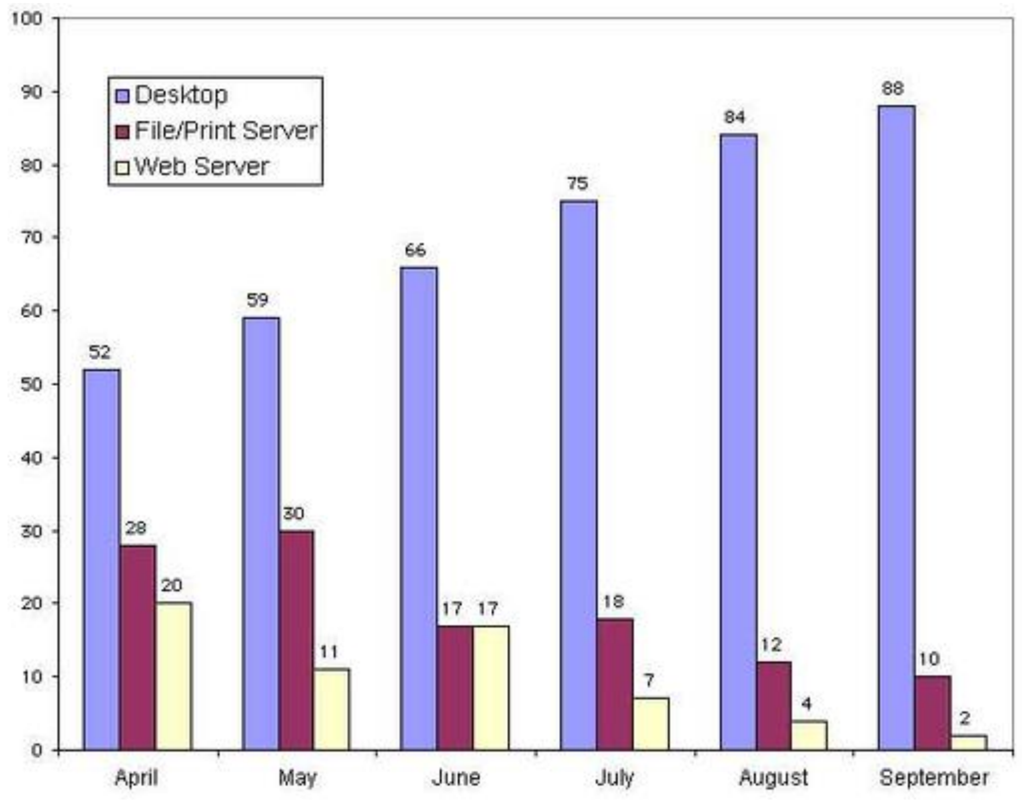


User Friendly

O'Reilly & Associates has published their first cartoon book, *User Friendly* by Illiad. This is a collection of the very same cartoons printed each month in *Linux Journal*. It offers funny, offbeat and original comic strips to tickle the funny bone of all computer users, and Illiad is a supporter of Linux, too. Get your copy today.

TO THE DESKTOP! AND BEYOND!

Another quarter, another major advance in Linux' move to the desktop—at least if we judge from service calls. Last September, we reported that Linuxcare noticed a 27% increase in the number of desktop incidents, while service calls for file, print and web servers went down. Now we have the third-quarter numbers, and the trend continues. To show the dramatics, here's a graph.



Type of Service Calls Graph

Source: Linuxcare

EXAGGERATION IS CHEAP

The press loves a David & Goliath “war” of any kind. Bill Gates has played the Goliath role for quite a while, of course, but the Davids come and go. For most of the late 1990s, David was played by Netscape's Marc Andreessen. Now Linus Torvalds has been cast in the David role.

Thus, the headlines read like a play-by-play fight between Linus and Bill, or Linus and other big/bad tribal chieftains. Here at *Linux Journal*, we found ourselves supplying material for this sports story when we hosted “Linux for Suits: The Linux and Open Source Executive Forum” at Internet World on October 6 last year.

Headline:

Torvalds swings at Linux wannabes: Linux inventor dismisses moves toward open source by Sun, Microsoft... —ZD Net story, October 6, 1999

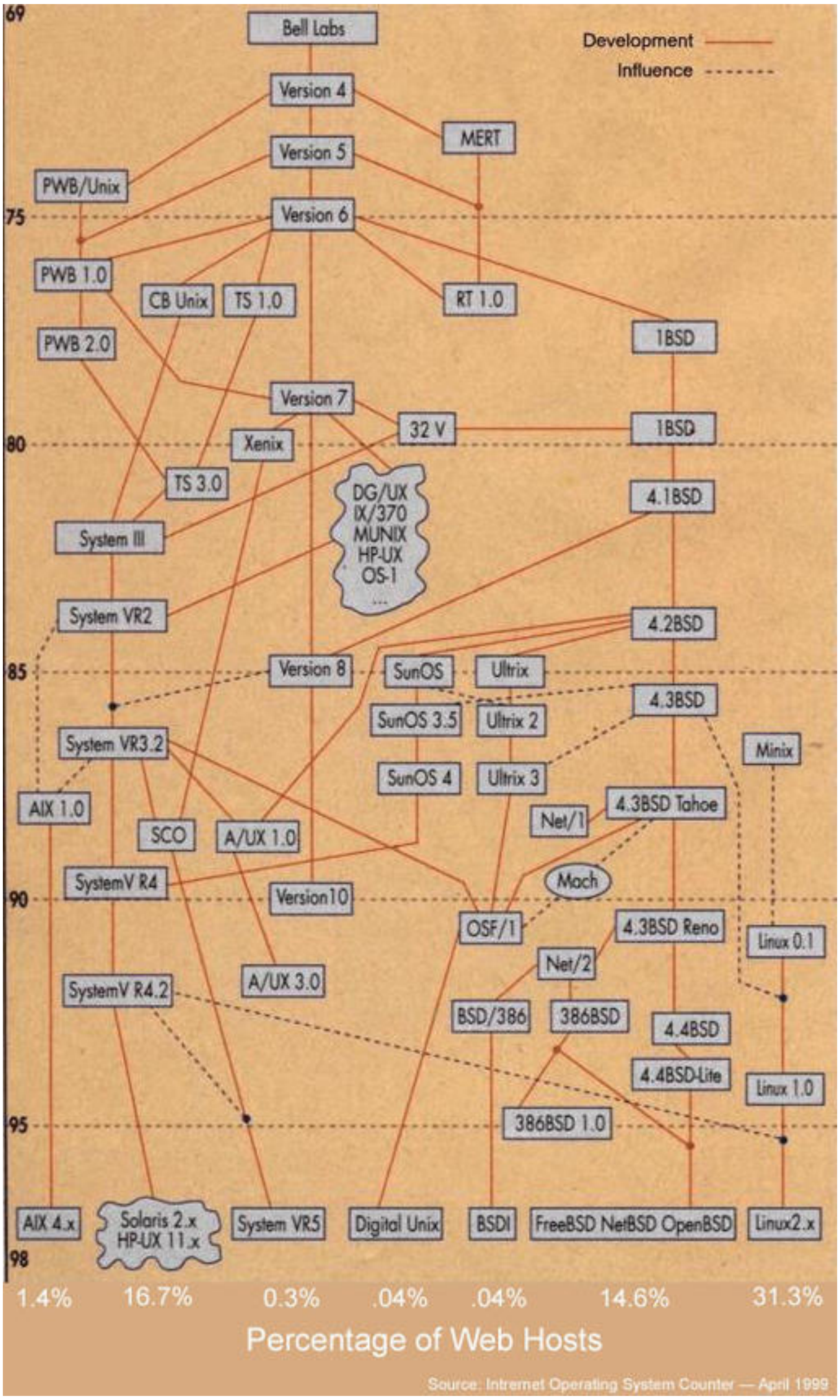
Actual Words:

Question: Microsoft has been talking a lot about opening up their own code. Whether or not they actually do that is a big question; but if they should do that, what does it mean for Linux?

Linus' Answer: I'd like to start off by saying talk is cheap, and I think this is a fairly theoretical question. However, as a theoretical question, I'd be more than happy to see more and more people open up their source. I'm not convinced, for example, that the Sun community license is a very good license. But it still makes me very happy to see that Sun is opening up and making their knowledge available to others. And if Microsoft were to open up, I'd be more than thrilled. I don't think it is very likely, or if it does happen, it will most likely be in niche markets. People think there is more of a Microsoft bigotry in the Linux community than there really is. I'm more than happy to use Microsoft products. It is just that I am selective about the products I want to use. For example, I've always liked PowerPoint, and I've always thought that Visual Basic was a good product. It's just been hard to use them because I've always thought the platforms they run on aren't good enough. I used to do my slides with PowerPoint for the longest time with a Windows Installation package. This is true, I find of all the developers I'm in contact with—that maybe we take up Microsoft as a bad example in specific areas, but we're not as anti-Microsoft as the press makes us seem.

Find the full transcript here —or listen to the whole day's proceedings at the ON24.com site. Go to www.on24.com and find it among the Linux search listings.

THE UPSTART AT 30



UNIX Systems Diagram

One of the old objections to UNIX was there were so many variants. The fact that lex, yacc, vi, troff and other commands ran uniformly across all of them didn't cut the ice. The plate of spaghetti here represents most of the variants and the cross-fertilization. Minix shouldn't be free-standing: it arises out of V7 from Bell Labs. I miss Dell UNIX, the best n86 version of SVR4.

The really important thing is the fact that every one of these streams is used for web hosting, and that the UNIX+Linux systems total a remarkable 64% of all sites. In other words, just about two-thirds of the Web is powered by UNIX or Linux.

Not bad for an upstart.

—Peter Salus

WORK STILL CUT OUT

After I took part on a panel at the FreeBSD conference in October, an attendee invited me to test the honesty of both *Linux Journal* and its community by looking at exactly which operating systems hosted the most popular Web sites. So I did. The server and host columns below contain exactly the data yielded by Netcraft at its “What's That Site Running?” page (http://www.netcraft.com/whats/) on October 26, 1999.

The source list of top 25 sites (in the U.S.) comes from Media Metrix, which is the primary source of web-site popularity figures.

Indeed, the results look good for BSD (5). They look better for Solaris (10). They perhaps look best for Bill Joy (18), whose genius is behind the Berkeley, Digital (3) and Sun breeds of UNIX. Although only Real Networks and Angelfire were found to be running on Linux (2), Netcraft informs us that other Linux sites include Deja.com, eToys, Go2Net and The British Royal Family (we knew those folks had taste).

Since we're being utterly fair here, Microsoft's NT (5) doesn't do too badly, either.

Next, the necessary disclaimer. “There's no sure way to tell what OS a host is running,” says our top tech wizard, Dan Wilder. “Indeed, there are lots of ways to fool people about what host is running. OS names are easily spoofed. We used to spoof them here at SSC, to foil hackers.” To find a fun hack on the supply side of this information, look at the server finding for Real.com, below.

Consolation: as we see elsewhere in this section, Linux is gaining rapidly overall and now hosts about one-third of all web domains (according to The Internet Operating System Counter).

—Doc Searls

Table

ATLANTA LINUX SHOWCASE

ALS '97 had about 20 vendors and a few hundred attendees.

ALS '98 was like reverse time travel. It had both the atmosphere and the feeling of USENIX conferences before the mid-80s, with about 40 vendors and over a thousand attendees. (Atlanta USENIX in 1986 had 1200 attendees.)

Slashdot Booth

ALS '99 was still full of talks and demos and corridors and dinners. But now, after only three years, there was active cooperation between the Atlanta Linux Enthusiasts and USENIX: it had two days of technical tutorials; the show floor was over double the size of what it was in 1998; and there were 3,000 attendees.

The discussions were intense; the atmosphere was friendly; more got done in the hallways, bars and lounges than in the sessions. There was a feeling of intellectual ferment.

Eric Raymond was there. Maddog was there. This was the Linux geek event not to miss. My favorite hardware display was Compaq's Beowulf cluster; software was from Hummingbird Communications, Ltd., (http://www.hummingbird.com/), a Linux version of their Fulcrum search server and OpenSales (http://www.opensales.com), a neat e-commerce solution.

User Friendly Booth

But I have to admit that the product of choice was Raymond's *The Cathedral & the Bazaar*, just published by O'Reilly (see my review in this issue of *LJ*).

Marc Torres and his crew did a splendid job. I hope they invite me back next year.

—Peter Salus

DISTRIBUTION WATCH

A while ago, *Linux Gazette* editor Mike Orr and I considered founding an organization dedicated to the proposition that each user should have his or her very own distribution of Linux. That would make for several million new versions of Linux, a number we're steadily approaching. Here is one of the newer, more interesting distributions; quite exciting, n'est-ce pas?

kha0S—better living through extreme paranoia—aims to be the most secure Linux system ever. It's so paranoid, in fact, that right now it has stateside distribution "issues" on account of cryptography protecting privacy which is disagreeable to a certain government. kha0S is in version 0.99 right now, very close to the magical 1.0. Although it has no pretension of displacing market leaders (kha0S is expressly *not* a commercial project), it is vastly more secure and may raise standards across the board while filling its particular niche. This is a project not necessarily for people who need security, but for people who are intellectually fascinated by it. The kha0S team has surprisingly strong credentials for a fringe project like this, so they're probably going to turn out something rather impressive. It's nice to see brain power for Linux existing in areas which aren't Linux-specific, such as cryptography.

Kha0S employs a low-level-up approach: each component is tested to make sure it is completely free of security holes, then the kernel itself is hardened. Security packages to be integrated by default include the Cryptographic File System, Kerberos, IPsec, IPv6, VPN, ssh/lsh and others. The Cryptographic File System is one of the polishing stones of the kha0S project and should provide transparent encryption of files on the system (hopefully encrypting swap as well).

I suppose there's a limit to how secure a system can be, but the fun is in the project anyway, isn't it? It's a lot like hacking in reverse (or cracking in reverse, if you prefer), although I wonder what these folks were up to in their teenage years while becoming such security experts.

For more information on this exceedingly cool distribution, check out http://www.kha0S.org/, especially if you're very good at breaking encryption schemes (they may want you). I wonder how quantum computers will affect the kha0S project—I probably won't have to worry for a very long time.

—Jason Kroll

KEYBOARDS—>CAN'T GET ENOUGH OF THOSE PENGUINS!

The first time I saw Windows, I thought it was a joke; I actually laughed. I didn't look at it again until I bought a PC to install Linux. That's when I noticed something horrid on my keyboard: little keys with Windows on them. Well, now Linux has its own keyboards—magical custom hardware—without those Windows logos.

Linux CoolKeyboards' flagship product, featuring the uber-ubiquitous Tux, is a keyboard appropriate for mainframes. It is serious, heavy, and it clicks, a genuine “Old Skool” design from the days when machines were machines and coders ate pizza (wow, it seems like only yesterday). It is slightly oatmeal in color, with a 70s brown hard plastic dustcover (which I recommend removing or decorating) and *lots* of keys. The keys are quite perky, among the least gummy in history. The feel is a question of taste: do you prefer, for example, Mac keyboards, soft-touch keyboards, laptop keyboards, etc.? If you believe the only real keyboards are ones with key-punch, microswitch action, this is just your thing. Ah yes, the Linux point of this keyboard is the penguins—two penguins and a Linux logo—replacing the Windows keys. As an added bonus, all twelve function keys are labelled twice (once on the keys and once on the keyboard, so you can bring it to parties as a conversation piece) just like in the “old days”, but then again there's a Euro mark on the 5, a sign of changing times (along with our dear penguins). I'd like to see non-oatmeal colors and a more specialized Linux setup (change caps-lock to control, etc.), as well as rubber on the feet which are lethally sharp.

Consideration for the DVORAK layout would also be nice. All in all, though, CoolKeyboards is on to a good thing (and trying to patent it for whatever reason). Now, on to smaller things.

Happy Hacking Keyboard Lite—you've seen those very happy people; this is why they're so happy—it's a *tiny* keyboard, with only 60 keys, optimized for Linux users. This means that control replaces caps-lock, escape is next to 1 where ~ usually is, delete/backspace is large, the number pad has been removed, and the weird keys (function, cursor, page keys, etc) are accessed with a special Fn key (as found on laptops). Specialized configuration of keys can be achieved by way of dip switches and a paper clip. Few people will realize how tiny this keyboard is until they see it; the keys are full size, but the board itself is even smaller than those found on laptops. Strangely, everything fits! This has got to be the cleverest keyboard design going, especially for vi and Emacs users. It's much more elegant than a traditional keyboard, but it also relies heavily on the right hand. The key action has been described as “silky”, and is actually very soft, and again not gummy. The keys are quiet enough, though not quite as perky as on a laptop or the aforementioned board. As for layout, PFUCA says DVORAK will be a critical issue for the power user, but there are no plans for one as yet. PFUCA is delivering a black HHKL soon (what about translucent purple?), which would be nice paired with a flat-screen monitor and a tiny computer like a Netwinder (ah, StrongARM). Good for apartments where space is at a premium and for people who like small things. No Windows keys, and no penguins. (Try your local aquarium or zoo for penguin stickers, and go on a rainy day so there aren't so many people scaring the animals.)

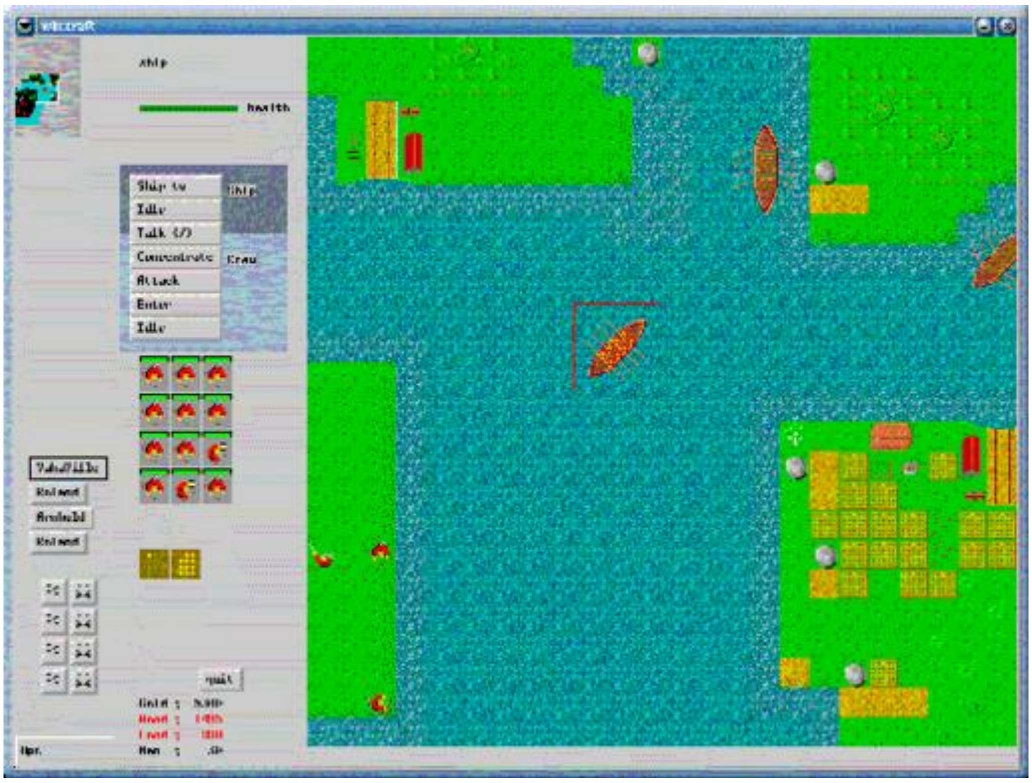
If you've worn out your space bar from too much ZBlast, not to mention what you've done to your cursor, (**ALT**), and **CTRL** keys on account of Quake, maybe it's time for a new keyboard (and/or a joystick).

—Jason Kroll

GAMES FOCUS

In keeping with the theme of world domination, it's time to indoctrinate Linux users in our quest by beginning their training in various ways of taking over the world. Strategic conquest, of course, is horribly addictive, and ever since the first world-conquering games started appearing several years ago, take-over-the-world power gaming has practically taken over the world. Indeed, it seems as if the two genres of computer games these days are strategic war games and 3-D Wolfenstein descendants. Since we're so sophisticated, let's have a look at the former.

Craft—The Vicious Vikings



One day can actually change the course of your life. For example, a bit too much to drink, a quibble with your chief, and suddenly you find yourself out to found a new nation with two of your drinking buddies. Craft author Uwe Breyer adds, “since you are obviously even too lazy to work, your companions vote you to be King.” So that's how it happens—the path to world domination begins with three drunken Vikings. Craft features up to four players, human (with network support) or computer, wandering about in a world which is a bit Warcraft, a bit Civilization, but rather less cluttered and faster-paced, with a minimal learning curve. You've got access to knights, archers, scouts, workers, merchants, scientists and pawns, as well as town halls, farms, camps, mills, smiths, universities, forts, markets, docks, ships and catapults. All in all, you can really go nuts killing things if you like (and if you want to win). This game is a Linux gem, fast-paced (real-time as opposed to turn-based) and it doesn't degenerate as quickly as so many strategic conquest games. Look for it at http://borneo.gmd.de/AS/janus/craft/.

FreeCiv—'Cause Civilization Should Be Free…

Open-source developers are truly amazing. This time, they've managed to create a free and improved Civilization I/II for Linux that's more fun than the originals. The graphics are really nice, a bit on the dark side, and the game feels more serious. FreeCiv supports AI and network play with 32 nations; the newest release should support an infinite number. Maps can range in size from very small to very large, and one could, if so inclined, devote a weekend or longer to a massive clash of civilizations over the Net. There are 47 different units available, as well as several different tile sets, so that you can choose your graphics (even in different sizes). It's a typical client-server model, easy to set up and play. Here it is, only one download away—you don't even have to buy it: http://www.freeciv.org/.

Anachronism—Magic, Blood and Action

Anachronism is Nikos Vasiliou's contribution to Linux gaming, made when he felt Linux didn't have enough games. If steep learning curves are discouraging, check out this one. There's very little to worry about except for troops and killing, so you can concentrate on war. It has rendered graphics (characters and terrain), two civilizations with ten armies, multiple scenarios (and a map editor), multiple player support, sound effects and music. The happypenguin.org site says, “just take your troops and slaughter.” Quite right. Find it at students.ceid.upatras.gr/~nbasili/anachronism.html.

In Development

Developers who hope to contribute to global conquest might want to get in contact with the Boson development team. Released under the GPL, Boson is a real-time strategic war game with absolutely fantastic graphics, only it's not quite playable yet. Right now the developers are looking for more people to help with graphics, documentation and the like. It looks really neat, especially the machinery: http://aquila.rezel.enst.fr/boson/.

TUD (The Urgent Decision) is another very promising game which may need some developers. It's another take-over-the world game, but it's network-playable and has a very military approach. For example, if you want to lead Viking settlements or win a victory of civilization by controlling all elements of society, Craft and FreeCiv are for you, but if your brilliance is limited to the military sphere (or you just like it better—fast planes, heavy machinery), TUD might excite, as it's quite mechanical and very close to that magical 1.0 version. Check out the web page, maybe it's just the project you're looking for: www-ti.informatik.uni-tuebingen.de/~thiele/tud.html.

Conclusion

We all know we're already geniuses (just look at what OS we use—our IQs must be over 2000), so now we're ready to go out and conquer the world (a tad gruesomely at that). There are dozens of strategy games for Linux; these are only a few highlights. All these games and more can be found, as usual, on http://happypenguin.org/, or you could try http://linuxgames.com/ for general gaming information.

—Jason Kroll

STOP THE PRESSES: Red Hat Buys Cygnus

On November 15, Red Hat announced they were buying Cygnus Solutions for a reported $674 million (US) in a stock-for-stock merger. I don't think anyone was surprised to hear that Red Hat was acquiring a company. Ever since the IPO, people have wondered who it would be and when it would happen. After all, what else were they going to do with all that money? Plans consisting of building a bigger and better portal and nothing else is certainly not the way to inspire confidence in your investors. However, I for one was certainly surprised that Cygnus turned out to be the company Red Hat bought. Of course, I had recently heard the rumor and hoped it would prove to be false.

Cygnus Solutions has been in business for ten years now. They are successful. They market good products based on open-source software, mainly the GNU tools such as gcc and gdb. Recently, they have marketed an integrated developer tool, called Code Fusion, which won our Editors' Choice Award for Best New Application for the Developer. They have always been vendor independent, ensured all their products ran on all Linux distributions and given back to the community. Their improvements to GNU tools have always been open source, giving everyone the chance to benefit from the advances rather than just those who could afford to pay for them. For these reasons, they have held a high standing in the regard of all those in the Open Source community. They have held this same high regard in the business community, because they have shown that you can indeed make money from open-source software.

Now that Cygnus is owned by Red Hat, what will happen? Will Cygnus products become vendor specific? Will upgrades always be available to Red Hat first? Will they continue to give back to the community? At least for this last question, I don't think we have cause to worry. Red Hat has been good about giving to the community too—I don't think that is going to change. But I do worry about the first two. Now that Red Hat has stockholders to report to, changes are inevitable—a definite market edge must be demonstrated and profits must be made. Red Hat's commitment to the Open Source movement is going to be put to the test.

They have bought a good, strong company, one that will help them show profits and continue to grow. It was a good business decision on their part. Still, I wish they hadn't made it so soon. I have no doubt that Cygnus is only the first company to be bought by Red Hat—there will be others. I feel it would have been best if they had first acquired a company that was not as strong; that is, one that would have benefited more from the merger. Let's not kid ourselves; Cygnus doesn't need Red Hat to be successful.

This merger looks like a bid for money and market domination, not a step taken to further the future of free software. Still, only time will tell if the Linux community has actually been slimed or if it just feels that way.

—Marjorie Richardson

VENDOR NEWS

Corel announced an alliance with **PC Chips** to bundle Corel Linux with more than 20 million PC Chips motherboards. PC Chips will also ship Corel WordPerfect 8 for Linux and WordPerfect Suite 8 OEM for Windows with its motherboards.

SuSE, a distributor of the Linux operating system, announced an agreement with IBM to distribute, market and support selected IBM e-business software solutions for the Linux platform. SuSE is working with IBM to meet the demands of its customer base for software products running on the Linux platform. Initially, SuSE will concentrate on the following IBM and Lotus technologies for the Linux platform: Lotus Notes/Domino, MQSeries, Transaction Series, Component Broker, IBM Data Management Software, Websphere Family, IBM Electronic Commerce Software, VisualAge, San Francisco Framework and SecureWay.

Motorola announced the appointment of Metrowerks' two most senior executives, Jean Belanger and Greg Galanos, to strategic roles within Motorola Semiconductor Products Sector (SPS) and the appointment of David Perkins as President of Metrowerks. The changes in these executives' roles were made to expand Motorola SPS' software expertise and accelerate the integration of Metrowerks and SPS. Belanger, Galanos and Perkins all remain on the board of directors of **Metrowerks Inc.**, which became a wholly owned subsidiary of Motorola on September 24, 1999.

Caldera has announced they have chosen Multi User Solutions to be their first Authorized Support Center. Multi User Solutions has over nine years of experience in supporting the UNIX environment, and currently supports over 10,000 sites nationwide. They offer a unique solution covering both OpenLinux and hardware support, and thus will be able to give Caldera customers the extensive coverage they need in phone, e-mail and on-site Linux and hardware support.

Ariel Corporation announced that the source code for its Linux remote access drivers is now freely available to the public through the Open Source Initiative. The drivers make it easy for service providers, OEMs targeting service providers and corporate enterprises to add high-density V.34, 56K and ISDN remote access to a broad range of PCI- and CompactPCI-based Linux systems.

Linuxcare announced an agreement with **TurboLinux** to partner on enterprise support, services, training and hardware certification and testing for TurboLinux software, including TurboCluster Server. Linuxcare also announced it will be providing technical support in Japanese for Linux software developers at NEC Software.

esoft Inc., a developer of Linux-based Internet appliance solutions for small- to medium-sized businesses, announced a distribution agreement with IT Resources Pte Ltd., a Singapore-based distributor. IT Resources will distribute the TEAM Internet family of products and will be selling them to value-added resellers throughout Singapore and India.

Intel Corporation announced it will bundle the Red Hat Linux operating system with its server platforms marketed through its recently created Internet Service Provider program. The inclusion of Red Hat Linux with Intel's ISP program offers customers the power of open-source software in the demanding Internet server environments in which it performs best. Comprehensive services and support will be provided by both organizations.

Activision, Inc. and **Loki Entertainment Software** announced they have joined forces to bring more PC and Macintosh games to the Linux platform. Loki will be developing, publishing and supporting the Linux ports of five Activision games, including Heretic II and Heavy Gear II, over the next two years.

National Semiconductor Corporation has appointed **INFOMATEC AG/IGEL Technology Labs** to serve its Asia Pacific customers. INFOMATEC/IGEL will develop Linux-based firmware to port to National's set-top box and thin-client platforms, providing highly integrated solutions for the information appliance market.

TurboLinux announced that **Sanyo Electric Co.** in Japan will be using TurboLinux as the base operating system in 20,000 Newve medical workstation products expected to ship over the next four years.

SMC Networks, a manufacturer and supplier of LAN products for small- and medium-sized networks, announced they will be providing a complete TurboLinux workstation solution to North American resellers and end users with every SMC single-pack network adapter purchased.

SCO and **TurboLinux** announced a worldwide services initiative to provide TurboLinux customers with Linux Professional Consulting Services from SCO. SCO will support TurboLinux customers in planning, cost analysis and deployment of their systems. SCO will also develop specific service offerings for the newly announced TurboCluster Server.

Dell Computer announced it is shipping its Precision WorkStation series in Japan bundled exclusively with TurboLinux. Customers can also purchase Linux support services from TurboLinux through Dell's DellWare program.

Covalent Technologies, a supplier of commercial software for the Apache web server platform, announced the availability of several commercial support options for the Apache web server.

Fujitsu Ltd. announced it will be distributing Caldera OpenLinux 2.3 (and future versions) on many of its servers, including the GranPower series. Both companies will work to provide a comprehensive hardware certification program for OpenLinux products on Fujitsu systems.

Red Hat announced an expansion of its services program that will provide the consulting, and support enterprise organization's need for nearly all popular, powerful, open-source software applications. As the first step in the program, Red Hat's worldwide services group will immediately offer expanded Service Programs for popular open-source software solutions, including the Apache web server, Sendmail and Postfix.

Quote: I am basically a very lazy person who likes to get credit for things other people actually do. —Linus Torvalds (*The Cathedral & the Bazaar* by ESR)

Factoid: Pierre Laffitte, a French senator, has proposed a law that would make the use of open-source software mandatory for most government use! (Slashdot, October 28, 1999)

Joke: Windows 2000 is set to debut February 17, 2000—Groundhog's Day. If Windows sees its shadow, expect six more weeks of beta testing! —Patrick Hair, Panama City, FL.

Archive Index Issue Table of Contents

Advanced search

January 2000

Phil Hughes

Issue #69, January 2000

Phil evaluates his predictions.

Back in *Linux Journal* issue #2 handy, the title of my “From the Editor” column was “Linux 2000...” The abstract goes on to say:

> I see Linux as a progressive movement as well as an operating system. And, with any movement you need to chart your direction. To help with that charting I decided that rather than write the April, 1994 editorial I would just go ahead and write the one for January, 2000.

“Well, here we are and it's time to see how well I did. If you don't happen to have issue #2, you can check out the original article here.”

My first idea, that Linux would turn into a movement to provide affordable, reliable, multi-tasking software, was certainly right on. Looks like my growth prediction was a bit off, though. I estimated Linux would be in 100 million homes—seems a bit high.

I also implied that few would remember an old program loader called MS-DOS. Well, I think that has essentially happened. While it continues to live under Windows, not many people seem to realize it.

Another miss on my part was personal satellite stations in the late 1990s. They exist—you can buy a system that offers a bidirectional connection to the Internet—but the price of that connection is still too high to be practical for personal computing. What *has* happened is that the cost of wired Internet connectivity, most specifically DSL and cable modem, has offered significant bandwidth to many at a very low cost.

I had some predictions about *Linux Journal*. Specifically, I projected that 90% of our subscribers would receive their magazines via the Internet rather than on paper. This could have happened, if not for two reasons: most readers want the print version of the magazine, and our advertisers want you to see their ads. We do offer an electronic version of the articles to our subscribers at http://interactive.linuxjournal.com/, but we continue to print over 100,000 magazines each month.

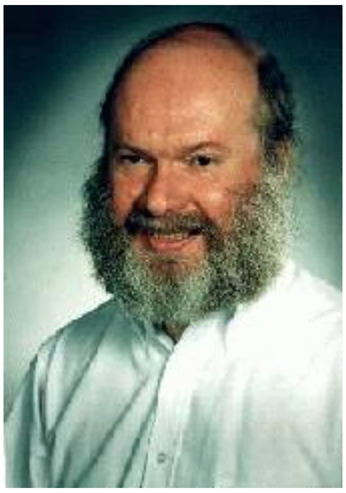
I predicted two events that would make Linux a huge success. The first was the formation of MoAmI Semiconductor, a company that produced a chip where Linux was the first OS to run native. We're still waiting on this one. Linux will most likely be the first OS on Intel's Merced chip, with VA the first vendor to ship systems. And, well, keep watching Transmeta.

The second event was Linux becoming the most popular OS in computer science classes. I don't have the numbers, but it sure seems like that could have happened.

I also predicted that NT would become a niche operating system. Did it? We don't know yet, as its name is now Windows 2000 and, well, it really hasn't shipped yet. I still feel that most corporate customers waiting for Windows 2000 will, once they see it, decide Linux is the right answer for servers.

I also predicted those old Linux developers would still be writing code or books on Linux, rather than being CEOs of multi-billion dollar corporations. Generally, this seems to be true, with some also writing articles for *Linux Journal*. Why? Because these people are doing what they want.

I'm fairly happy with how things have turned out. I took some rather SWAGs (Scientific Wild Guesses) and wasn't totally out of line. The good news is Linux is still here, you are still here, and we are still here. Hopefully, the only big arguments of the next five years will be about programming in Perl or Python and whether to use GNOME or KDE on your Linux system.



A Look at IPv6

Peter Salus

Issue #69, January 2000

With IPv6, IP addresses go from 32-bit to 128-bit. Here's why the change is being made.

Vinton Cerf and Bob Kahn came up with the original version of TCP (Transmission Control Protocol, RFC 675; December 1974) and Jon Postel with that of IP (Internet Protocol, RFC 760; January 1980) 20 years ago and more. These increased the network “address space” to 32 bits, but the structure of the ARPANET was “classless”, that is, the hierarchical distributed database we are familiar with came about only with Dave Mills' conceptualization of the Domain Name System (DNS; RFC 799; September 1981) and its implementation by Paul Mockapetris (RFCs 882 and 883; November 1983). Mockapetris' implementation was called Jeeves. BIND (Berkeley Internet Name Daemon; written by Kevin Dunlap, maintained by Paul Vixie) is currently the most-used.

Thus we achieved 32-bit addressing and a hierarchical array of classes of networks: A, B, C, D and E. There are 128 Class A addresses, each of which can have 16,777,216 unique host identifiers. There are 16,384 Class B addresses with 65,536 unique identifiers, 2,097,192 Class C addresses and over 268 million Class D groups. Class E addresses have never been available for general use.

Using this scheme, DNS allowed for about four billion hosts on 16.7 million networks. This seemed like a very large number of addresses. But the expansion of Internet use over the past decade has been explosive.

In August 1990, during the Vancouver Internet Engineering Task Force (IETF) meeting, Frank Solensky, Phill Gross and Sue Hares projected the current rate of assignment would exhaust the Class B space by March of 1994.

Classless Inter-Domain Routing (CIDR, RFCs 1518 and 1519; September 1993) was introduced to improve both routing scalability and address-space utilization in the Internet. By eliminating the notion of “network classes”, CIDR allows for a better match between address requirements and address allocation. CIDR has enabled the Internet to function while growth continues.

Even with CIDR, it was revealed at the July 1994 Toronto meeting of the IETF that the Internet would exhaust the IPv4 address space between 2005 and 2011. With several more years of experience, we can push these dates out a bit, but exhaustion will come.

The Internet has grown with the number of intranets (what we used to think of as “internal corporate networks”) and the number of different uses to which they are put (Internet radio, telephone, mobile computing, etc.).

The Toronto IETF meeting set up an “IPng” (Internet Protocol Next Generation) or “IPv6” task force, cochaired by Scott Bradner and Allison Mankin. Recommendations from that task force were released in October 1994 for discussion at the December 1994 IETF meeting. The basic goal was to have something in place before 2000, so that the time limit would not be pushed. Unfortunately, as Bradner and Mankin put it in their recommendation:

> Some people pointed out that this type of projection makes an assumption of no paradigm shifts in IP usage. If someone were to develop a new “killer application” (for example cable TV set-top boxes), the resultant rise in the demand for IP addresses could make this an overestimate of the time available.

IPv6 provides for 128-bit addressing. This is a gigantic number, larger than the estimated total number of molecules in the moon. Just how this will work is still unclear; as I write this, the new protocol has yet to be widely deployed. Among other things, going from 32 to 128 bits will entail renumbering a large number of addresses already in use.

However, it is absurd to state that address space depletion is the only driving force behind IPv6. While the address space now provided for is enormous, it's not everything. A number of other abilities “have been developed in direct response to current business requirements for more scalable network architectures, mandatory security and data integrity, an additional field for quality-of-service (QoS), autoconfiguration and more efficient network route aggregation at the global backbone level.”--IETF draft; no longer on-line.

A business or private user might well say “So what?” to this, thinking that IPv6 support for a large variety of network devices just isn't an end-user or business concern.

Over the next few years, conventional computers on the Internet will be joined by a variety of new devices, including palmtop personal data assistants (PDA), hybrid mobile-phone technology with data processing capabilities, smart set-top boxes with integrated web browsers, and embedded network components in equipment ranging from office copy machines to kitchen appliances. Many devices requiring IP addresses and connectivity will be consumer-oriented, such as your coffee machine, dishwasher, etc.

IPv6's 128-bit address space will allow businesses to deploy a huge array of new desktop, mobile and embedded network devices in a cost-effective, manageable manner. Furthermore, IPv6's autoconfiguration features will make it feasible for large numbers of devices to attach dynamically to the network, without incurring unsupportable administration costs for an ever-increasing number of adds, moves and changes. The business requirement for IPv6 will be driven by end-user applications.

Peter H. Salus, the author of *A Quarter Century of UNIX* and *Casting the Net*, is Editorial Director of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Running the Numbers

Doc Searls

Issue #69, January 2000

According to Benjamin Disraeli “There are three kinds of lies: lies, damn lies and statistics.” But, according to Jon “maddog” Hall, “There are three kinds of lies: lies, damn lies and benchmarks.”

Businesspeople and journalists have at least one thing in common—they love numbers. Their appetite for numbers constitutes the enormous demand market that keeps think tanks, research firms and other professional guessticians in business.

The “suits” have a legitimate need for data to make informed business decisions. Good data often mean the difference between life and death for a company. We writers have a need no less compelling, though far less legitimate. We need to tell stories, and numbers make great story material—especially when we turn number-packed spreadsheets into pretty pictures worth a thousand words. Hey, it saves us from writing, and as the graphics people will tell you, readers look at pictures first.

Admit it; you looked at this graphic first, didn't you? Of course you did. Well, the graphic tells a lie. I made up the numbers, the spreadsheet made up the graphic, the artist made it pretty, and here we are, making a point. It's not just that “figures lie and liars figure”. It's that we rely a great deal on both. Take it from an old liar, or as I hate to admit, a PR guy. Let me tell you a true PR story.

The year was 1988, and I was working for a hot networking company that sold more than half the network connectors in its market. Since that market consisted of boxes sold by one company and sales figures for it were easily (though not precisely) estimated, we could figure out our market penetration—or something close to it.

However, there was a hitch: we competed directly with that box company (and lived at their mercy), so we didn't want to release our actual sales figures. Yet we wanted to publicize our success. We knew our story would be easier for editors to accept if our numbers were “objective”. Since editors go to industry research firms for their objective data, we had to get those firms to OEM our numbers for us.

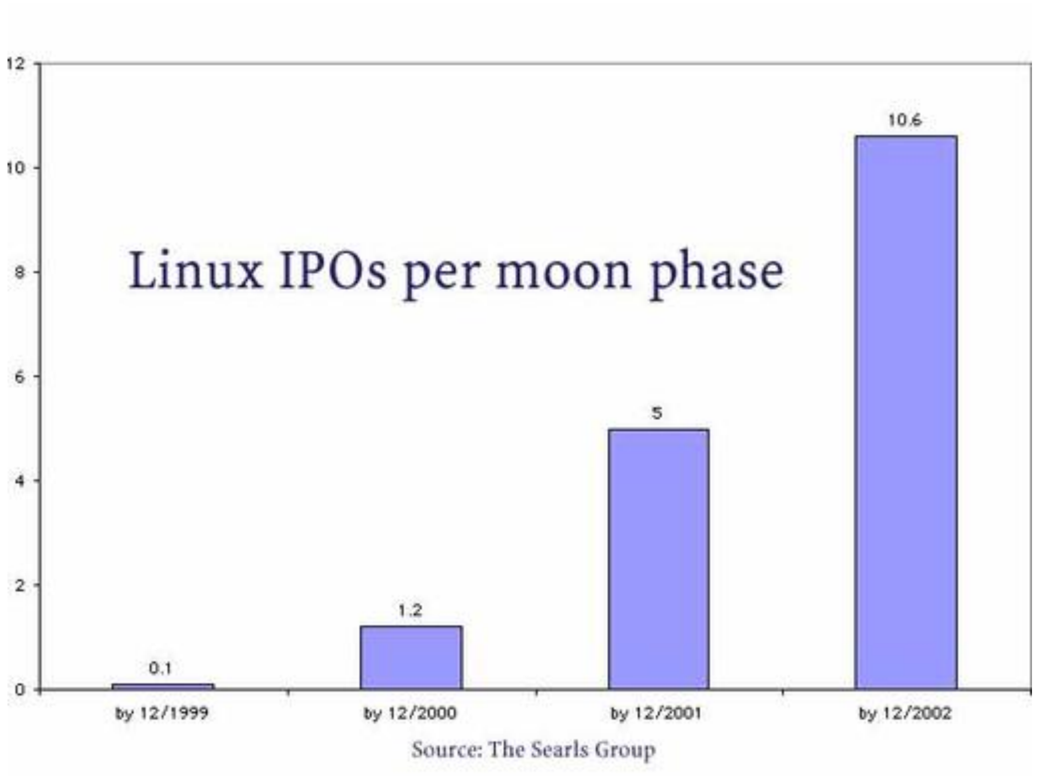
We created a nice “internal” graphic showing our best guess—55% market penetration—and called Analyst A at the biggest research firm. Would he like to know how we were doing? Of course! So we sent him the graph. A few days later, after the firm finished digesting this “fact”, we started referring editors to the firm. Sure enough, new graphs began to appear with stories of our success, all listing the research firm as the “source”.

Were we lying? No. We were simply dropping our best facts into the bottom end of the data food chain, having faith that it would find its way to the top. Now here I am, at the top of the same food chain, and I see very few Linux companies which know how this system works. I think the problem is techies are too honest and too literal. Most Linux companies are run by techies—guys like you—if we trust our own readership figures (from an “objective” source, of course).

Take an issue like host web servers and operating systems. At Netcraft's site, http://www.netcraft.com/whats/, you can discover, for example, that the Microsoft Network is running Microsoft-IIS/4.0 on NT3 or Windows 95. That information is returned when Netcraft interrogates the site host, copied out of the browser and pasted right into this text. Can we trust Netcraft, or the information it automatically obtains? Our techies here at *Linux Journal* say “No.” In fact, one techie believes MSN may actually be hosted by UUNET using BSDI. There's no way to tell by using Netcraft's method, because it's too easy for the host to spoof out a wrong answer, just to be perverse.

I still wanted that kind of data, however qualified, so I went ahead and put a chart together with the results of Netcraft's interrogations of the top 25 U.S. hosts (see “Work Still Cut Out” in upFRONT). Highly qualified findings are better than no findings. Some interesting data are there, such as Hotmail, a Microsoft property, running Apache on FreeBSD; and some significant ones, such as Linux running on only two (Real Networks and Go2Net). These findings address the concerns of several parties—regular readers of *Linux Journal*, BSD advocates, “suits” who need unbiased numbers and others—who feel that *Linux Journal* should remain as informative and unbiased as possible, while still advocating Linux to the world.

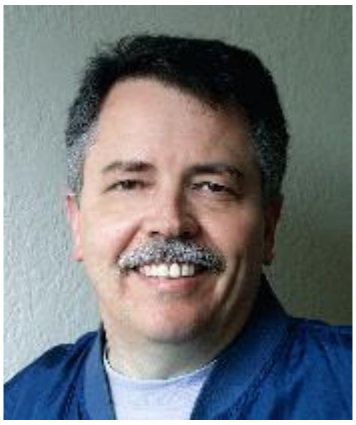
No problem with doing that, but we need the numbers. The numbers we get from research firms will be no better than those obtained from Linux vendors and other involved parties.



Linux IPOs per Moon Phase

Let's look at two of the oldest and largest research firms operating today: Gartner Group and International Data Corp. In recent months, the Linux community has made the most of IDC's very positive numbers, which show Linux as the fastest-growing server operating environment. At the same time, many in the Linux community bristled at Gartner's research findings, which called Linux “hype du jour” (among other unkind things) in its report, “Will Linux Be Viable Competition for Windows Desktops?” A Google lookup finds over 2,000 pages mentioning both Gartner and Linux. Most of those flame Gartner for its cluelessness. One anonymous coward on Slashdot writes, “The reality is, while the Gartners of the world are fudding away, Linux is already installed at all levels. Get a clue, PHBs [bosses]. Now, why would anybody pay Gartner for advice?”

The answer is simple: because they need objective information from somebody, and they're not getting it from you. Flames are easy. Facts are hard. Like them or not, Gartner and IDG traffic in facts or the most educated possible guesses. If you have good hard information on how Linux is changing the business world, let them know. Or skip them, and let us know. We're in the same business. That's what you—our readers—pay us for. No lie.


Doc Searls is the Senior Editor for *Linux Journal*. He can be reached via e-mail at info@linuxjournal.com.

Best of Technical Support

Various

Issue #69, January 2000

Our experts answer your technical questions.

X Marks the Spot

My system worked fine for a few months, but now I'm facing some problems with it. When using X, the system suddenly freezes. Neither the keyboard nor the mouse responds. I can't **telnet** to my computer—nothing works. Log files tell nothing—just as if nothing happened—but I have to boot my computer and hope **fsck** can fix everything. This happens about once a week and I don't have the slightest idea why. The system is Red Hat 6 out-of-the-box with no updates. Tried another video adapter, but that didn't change anything. Hardware: AMD-233, 64MB, ATI 3d Rage Pro, 3c509, Seagate 32. —Tony Söderudd, tsoderud@cc.hut.fi

It's hard to tell what the problem is. One possible reason is a bug in X; try upgrading to the latest stable XFree86 (3.3.5 at the time I write this). Another possibility is some kind of hardware/resource conflict, although since you say the problem happens only once a week on average, it's not too likely. The last option is simply bad hardware, like an overheating CPU (bad fan?) or flaky memory. I believe that is the most likely alternative and the freeze is actually completely unrelated to X. See http://www.bitwizard.nl/sig11/. —Marc Merlin, merlin@varesearch.com

A Communications Problem

For the last couple of years, my computer has been a small server on the Internet. Its connection is via modem. I had no problems the first year or so. On April 16, 1999, my /var/adm/messages file filled up with garbage. After a couple of weeks, I determined the phone company came to the wrong residence and “played” with my phone lines while my modem was connected. Ever since that time, my modem will randomly hang up without dropping carrier. Linux still thinks it is connected to the Internet, i.e., the **route**, **ifconfig**... still show a connection even though there isn't one. This doesn't happen every time the modem disconnects. Sometimes this will happen a couple of times a day; sometimes only once a month. Once it does happen, there is no way to fix it, other than physically turning the modem off and back on. I've tried replacing all hardware: modem, modem cable, serial port(s), phone wiring, phone jack. I've also tried compiling a new kernel and **pppd**. Nothing has helped. What in the world is happening and how do I fix it? —Eric Trimmer, eric@et.trimmer.org

There is a relatively simple solution, but you may need to upgrade your kernel and PPP package to support it, depending on the age of your actual installation. Use the **lcp-echo-interval** and **lcp-echo-failure** options to tell pppd to terminate when it doesn't receive a certain number of **ping** request responses. **pppd** will then terminate and cycle the connection when your physical layer goes down, whether or not your modem is correctly telling Linux it has lost its link. —Chad Robinson, Chad.Robinson@brt.com

Those Pesky .tgz Files

The instructions Software Forge Inc. provided for installing LinuxCAD are not really applicable to OpenLinux 2.2. What is the best way to install software in a .tar or .tar.gz file on an operating system that uses RPM? —Wayde C. Gutman, wcgutman@mwpower.net

You can actually install a **tar** file just fine on an RPM-based system. It is, however, better to have an RPM, since it keeps track of the files installed. I believe you should be able to use **alien** to convert your tar file into an RPM: http://kitenet.net/programs/alien/. —Marc Merlin, merlin@varesearch.com

The tar.gz format is somewhat similar to .ZIP under other operating systems. Usually, you'll just unpack the tar file, following instructions found in the included README or equivalent file.

Use **tar tvzf** ***`filename`*** to list the contents of a tar.gz file, and **tar xvzf** ***`filename`*** to extract it. **tar** is one of the most important UNIX commands; check the introductory UNIX guides or check **info tar** and **man tar** to get detailed information. —Alessandro Rubini, rubini@prosa.it

To Load or Not to Load

I have tried to use function **request_module** in my code to load dynamic kernel modules. The request_module function itself is not a problem, but the function **kerneld_send**, which is called by request_module. Every time I try to insert (**insmod**) my module, which is supposed to request the other module, I get the following error: “helloworld.o: unresolved symbol kerneld_send”. Why can't request_module find the kerneld_send function? I use Red Hat 5.0, kernel 2.0.36RTL1.1. —Kristiina Valtanen, Kristiina.Valtanen@vtt.fi

At insmod time, the module is linked to the running kernel, using the public symbol table exported by the kernel (shown in /proc/ksyms). The function request_module (defined in linux/module.h) is actually an in-line function, and it gets expanded in-line at compile time. However, kerneld_send is not in-line and must be located at link time. Since kerneld_send is not exported, your module can't be loaded. The function is not exported, because no module normally asks for kenreld functionality. Unneeded internals are not exported. You can export the function by hacking the file kernel/ksyms.c and rebooting a new kernel. —Alessandro Rubini, rubini@prosa.it

Support Your Local CD-ROM

I am trying to install Linux 6.0 on my 486. I have run into serious problems. Red Hat Linux 6.0 does not seem to recognize my Sony CD-ROM. Several e-mails and phone calls to Technical Support at Red Hat have not solved my problem. The error message I keep getting is “I could not mount your CD on device hdk”. I was asked to download the new boot image and try installing it again, but was not successful. I was asked to give some parameters at the boot prompt, again in vain. The last suggestion I got from tech support is to buy a new CD-ROM. The CD I have is Sony Atapi CD76E. It is an IDE controller. Is there anyone who could let me know how I can work around this problem so that I don't have to buy another CD-ROM? —Srilakshmi, RSSri@aol.com

You say the installation is trying to mount the CD drive as /dev/hdk—that's the 11th drive on your system! I suspect this is not correct, and the kernel merely needs to be nudged toward the correct device. Consider first the naming conventions for IDE drives (including CD drives):

- First IDE drive on 1st controller: /dev/hda
- Second IDE drive on 1st controller: /dev/hdb
- First IDE drive on 2nd controller: /dev/hdc
- Second IDE drive on 2nd controller: /dev/hdd

You should see the pattern. What you probably need to do is determine which /dev device points correctly to your drive, and tell the Linux kernel what it is at the **boot:** prompt. For example, if you have two IDE controllers on your system (very common), the first controller has one IDE hard drive and the second controller has your CD drive. Thus, you should point your kernel to /dev/hdc. At the **boot:** prompt, type:

```
hdc=cdrom
```

There will be something before this on the **boot:** line that needs to be typed in; probably **linux** or **install**. Red Hat support should be able to tell you what needs to precede the line above. —Erik Ratcliffe, erik@calderasystems.com

Control Issues

I'm using Linux and I'm wondering how I can control a TELNET session from a shell script. In DOS, I would use a keyboard buffer, but it seems this is not available in Linux. I would like to send characters, wait for a special response, then send something again. How can I do that? —Thomas Lienhard, tl@tlienhard.com

You can open a pipe to telnet and send data to its STDIN and read responses from STDOUT. Using the keyboard buffer makes no sense in UNIX, as no process accesses the keyboard; they just read from standard input. If using the telnet command doesn't work, you can try **netcat** (available within Debian). Also, note that most languages support TCP connections; **expect**, for example, is a Tcl extension designed to perform exactly the task you need to accomplish: sending input and expecting responses. —Alessandro Rubini, rubini@prosa.it

What you are looking for is expect. This is exactly what it was designed for. **man expect** on any Linux box with it installed will give you the information you need, or you can download it from http://expect.nist.gov/ if you don't already have it. —Mark Bishop, mark@bish.net

Executions 'R' Us

I'm using the KDE window manager, KDM and tcsh. My distribution is Mandrake 6.0. In my home directory, I'm keeping .cshrc and .login for my own initializations, but the problem is that after logging to an X session and invoking any terminal (xterm or kvt), the .login is not executed at all (.cshrc executes fine). Both files (.cshrc and .login) have execute permission and properly execute in a normal (batch) session. Where and how do I need to tell X (and/or KDM) to execute my .login after start-up? —Valentine Kouznetsov, vkuznet@fnal.gov

When you log in to the computer using a display manager, no login shell is started, so no .login or .bashlogin is sourced. The display manager handles the authentication and runs your display configuration without passing through a shell. You need to perform all of your initialization from within .cshrc. That's one of the reasons I don't run a display manager at all. —Alessandro Rubini, rubini@prosa.it

Need Fire Protection

Is it possible to run a Linux firewall using IP mask connecting a small network to the Net using a cable modem? —Tom, tlross@home.com

Yes, it is the easiest firewall configuration. Just run

```
ipfwadm -a accept -m
```

or an equivalent **ipchains** command. —Alessandro Rubini, rubini@prosa.it

Advanced search

New Products

Jason Schumaker

Issue #69, January 2000

Quick-Connect, CoolKeyboards, ESP Print Pro and more.

Quick-Connect

BASCOM Global Internet Services, Inc., announced the 2.2 release of Internet Quick-Connect (IQ2), a “plug and play” network connectivity solution designed to provide small businesses with robust, big-business performance while offering the quick set-up features found in BASCOM's family of Linux-based thin-server applications.

Contact: Bascom Global Internet Services, Inc., 275-R Marcus Blvd., Hauppauge, NY 11788, 516-434-6600, 516-434-7800 (fax), info@bascom.com, http://www.bascom.com/.

CoolKeyboards

CoolKeyboards Corporation has produced the world's first piece of computer hardware specifically for the Linux operating system. The keyboard replaces other logos with those of Tux the Penguin and the logo of Linux itself. Each keyboard comes with an easily removable dust cover and is backed by a one-year warranty.

Contact: CoolKeyboards Corporation, 741 North Pine Island Road, Plantation, FL 33324, 954-533-1314, sales@coolkeyboards.com, http://www.coolkeyboards.com/.

ESP Print Pro

Easy Software Products announced the first production release of ESP Print Pro, a complete printing solution based on the company's common UNIX printing system technology. ESP Print Pro supports Digital UNIX, HP-UX, Irix, Linux and Solaris, and provides PostScript and image file RIPs to support non-PostScript printers.

Contact: Easy Software Products, 44141 Airport View Dr., Suite 204, Hollywood, MD 20636-3111, 301-373-9600, 301-373-9604 (fax), info@easysw.com, http://www.easysw.com/.

EnlightenDSM version 3.4

Enlighten Software Solutions, Inc. announced the release of EnlightenDSM version 3.4, the latest version of its flagship multi-platform system administration product. EnlightenDSM provides a single graphical interface to simultaneously manage mixed environments comprised of multiple Linux, UNIX, NT and Windows 95/98 machines. EnlightenDSM 3.4 is available for download and supports TurboLinux Workstation version 3.6.

Contact: Enlighten Software Solutions, Inc., 99 Baker Way, 5th Floor, San Mateo, CA 94404, 650-578-0700, 650-578-0118 (fax), info@sftw.com, http://www.TurboLinux.EnlightenDSM.com/.

RTSP/RTP Server

Entera, Inc. unveiled its new RTSP/RTP (real-time streaming protocol/real-time protocol) server for Apple's QuickTime 4 technology using the Linux platform. The new RTSP server from Entera allows end users, content providers, and ISPs to stream movies, music and interactive content. A beta version of the software is now available for free download.

Contact: Entera, Inc., 40971 Encyclopedia Circle, Fremont, CA 94538, 877-4-ENTERA, 800-726-5762 (fax), info@entera.com, http://www.entera.com/.

EZTerm 1.3 for Linux

CSV Technologies Inc. announced the release of EZTerm 1.3 for Linux. EZTerm is designed to eliminate the complexities of the Linux command-line interface by allowing the user to build and modify commands. The software is a complete Linux command reference and will run with all Linux distributions. All products, including EZ Linux mouse pads and EZ Linux command cards, are currently available.

Contact: CSV Technologies, Inc., 1113 Blanchard St., Victoria, BC V8W 2H7, Canada, 250-386-4689, http://www.csvtech.com/.

OpenMerchant

Idealab unveiled its first Linux venture, OpenSales, Inc., to offer an open-source e-commerce software solution called OpenMerchant. The free retailing catalyst runs on Linux, Solaris, UNIX and Windows NT platforms and will be bundled with VA Linux Systems servers late in 1999. The company will also offer a wide range of support and complementary services.

Contact: OpenSales, Inc., 2955 Campus Drive, Suite 250, San Mateo, CA 94403, 650-372-5230, 650-357-7621 (fax), information@opensales.com, http://opensales.com/.

High-Density T1, E1 WAN Cards and ImageStream Routers

ImageStream Internet Solutions announced the release of four new WAN cards for Linux. The new cards include T1 and E1 versions of the 4-port WANic 654 and the 8-port WANic 658. ImageStream is also releasing CompactPCI versions of the cards, the Aries 654 and Aries 658. The WANic 654 and 658 can be installed in any Intel-compatible PC that has PCI bus support.

ImageStream Internet Solutions also announced gigabit Ethernet support for the Rebel, Gateway and Enterprise Router, as well as the upcoming Rocket Router. Gigabit support includes a full-duplex SX-style fiber interface that connects to a gigabit Ethernet switch on the LAN side of the network, or it can be connected to another router with gigabit Ethernet support.

Contact: ImageStream Internet Solutions, 7900 East 8th Road, Plymouth, IN 46563, 800-813-5123, 219-935-8488 (fax), info@imagestream-is.com, http://www.imagestream-is.com/.

NetMax Thin Server Series

Cybernet Systems Corporation announced NetMax, its Linux-based thin/Internet server software suite. The NetMax suite provides web server, file server and firewall capabilities. The NetMAX servers use custom configuration code, tight component integration and an advanced graphical HTML interface to simplify Linux installation and management.

Contact: Cybernet Systems Corporation, 727 Airport Blvd., Ann Arbor, MI 48108, 734-668-2567, 734-668-8780 (fax), info@cybernet.com, http://www.cybernet.com/.

OpenMail 6.0 for Linux

Hewlett-Packard Company announced the release of OpenMail 6.0 for Linux. OpenMail, HP's business messaging and collaboration solution for UNIX systems, provides upgraded functionality and e-service capabilities to the growing number of Linux-based businesses. OpenMail is now available for download.

Contact: Hewlett-Packard, 3000 Hanover Street, Palo Alto, CA 94304-1185, 650-857-1501, 650-857-5518 (fax), http://www.hp.com/go/openmail/.

VerteX 400-2U Server

EIS Computers announced the VerteX 400-2U, a server designed and optimized for ISPs that can be easily maintained in rack environments. The VerteX 400-2U has a Pentium-III Intel motherboard capable of operating with single or dual 600MHz processors and up to 2GB of PC-100 RAM. The integrated motherboard includes two Ultra-2 LVD SCSI disk channels, 10/100 base-T Ethernet and a 2MB SVGA video port, all on-board. The 2U enclosure can hold up to six hard drives, providing over 200GB of total storage and includes hot-swappable power components.

Contact: EIS Computers, Inc., 207 West Los Angeles Avenue, Suite 303, Moorpark, CA 93021, 800-351-4608, 805-383-1470 (fax), http://www.eis.com/.

VSI-FAX 4.0

V-Systems, Inc. announced the latest version of its flagship product, VSI-FAX 4.0, which uses a central fax engine running on UNIX-based or Windows NT-based network hosts. The TCP/IP-based server integrates across Linux, UNIX and Windows NT platforms to fax-enable applications and work-flow processes. VSI-FAX features improved Fax Merge and Print-to-Fax support for StarOffice, WordPerfect Office and Microsoft Office.

Contact: V-Systems, Inc., 32232 Paseo Adelanto, Suite 100, San Juan Capistrano, CA 92675, 800-556-4874, 949-489-2486 (fax), info@vsi.com, http://www.vsi.com/.

Xpresso Linux 2000

Xpresso Ltd. announced the release of Xpresso Linux 2000, a fully integrated Linux distribution based on Red Hat 6.0 (kernel 2.2). Xpresso Linux uses KDE 1.1 as the default GUI, Netscape 4.51 for Internet browsing, Corel's WordPerfect 8 and StarOffice 5.1 for word processing, and even includes a few games.

Xpresso Linux 2000 is available for free download, or the complete distribution can be purchased.

Contact: Xpresso Ltd., PO Box 11, Woodford Green, Essex, IG8 OSQ, United Kingdom, 44-208-559-2600, 44-208-498-9834 (fax), mail@expresso.org, http://www.xpresso.org/.

iMigo

Lineo, Inc. announced that Embrowser, Lineo's embedded micro web browser, and DR DOS, one of Lineo's embedded operating system platforms, have been selected for Multicode's iMigo, a Latin American-bound Internet set-top device that provides low-cost browsing and e-mail over televisions. End users of iMigo will be able to connect their homes to the Internet for a fraction of the cost of a full desktop system. The first version of Embrowser is based on DR DOS and leverages the broad set of existing drivers, development tools and engineering knowledge base associated with DOS. The port of Embrowser for Embedix, Lineo's embedded Linux platform, will ship during the first quarter of 2000.

Contact: Lineo, Inc., 383 South 520 West, Lindon, UT 84042, 801-426-5001, 801-426-6166 (fax), info@lineo.com, http://www.lineo.com/.

Storm Linux Beta

Stormix Technologies announced the beta version of Storm Linux. Features include GUI modules for networking, dial-up and adding users. Another major feature of Storm Linux is the Storm Hardware System. SHS automatically detects PCI devices, including video and network cards, SCSI devices and USB bridges. The final release is scheduled for November 1999.

Contact: Stormix Technologies Inc., 555 West Hastings Street, Suite 2040, Vancouver, BC V6B 4N6, Canada, 877-STORMIX, 604-688-7317 (fax), http://www.stormix.com/.

Linux Means Business: Linux and Banking

Josip Almasi

Issue #69, January 2000

A bank in Croatia is using Linux for its development platform. Here's the story.

We are a small bank in a small country. Up until two years ago, our management policy regarding software was internal development—standard banking software price is about $1 million, and we were small enough to handle internal development. Our servers were Intel-based machines, running SCO and Informix SE.

Then management decided to change our business strategy, and soon we started to grow very fast. In less than a year, the number of employees doubled, and so did the number of PCs. Our LAN became a WAN, and the number of transactions multiplied five times and continues to grow. We could not ensure efficient software solutions for all new requests, so the decision was made to buy commercial applications. We also bought two IBM RS/6000 servers running AIX, connected in a high-availability cluster.

A year later, this decision may not seem too smart. We have our software, which handles 70% of our business, and a bunch of applications, communicating between each other with shell scripts, 4GL and C programs that perform needed data conversions. But development takes time, and we cannot employ a new application without at least three months of extensive testing.

How Linux Fits In

More and more users needed Internet access, and we needed a dial-on-demand router. We didn't want specialized hardware. We wanted to be able to customise logs, schedule access time on a per-user basis and use an existing PC for that purpose.

After a week running NT Workstation with WinGate demo, we gave up. WinGate used to die two or three times a day. That would not be a problem in and of itself, since it was a demo anyway, but the COM port stayed locked every time it happened, remaining locked until reboot. Obviously, NT did not release allocated resources after process termination. Well, it did not look like a serious operating system to us.

Linux came in our focus accidentally. A friend of mine did quite a bit of advocacy and convinced me to try it, giving me precise directions and a Slackware 3.2 CD. Why not? If it works, fine; if it doesn't, it costs me nothing. After all, I am an experienced UNIX administrator—I can do it. So I took a 486 with a 1GB SCSI disk and 16MB RAM and started to play with Linux.

A few hours after inserting the CD into the drive, I had a working proxy, and I still have it. It has been working for a year now. All we have to do with it are the usual administration jobs: adding new users, kernel patches, etc. Furthermore, it became a caching proxy and news server. To be quite specific, there is **diald**, socks5 proxy, caching DNS, **tcpd**, Apache as caching HTTP proxy and internal web server, and **inn** as a news server for both local and Usenet newsgroups.

My colleagues were very suspicious of Linux, but started respecting it after a few months of uptime.

After the initial experience, I was very pleased with Linux and started to explore. As I discovered the purpose of the iBCS module, I had to run the SCO version of Informix on it. So, I copied Informix, our database and applications, tested it, and again it worked from the first try. I even found Linux a better development platform than SCO—it didn't require additional kernel parameter tuning to run complex database queries, it worked faster and was more stable, and it had better development tools.

Well, it's not quite fair to compare Linux to an obsolete product (SCO 4.2), but I tested concurrent queries and updates on millions of records, shut down the machine in the middle of work, killed system processes such as **kswapd**, tested power-off behaviour, and even pulled off the IDE disk cable on the working machine. Although I got database corruption, I didn't manage to produce unrecoverable file system errors. When **fsck** failed, **fsdebug** worked. For the record, we had unrecoverable file system errors on SCO. The result is that we now use Linux as our development platform.

Switching to Linux may not be easy for a Windows user, but we are used to working in a UNIX environment. We have set up Linux on a PC as a server for development and testing purposes and on three developers' PCs instead of Windows. We used Windows mostly as a task switcher for a few terminal sessions and for printer sharing. The only reason we used it was that it came pre-installed on every PC. Linux gave us additional possibilities: we have our native development environment on every PC, with good development tools such as Emacs, **ddd** and other good stuff. Since we are used to GNU tools, we installed them on both AIX and SCO. From a user's point of view, our AIX looks more like Linux now.

It's hard to believe that you can get a good development environment free when you used to pay thousands of dollars for one, but it's true.

When our users wanted to share Windows files across the WAN, we didn't even think about an NT Server, as Linux has already proven its reliability. So, we installed Samba as a WINS server. We did not set it up on AIX servers because they handle our critical applications and we just don't want to experiment on them. Today we use Linux/Samba for printer sharing, too, as it gives our UNIX systems access to Windows print servers, and Windows workstations access to UNIX printers. Basically, when a user wants to print something from UNIX, he can either print to a dot matrix or laser printer in the system room, or to a printer in his room, which may be connected to his PC or Windows-shared. When he chooses a local printer, the appropriate Windows print queue is determined by his IP address.

When we realized Samba's possibilities, we wanted to implement a centralized backup, based on the following idea: all users have their accounts on the Linux box, their Windows boxes mount their home directory upon startup, and all their documents are saved on Linux. Most of our users don't even know where their files reside; they just click on the icon and use the application, so we could change the working directory for Windows applications on every PC.

This policy has some other security considerations. First, only the logged-in user can read documents. Second, we receive some documents from the National Bank and other government organizations as Microsoft Office files. Once we even received a macro virus with them—it damaged only the user's local disk, but the user lost some important files. Before that happened, we had not been worried about viruses, since they come with cracked games, but now they may come with business data. This way, we can also do periodic virus scans, and file permissions are set in such a way that a user can write to only his home directory, thus a potential virus can't spread around. But we haven't fully implemented this yet, because we have to change every PC's network settings to mount the Linux drive and move all files to it. We also have problems forcing our users to log onto Windows with their Linux user names and passwords and explaining to them how to change the Windows password every time their Linux password expires.

One day, we received a request to allow access to an old Clipper application from a remote location. If we had just installed it on the remote PC and mapped the network drive to our machine, the Clipper database would go over the modem line, so we would have to increase bandwidth. What's more important, we would risk database corruption, and it's also the same story with viruses, since we have no physical control over remote locations. Linux took a role again; we installed the DOS Emulator. A few smaller problems arose. Clipper is never idle, so DOSEmu takes all available CPU time. Since we have only a few Clipper users, we just gave DOSEmu a smaller priority. This way, only screens are traveling the network, and with **ttysnoop**, we can see exactly what a user is doing. Also, there's no database corruption when Windows dies—Linux doesn't die.

Linux has made our life much easier, especially on that Monday morning when the hard disk on our SCO server didn't wake up. We had no up-to-date database on our backup server, since the previous weekend the entire bank moved to a new location. The backup server was moved first, so there was no overnight database copying. We had a tape backup, but it needs at least an hour and a half to restore it. We had the Informix log file, which takes at least the same time to roll forward the database. Luckily, we had an up-to-date database on our development server; unfortunately, copying over the network takes about an hour. Our customers were waiting, so we chose to drive the bank on Linux for a day. The server was not really a server but a PC with P133, 16MB RAM and an IDE disk, so there was a lot of swapping, but most users didn't notice any difference.

Today, Linux does quite a good job for us, and so do GNU programs. Our complete development is done on Linux. C programs are then recompiled with the GNU C compiler on either SCO or AIX; 4GL programs are just copied to the proper place. Bash became our standard shell on both AIX and SCO. CVS takes care of the version control. Communication with remote locations is also handled by Linux, and so is printer spooling. With the proxy, the internal web and news, Linux played a significant part in our job. Anyway, it's not yet ready to be used as the database and application server for our purposes, because it still lacks a few features needed in our mission-critical applications: the high-availability cluster software and a journaling file system. We cannot afford any data loss, and that's the main reason we have chosen AIX for certain applications.

At the end of this year, we plan to migrate our database, applications and development completely to AIX, so we'll stop using SCO, but Linux will stay. It works very well and has greatly improved our security and administration.

Josip Almasi (jalmasi@partner-banka.tel.hr) has been in database design/ programming for ten years, and in Novell and UNIX system administration for

the last five years. He is currently working as a database/system administrator at Partner Bank, Croatia. His hobbies are Aikido and blues harmonica.

Archive Index Issue Table of Contents

Advanced search

Writing an Alphanumeric Pager Server for Linux

Erik Max Francis

Issue #69, January 2000

For those thinking of setting up an alphanumeric paging service or gateway, this article explains the protocols and methods involved.

Alphanumeric pages—at least the type we will consider here—are delivered via modem. To deliver a page, we connect to our service provider, communicate with it through its specified protocol, receive a confirmation that the page has been delivered, then disconnect.

To send a message to someone, we need to know two things about them: their pager service dial-up number and their PIN (personal identification number). Alphanumeric pager owners should be able to contact their pager service provider and get this information with little trouble.

Protocol

Several protocols are used for delivering pages. The most popular (and among the simplest) is IXO, named after the company which invented it. It's also been called TAP (Telocator Alpha-entry Protocol) and PET (Personal Entry Terminal); these names were assigned to it by other companies over its history.

In this article, I will talk solely about the IXO protocol. It is a bit involved so I won't go into great detail, but it is relatively straightforward. In brief summary, here is a description of the overall transaction between the remote dial-up site (us, when delivering a page) and the pager service provider.

First, we establish a connection and get the service's attention. After the service acknowledges our intent to transmit a message, we declare our wish to enter automatic mode—this is the simplest method through which to transmit a message, although in rare cases we may want to use manual mode.

We wait for the service to acknowledge our request. When we get the go-ahead signal to send the content of our message, we send a series of fields. Usually we would send two fields, consisting of the PIN of the pager to which we wish to send a message, and the message. We follow the message with a checksum, so the provider can verify (to some degree) that our message has been faithfully received. Finally, once we receive an acknowledgement that our message has been received, we politely request to be disconnected and drop carrier.

The IXO protocol (see Resources) supports sending multiple pages to the same service, but not necessarily the same pager, during a single transaction. Thus, this method provides the possibility for optimization in a large-scale service, if we find many pages are being sent concurrently to people using the same service, i.e., the same dial-up number.

Profiles and Devices

To start with, we'll need a mechanism for identifying each pager to which we want to send messages; we'll call them *profiles*. As mentioned earlier, there are two things we need in order to be able to send messages to a particular pager: the phone number of its pager service and the PIN of that particular pager. These may be kept in configuration files, which an administration program can edit and delete.

If we want to support multiple modems—that is, the ability to deliver multiple pages simultaneously—we also need to know about and keep track of the physical modems. We'll call them *devices*. As with profiles, we'll keep a list of the devices allocated to our server in a configuration file.

Concurrent page delivery should take advantage of multiprocessing; use **fork** to send multiple pages simultaneously with different devices. This makes things much simpler in terms of the overall flow control and doesn't pose any significant problems, since we can simply check the exit status (succeed or fail) of the subprocess through **waitpid** and take action accordingly.

We'll keep the list of devices handy, and when we are delivering a page using one, flag it as busy. Unless we're going to be running on a system where the modems are dedicated solely to our server, we will almost certainly want to use standard lockfiles, in addition to flagging the devices as used. (Even if the modems truly are dedicated, it can't hurt to be extra careful.) This will prevent other processes from interfering with our server while it is actually in the process of delivering a page.

Filters

Alphanumeric pagers tend to be limited in terms of how many characters a message can display; the typical maximum-length support with the IXO protocol is 256 characters, including the PIN expressed as a string, so in practice it's usually closer to 250.

People who get a lot of pages frequently run into this limit, particularly when longer messages (such as e-mail or notes) are being sent. A useful solution is to implement a *filter*, which processes messages to be sent and does simple searches and replaces in order to substitute shorter equivalents for commonly appearing words and phrases. For instance, the word “and” might be represented with an ampersand (&), “talk to you later” might be abbreviated TTYL, and so on.

Filters could also be used to provide a form of security through obscurity. With most pagers, messages delivered over the airwaves are not secure; anyone with the right equipment can intercept them. This means, obviously, that the average person would probably not want their name, address and other private information being transmitted over the air. A simple choice of filters could replace this sensitive information with abbreviations recognizable to the party receiving the page, but meaningless to anyone else.

It would also be straightforward to implement a filter based on regular expressions, using the POSIX regex library. Since both the simple and regular expression searches are relatively straightforward, I will leave them as an exercise to the reader.

Server

The *server* is the workhorse of our operation. Servers process pager delivery requests (we'll call them *jobs*) and then deliver them via the IXO protocol over an available device.

Jobs consist of two very simple things: the profile to use and the message to send. How the server receives each job depends on how we want to structure it. I'll discuss two basic approaches here: TELNET-based and file-based servers. Each has advantages and disadvantages, and your choice of which to implement will depend largely on what sort of clients you want to support.

TELNET-based servers listen to a specific port on the host machine and use a simple protocol to send the profile and the message. A protocol such as:

```
profile
```

should suffice (provided, of course, that the strings identifying protocols don't contain newlines).

For security reasons, such servers should require some sort of authentication before accepting a job for delivery. The type of authentication we should use will depend strongly on our particular security needs, and is beyond the scope of this article.

A file-based server, on the other hand, watches a specific directory on the host machine (pausing for a brief length of time, then scanning its contents) and reads each new file, creates a job for it, then processes it; the file can be deleted when the server no longer needs it. The format of this file need be no different from the protocol of the TELNET-based server:

```
profile
```

As with TELNET-based servers, file-based servers need to consider security issues. The server would run as a specific group, and only programs which are part of that group may submit a job. When job files are created, they must be owned by the group and given group-writeable permissions (in a group-writeable directory), so that the server may delete files when it is done with them. An authentication scheme would also probably be useful here; at the very least, some simple encryption may well be in order.

Both TELNET- and file-based servers have their respective strengths. The most obvious for a TELNET-based server is it is quick and simple, and a client can be run on any machine that can be reached by the server using the **telnet** command. (Without adequate security considerations, this can also be a liability.)

The clients to file-based servers must run on the same machine (disregarding NFS-mounted directories for the sake of simplicity) and must be run by a user who is a member of the appropriate group. The most obvious advantage to a file-based server is that in the case of a server shutdown or a machine reboot, the pending jobs will still be waiting, whereas TELNET-based jobs must offload their pending jobs to disk (or to another server) if they are shut down.

Clients

Clients are the programs we use to send jobs to the server. At this point, writing the clients should be easy, since by choosing the type of server we are going to implement, we have implicitly chosen the type of client we must use to communicate with it.

Clients need to know how to contact the server, as well as verify that a profile is valid (the client should know when it has been asked to send a message to a nonexistent pager). To start with, we need a basic client, one we can run using the command line (or perhaps STDIN) to specify the profile and message and deliver a job—no frills, no fancy interfaces.

For file-based servers, this is a program (executable only by members of the appropriate group) which creates a job file in the appropriate directory with the appropriate permissions. With TELNET-based servers, the program can be run on any machine which has TELNET access to the machine on which the server runs and delivers the message via TELNET.

Once we have the basic client, we can move on to other types of clients. Here are two we might want to support:

- An e-mail-based client: something like procmail could help us do the difficult work here—procmail can selectively process incoming e-mail and route it to a program of our choice. That program could then pick out the relevant headers we want transmitted (e.g., From, To and Subject) and the body of the e-mail, then pass it on to the basic client.
- A web-based client: a CGI script can do the job here, picking out appropriate form entities and passing them on to the client. In the file-based server, we must make sure that the user the web server runs as, often "nobody", is also a member of the pager group. (Be sure to send an HUP signal to the server after adding it.)

In both cases, there are security issues to contend with. We certainly wouldn't want just anyone who has discovered the e-mail or web gateway to be able to send any page they wanted to anyone. The specifics regarding what sort of security precautions are advisable will depend strongly on the type of service we are running, so I will not go into detail about them here.

Putting it All Together

Now we can fit the pieces of the puzzle together to come up with our server.

The main server will read in configuration and data files (where to find things, the list of protocols it knows about, the available devices, etc.). It then goes into its main loop.

If we support multiple devices, use waitpid and its **WNOHANG** option (so that the server will merely see if any subprocesses have finished, as opposed to waiting indefinitely—and blocking all other server activities until a subprocess *does* finish) to determine whether any jobs have finished being delivered; remove them from the queue if they have.

Then, check to see whether a device is available and a new job is ready to be delivered; if both are true, then take the following steps. First, determine the profile and the message from the job. Then, get the next available device and lock it, so our delivery will not receive any interference. After that, filter the message (if appropriate), and finally, start up a subprocess to deliver it.

Summary

Resources

Erik Max Francis is a UNIX engineer who lives in San Jose, California. His main interests are programming, Linux, physics and mathematics. He's been using Linux at home exclusively since kernel version 1.2.8 and has been reading and contributing avidly to Usenet since 1989. He can be reached via e-mail by max@alcyone.com.

Archive Index Issue Table of Contents

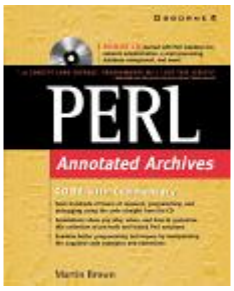
Advanced search

Perl Annotated Archives

Paul Dunne

Issue #69, January 2000

The book is full of clearly explained code.



- Author: Martin Brown
- Publisher: Osborne/McGraw-Hill
- E-Mail: customer_service@mcgraw-hill.com
- URL: http://www.osborne.com/
- Price: $49.99 US
- ISBN: 0-07-882557-1
- Reviewer: Paul Dunne

This is a fine book and a good resource for learning Perl. The subtitle, “Code with Commentary”, is apt. The book is full of clearly explained code. Studying such annotated examples of practical code is an ideal way to learn a language or increase one's knowledge of a language. The author takes a cross-platform approach, which is nice to see. Even though most of the examples are inevitably UNIX-based, some code is also demonstrated for fixing NT and Macintosh problems. The usual CD with code examples is included. One minor annoyance is the lines are terminated with “CRLF”, but I suppose it's easier for Linuxers to deal with this than for those using NT to handle bare newlines.

The book is divided into four parts. It begins with “Text Processing”, which also serves as a basic introduction to Perl. This section starts by rounding up the usual filters, then examines simple file manipulating, and concludes with various types of text database work. After that, “Networking and E-mail” does a good job of illustrating and explaining Perl's excellent support for network programming. Part 3, “World Wide Web”, covers the ground one would expect: HTML munging and CGI scripts. The CGI coverage ranges from simple examples to an entire on-line shopping application. Finally, “Administrator's Toolkit” supplies a useful set of tools for systems and network administration. All in all, 11 chapters, with over a hundred clearly and closely explained scripts—there is nothing I can find fault with here.

I have one issue to raise, however. That is, “everything with Perl” is not always the best way. While one can hardly fault a book about Perl for taking this approach, I would still like to see more emphasis on how Perl can co-operate with existing UNIX utilities. For example, my ISP uses dynamic IP addressing, so when I need to find out what my IP address is this time, I use a little shell script that reads as follows:

```
ifconfig | grep ppp -A1 | domain.pl | head -1
```

Where domain.pl is the following script:

```
#!/usr/bin/perl -n
while (m/(([-a-zA-Z0-9]+\.[-a-zA-Z0-9]+)(\.[-a-zA-Z0-9]+)*)/g) {
    print "$1\n";
}
```

I know this doesn't cover all cases, but it works where required. I guess I *could* have implemented the whole pipeline in Perl; but for me, and I suspect many other UNIX users, it is easier just to bolt a few tools together than to write a program, even in a language as easy to get started in as Perl. Why reinvent the wheel? Much better to use Perl as part of the UNIX toolkit, not to replace it.

In conclusion, as Perl is best-known as the “glue” that holds many web sites together, this book, if aimed at any specific audience beyond Perl learners and users, it is aimed at the web site administrator or programmer. However, it is well worth reading by anyone interested in Perl.

Paul Dunne (paul@dunne.ie.eu.org) is an Irish writer and consultant who specializes in Linux. The only deadline he has ever met was the one for his very first article. His home page is at http://www.cix.co.uk/~dunnp/.

The Collaborative Virtual Workspace

Stephen Jones

Issue #69, January 2000

Attend business meetings and impromptu gatherings with friends without ever leaving home—enter the virtual world.

You walk down the hallway and go into the design room. It's as hectic here as always; stuff is lying all over the floor. Alex, the lead architect, is here with two other designers, Brad and Lynn. They have been discussing the architecture for the new project.

“I think we've almost got it,” says Alex, “take a look at our design.” He hands you a document, which you duly examine.

“Not too bad,” you reply, “but you might want to look at something.” You walk over to the whiteboard and draw out a rough sketch. Brad sees where you're heading, grabs a marker, and adds some components to your drawing.

Lynn notices a problem and says “We may have some problems implementing the security interface with that. Let me give Rachel in security a call.” She grabs the phone, and after a quick conversation, tells everyone the impact the security team sees from this implementation.

“Okay,” you say, “sounds like you guys are on the right track. I'll carry this preliminary design over to the programmers and get a rough estimate on the time they'll need to implement it.” As you walk out, you notice your beeper has a message from your secretary reminding you of a meeting at 3:00 with the division head.

Activities like these occur in work environments every day; however, in this scenario the individuals involved never physically moved. In fact, one person sits down the hall from you, another works at a remote site, and the third was at home. Yet everyone was able to interact as though individuals, objects and documents were physically colocated.

As computers move to take a dominant position in the workplace and at home, the interactions between people change significantly. Instead of gathering around a water cooler or at a small cafe, meetings have moved to the Internet. However, with this change of venue come certain restrictions. No longer can you read a person's body language or infer meanings from the tone of someone's voice. With many Internet collaboration tools, such as Internet Relay Chat (IRC) or instant messengers, you can only interpret the words others type.

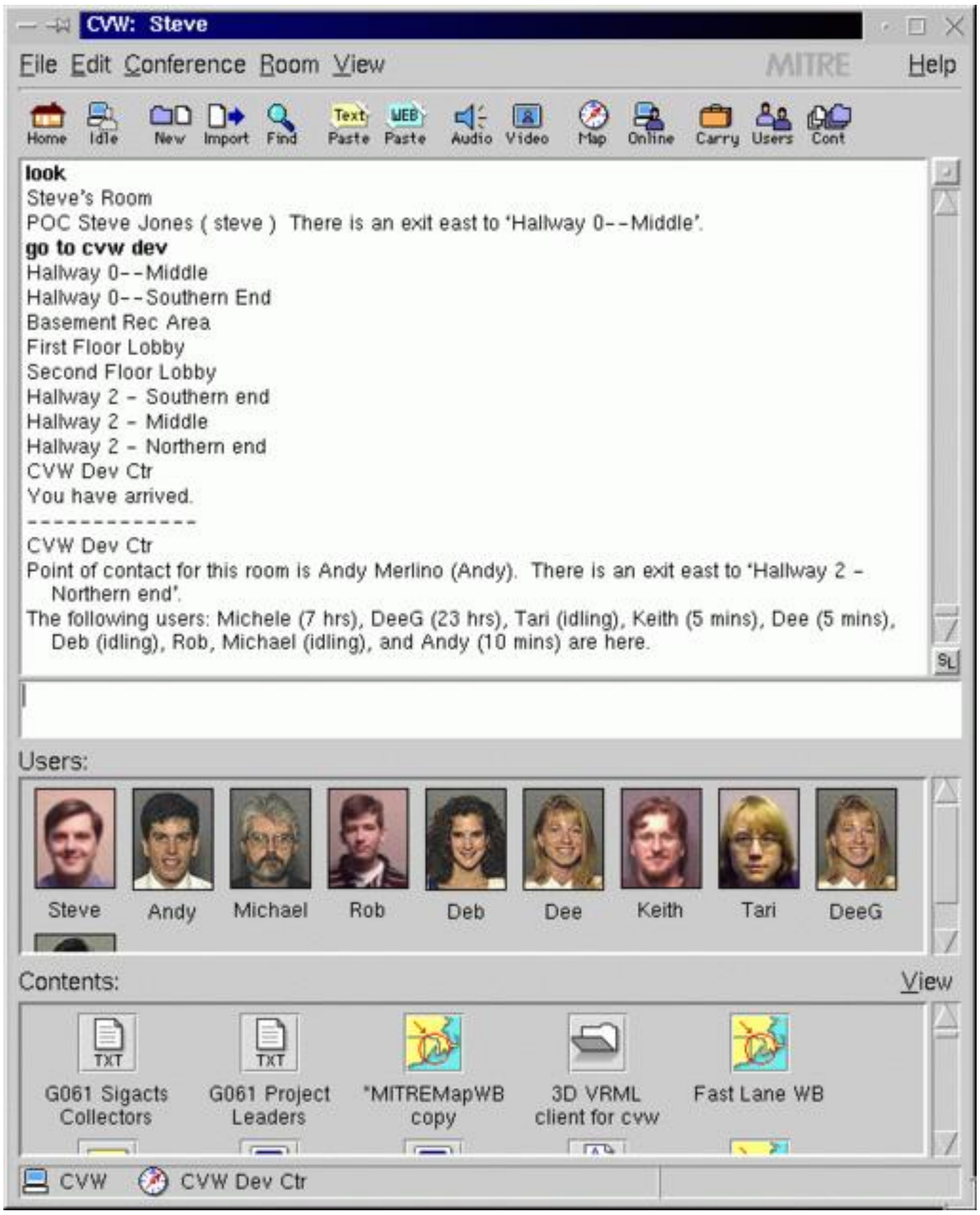
The project on which I currently work, the Collaborative Virtual Workspace (CVW), looks to add more meaning to computer-based social interactions. Instead of relying solely on spoken words, CVW adds actions, audio and video, and a more descriptive virtual environment to social settings. Additionally, users within the CVW environment can create and pass around various artifacts, such as documents and pictures. This is because CVW implements a persistent storage mechanism in conjunction with the environment. Anything created within CVW can be saved for others to see whenever they like.

Best of all, CVW is freely available. To further encourage innovation in collaborative systems, The MITRE Corporation has made CVW an open source software product. By disseminating CVW in this manner, different communities can begin to experiment with and understand synchronous computer-based collaboration. We believe collaboration systems such as CVW will continue to evolve, and the Internet community will have much to offer.

Let's look at where CVW comes from and what makes up our virtual environment.

Servers

From it's inception in 1994, CVW was built on a client-server architecture. At the heart of the server is a MUD, Object-Oriented (MOO) from Xerox PARC (ftp://ftp.lambda.moo.mud.org/pub/MOO/). Once only within the domain of dungeons and dragons players, MUDs and MOOs have progressed to incorporate numerous social settings. Although providing a very descriptive environment, MOOs present one challenge to the average computer user today—they are predominantly text driven. With no fancy graphic interfaces or elaborate designs, a user simply connects to a server using a MOO client program or **telnet**, logs in and begins interacting within the environment.


CVW: Steve
File Edit Conference Room View
MITRE
Help
Home Idle New Import Find Text Paste WEB Paste Audio Video Map Online Carry Users Cont
Users:
Steve Andy Michael Rob Deb Dee Keith Tari DeeG
Contents:
View
G061 Sigacts Collectors
G061 Project Leaders
"MITREMapWB copy
3D VRML client for cvw
Fast Lane WB
CVW
CVW Dev Ctr

```
look
Steve's Room
POC Steve Jones ( steve )  There is an exit east to 'Hallway 0--Middle'.
go to cvw dev
Hallway 0--Middle
Hallway 0--Southern End
Basement Rec Area
First Floor Lobby
Second Floor Lobby
Hallway 2 - Southern end
Hallway 2 - Middle
Hallway 2 - Northern end
CVW Dev Ctr
You have arrived.
-------------
CVW Dev Ctr
Point of contact for this room is Andy Merlino (Andy).  There is an exit east to 'Hallway 2 -
   Northern end'.
The following users: Michele (7 hrs), DeeG (23 hrs), Tari (idling), Keith (5 mins), Dee (5 mins),
   Deb (idling), Rob, Michael (idling), and Andy (10 mins) are here.
```

Figure 1. Room Screenshot

One of the true strengths of a MOO server is its extensibility. The server contains an object-oriented programming language called "MOO". Using the MOO language, programmers can create new rooms, implement new actions, and provide various means of interacting with others within the virtual environment. Add the ability to allow a large number of people to connect simultaneously, and it's easy to see why MOOs have become a popular collaboration environment.

If you download the LambdaMOO source code and run it on your Linux box, you'll notice not much exists in that environment by default. Each MOO administrator must either build a new virtual environment, or obtain the code someone else has written.

CVW's MOO server already contains the code for our virtual environment. We have extended our MOO server in order to create a large building, the floor plan of which displays graphically in the client. Although this tightly couples the client to a CVW server exclusively, it does provide a more intuitive interface for new users. The building consists of several floors, each with seven rooms and various other meeting places (hallway, lobby). These can be modified to suit any group's needs.

Using a room-based metaphor provides two other distinct advantages. First, session management can be logically partitioned. Each room will administer the particular session requirements for the users in that area. By having this controlled at the server side, any clients connecting to the server do not have to configure their audio, video or document settings to communicate with the other users. Secondly, security can likewise be easily controlled. As each room may implement an individual access control policy, private meeting places can be created. Again, this provides a very intuitive means of implementing security—if the server denies a user access to a room, they usually understand they don't have permission to enter that area.

In addition to the MOO server, we incorporated document persistence into our collaboration system. This allows CVW to encompass the role of a collaboration framework. If individuals meet to discuss a topic, they may create documents or graphics using whatever tools that may be available to them (document editors, drawing programs). Using CVW, users can import these external documents and allow others to view or edit them. Although MOOs do allow users to create objects, these are normally simple text documents that can't be easily exported to other external applications. This persistence capability allows for asynchronous collaboration—users may enter or leave the room whenever they desire, but they would still have access to the work being performed.

Since the addition of persistence incorporated a significant change to a MOO environment, we implemented this functionality as a unique server process, the Document server. This server does not have to run on the same machine as the MOO server, but it should have enough disk space available to store all the documents created by the users. Storage of documents is quite simple—basically, a flat database directory. To date this has proven sufficient, but a large implementation of CVW may want to look to re-engineer this process.

Living in Your Own Virtual World

The first step in creating a CVW environment is to obtain and install the CVW MOO server and document server software. If you browse the web site (http://cvw.mitre.org/) you'll find both a binary and a source build available for downloading. I'll step through the binary release installations, although the source releases require only a few more steps. Both downloads include detailed documentation, so I'll only highlight the important steps and try to point out any possible stumbling blocks for getting the server running.

As for system requirements, the size of your CVW system depends on the size of the virtual community you wish to build. My Linux system contains a Pentium 300 with 64MB RAM running Red Hat 5.1 and a 2.2 kernel. Although we haven't tested this configuration under a load, it should suffice for 40 to 50 simultaneous users at a minimum. We have seen over 250 users simultaneously on a Sun Ultra 2 dual-processor machine, so an appropriately configured Linux workstation should perform likewise. Additionally, the server running the document server process requires installation of the Java 1.1 JRE.

During the CVW software download process, after accepting the license agreement you'll be asked for some personal information. MITRE is a not-for-profit organization, and as such, does not use advertisements or sell user lists to other organizations. Our only reason for maintaining a user list is to inform you of any changes in the software and any other activities that might occur with the CVW software.

To begin installation of the CVW server, download the two binary packages, cvw-moo-server-3.2.0-x86-Linux2-libc6.tar.gz and cvw-doc-server-3.2.0-any.tar.gz, and place them in a temporary directory. Extracting these packages using the command **tar xvzf** ***`package_name`*** will create separate directories with the required binary files.

I decided to make /opt/CVWserver my installation directory. Before creating this directory, create a user named “cvw” on your system. This user will be the owner of the installation, so the binary won't execute as root. Although we haven't discovered any real problems running the software as root, this tends to be a good rule-of-thumb. After creating the user, make your installation directory using **mkdir** and assign your cvw user as its owner.

Move the following files into your installation directory from the moo-server binary directory—the moo executable, the restart script and the CVW.db file. If you like, you may rename CVW.db to something that better describes your environment. The name you choose doesn't matter, but you need to remember it, as you will be using it later on.

The Document server also requires a document repository. This repository directory must be large enough to store all your documents within CVW. Depending on your potential use, you may create a directory under the server directory you just created or mount a separate partition on your system. I created an /opt/CVWserver/docstore directory since I expected to have a small number of documents. You could also create a /docstore directory and mount another disk to it. The route you take depends a great deal on your projected use.

Next, the document server requires files to be installed from the document server binary directory. Move docserver.jar, docserver.cfg and the start-dserver script into your installation directory. You will need to modify **docserver.cfg** to point to your document repository. Finally, change all file ownerships within the installation directory to be owned by the cvw user.

To start the processes automatically when I boot my system, two init scripts have been provided. These include **cvw.boot** for the MOO server and **cvwds.boot** for the docserver. Move these scripts to /etc/rc.d/init.d and edit them for your personal configuration. By default, we use 8888 as our MOO server communication port and 8889 for the docserver; if any other programs on your system use these ports, change the numbers in the scripts to an unused port. Links can then be created from /etc/rc.d/rc3.d/S99cvw and /etc/rc.d/rc0.d/K99cvw to cvw.boot, and also /etc/rc.d/rc3.d/S99cvwds and /etc/rc.d/rc0.d/K99cvwds to cvwds.boot to allow the startup and shutdown to occur automatically. To start the processes manually, simply type:

```
/etc/rc.d/rc3.d/S99cvw start
/etc/rc.d/rc3.d/S99cvwds start
```

To test whether the server processes are operational, a TELNET session can be used. See Listing 1 for information on running these tests.

<u>Listing 1</u>

To present your users with a complete CVW experience, you'll want to build a repository of user images. These images will be presented on the client software as the users move from room to room to allow a virtual “face-to-face” feeling. The user images should optimally be 50x60 dpi gif images (although other sizes should scale within the client) that may be stored in one of two locations. First, if you have a distributed file system (e.g. NFS), the images may be stored there to allow for global access. The second option is to make the images available from an HTTP server. In either case, as long as the image repository is globally accessible, the client software can be configured to find the images.

Let me In!

Now we need to get a client. Since many computer platforms normally exist within any given area, we've built client programs that run under Unix platforms as well as Microsoft Windows. Even though users may prefer to work on different systems, they can still collaborate using CVW.

Two versions currently exist that can run on a Linux system: the original Tcl/Tk client and a newer Java client. Both are available in source distributions, but only the Tcl/Tk can be downloaded as a binary. Additionally, only the Tcl/Tk client can support administration functions, although we hope someone will want to fix this soon. For this reason, I'll discuss the installation of the Tcl/Tk binary distribution. However, I will touch on the Java source distribution later.

Download the Linux client distribution and save it into a temporary directory. For the Tcl/Tk client, retrieve the package named cvw-tk-client-3.2.0-x86-Linux2-libc6.tar.gz. Move to the directory where you want the client software installed, then run **tar xvzf** ***package_name***. In addition to the client software available from our web site, several tools can be obtained from other sites. These include **vat** for multicast audio-conferencing and **vic** for multicast video-conferencing originally developed by the Lawrence Berkeley National Laboratory (www.lbl.gov). Currently, we don't include these tools because we don't have permission to redistribute them. Once this hurdle has been cleared, we hope to include everything within the client package.

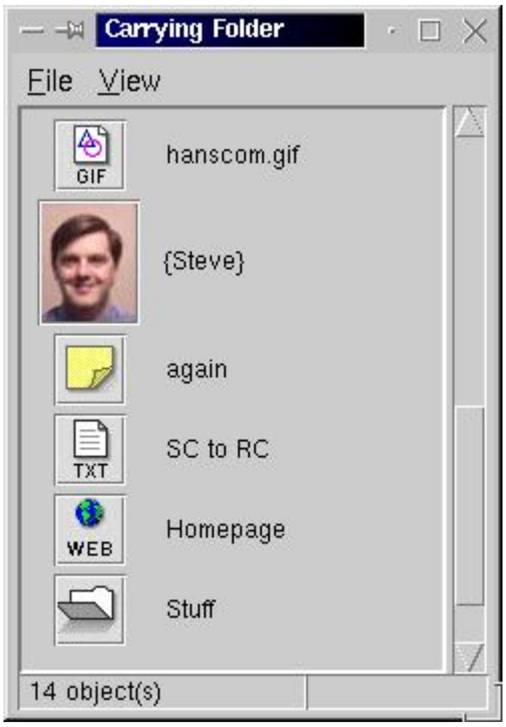
Another important configuration step for the client is to modify the mime types. CVW uses a simple mime lookup to associate file types with their parent application. This file can be found in the installation directory under lib/mime-db. To add new mime associations, simply provide an application to open a given document type for both reading and writing.

The README file fully details the remaining configuration required to connect to your CVW server. Once completed, executing bin/xcvw from your installation directory will start the application.

A default login of “User: admin, Password: admin” will allow you to log into your CVW server and CVW. After snooping around for a minute, you need to accomplish a couple more configuration steps.

The “Admin” menu on the main menu bar provides access to the system administrator functions. Open the “System Settings” menu selection and provide the requested information. Note you should provide a multicast prefix that will not interfere with other multicast users on the network. Multicast prefixes can range from 224.0 to 239.255.

Before proceeding with commands you can use in CVW, I'll describe more about your virtual user. Individuals will usually be assigned a normal user within CVW. This user has the ability to do a variety of things: move around the environment, create and read objects, place objects in a room or carry them along in a carrying folder, and communicate with others within the MOO. The users may also modify their description to personalize their appearance to others. Additionally, a user may be made the owner of a room, allowing him or her to modify the access list and the description of the room.



Carrying Folder

A user can be given additional privileges to become a programmer within the CVW environment. Although CVW doesn't recognize a programmer as having additional capabilities, a programmer can modify code and create objects within the MOO environment.

The CVW admin is the analogue of a MOO wizard. In other words, this user has the most capabilities within CVW. The admin has the ability to: create rooms and users, modify objects, assign privileges, and configure the CVW environment. Although admin capabilities can be assigned to anyone, it's advisable to limit the number of people having this permission.

After logging into CVW, you enter a building with several floors. On each floor, you'll find eight rooms and connecting hallways. Unless permissions intercede, you may move freely around the building. You may also pick up and examine objects that exist within CVW.

Now let's try it out. For the following examples, enter the MOO commands in the input window just below the scrolled text area. To find both extra information and examples, use the **help** command, either alone or in conjunction with the command you want to use:

```
help say
```

First, to speak to everyone currently in the room, use the **say** command:

```
say Hello, World
```

You may also direct speech to a particular person. Although others may see this conversation, the receiving party will notice you are directing the conversation to them.

```
to admin Hello
```

Finally, to communicate privately so no one else sees the communication, use the **whisper** command, or if the user is in another room, the **page** command:

```
whisper Hello to admin
page admin Hello
page admin !Hello
```

The last example displays a pop-up to the admin user. This command makes your comments more noticeable to the receiving party.

As in everyday human communications, actions sometimes speak louder than words. MOOs by default allow for these kinds of communications. Called an **emote**, a user may let their actions speak for them:

```
emote types furiously at the keyboard
```

Shortcuts for many of these commands can be found on the user images. By right-clicking on a user image, you will find a pop-up menu with several communication methods, along with an information selection. This menu will generate directed communications with the selected individual that can be seen by everyone.

Two other popular communication methods supported within CVW are audio and video. CVW implements these capabilities by tying in two common **mbone** tools from the Lawrence Berkeley National Laboratory: vic and vat. To launch these tools, press either the Video or Audio button on the top toolbar. You will notice no configuration will be needed to initiate conferences with anyone else in the room. CVW automatically handles session management tasks. Naturally, your Linux kernel will need to support the audio and video drivers necessary for your particular hardware to use this functionality.

You may not always be able to converse with others over CVW, either because you're away from your desk or busy on something else. CVW provides commands that allows you to let other users know you're unavailable:

```
idle gone fishing
busy working on project proposal
```

CVW provides two methods for moving between rooms. First, you can use the navigation commands (north, south, east, west, up, down) to simply direct where you would like to go. The **go to** command can also move you automatically to your destination:

```
go to CVW Help Desk
```

However, you may have noticed the CVW map button. Clicking on this button will open a separate window with a blueprint of your building. This map will also allow you to navigate around the building. You can choose the floor you wish to see by using the slide bar at the bottom of the window, and double-clicking on a room allows you to move quickly between rooms.

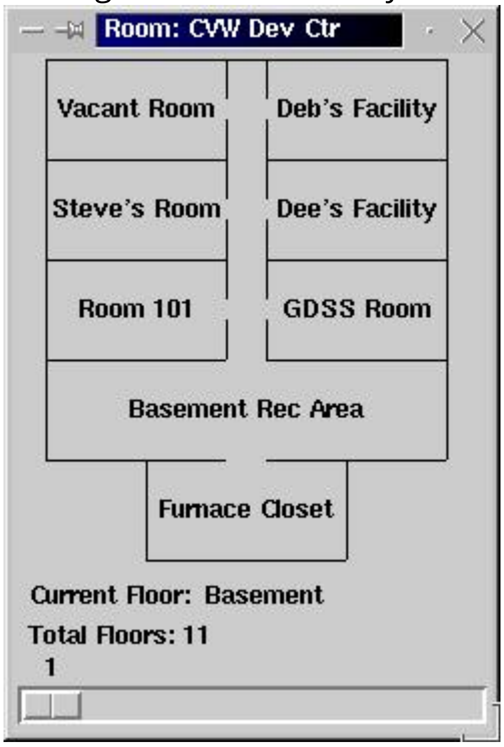
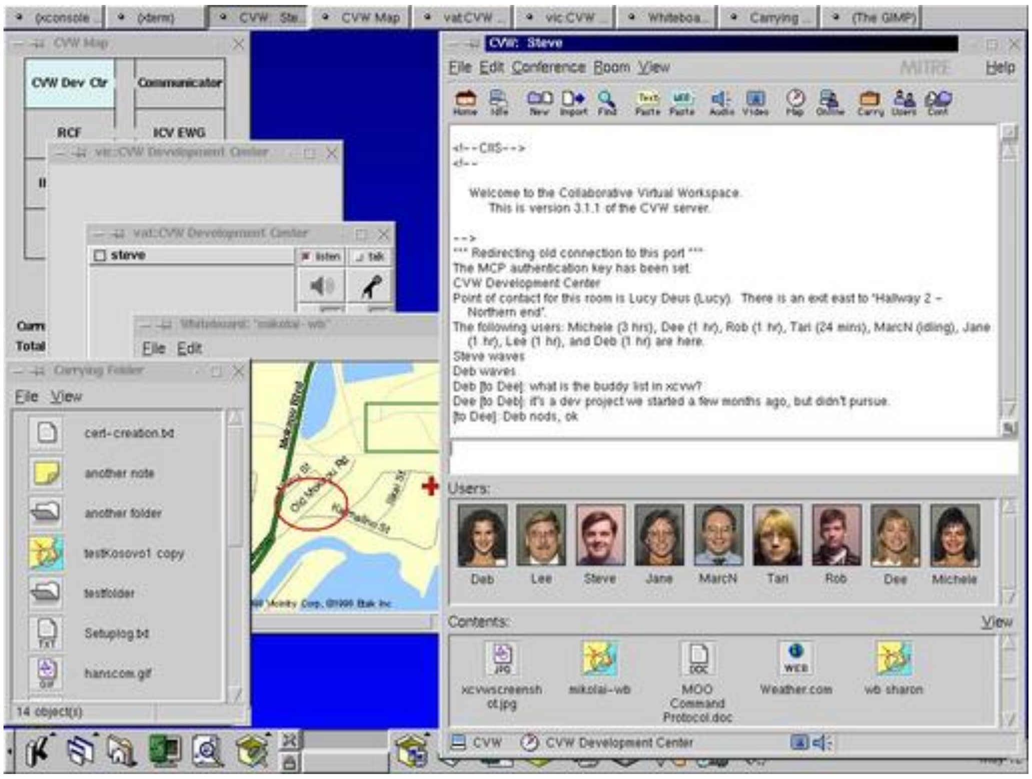


Figure 3. Building Map

Once in a room, objects currently residing in the room will be listed in the Contents window. Some of the objects you'll find include folders, documents, whiteboards and URL links. Menus for these objects can once again be found by right-clicking their icons. Options normally found on these menus will allow the user to view, copy, make a shortcut (link) or delete a document depending on the user's privileges for the item. You may also take objects from the room, which will move them to your carrying folder, and also drop items from the carrying folder back into the room.

The CVW client will allow you to perform many more actions with documents and people. These include importing and exporting documents, getting a listing of people connected to the server, finding particular people or items, and even creating groups of people to make communication easier. Rather than stepping through the myriad of options available (which can take four hours to teach), I'll instead point to our user-guide, *CVW 3.0 Unix Client Quick Reference Guide*, which can be found at our web site.


```
<!--CIIS-->
<!--

    Welcome to the Collaborative Virtual Workspace.
        This is version 3.1.1 of the CVW server.

-->
*** Redirecting old connection to this port ***
The MCP authentication key has been set.
CVW Development Center
Point of contact for this room is Lucy Deus (Lucy).  There is an exit east to "Hallway 2 -
   Northern end".
The following users: Michele (3 hrs), Dee (1 hr), Rob (1 hr), Tari (24 mins), MarcN (idling), Jane
   (1 hr), Lee (1 hr), and Deb (1 hr) are here.
Steve waves
Deb waves
Deb [to Dee]: what is the buddy list in xcvw?
Dee [to Deb]: it's a dev project we started a few months ago, but didn't pursue.
[to Dee]: Deb nods, ok
```

Figure 4. CVW Screenshot

The Future of CVW

As I mentioned earlier, a Java source client also exists for CVW. This client uses the Java 2 port recently released by the Blackdown organization (http://www.blackdown.org/). Although this client does provide some newer and more advanced features and also works on Linux, it can be considered a beta-release and requires some bug fixes. We do believe, however, this client will eventually become the standard CVW client. We hope that many users will find it interesting and build it up to match the capabilities of the Tcl/Tk client.

Some investigation into replacing the document server has begun. Although the current server performs sufficiently, it has several drawbacks: the interactions between the client and server can't be made secure, the repository consists of one large directory, and no real authentication mechanisms exist. We hope to address these issues with a new document server that provides greater capabilities and performance.

Since CVW has been made into an open-source project, MITRE will be concentrating more on developing a next-generation collaboration system. We hope many people will find CVW as useful to them as it has been to us and continue to improve on it. Since it is a relative newborn as an Internet project, many development possibilities exist. We hope CVW will continue to expose many people to the possibilities of computer collaboration.

Stephen Jones currently works as a software engineer for The MITRE Corporation. When he's not out on his Harley, he enjoys hanging out with his wife Lynn, two-year-old computer-fiend son Bradley, and new arrival, son Derek. He can be reached via e-mail at srjones@mitre.org.

Archive Index Issue Table of Contents

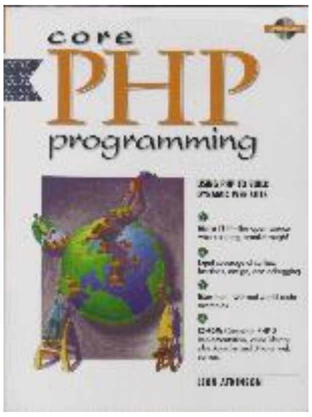
Advanced search

Core PHP Programming: Using PHP to Build Dynamic Web Sites

Allen Riddell

Issue #69, January 2000

The book, having already sold out of its first printing, has placed consistently in the bestsellers at the on-line computer book retailers.



- Author: Leon Atkinson
- Publisher: Prentice Hall
- E-mail: ptr_feedback@phptr.com
- URL: http://www.phptr.com/
- Price: $39.99 US
- ISBN: 0-13-020787-X
- Reviewer: Allen Riddell

The word “coattails” often comes up in political races, describing situations where a popular candidate, usually a presidential one, can carry other members of his party into office on the basis of his popularity—on his “coattails”. Such an analogy fits this year well. Instead of a year of elections, we have a year of Linux and Open Source, each carrying a wide array of programming languages and applications into the mainstream through association.

One language of note riding the Penguin's wake is a web scripting language called PHP that has gathered users at an impressive rate. PHP competes against a sea of languages used to generate “dynamic web pages”--pages that consist partly or wholly of code and are interpreted every time they are viewed. Rival languages include Microsoft's Active Server Pages (ASP), Allaire's Cold Fusion and even Perl. More established competition would be hard to find.

PHP's popularity is further evidenced by sales of the book *Core PHP Programming* by Leon Atkinson, one of the few available books on the language. The book, having already sold out of its first printing, has placed consistently in the bestsellers at the on-line computer book retailers. And, according to Atkinson's web page, a second edition is being considered.

With only one PHP book available, such popularity might be attributed to the subject rather than the quality of the book. In this case, PHP might as well be BASIC, for Atkinson's writing stands among the best in the expanse of computer books, bucking the trend of quantity over quality that plagues so many thousand-plus-page books. Atkinson's writing is uncommonly concise and comprehensive.

Atkinson begins sensibly with an introduction to the language, detailing its advantages over the competition. With something as intentionally transparent as a web scripting language, such discussion is vital. Surfers have little chance of knowing which sites employ any web scripting language. For instance, the popular software site for Linux, freshmeat.net, uses PHP exclusively, although users cannot detect it viewing the opening page. Atkinson advocacy is traditional for open-source software, arguing that it beats the competition in price and flexibility—notably, it can be modified freely and distributed.

PHP's introduction and language basics comprise the first of four sections in the book. The introduction is quite rudimentary, suitable for complete beginners to programming. Variable declaration and other important but introductory topics are discussed exclusively in the opening section. For many —if not most—readers familiar with Perl, a language that bears substantial similarity to PHP, drifting through basic concepts like type casting and **for** statements to find the key differences between languages may be frustrating. A section or two comparing PHP and Perl would have been nice. It is, however, not too problematic, as most readers will soon become familiar through examples or reference sections if they do not slog through the opening chapters.

Atkinson calls the second section a “functional reference”, combining a traditional function reference with detailed examples on each function. Despite the redundancy of nearly identical functions and examples for SQL database calls in MySQL, mSQL and PostgreSQL, the design serves another purpose. By giving an example with each function, Atkinson allows the reader to learn by example. Such instruction should suit the type of programmer who is familiar but not proficient with the language, assuring him with each example that he is using both the function and related functions correctly. And while the reference does take up the majority of the book, Atkinson keeps examples brief but useful.

For beginning users, the third section, entitled “Algorithms”, provides an interesting path for readers, building from stable searching and sorting methods like bubble and quick sort, to more functional examples like retrieval from SQL databases. Such a design may strike more advanced users as somewhat silly, since no need for a bubble sort exists when using SQL or most other databases commonly accessed by PHP. Indeed, actually using PHP to sort a large database would be inadvisable—such a scheme would essentially entail compiling and executing code to sort the database every time a browser viewed the page. While parts of the chapter do seem better suited for an introductory computer course, struggling users may find familiar computer-science algorithms implemented in PHP helpful.

In the final section, Atkinson shines, taking the crucial step away from teaching the language to applying it in real-world examples and advising how to make the language a true alternative to the heavyweights of ASP and Cold Fusion. He calls the section “Software Engineering”, in reference to the increasingly popular term “Web Engineer”--comparing the construction a programmer does to that of a civil engineer. Discussing issues of design, efficiency and debugging, Atkinson outlines the important aspect of using PHP, doing useful things with it, namely building web sites.

Considering PHP's rivals, discussion of web engineering with PHP serves one purpose beyond instruction: it adds to the credibility of the language as a serious rival to its high-priced and high-powered competitors. Through discussing how to efficiently design an expansive and maintainable site—something quite uncommon in an introductory text—Atkinson gets to the heart of one possible misconception of PHP, that it bears more resemblance in ability and function to something like JavaScript than a more “powerful” and “mainstream” package like ASP. The criticism has as little truth here as when levied at Linux and open-source applications, and in his discussions of databases and of efficiency in coding, Atkinson develops ways of unifying a site under PHP, employing it as a tool to build an underlying framework rather than as a method of enhancing a pre-existing one.

For his part, Atkinson has done a wonderful job. His precision and organization makes reading and learning PHP simple. What should concern those interested in PHP is not how to learn it—Atkinson has solved that problem—but where PHP, the language itself, is headed. As with many open-source applications, PHP is evolving quickly. One rather drastic change currently taking place is a rewrite of the underlying “engine” of PHP. The project to do so is called “Zend” and also involves a licensing change. While the promise of backward compatibility exists, the prospect of new and better functions or methods of coding looms. For some, this may be a reason to use caution; for others, another reason to start learning the language.

PHP's future is bright. Atkinson's book is an excellent companion for anyone interested in learning a promising language—fast growing coattails of its own.

Allen Beye Riddell (ariddell@concentric.net) currently attends Stanford University and has worked as a technology consultant for two years. PHP has been his language of choice for the past year—running on a Linux server, of course.

Advanced search

AppSwitch: Network Switching with Ada from Linux

Ann S. Brandon

Issue #69, January 2000

Software to automatically sort network communications to your specifications.

Managing network performance is like keeping an ant colony orderly: the network has its own criteria for what to do next, the logic behind which humans often cannot fathom. The challenge is actually even more impossible, for in nature, ant colonies consist of the same species. In networks-as-ant-colonies, the database transmissions and e-mails are all different species. Some are compatible, some not. Most are friendly, some hostile, a few the equivalent of anti-personnel mines.

In order to control data communication networks, Top Layer Networks, a new company located in Westborough, Mass., developed the AppSwitch. The software automatically sorts network traffic according to both a user's priorities and the application that is generating the messages. It penetrates up to Layer 7 of an e-mail's headers, for example, which can mean translating the code of seven different “species” of applications and priorities.

The prototype for the AppSwitch 2000 was written on top of a Red Hat 5.1 Linux system. The Top Layer executives assessed Linux as too rich for the application, however, and chose to program the fielded system in Ada95. The GNAT compiler from Ada Core Technologies (ACT) allowed a seamless and fluid transition from Linux to an Ada95 system. The New York City company's stable of Ada compilers is all open-source software and available at http://www.gnat.com/.

Linux was chosen as the prototype platform OS because ACT targeted a compiler to it. Also, the Top Layer software engineers are familiar with UNIX-like operating systems. The operating system for the Back Engine of the AppSwitch is a modified version of the GNAT runtime system. As part of its modification, Top Layer also created an OS kernel that supports a POSIX interface, which Linux also supports.

After the simulator proved feasible, Ada95 was used for 90 percent of the AppSwitch's code, with the remaining ten percent being low-level commands implemented in C and assembly. Top Layer still uses Linux as the “front-end” console, i.e., the gateway between the AppSwitch and the software engineers' workstations as they further develop and test the application's next release. A new AppSwitch 2500 and 3000 are expected in fall 1999 and winter 2000.

Does building a network in Ada, a field which C dominates, make sense? Top Layer evaluated their options and decided “yes”.

First, the AppSwitch software is expected to have a long lifetime. The programming language must be portable among different target machines and generations of target machines, including those not yet invented. In other words, it must be a national and international standard, as Ada is. Yet it must not make their users or themselves hostages to one target or compiler. Again, Ada meets that need.

The Ada standard is tweaked through the following bureaucracy. The ISO's Working Group 9 responds to software development tool vendors' comments and arguments about the standard's interpretations of Ada. If the WG 9 votes on the side of the vendor, the change is reflected in a Validation Test Suite. If a compiler passes the majority of the tests, it receives a certification from the Ada Resource Association (ARA), which is an organization of software development tool vendors who cover over 90 percent of the Ada market. When an developer buys an ARA “Certified Ada” compiler, she knows the compiler has been tested to interpret “standard” Ada. The interpretations have been agreed upon internationally.

Top Layer's second criterion in selecting a language was that the software must work right the first time and continue to execute in the presence of faults and software reconfiguration. The company judged Ada to have the best combination of language features for the two requirements of high reliability and portability, including strong typing, object-oriented features, multitasking and exception handling.

Ada is not “another damned acronym”. The language is named after the first published computer programmer, the Countess of Lovelace, Ada Byron King. Yes, that Byron; she was the poet's only legitimate daughter. A mathematician, she worked with Charles Babbage on the Difference Engine, and prophesied the computer's many applications, such as writing music, calculating variables, etc., as well as the potential of reusing code, which the Ada language was designed to fulfill.

GNAT was chosen as the tool chain because it is based on GCC and GCC is targeted to the ARC processor. The Ada tools are hosted on Solaris and Linux and targeted to the Motorola MPC 860, the ARC and x86.

The AppSwitch is a multiprocessor system. The software architecture is designed to take advantage of Ada95's distributed system features as the implementations mature. The AppSwitch analyzes network application traffic and enforces quality of service policies automatically, i.e., with no human intervention and no changes to PCs or servers. The AppSwitch coordinates both hardware, to examine and direct traffic at wire speed, and software, to guide the switching and the network's performance. Clients' business objectives decide the AppSwitch's policy criteria, a process that occurs with quotidian emergencies and projects, yet proves evolutionary as the business's function and size change. AppSwitch immediately benefits a business by enforcing criteria that allow the network to resemble a human messaging system more than an ant's.

With its integrated hardware and software architecture, the AppSwitch lets a network administrator monitor and understand all the elements of the network. The administrator adapts the network to changes through the AppSwitch's web-based graphical user interface.

The hardware and software components that constitute flow classification technology are at the core of the AppSwitch. With this technology, the AppSwitch goes deeply into the incoming packets, up to the application's Layer 7 headers, to determine the packets' source, type and destination addresses. The pre-configured Application Definition Library (ADL) supports the AppSwitch flow classification. The ADL is built into every switch and holds the rules for flow classification and prioritization.

When the first packets of a flow arrive, the AppSwitch uses a connection-setup-process subarchitecture to select the appropriate “tailored” profile in the ADL. It also establishes pertinent information on the new flow, which is captured in the session database. After that, the AppSwitch routes remaining packets in the flow through the coordination of the session data and the selected profile. The state data captured by each flow in the session database can intelligently adjust the quality of the service as needed to upgrade or downgrade priority. On the output side, thousands of queues are available to expand the number of flows and priorities among those flows.

The AppSwitch works behind a router or in a LAN environment. Incoming packets are subjected to proprietary “triage” by the media-access-control function (MOM) components to broadly categorize the packets. It also puts the packets into a canonical form that is data-link-independent. From there, the “categorized” packets flow over the TopWire to the Forwarding Engine (FE Chip + QM Chip) where the application control is performed.

Ada was applied in an uncomplicated manner to both the Forwarding Engine architecture and the Background Engine. While meeting deadlines is important, the AppSwitch is not strictly a hard real-time application. Top Layer's engineers eliminated some restrictions on the software from a traditional real-time application. They therefore permitted greater flexibility in the software architecture to accommodate the dynamic nature of data communications.

Ada contributed to Top Layer's success in meeting an aggressive nine-month development schedule. That is fortunate, for the decision to program in Ada met with resistance from the predominantly C shop. The engineers' learning curve steepened and grew icicles as they wondered how “Ada” will read to future employers looking for the letter “C”.

Linux users everywhere will sympathize with management's challenge to make Ada not only palatable but also preferable to the reigning zeitgeist of Microsoft, which includes C and C++ as well as the Windows OS.

They succeeded. Once Top Layer corporate executives decided Ada was the best choice, they stuck with it. Their commitment to Ada was top down and unwavering. They hired a full-time Ada expert, Mike Kamrad, to provide intensive internal training in and advocacy of Ada. The result was a personal education for all the software engineers. Finally, Top Layer took extra steps to make Ada work technically on the target systems, thereby providing a trusted and well-performing tool chain and avoiding further resistance and frustration to learning.

Like the AppSwitch itself, the engineers found that Ada speaks a sensitive, sophisticated and robust language. Ada enabled Top Layer to create an Esperanto for network administrators. The AppSwitch allows them to interpret the e-mails and transmissions' unique signals into the international and human-friendly language of graphical interfaces. The goal is to direct the multitudes of unique ant-like bytes into the tunnel-like channels that will most efficiently build and strengthen a business. The AppSwitch makes that goal possible, but only because Top Layer, in using Linux and Ada, bushwhacked their way off the worn trails of network switching systems.

For more details, see http://www.adaresource.org/ and http://www.toplayer.com/.

Ann S. Brandon (abrandon@sover.net) has written about the Ada language for the last eleven years. As Ann Eustice, she worked as an editor for the Daily American in Rome, and a monthly business magazine also in Italy, and in PR at United Press International (UPI) in Washington, DC. UPI being her third employer to go bankrupt, she decided to learn about computers, and has not had a paycheck bounce on her since. She is married to an Ada and Physics professor, Carl, and they live in Vermont. As a writer, Ann is also an Associate Editor for Invisible Cities Press.